Master Thesis

What are my chances?
Predicting social housing allocation with AI

Author: A.J. Schaefers (2730261)

1st supervisor:	Prof. dr. A.E. Eiben
daily supervisors:	M. van den Bos & B.J. Koerhuis
2nd reader:	Prof. dr. R.D. van der Mei

A thesis submitted in fulfillment of the requirements for the VU Master of Science degree in Business Analytics

July 19, 2024

"I am the master of my fate, I am the captain of my soul"
from Invictus*, by William Ernest Henley*

Acknowledgements

This thesis has been written for the completion of my Master of Science degree in Business Analytics, specializing in Computational Intelligence, at the Vrije Universiteit Amsterdam. Over the past six months, I was privileged to be an intern at WoningNet, writing this thesis on predicting individuals' chances of being allocated social housing. I applied a hybrid approach using both traditional and machine learning models.
I especially would like to thank Bas Koerhuis and Marcel van den Bos for their guidance and support at WoningNet. Their advice, sector knowledge, and insights into WoningNet were invaluable. I truly felt supported throughout this journey, where brainstorming sessions and stand-ups were particularly helpful. I am also grateful to Guszti Eiben for his supervision through the study group at VU. His mentorship and feedback were instrumental in the development of this thesis. Lastly, I wish to thank Rob van der Mei for being the second reader of my thesis.

Annemarie Schaefers
July 2024

Abstract

Context. In our time many seekers for housing miss a clear insight into their chances of securing a property and how to adjust their housing preferences to access better listings on the website. In addition, the scarcity of social housing exacerbates this situation, as high demand and limited supply of social rental properties further reduce their chances.

Goal. Supporting prospective tenants is crucial for WoningNet to maintain the satisfaction of current customers and distinguish itself from its competitors. WoningNet's Customer Contact Center (KCC) currently handles inquiries regarding the allocation chances of prospective tenants, a time-consuming process. This study aims to predict the allocation probability for prospective tenants. WoningNet intends to display this information on its website to assist tenants in identifying houses with better prospects or to effectively adjust their search criteria.

Method. The allocation probability is closely related to the final position on the applicant list in the allocation process; therefore, the final position is predicted. Due to the absence of reproducible studies in this area, a hybrid approach is utilized, combining mathematical methods for predicting the final position with machine learning techniques for predicting the number of responses to a listing. The mathematical methods compared include a simple mean approach and a novel sorting ratio approach. This study demonstrates that accurately predicting the final position requires predicting the total number of responses on a listing. To predict this, the performance of five machine learning models (SVR, Random Forest, Extra Trees, LightGBM, and CatBoost) is compared, with models tuned using Random Search. Additionally, different feature sets are assessed for their effectiveness. The performance of the models is evaluated using RMSE and R-squared metrics.

Results. The results show that the mean approach without scaling performed poorly (R-squared = 0.12). Incorporating the number of responses significantly improved its performance (R-squared = 0.99). The sorting ratio approach achieved an R-squared value of 0.97. All models for predicting the number of responses demonstrated similar performance, with R-squared values between 0.84 and 0.89. The minimum age selection criterion emerged as the most important feature across most models. CatBoost achieved the highest performance with the fewest features. When using the predicted number of responses in the position prediction models, the scaled mean approach achieved R-squared values between 0.79 and 0.86 and RMSE values between 126 and 156. For the sorting ratio approach, the R-squared values ranged from 0.75 to 0.81, and the RMSE values ranged from 130 to 151. Among the models, LightGBM achieved the highest performance, with R-squared values of 0.86 for the scaled mean approach and 0.81 for the sorting ratio approach.

Conclusions. The simple mean approach for predicting the number of responses performed inadequately on the test dataset. Although incorporating the number of responses made the model highly accurate, this approach is difficult to scale, necessitating the development of a second model: the sorting ratio approach. The sorting ratio approach also achieved high performance. However, it requires the number of responses per listing, which is unknown in practice. Machine learning methods for predicting the number of responses performed reasonably well. By integrating these machine learning predictions with the position-based approaches, the performances were reduced but remained fairly good, with an R-squared value of 0.81 for the scalable sorting ratio approach. Among all the models, LightGBM achieved the best results.

Contents

1 Introduction

1.1 Housing shortage in the Netherlands

In the Netherlands challenges regarding housing availability are persistent with shortage reaching 390,000 houses in 2023 ([30]). Experts attribute the housing shortage primarily to the collapse of new construction following the 2008 credit crunch ([23]). From 2000 to 2008, there were huge construction efforts, with approximately 790,000 new homes built annually ([6]). After this period this positive trend took a downturn. Post-2013, construction rates drastically fell to a meager 50,000 homes per year, increasing the housing shortage ([6]). Concurrently, the growing population and immigration strained housing resources further. Moreover, building permit issuance in 2021 faced obstacles due to nitrogen and PFAS issues. The introduction of the Code of Conduct for Mortgage Loans in 2011 added complexity, particularly impacting middle-income earners, who found themselves underserved in both the social housing and homeownership markets. Soaring rents within social housing and economic instability further widened the socioeconomic gap, leading to differences between lower-income renters and wealthy homeowners. This led to increased spatial segregation within the social housing sector, primarily serving low-income and retired individuals (64% of residents in 2018).
The shortage of social housing is a significant contributing factor to the housing scarcity in the Netherlands, particularly impacting individuals with limited income. Housing corporations increasingly use priority criteria and urgency categories to allocate social housing. According to a survey conducted by the newspaper de Volkskrant across Dutch municipalities ([38]), this puts less emphasis on waiting lists, especially in areas with severe housing shortages. Research by the news channel NOS in 2021 showed alarming statistics: in 25% of municipalities, seekers for social housing faced waiting periods exceeding seven years.

In Amsterdam, not even ranking among the top 10 municipalities with the longest waiting times, the waiting period stretches to almost 14 years ([24]). The decrease in the proportion of social rental homes in the housing stock over the past decade exacerbates this situation. Housing corporations did not build enough new social rental houses compared to owner-occupied and free-sector rental houses to compensate for this decline. According to housing corporations, there is too little flow between homes. This is often due to the unavailability of suitable housing options, compounded by the fact that moving is often not an attractive prospect. Also, imposed carbon reduction goals ask for increased renovation budgets and in 2013, the government introduced a special tax for housing corporations, known as the landlord levy ([30]). This put considerable financial pressure on them, reducing their ability to invest. Where municipalities previously mainly relied on housing associations to build affordable rental properties, they have increasingly turned to commercial parties to bridge the gap in affordable rental properties ([1]). Since last year, this levy has been abolished and while the abolition offered hope, housing associations are still struggling to regain this lost ground. The national association of housing corporations, Aedes, is concerned. Aedes does not oppose other parties building affordable rental properties, it highlights that commercial parties operate outside the allocation rules followed by housing associations, including imposing income requirements on renters. Moreover, affordable rental properties built by commercial parties frequently shift to the private rental sector after 10 or 15 years, with higher returns for landlords. Aedes has requested the Kadaster (Land Registry) to assess the types of homes built from 2017 to 2020, revealing that only 16% were corporation-owned, a much lower number than the typical municipal targets of 30 to 40% social housing ([1]). In several municipalities, particularly around major cities, there are indications that hardly any rental properties have been built in the regulated segment in recent years ([1]).

1.2 Role of WoningNet

Facing those challenges, organizations like WoningNet have a crucial role in bridging the gap between housing supply and demand. Through its website, WoningNet facilitates connections between housing supply from corporations and prospective tenants, with a primary focus on social housing. Over 200 housing corporations are currently affiliated with WoningNet. Regarding housing allocation, this encompasses roughly 100 housing corporations. The organization aims to provide comprehensive support to prospective tenants while maintaining strong connections with municipalities and housing corporations.

With advice, WoningNet offers guidance to collaborating housing corporations on reaching their goals and aids municipalities in translating policies into allocating rules. These goals frequently involve allocating homes to the appropriate target group. WoningNet strives to support prospective tenants in responding to suitable offers and managing expectations. This includes ensuring that their websites are user-friendly and easy to navigate.

When allocating houses, a set of criteria governs the process, including selection and sorting criteria. These criteria are in place to ensure fair distribution of homes among prospective tenants, given the current situation of demand exceeding supply, and following European, national and local regulations. During the application period, prospective tenants can only apply for a house if they meet the selection criteria. Those who meet these criteria and actively seek housing are termed suitable home-seekers. Each suitable home-seeker is assigned a minimum position on a list based on the sorting criteria. This minimum position represents the position a person would hold if every suitable home-seeker applied for the house. Sorting criteria vary by region, municipality, and property. Additionally, properties with higher demand may have more decisive rules, ensuring fairness in applicant selection. A final applicant list is compiled from all respondents to the house publication. The list ranks applicants from 1 (the best match) to the lowest position (the least suitable match).

1.3 Impact of uncertainty in social housing

Besides the rational complications of allocating a house, there is also the profound emotional impact on individuals waiting for social housing. Research has been conducted in Australia on the impact of waiting for social housing ([29]). Although focused on the Australian housing market, the findings have global relevance. The researchers found that waiting for social housing is marked by triple uncertainty, adding another layer of unpredictability to individuals' already unstable housing and job situations. This prolonged wait significantly affects individuals. The housing crisis results in long waiting lists, especially for those with disabilities. During the wait for social housing, individuals often face difficult living conditions, with rental stress and insecurity in the private rental sector, possibly resulting in homelessness. Pursuing social housing may discourage them from pursuing a career, as higher income might negatively affect their position on the waiting list. Lastly, prolonged waiting contributes to poor physical and mental health among those waiting, exacerbated by the uncertainty and lack of control over their future.

1.4 Problem statement

WoningNet aims to provide comprehensive support to prospective tenants, with user-friendly websites. Currently, prospective tenants often lack indications of their likelihood of securing a property and the waiting time involved. This leads to numerous inquiries received by the Customer Contact Center (KCC) of WoningNet. WoningNet seeks to address this issue by improving the information available on its website, ultimately providing prospective tenants with an indication of their chances of securing a property. This additional insight could help prospective tenants adjust their search criteria, such as the number of rooms, resulting both in better chances and a reduction of the volume of questions handled by the KCC. This research aims to develop an algorithm to predict the likelihood of a housing seeker being allocated a home. And offer prospective tenants support by providing comprehensive and effective information on the website.

1.5 Research question

This study aims to predict an individual's likelihood of securing a social housing unit before the final applicant list is known. An individual's likelihood is closely related to an individual's position on the final applicant list. This research aims to assess why the number of responses is needed to predict an individual's final position accurately. Therefore, this study aims to answer the research question:

> *"How does incorporating the predicted number of responses affect the likelihood or final position prediction of a prospective tenant securing a social housing unit?"*

To answer this, a hybrid approach will be utilised. This involves machine learning algorithms to predict the number of responses, which is necessary for accurate predictions. Furthermore, the individual's likelihood will be determined based on their final position on the applicant list using traditional modelling techniques.

1.6 Outline

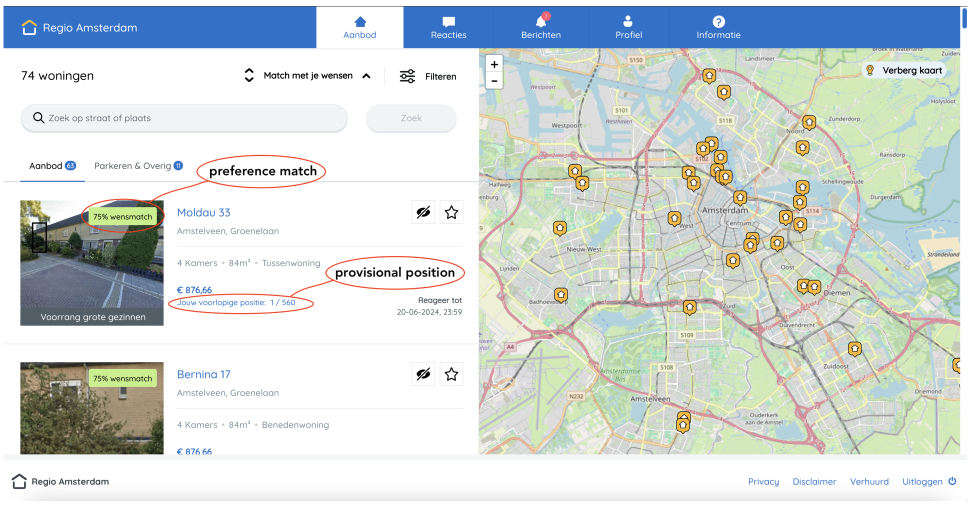
The thesis is divided into several chapters. Chapter 2 describes the background of the problem and the complexity of the problem. Chapter 3 reviews the existing literature, which highlights the scarcity of research on this specific topic and draws parallels with university admissions studies, noting their lack of reproducibility. Chapter 4 describes the data used in the study, while Chapter 5 explains the methodology, including the pros and cons of the used models. Chapter 6 presents the research results, followed by Chapter 8, which concludes the findings. Finally, Chapter 7 discusses the implications of the results and possible future research directions.

2

Background

2.1 Website

A prospective tenant visits the WoningNet website and logs in. Based on the entered data and preferences, the user sees a selection of properties that meet their criteria. The order of listings received is based on the so-called "preference match", meaning how closely the users' filters match the listing. This preference match can also be used to find parking spaces or houses in the free sector. However, the majority of WoningNet's listings cover social housing. This study therefore only focuses on social housing.



Moldau 33
Amstelveen, Groenelaan
75% wensmatch
Voorrang grote gezinnen
4 Kamers • 84m² • Tussenwoning
€ 876,66
Jouw voorlopige positie: 1 / 560
Reageer tot 20-06-2024, 23:59

Bernina 17
Amstelveen, Groenelaan
75% wensmatch
4 Kamers • 84m² • Benedenwoning
€ 876,66

Figure 2.1: Website WoningNet (https://amsterdam.mijndak.nl/)

The preference match provides an interesting overview. However, in scarce supply, the prospective tenant will probably only see listings that do not entirely meet their preferences. WoningNet wants to provide prospective tenants insights into their chance of allocation and how preferences entered influence the properties displayed. For instance, would they have a better chance if they specified one less bedroom? Prospective tenants can respond to a maximum of two listings simultaneously. Once a listing period ends, the person receives their position on the final applicant list, where position 1 has the highest priority. As described in the introduction, it is aimed to both support prospective tenants and alleviate the KCC workload. WoningNet aims to prevent withdrawals from listings before the end, as this confuses the provisional position of others. The provisional position provides an indication based on all prospective tenants who have responded to the property at that moment. However, applicants that respond early may see a high position which is inaccurate compared to the final position. Additionally, responding for an applicant is limited to two listings at a time. Displaying predicted responses, predicted final positions or predicted chances can support applicants in making informed choices.

2.2 Allocation rules

Properties can be allocated based on availability, direct mediation, or a lottery system. Direct mediation is not conducted through the WoningNet website and is excluded from this research. In the availability model, prospective tenants can respond to listings that meet selection criteria and the property is assigned based on sorting criteria. In contrast, lottery properties are allocated randomly but may have sorting criteria, such as prioritizing residents of a particular region. Each property has a unique set of selection or sorting criteria. These selection and sorting criteria together constitute a single allocation rule for each property. Therefore, every property is governed by one specific rule. The allocation rules for property distribution, are determined by policies at European, national, regional, and municipal levels. These rules pertain to the distribution of properties, such as the percentage allocated to specific target groups. Consequently, different types of properties may target different groups and receive varying numbers of responses. For instance, an apartment might be open to all target groups, while a larger house with a stair lift may prioritize individuals with mobility issues due to the high cost of modifying properties. Selection criteria define how many prospective tenants can respond to a property, but properties with the same selection criteria may still receive different numbers of responses based on location and other characteristics. Once a listing is selected, sorting criteria determine the applicants' positions. Municipal and regional policies create differences in these selection and sorting criteria, leading to numerous local variations. The order of the criterium is not fixed either; a criterium might be in the first position in one listing's sorting text string but in the fifth position in another. Often, the registration date is the final criterion. Most sorting criteria are binary—either met or not met. However, not all listings with the same selection criteria receive the same number of responses, as factors like neighbourhood can significantly affect attractiveness. The number of sorting criteria also varies, with some listings having as many as eight sorting criteria. The order of these sorting criteria is essential, another order can result in an entirely different final position. After the listing closes, the final applicant list is created, listing everyone who responded to the property and did not withdraw before the listing end date. Each person has a sorting string, which is used to order the applicant list. While it may seem obvious that only one admission is granted per property, it's important to note that the number one on the list doesn't always secure the property. This can occur if the applicant declines for various reasons, such as dislike of the property or being listed as number one for another property. Additionally, corporations may reject an applicant if inaccurate personal information is provided, resulting in individuals with a lower position on the list to secure the property.

2.3 Complexity

Listings can be highly diverse; not every similar type of housing has the same selection and sorting criteria. Listings include a mix of senior, youth, and regular housing, with significant differences in response numbers due to varying selection criteria. Each region has grouped criteria, but these can vary for each listing. Criteria can be enabled or disabled, the order of importance can be changed, and quantities can be adjusted. For instance, priority given to families larger than two members can be adjusted to families larger than four members. House scarcity can result in thousands of responses for specific property listings, while other listings receive few responses due to stricter selection criteria or less attractive features. The total number of responses to a listing is important since a higher number of responses decreases the chances of success for an applicant.

When predicting an individual's chance of housing allocation, the final position on the list is crucial as it determines the order in which offers are made. Since the order of sorting criteria is predetermined, the key question is how many and which people will respond to a listing. One approach could be to predict the final position (important for chances) by forecasting who will respond. Data such as all active prospective tenants and the minimum position are calculated but not stored because they require substantial data storage. Theoretically, it should be possible to connect multiple data sources and determine who is an active seeker however, scarcity leads to a tremendous amount of active seekers per listing. The unavailability of suitable data resources made this process too complex and time-consuming for the scope of this research. The available data includes the applicant list per property and the characteristics of these properties. Creating a single model that works across all properties, regions, target groups, and rule sets would be ideal for model implementation. It is assumed that at least one response is received per property, as no applicant lists have zero length.

3

Literature Review

This chapter discusses literature relevant to this research, comparing the distinctions between machine learning and traditional models. It also describes similar research within university admissions and social housing domains, highlighting the rationale behind adopting a hybrid approach.

Traditional programming has long served as the basis for software engineering. In recent years, more and more companies have been using machine learning (ML) because of its remarkable performance in various domains. ML models are trained to learn from data and make predictions or decisions. Unlike traditional programming, ML can deal well with complex problems and automate tasks that are difficult to program explicitly. ML excels in scalability, handling vast amounts of data, and can improve models continuously through retraining. However, the performance of ML models relies heavily on the quality and quantity of training data. Despite their remarkable performance, some models, especially those in deep learning, are criticized for their 'black box' nature and lack of interpretability. Training complex models can be computationally expensive and can lead to overfitting, where models perform well on training data but struggle with new, unseen data. ML algorithms are generally more accurate than human-made rules because ML algorithms consider all data points in a dataset without human bias due to prior knowledge. Traditional programming explicitly defines the rules and logic that computers must follow. It is best suited to tasks with clear, deterministic rules. Traditional methods are predictable and consistent. This gives developers complete control over the outcome of models. However, they face challenges in terms of scalability and processing huge datasets or complex patterns. These models require manual updates to adapt to new scenarios, which can be difficult for problems with high variability or uncertainty.

In the context of social housing, only a handful of studies leveraging machine learning techniques have been conducted and published. However, these studies focus on predicting mental health and analyzing floor plan images for design ([4]) ([25]). Despite efforts, searches with terms like "affordable housing," "social housing," or "public housing" in combination with “machine learning”, “data science”, and “AI” did not yield significant findings. While an industry blog highlights the potential value of artificial intelligence (AI) in the Dutch social housing market ([21]), there is a noticeable absence of papers or articles on the subject.

When examining areas of research that parallel the challenges of housing allocation, university admission emerges as a prominent field of comparison. The research area into university admissions mirrors social housing allocation, as both face constraints of scarcity and adhere to predefined regulations which provide a sorted list. Furthermore, in both scenarios, individuals who are admitted may later opt to decline, thereby creating an opportunity for someone else. The competition for spots in esteemed programs like MTech or MBA is intense due to high student demand. Multiple studies emphasize the value of utilizing machine learning to forecast admission probabilities. This aids students in selecting the most promising universities to apply to and assists institutions in effectively handling application volumes. Several research papers delve into predicting admission probabilities using various machine-learning techniques. Joshi Padma et al. ([18]) employed Linear Regression, Decision Trees, and Random Forest to forecast the likelihood of student admission, with Linear Regression yielding the most accurate predictions at an 82% accuracy rate. In a study of Sivasangari et al. ([33]), the CatBoost algorithm, without tuning, emerged as the most accurate predictor with a 95% accuracy rate, surpassing Linear Regression and Random Forest models. El Guabassi et al. ([12]), Omaer Faruq Goni et al. ([31]), Chakrabarty et al. ([8]), and Maulana et al. ([27]) all focus on predicting the chance of admission El Guabassi et al. ([12]) demonstrated that Random Forest Regression outperformed other algorithms with an 89% accuracy rate in predicting college admissions. Omaer Faruq Goni et al. ([31]) utilized a DNN model and showed that it scored better than previous models, achieving a 0.8538 accuracy. Chakrabarty et al. ([8]) and Maulana et al. ([27]) both evaluated Gradient Boosting Regressor and Random Forest algorithms, with Gradient Boosting Regressor achieving an R-squared score of 0.84, and Random Forest demonstrating superior performance with an accuracy of 0.816. Additionally, Sridhar et al. ([34]) and Kaynar and Özçiloğlu ([20]) focused on binary admission prediction. Sridhar et al. ([34]) proposed a stacked ensemble model for predicting admission probabilities, outperforming other algorithms with a 91% accuracy rate, while Kaynar and Özçiloğlu ([20]) compared SVM and SDT models, with SVM yielding the best accuracy at 93%. While these studies highlight the efficacy of machine learning approaches in predicting admission probabilities, it is important to note that none of these papers define how they compute the "chance of admission". Recognizing the fundamental difference between university admissions and social housing allocation processes is crucial. While both involve selecting applicants from a pool of applicants, it is important to note that all the studies mentioned focus on admissions data from a single university. This differs from social housing allocation, where predicting outcomes for a single house is insufficient. In social housing, each house can only be assigned to one person or family, unlike university admissions where multiple students get admitted. Moreover, the number of responses is higher for a single university than for a single house. This distinction underscores the complexity inherent in social housing admission.

A recommender system is an ML algorithm that provides an ordered list to users. When predicting someone's position on the applicant list, the initial instinct may be to utilize recommender systems. However, certain aspects of applicant list determination indicate that this may not be the most suitable approach for this problem. Recommender systems are primarily designed to suggest items to users based on various filters, including user preferences, behaviour, and similarities with other users. Typically, their objective is to personalize the user experience by recommending products, services, or content based on collective user data. However, in the context of housing applicant lists, the goal is not to recommend individuals for housing. The purpose is not to rank people for a house; the sorting criteria are already predetermined. Additionally, before posting the house listing, the final length of the applicant list is unknown. This lack of prior knowledge poses a challenge in predicting an individual's final position, as it introduces uncertainty about the number of applicants and their characteristics relative to the sorting criteria that may respond to a listing in the future. While recommender systems' performance measures could be adapted for this task, as they involve comparing two positions, the direct application of recommender systems to predicting final positions on housing applicant lists is challenging due to the fixed sorting criteria and uncertainty about the number of future applicants. Therefore, alternative methodologies may be more suitable for this particular forecasting task.

Research on real estate within Artificial Intelligence mainly focuses on predicting house prices using regression models. While this is not the main focus of the present paper, reviewing the literature provides useful insights into feature selection and engineering. Additionally, the comparison of these models is interesting for predicting the total number of responses on a house listing, as this also involves recent regression techniques. Truong et al. ([36]) evaluated several machine learning models for predicting housing prices in Beijing. It was observed that, despite the rapid results of XGBoost and LightGBM, a Hybrid regression model excelled in performance. Data preprocessing steps included the removal of features with more than 50% missing data, integration of the distance to Beijing's city centre, replacing construction time with the age of buildings, and setting minimums for price and other attributes. The study also segmented floor types and heights and standardized the number of living rooms to refine the data. Jha et al. ([16]) compared the performance of various regression models, highlighting XGBoost's efficacy, especially with target binning, and pointing out the influence of seasonal trends on property purchases. Particularly, it noted a peak in sales during summer months and a preference for homes with two or three bedrooms. Chen et al. ([9]) compared five popular machine learning and deep learning approaches to predict house prices using a dataset from Kaggle. The methods included Linear Regression, Bayesian methods, Backpropagation Neural Networks, Support Vector Regression (SVR), and Deep Neural Network (DNN). The results showed that Bayesian methods, Backpropagation Neural Networks, and SVR were particularly effective for predicting house prices. SVR achieved the highest R^2 value, indicating strong performance. However, DNN performed poorly due to the small dataset, which is insufficient for deep learning models. The paper highlights that SVR is especially effective for predicting house prices due to its robustness and ability to handle nonlinear data. Begum et al. ([3]) found that among the algorithms tested for house price prediction—decision tree regression, random forest regression, and linear regression—random forest regression was the most effective. Similarly, another study on bird distribution density reached the same conclusion when comparing these algorithms ([39]). Mohd et al. ([28]) attempted to predict house prices in Malaysia by comparing random forest, decision tree, ridge, linear, and lasso regression. Random forest was preferred for its overall accuracy, as evaluated by the root mean squared error (RMSE). Another study assessed algorithms like Linear Regression, Random Forest, Gradient Boosting, and Extra-Trees on the Kaggle house price dataset ([19]). Through feature extraction, the study identified the top 30 features impacting house sales prices. The feature extraction process was critical for the algorithms to achieve an R2 score over 0.84, with Extra Trees and Gradient Boosting standing out with scores around

0.925 due to their robust handling of complex interactions within the data. Lastly, a study discussed the challenges XGBoost faces with large datasets and contrasted it with LightGBM's superior ability to process vast amounts of data quickly while maintaining high accuracy, evidenced by an accuracy rate of 90.96% ([17]).

In summary, machine learning (ML) has gained popularity for its ability to handle complex tasks and automate processes that are difficult to program explicitly in the past decade. ML models can continuously improve by learning from new data through retraining. Conversely, traditional programming remains valuable for tasks with clear logic and in environments with limited data or where interpretability is crucial. This study aims to address the gap in the literature regarding the prediction of an individual's probability of authorising a specific house before the listing closes. Since individuals responding to a house listing are ranked on an applicant list based on predefined sorting criteria, the challenge lies not in the ranking itself but in the uncertainty of who and how many people will respond to each listing. Existing research in similar domains, such as university admissions seems promising but lacks reproducibility regarding the definition of admission probabilities in their datasets and there is hardly any research available in the field of social housing, possibly because organisations opt to retain such information internally. This research therefore proposes a new approach for predicting the probability of admission, utilizing both traditional programming and machine learning techniques.

4

Data

Before describing the methodology in Chapter 5, the data exploration and preparations are outlined in this chapter. The exploration was necessary in developing the methodology, and the preparations were essential before executing the methods. This chapter provides an overview of all data sources and preprocessing steps. Data from the WoningNet platform is transmitted to WoningNet's Data Warehouse (DWH) daily. Before integration into the DWH, this data undergoes anonymization and classification. Initially, it is deposited into a source layer of the data warehouse. Subsequently, the data undergoes several transformations to prepare it for reporting, including the establishment of facts and dimensions. These facts and dimensions are then accessible through the MicroStrategy reporting tool, where users can utilize both standard dashboards and conduct customized analyses. While MicroStrategy generally contains cleaner data than the DWH, however, inaccuracies may still exist within it. For this study, a total of 6 datasets are utilized:

- 1 dataset is derived from the DWH by combining multiple tables about the candidate lists.
- 2 datasets are sourced directly from MicroStrategy concerning Publications and Allocation Rules.
- 2 datasets are obtained from Statistics Netherlands (CBS) regarding zip codes.
- 1 dataset is self-scraped from WoningNet's publication website (https://woongaard.mijndak.nl/).

In Section 4.1, all datasets will be described along with the initial cleaning process. The initial cleaning process involved optimizing computation time for the 6 separate datasets. Not all cleaning processes were performed immediately. To prevent data leakage, certain data cleaning steps were executed only after the datasets were merged and the training and test data were split. For determining the position, the first 3 datasets are used. This includes the applicants' lists per listing, allocation rules and the end date of the listing. For determining the number of responses, the last 5 datasets are used and contain information about each listing. Section 4.2 describes the pre-processing steps for the first part of the study, including editing the sorting text and calculating success ratios. Section 4.3 describes the pre-processing steps for the second part of the study, including removing outliers, imputing missing values, transformation, normalisation, feature engineering and removing missing values. An overview of the merging process is also given.

4.1 Data description and Initial cleaning

4.1.1 Applicant list

The first dataset was extracted from the data warehouse and includes applicant IDs along with their sorting strings and status for each listing ID. One of the objectives is to predict the total number of responses before the listing closes. If an applicant's detail status is 5, it indicates that the applicant has withdrawn before the listing closes, and thus, the candidate is excluded from the dataset. Consequently, the applicant's position at the time of listing closing is determined by sorting the sorting text of applicants per listing. By summarizing the data for each listing, the allocation position is determined, as well as the total number of responses for that listing. After mutating the dataset contains 10 columns and 1,967,154 rows. The dataset covers only listings of the partnership "Gooi en Vechtstreek" in the years 2020 up to and including 2023, covering 5579 listings.

4.1.2 Listing

The MicroStrategy retrieved Listing dataset contains information on published social housing advertisements. It contains details such as the number of responses to the listing, property characteristics (e.g. number of rooms, living area), address details (street name, house number, postcode, city, municipality), rent, service charges, target group and the method of housing sorting (e.g. sorting rules or lottery), and the allocation rule set. Furthermore, the dataset includes three regional partnerships: 'Gooi and Vechtstreek', 'Woongaard' and 'Eemvallei', for all publications from 2020 to 2023. To prepare the dataset before merging, several pre-processing steps were performed. Initially, redundant, or irrelevant columns were removed, including columns with inconsistent data or with more than 80% missing values. Address-related information spread across multiple columns was consolidated into complete addresses and a separate column for zip codes. Entries labelled ‘Empty’ or ‘Unknown’ were converted to ‘Not Available’ (NA). Columns with only two categories were converted to Boolean variables. Columns with specific costs were removed due to annual variations and overlaps with categorical attributes. In addition, some columns were renamed or merged to make the dataset clearer and more coherent. Rows with missing values in the ‘Response’ column were removed, as this is the target variable. Entries corresponding to very rare housing types and difficult to merge with other dwellings were also excluded: trailer stands and temporary housing (together only 3 rows). Missing values were imputed from row data where possible to avoid data leakage. These pre-processing steps reduced the shape of the dataset from 61 columns and 20604 rows to a refined ‘Listing’ dataset comprising 37 columns and 20601 rows. The dataset is presented in Tabel 8.1 in the Appendix.

4.1.3 Allocation Rules

The third dataset retrieved from MicroStrategy provides insights into the allocation rules alongside each listing. This dataset contains details such as the Listing ID, partnership, allocation criteria, criteria type (selecting or sorting), order number, allocation rules set, and if the criterion was selected. The criteria that were not selected were deleted. Additionally, it includes the allocation sorting criteria, which remains empty for rules based on binary conditions, while containing specific values, such as registration periods, for date-based criteria. In Tabel 4.1 an example dataset is shown, containing only the sorting criteria for one listing.

Table 4.1: The sorting criteria of one listing

Listing Id	Partnership	Allocation Criteria	Criteria Type	Criteria sorting variable	Rule set	New order	Allocation Sorting
25302	Gooi en Vechtstreek	Urgent GEV (Calculation date ascending)	Sorting	Calculation date	Social rent: Regular GEV	1	sort_urgent
25302	Gooi en Vechtstreek	Relocation urgent (Calculation date ascending)	Sorting	Calculation date	Social rent: Regular GEV	2	sort_others
25302	Gooi en Vechtstreek	Regional ties Gooi and Vechtstreek	Sorting	Empty	Social rent: Regular GEV	3	sort_region
25302	Gooi en Vechtstreek	Search value	Sorting	Registration Date	Social rent: Regular GEV	4	sort_registration_date

The allocation rule dataset is connected to the sorting text of every applicant in the applicant list dataset. An example of a sorting text associated with Tabel 4.1 is:

"0_20191202_1_00000000_1_20191001".

In this context, unlike typical boolean values, 0 indicates that the criteria have been met, while 1 indicates that they have not been met. A calculation date holds two places in the sorting string, whereas other criteria only hold one place. This is because calculation dates are associated with urgency criteria, which can either be satisfied or not, resulting in the presence or absence of a corresponding date. Some publications had too many sorting rules

in the allocation rule dataset, causing mismatches between the sorting text length from the candidate list and the sorting criteria in the allocation rules dataset. These instances were removed. In addition, some listings contained duplicate rules but were written differently. In such cases, the duplicates were removed: for lottery properties, the last duplicate was removed; otherwise, the first was removed. The criteria were then sorted and a new sort value was calculated.
As described earlier the allocation rules play a critical role as they dictate the number of individuals permitted to respond to a publication. However, they lack precise specifications. Criteria can contain variables that can be changed in the system before publishing the listing. For example the criteria: "minimum family size of 2". However, this requirement may have been adjusted to a minimum size of 4, this reduces the pool of suitable prospective tenants and potentially decreases the response rate. Yet these modifications are not currently recorded in the Data Warehouse. For example, all criteria about income were consolidated into one group, while those concerning youth housing (up to 28 years) were grouped separately. Due to the non-specific nature of allocation criteria and to enhance manageability, the criteria were grouped. This consolidation reduces the total number of unique criteria from 261 to 29, enhancing manageability. In the last column of Tabel 4.1 these grouped criteria are shown.
To prepare for the second part of the research before merging a summarized version was made of this dataset. For every listing the number of sorting and selecting criteria was calculated. Moreover, for every listing it was indicated if a grouped criteria was present. This grouped Allocation Rules dataset contains 32 columns and 20601 rows. The data is presented in Table 8.2 in the Appendix.

4.1.4 CBS Zip Codes

The fourth dataset, sourced from CBS (2022), comprises information on every postal code in the Netherlands. Dutch postal codes consist of four numbers and two letters, with the numbers denoting broader areas and the letters specifying neighbourhoods or segments of streets. This dataset includes details about the total population, gender, age demographics, origin of the residents, composition of families, housing stock and ownership status in each area. Additionally, it provides information on the density of addresses per square kilometre. To enhance readability, column names were shortened. Furthermore, incomplete data was present in areas with sparse populations due to privacy reasons, denoted by -99997 in the dataset, which was replaced with NA. Missing values were imputed based on available data in other columns. Features containing counted data were converted to ratios to facilitate easy comparison across different areas. Zip codes with missing data were replaced by the closest zip code. This dataset contains 38 columns and 4053 rows. The data is presented in Table 8.3 in the Appendix.

The fifth dataset, also from CBS but from 2020 because this version of full six-digit zip codes contained more data than the 2022 version. The dataset includes the same data as the fourth-digit zip code dataset and more metrics such as welfare recipients, gas and electricity consumption, and various location-based indicators such as the number of supermarkets within various km radii and distance to the nearest one. Preprocessing steps mirrored those of the four-digit postal code dataset, with some columns dropped due to a high number of missing values. This dataset contains 124 columns and 419131 rows.The data is presented in Table 8.4 in the Appendix.

4.1.5 Web scraping

The sixth dataset was self-scraped due to limitations in the allocation rule dataset that were described earlier, and because certain data couldn't be obtained internally. With this dataset, an attempt was made to improve the model's performance. The dataset was scraped from each listing on WoningNet's website. The scraped values from each listing included the number of pictures, construction year, energy label, energy index, target demographic, minimum and maximum occupancy, requirements regarding children, and a column about occupancy with an undetermined minimum or maximum. The undetermined minimum or maximum column arises from changes on the website between 2020 and 2023. In 2023, most houses had clear and consistent criteria, whereas in previous years the location of the criteria on the website was ambiguous and sometimes even missing. The scraped data from earlier years may lack precision, as some descriptions often include terms like "senior," referring to the surroundings rather than the house itself. However, as will be shown, including this information improved the results, hence its inclusion. Additionally, WoningNet seeks input on what information to include in their new data warehouse. Some data, such as energy-related information and details about gardens, was incomplete in certain listings and was therefore excluded from the analysis. This dataset contains 11 columns and 20601 rows. The data is presented in Table 8.5 in the Appendix.

4.2 Position prediction

4.2.1 Preprocessing and exploratory data analysis

The applicant dataset was combined with the allocation rules and subsequently split into 60% training data, 20% validation data, and 20% test data. The positions of urgency dates were determined and a new sorting string was created, splitting it into separate columns based on the positions of the urgency dates. If a date was unknown or empty, it was represented as either '00000000' or '99999999' and converted to 01-01-2030. By separating the sorting text, it became visible which criterion corresponded to each part of the sorting text, refer to Table 4.2. This process allowed the retrieval of registration dates, urgency dates, and other relevant dates into new columns. From the listing dataset, the end listing date was retrieved and merged, along with a boolean column indicating if a listing was a lottery.

Table 4.2: Preprocessing Sorting Text

Listing Id	Sorting Text	ST1	ST2	ST3	ST4	ST5	ST6	ST7	Lottery	Urgent_date	Others_date	Registration_date	End Listing
25302	0_20191202_1_00000000_1_20191001	20191202	20300101	1	20191001	00000000	00000000	00000000	False	2019-12-02	2030-01-01	2019-10-01	2020-01-27
25302	1_00000000_1_00000000_0_20050128	20300101	20300101	0	20050128	00000000	00000000	00000000	False	2030-01-01	2030-01-01	2005-01-28	2020-01-27
25302	1_00000000_1_00000000_0_20051010	20300101	20300101	0	20051010	00000000	00000000	00000000	False	2030-01-01	2030-01-01	2005-10-10	2020-01-27
25302	1_00000000_1_00000000_0_20060828	20300101	20300101	0	20060828	00000000	00000000	00000000	False	2030-01-01	2030-01-01	2006-08-28	2020-01-27
25302	1_00000000_1_00000000_0_20070507	20300101	20300101	0	20070507	00000000	00000000	00000000	False	2030-01-01	2030-01-01	2007-05-07	2020-01-27

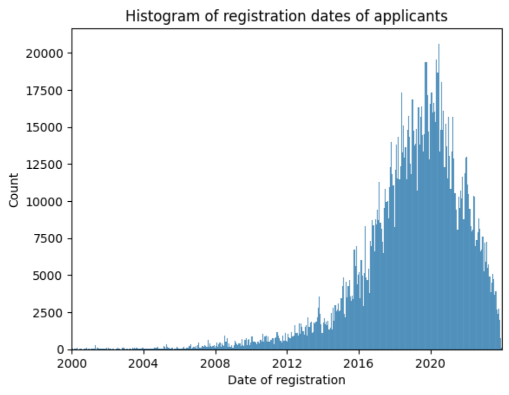


Figure 4.1: Histogram of Registration Date

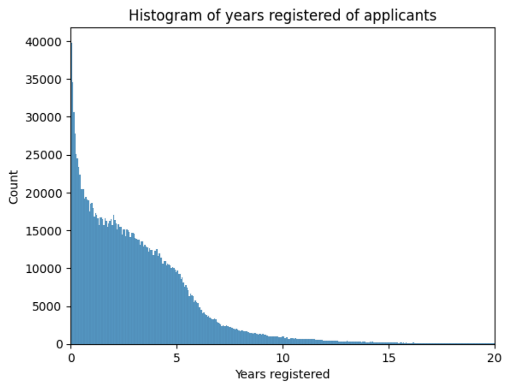


Figure 4.2: Histogram of Registration Years

Figure 4.3: Transformation of registration date into registration years (limited to a maximum of 20 years)

A new sorting text was created for the first approach because the initial sorting text included fixed dates. After the dates were extracted into separate columns, they were converted into the number of days between each date and the listing end date. Since applicants' sorting texts will be compared to other sorting texts of applicants in other listings, a fixed date would be incorrect. In Figure 4.3 on the left side the registration dates are shown. The data shows a peak in 2020 as it includes listings from 2020 to 2023. An applicant with a registration date from 2020 can apply for a property listed in 2023, but an applicant with a registration date from 2023 cannot apply for a property listed in 2020. In the right segment of Figure 4.3, it is evident that most applicants have a registration date closely proximate to the publication's end date. To ensure correct sorting, for the first approach, the days were transformed such that candidates with the longest registration period were assigned the lowest numbers. This was achieved by calculating the days until publication from a theoretical minimum registration date of 34332 days (approximately 94 years, representing the maximum possible registration period). When the number of days was negative, it was replaced by -1. The days registered until the end of the listing were subtracted from 34332 to ensure that individuals who had been in the system the longest had a lower number. The new sorting texts in Table 4.3 were created from the separate sorting columns: ST1, ST2, ... ST7 in Table 4.2, each containing only one value per criterion. This approach requires having a set of listings with the same sorting criteria in the same order. Therefore, the frequency of each combination of sorting criteria, termed a sorting rule, was calculated, identifying the most common one.

Table 4.3: Preprocessing for the first approach

Listing Id	Sorting Text	New Sorting Text
25302	0_20191202_1_00000000_1_20191001	34276_34333_1_34214_00000000_00000000_00000000
25302	1_00000000_1_00000000_0_20050128	34333_34333_0_28855_00000000_00000000_00000000
25302	1_00000000_1_00000000_0_20051010	34333_34333_0_29110_00000000_00000000_00000000
25302	1_00000000_1_00000000_0_20060828	34333_34333_0_29432_00000000_00000000_00000000
25302	1_00000000_1_00000000_0_20070507	34333_34333_0_29684_00000000_00000000_00000000

For the second approach, it was assumed that individuals always receive their final position based on a specific date, which could be either an urgency date (urgency or others) or the registration date. For each listing, the ratio of applicants meeting the sorting criteria was calculated. The distributions of these criteria are illustrated in Figure 4.4. Some criteria exhibit minimal variability (sort_others, sort_urgent, sort_maxpersons), while others show significant variability in their ratios (sort_seniorlow). The median ratio per sorting criteria was calculated on the listings of the training dataset and saved. These ratios will be used for the second approach. Additionally, to implement this approach, three columns containing lists were created alongside this dataset. The first column identifies

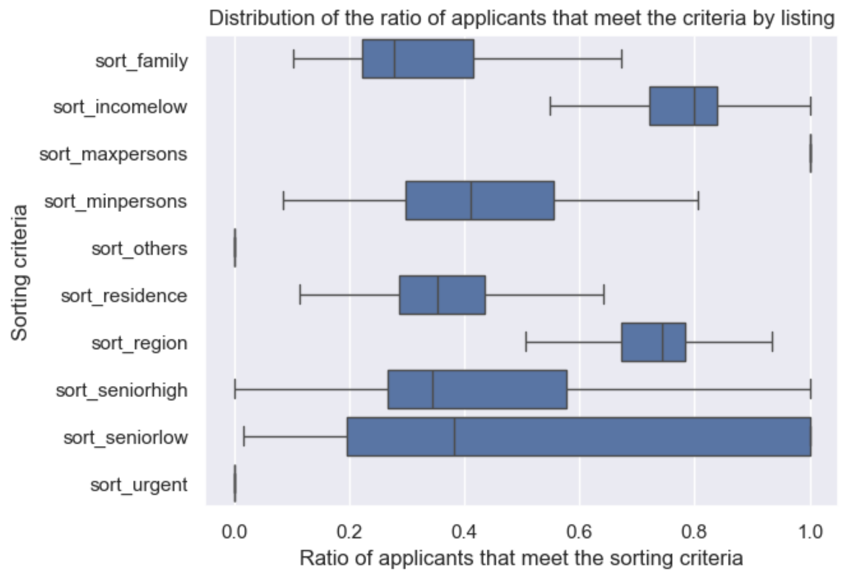


Figure 4.4: Distribution of the ratio of applicants that meet sorting criteria (calculated by listing (n > 20))

the values in the sorting area, the second column specifies which criteria pertain to these values, and the third column indicates if the criteria are binary, refer to Table 4.4.

Table 4.4: Preprocessing for the second approach

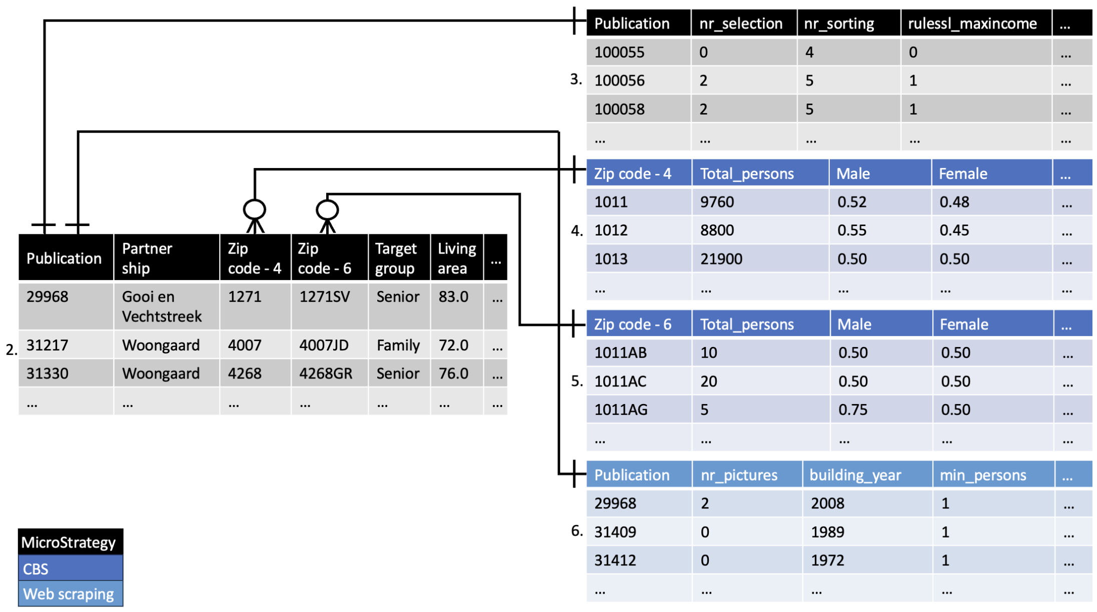
Listing Id	Sorting Text	processed_list	processed_list_name	processed_list_binary
25302	0_20191202_1_00000000_1_20191001	[0, 56.0]	["sort_urgent", "sort_urgent"]	[true, false]
25302	1_00000000_1_00000000_0_20050128	[1, 1, 0, 5477.0]	["sort_urgent", "sort_others", "sort_region", "sort_registration_date"]	[true, true, true, false]
25302	1_00000000_1_00000000_0_20051010	[1, 1, 0, 5222.0]	["sort_urgent", "sort_others", "sort_region", "sort_registration_date"]	[true, true, true, false]
25302	1_00000000_1_00000000_0_20060828	[1, 1, 0, 4900.0]	["sort_urgent", "sort_others", "sort_region", "sort_registration_date"]	[true, true, true, false]
25302	1_00000000_1_00000000_0_20070507	[1, 1, 0, 4648.0]	["sort_urgent", "sort_others", "sort_region", "sort_registration_date"]	[true, true, true, false]

4.3 Preprocessing and engineering

The last five datasets that exclusively display data related to publications or zip codes (from chapters 3.1.1 to 3.1.4) are merged, see Figure 4.5, and subsequently split into 60% training data, 20% validation data, and 20% test data. Categorical features are then transformed into dummy features. This section outlines the preprocessing and data engineering steps of the entire dataset, including outlier treatment, imputation of missing values, feature engineering, data transformation, normalization, and feature removal. These steps refine the dataset for predictive modeling.

4.3.1 Outliers

All features underwent visual inspection to detect outliers within the training dataset. Realistic value boundaries were established for each column and adjusted as necessary. These boundaries, derived from the training dataset, were applied consistently to the validation and test datasets. Any values falling outside these boundaries were replaced with the median value of the respective dataset. For instance, Figure 4.6 illustrates the adjustment of outliers in gross rent and living area. Certain minor outliers were retained if they represented realistic values but were less common. This process was done for 62 features.



2.

Publication	Partner ship	Zip code - 4	Zip code - 6	Target group	Living area	...
29968	Gooi en Vechtstreek	1271	1271SV	Senior	83.0	...
31217	Woongaard	4007	4007JD	Family	72.0	...
31330	Woongaard	4268	4268GR	Senior	76.0	...
...	...	...	...	...	...	...

3.

Publication	nr_selection	nr_sorting	rulessl_maxincome	...
100055	0	4	0	...
100056	2	5	1	...
100058	2	5	1	...
...	...	...	...	...

4.

Zip code - 4	Total_persons	Male	Female	...
1011	9760	0.52	0.48	...
1012	8800	0.55	0.45	...
1013	21900	0.50	0.50	...
...	...	...	...	...

5.

Zip code - 6	Total_persons	Male	Female	...
1011AB	10	0.50	0.50	...
1011AC	20	0.50	0.50	...
1011AG	5	0.75	0.50	...
...	...	...	...	...

6.

Publication	nr_pictures	building_year	min_persons	...
29968	2	2008	1	...
31409	0	1989	1	...
31412	0	1972	1	...
...	...	...	...	...

Figure 4.5: Publication related datasets: 2. Publications, 3. Allocation Rules, 4. CBS Zip code-4, 5. CBS Zip code-6, 6. Web scraped.

4.3.2 Missing values

While many missing values were previously eliminated, some columns still exhibit this issue. For those with less than 20% missing data, imputation was performed using the training data median. The median is more stable when the data is not distributed normally. In the case of the eligible rent class, a categorical variable, missing classes were deduced by referencing the nearest median eligible rent. The following pseudo-code illustrates the algorithm employed for this process.

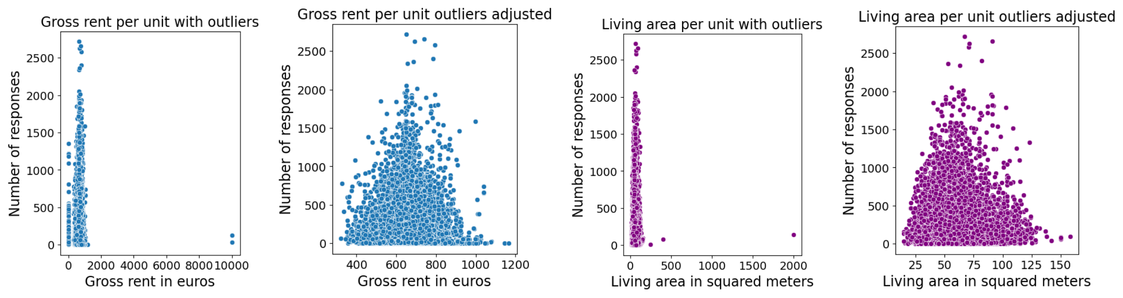


Figure 4.6: Adjustment of outliers for gross rent and living area

Algorithm 1 Impute missing rent_class

```
Require: rent_class = NaN
 1: median_rent_class ← median rent by class
 2: for each row with missing rent_class do
 3:     min_difference ← $\infty$
 4:     closest_rent_class ← None
 5:     for each rent_class in median_rent_class do
 6:         abs_rent_diff ← $|\text{rent} - \text{median\_rent\_class}|$
 7:         if abs_rent_diff < min_difference then
 8:             min_difference ← abs_rent_diff
 9:             closest_rent_class ← rent_class
10:         end if
11:     end for
12:     df rent_class ← closest_rent_class
13: end for
```

4.3.3 Feature engineering

New columns were generated by combining existing columns to reduce the dataset dimension or extract more insightful information. For instance, a set of 28 features representing distances to various facilities, such as the distance to the nearest school and the nearest supermarket, were combined by summing their values. This aggregation process is illustrated in Figure 4.7. Interestingly, houses with numerous responses tend to have lower summed distances to facilities, while those with higher cumulative distances generally result in fewer responses. Other columns created include the sum of facilities within 1km and 20km, the scaled collaboration, house age instead of building year, number of persons, the age experiment where the minimum and maximum age were combined into one feature, a combination of less common housing types, young age selection, old age selection, and listings per street and place.

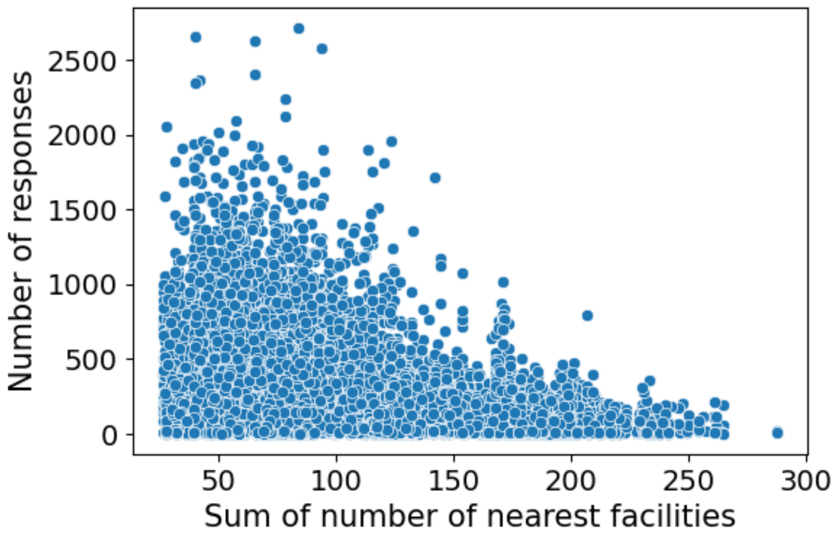


Figure 4.7: Feature engineering: combining the number of 28 facilities features

4.3.4 Transformation

The target variable was transformed and saved in a new column due to the prevalence of publications with only one response, resulting in an approximate exponential distribution. The use of a Box-Cox transformation was deemed appropriate, even though the variable is not entirely normally distributed, refer to Figure 4.8. This approach might improve predictive performance.

$$y(\lambda) = \begin{cases} \frac{y^{\lambda}-1}{\lambda} & \text{if } \lambda \neq 0 \\ log(y) & \text{if } \lambda = 0 \end{cases}$$

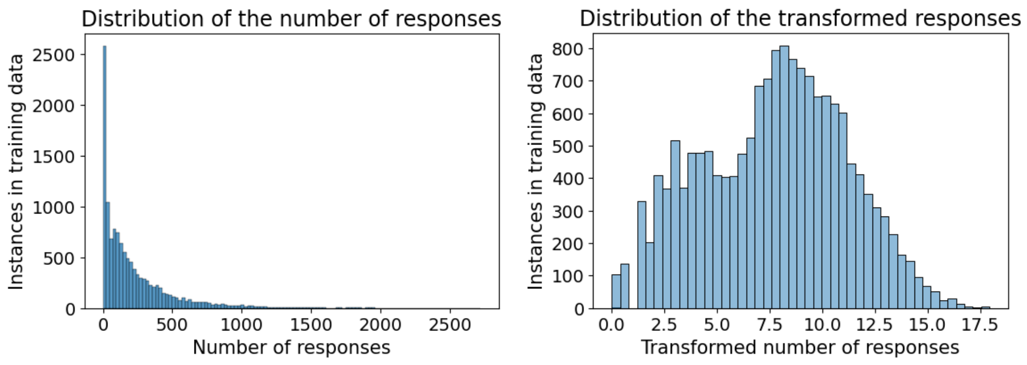


Figure 4.8: Transformation the target variable (number of responses) of the train data

4.3.5 Normalization

Normalization of features is crucial in certain algorithms, such as regression, for several reasons. It ensures that all features lie on a consistent scale, thereby preventing a feature from dominating solely based on its size. In addition, normalization can help improve convergence speed, leading to more efficient training. It also facilitates the interpretation of the importance of features. All features were normalized, with the equation:

$$z = \frac{x - \mu}{\sigma} \tag{4.1}$$

where:

z = standardized feature
x = original feature
μ = mean of the feature
σ = standard deviation of the feature

4.3.6 Removal

Features were removed in groups in an iterative process based on several reasons: domain knowledge, high intercorrelation and low correlation. Categorical features were converted into dummy features, except for CatBoost models where dummy features were omitted. Features that were removed based on domain knowledge were detected by visual inspection. This allowed features with noise, low variance or no observable patterns to be detected. Columns already used for a feature engineering feature were also removed. This led to the removal of 101 columns, containing mainly features from CBS data. Additional details are provided in the Appendix. Columns with a high mutual correlation (higher than 0.8) with other columns were removed to reduce possible adverse effects on model performance.

Of the correlated columns, those with the lowest correlation with the target feature were removed, resulting in 81 columns being removed due to high intercorrelation. Additional details are provided in the Appendix. An example of highly correlated features is net rent and gross rent, shown in the left segment of Figure 4.9. In the right segment of Figure 4.9, it can be seen that all columns show a weak to moderate correlation to the target variable (less than 0.45). Columns whose correlation with the target is very low were removed (0.05). In this step, 30 columns were removed. Additional details are provided in the Appendix. These steps reduced the total number of features from 281 to 80 columns. Although 80 columns still seem substantial, some models in the analysis phase include feature selection techniques. Therefore, the removal of additional columns was delayed so as not to compromise model accuracy.

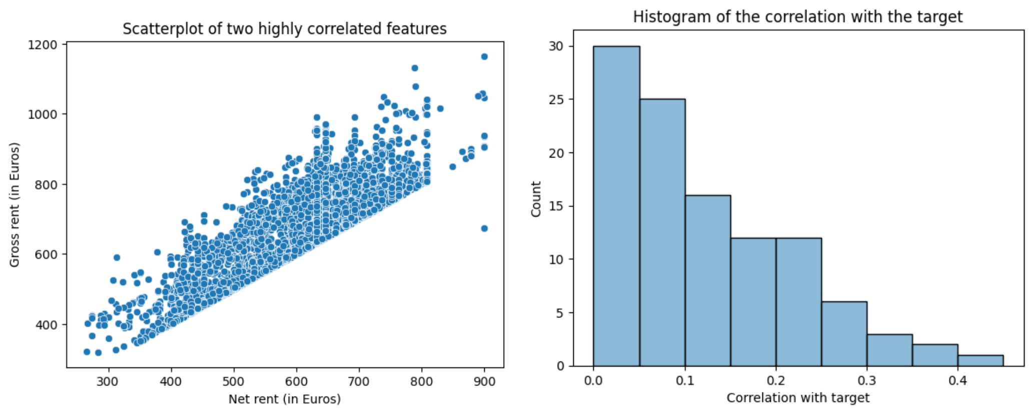


Figure 4.9: Features correlation

5

Methodology

This chapter describes the methodology of this study. It first discusses the methods used for predicting the final position of applicants. Next, various machine learning algorithms for predicting the total number of responses to a listing are explained. Finally, the experimental setup is described, including hyperparameter tuning. To fill the literature gap discussed in Chapter 3, this study proposes a hybrid approach using both traditional programming and machine learning techniques. Traditional programming will be used to determine an applicant's final position on the applicant list with 2 methods based on historical data: the mean approach and the sorting ratio approach. Relying solely on the mean to determine an applicant's final position proved to be insufficient, as the number of responses per listing plays a crucial role. It will be demonstrated how the number of responses is incorporated into the algorithms. Following this, the number of responses per listing will be predicted using various ML algorithms. ML is chosen over traditional programming because predicting an individual's position is complex and the number of responses per property can change over time, ML gives the ability to retrain easily.

5.1 Predicting the final position

Two novel methods for predicting the final position in the context of social housing are described in this section. The mean approach will be detailed first, including its incorporation of the number of responses. Then the more innovative sorting ratio approach will be discussed.

5.1.1 Mean approach

The mean approach uses all average positions of an applicant, including those from listings where the applicant did not apply, to estimate each applicant's final position. To determine the applicant's position on a listing they did not apply to, the transformed sorting text is used. This method applies only to listings with the same set of sorting criteria in the same order. The algorithm below describes how to derive the average position across listings sharing the same sorting criteria. Since an applicant can have multiple sorting texts due to data changes over time, and the sorting text is decisive, it was chosen to calculate the average position per transformed sorting text.

Sets:

L	Set of unique listing ID's in the train data
S	Set of unique sorting strings in the train data and input data in ascending sorted order

Variables:

p_{sl}	Position of an applicant with sorting string s on listing l, $\forall s \in S, \forall l \in L$
x_{sl}	Boolean, indicating if the position p_{ls} of sorting string s on listing l is known, $\forall s \in S, \forall l \in L$
f_s	The predicted or forecasted position for an applicant with sorting string s, $\forall s \in S$

Computations:

$\forall s \in S$:

$$p_{1,l} = 1$$

$\forall l \in L,\ \forall s \in S \setminus \{1\}$:

$$\begin{cases} p_{sl} = p_{s-1,l} + x_{s-1,l}, & \text{if } x_{sl} = 0 \\ p_{sl} = p_{sl}, & \text{if } x_{sl} = 1 \end{cases}$$

$\forall s \in S$:

$$f_s = \frac{\sum_{l \in L} p_{ls}}{L}$$

In addition, the number of responses was added to the algorithm to investigate its influence. This was accomplished by calculating the average position relative to the total length of the applicant list for each unique sorting string.

Added variables:

n_l	The total number of true applicants / total number of responses to listing l, $\forall l \in L$
r_{sl}	The ratio scaled position by the number of responses, $\forall s \in S, \forall l \in L$

Added computations:

$\forall s \in S$:

$$p_{1,l} = 1$$

$\forall l \in L$:

$$n_l = \sum_{s \in S} x_{sl}$$

$\forall l \in L,\ \forall s \in S \setminus \{1\}$:

$$\begin{cases} p_{sl} = p_{s-1,l} + x_{s-1,l}, & \text{if } x_{sl} = 0 \\ p_{sl} = p_{sl}, & \text{if } x_{sl} = 1 \end{cases}$$

$\forall l \in L, \forall s \in S$:

$$r_{sl} = \frac{p_{sl}}{n_l}$$

$\forall s \in S$:

$$f_s = \frac{\sum_{l \in L} r_{sl}}{L}$$

5.1.2 Sorting ratio approach

The second approach utilizes binary sorting criteria and the number of responses to determine a minimum and maximum position for each applicant and date sorting criteria to estimate the final position. This method assumes that a mean success ratio can be calculated for each binary sorting criterion and that the final position is always influenced by the first date in the sorting criteria. The mean success ratio indicates how many applicants, on average, meet the criterion per listing. After applying the success ratios, each applicant is assigned a minimum and maximum position. The final position is then estimated based on the application date relative to the distribution of dates in the training data. For example, in Figure 5.1, consider an applicant with the sorting values [0, 1, 0, 15] on a listing with 900 responses. Here, for each criterion 0 indicates success and 1 indicates failure. The applicant meets the first criterion (success), so their position is initially between 1-600. Failing the second criterion adjusts their position to 201-600. Meeting the third criterion further narrows their position to 201-240. The registration date distribution is then applied to determine the applicant's final position. Since there are 40 positions left in the position interval from 201 to 240, the registration date distribution of the training data will be divided into 40 equal intervals. Because a 15-day registration period is very short, and people with longer registration times get priority, the person will be in the first group and receive a final position of 240. The mathematical model that describes this approach is shown below.

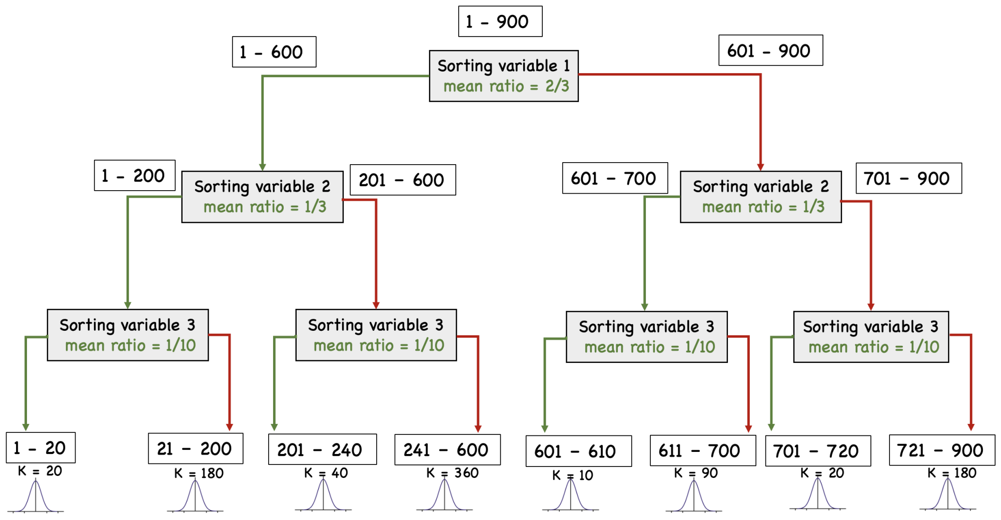


Figure 5.1: Example of the sorting ratio method

Sets:

A	Set of applicants in the input data
L	Set of unique listing IDs in the input data
G	Set of grouped sorting criteria
P	Set of possible positions in the sorting string {1, 2, ..., 7}
I	Set of the sorted registration time in days from the train data $\{i_{(1)}, i_{(2)}, ..., i_{(n)}\}$, $\forall a \in A, \forall l \in L$, where $\{i_{al} \| x_{al} = 1\}$, given $i_1 \leq i_2 \leq ... \leq i_n$ and $n = \sum_{a \in A, l \in L} x_{al} \cdot t_l$
U	Set of the urgency duration in days from the train data $\{u_{(1)}, u_{(2)}, ..., u_{(n)}\}$, $\forall a \in A, \forall l \in L$, where $\{u_{al} \| y_{al} = 1\}$, given $u_1 \leq u_2 \leq ... \leq u_n$ and $n = \sum_{a \in A, l \in L} y_{al} \cdot t_l$
O	Set of the other urgency duration in days from the train data$\{o_{(1)}, o_{(2)}, ..., o_{(n)}\}$, $\forall a \in A, \forall l \in L$, where $\{o_{al} \| z_{al} = 1\}$, given $o_1 \leq o_2 \leq ... \leq o_n$ and $n = \sum_{a \in A, l \in L} z_{al} \cdot t_l$

Variables:

t_l	Boolean, indicating if listing l is in the train dataset (1 = true, 0 = false), $\forall l \in L$
x_{al}	Boolean, indicating if an applicant responded to listing l (1 = true, 0 = false), $\forall a \in A, \forall l \in L$
y_{al}	Boolean, indicating if applicant a on listing l had an urgency (1 = true, 0 = false), $\forall a \in A, \forall l \in L$
z_{al}	Boolean, indicating if applicant a on listing l had another urgency (1 = true, 0 = false), $\forall a \in A, \forall l \in L$
v_{alp}	Value on position p in the sorting string of applicant a on listing l, $\forall a \in A, \forall l \in L, \forall p \in P$
b_{alp}	Boolean, indicating if the value on position p in the sorting text of applicant a on listing l is a boolean value, $\forall a \in A, \forall l \in L, \forall p \in P$

g_{alp}	The name of the grouped sorting criteria on position p in the sorting text of applicant a on listing l is a boolean value, $\forall a \in A, \forall l \in L, \forall p \in P$
r_g	Median ratio of the grouped sorting criteria g in the train data$\forall g \in G$
max_{al}	Maximum position of applicant a on listing l, $\forall a \in A, \forall l \in L$
min_{al}	Minimum position of applicant a on listing l, $\forall a \in A, \forall l \in L$
k_{al}	Difference between the maximum position and minimum position of applicant a on listing l, $\forall a \in A, \forall l \in L$
f_{alj}	Maximum boundary of j-th group in the entire train date distribution c_{al} for applicant a on listing l, $\forall a \in A, \forall l \in L$, where $j \in (1, ..., k_{al})$
c_{al}	The group where the date from the sorting text of an applicant a on listing l is located within the entire train date distribution.$\forall a \in A, \forall l \in L$
p_{al}	Predicted final position of applicant a on listing l, $\forall a \in A, \forall l \in L$

Computations:

$\forall a \in A,\ \forall l \in L$:

$$max_{al} = \sum_{a \in A} x_{al}$$

$$min_{al} = 1$$

$$\forall p \in P :$$

$$\begin{cases} max_{al} = \lfloor r_{g_{alp}} \cdot (max_{al} - min_{al} + 1) + min_{al} - 1 \rceil, & \text{if } v_{alp} = 0 \text{ and } b_{alp} = \text{True} \\ \\ min_{al} = \lfloor min_{al} + r_{g_{alp}} \cdot (max_{al} - min_{al} + 1) \rceil, & \text{if } v_{alp} = 1 \text{ and } b_{alp} = \text{True} \\ \\ D = (d_1, d_2, ..., d_n) = \begin{cases} I & \text{if } g_{alp} = \text{sort\_registration\_date} \\ U & \text{if } g_{alp} = \text{sort\_urgent} \\ O & \text{if } g_{alp} = \text{sort\_others} \end{cases} \\ \text{k}_{al} = max_{al} - min_{al} \\ \forall j \in (1, ..., k_{al}) & \text{if } b_{alp} = \text{False} \\ \quad f_{a,l,0} = 0 \\ \quad f_{a,l,j} = j \lfloor \frac{n}{k_{al}} \rceil, \text{ where } n = n(D) \\ c_{al} = \begin{cases} 0 & \text{if } k = 0 \\ j | (d_{f_{a,l,j-1}} < v_{alp} \leq d_{f_{a,l,j}}), \text{ where:} & \text{if } k > 0 \\ \quad d_m \in D, n = n(D), j \in (1, ..., k) \end{cases} \\ p_{al} = k_{al} - c_{al} + min_{al} \end{cases}$$

5.2 From the final position to the chance of allocation

The probability of allocation is closely related to the final position of an applicant. Based on the distribution of the final positions of allocated applicants, the probability that a property is allocated to an applicant with a certain final position can be estimated by dividing the observed frequency of that position by the total frequency. In addition, the probability that a property is allocated to an applicant with a certain final position or lower (with the highest position being 1) can be determined. This is equivalent to the probability of receiving an offer, assuming that the applicant receives the offer before declining or getting declined. For instance, if the applicant with position 15 is allocated the property, it means the property was first offered to all applicants with final positions ranging from 1 to 14. The probability of receiving an offer is calculated as described below.

Sets:

B	Set of allocated applicants

Variables:

p_b	Final position of the allocated applicant b, $\forall b \in B$
m	Maximum of the final positions that allocated applicants received
n_p	Number of allocated applicants with final position p, $\forall p \in (1, 2, ..., m)$
o_p	Chance of an offer for applicants with final position p, $\forall p \in (1, 2, ..m)$

Computations:

$m = max_{b \in B}\ p_b$

$\forall f \in (1, 2, ..., m)$:

$$n_p = \textstyle\sum_{b \in B} \begin{cases} 1, & \text{if } p_b = p \\ 0, & \text{otherwise} \end{cases}$$

$\forall o \in O$:

$$o_p = \frac{\sum_{i=p}^{m} n_i}{\sum_{j=1}^{m} n_j}$$

The likelihood of receiving an offer for a property is categorized into five equal classes, as shown in Table 5.1.

Table 5.1: Allocation Probability Categories

Chance group	Chance
Very High	0.8 - 1.0
High	0.6 - 0.8
Moderate	0.4 - 0.6
Low	0.2 - 0.4
Very Low	0.0 - 0.2

5.3 Predicting the number of responses

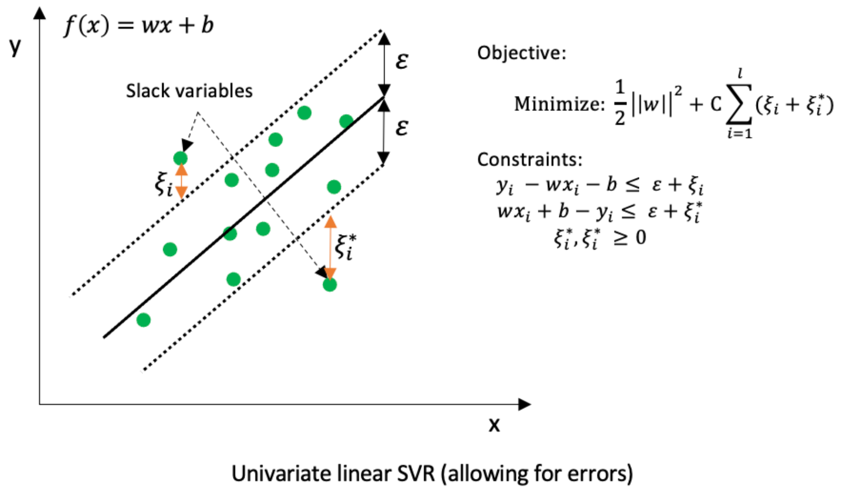
The methods for predicting the number of responses on a listing are explained in this section. The models that will be explained are support vector regression, random forest regression, extra trees regression, LightGBM, and CatBoost. The dataset used for this research is high-dimensional, and all selected models are capable of handling such data.

5.3.1 Support Vector Regression

As previously discussed Chen et al. ([9]) showed that Support Vector Regression (SVR) is effective in predicting house prices due to its robustness and ability to handle nonlinear data. SVR is a supervised machine learning method that tries to predict continuous values effectively and is a variant of Support Vector Machines (SVM) based on Vapnik-Chervonenkis theory ([2]), see Figure 5.2. SVR introduces an insensitive region ϵ resulting in an ϵ-tube around the estimated function, ignoring errors within this threshold. Consequently, the optimization problem aims to find the flattest tube that contains the most training instances. Training instances outside the ϵ-tube are called slack instances or slack variables. The objective minimizes the errors of those instances. High and low errors are penalised equally, due to the symmetrical loss function. Mathematically, the SVR model minimizes $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$, subject to the constraints $(y_i - wx_i - b) \leq \epsilon + \xi_i$, $(wx_i + b - y_i) \leq \epsilon + \xi_i^*$ and $\xi_i, \xi_i^* \geq 0$, where w is the weight vector, ϵ is the size of the tunnel, ξ_i are the slack variables with a negative error and ξ_i^* are the slack variables with a positive error and y_i is the target value for the i-th instance. The solution to this optimization problem can be expressed as

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(x_i, x) + b$$

where $K(x_i, x)$ is a kernel function that computes the inner product in the feature space, α_i and α_i^* are Lagrange multipliers, and b is the bias term. The kernel function K can be linear, polynomial, or radial basis function (RBF), depending on the complexity of the approximated underlying function. Furthermore, C is a regularization parameter.



$f(x) = wx + b$
y
x
Slack variables
ε
ε
ξ_i
ξ_i^*
Objective:
Minimize: $\frac{1}{2}||w||^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$
Constraints:
$y_i - wx_i - b \leq \varepsilon + \xi_i$
$wx_i + b - y_i \leq \varepsilon + \xi_i^*$
$\xi_i^*, \xi_i^* \geq 0$
Univariate linear SVR (allowing for errors)

Figure 5.2: Example of Univariate Support Vector Regression with a linear kernel, including the objective and constraints ([32])

Trzciński and Rokita ([37]) found that Support Vector Regression (SVR) with Gaussian radial basis functions provides precise and stable prediction results for online video popularity, as measured by the number of views. Although this study is in a different research area, it similarly addresses the concept of popularity. The study demonstrated that SVR outperforms existing methods due to its nonlinear nature and robustness by testing it on datasets with nearly 24,000 videos from YouTube and Facebook. The advantage of Support Vector Regression (SVR) lies in its foundation in Vapnik-Chervonenkis (VC) theory, which helps it generalize to unseen data. Using Vapnik's ϵ-insensitive approach, SVR forms a flexible tube around the estimated function, ignoring errors within a certain threshold (ϵ). This ensures that points outside the tube are penalized, while points inside the tube are not. The value of can be adjusted to balance complexity and prediction error. Another key advantage of SVR is that its computational complexity does not depend on the dimensionality of the input space, making it highly efficient for high-dimensional data. Additionally, SVR has excellent generalization capabilities and high prediction accuracy ([2]).

5.3.2 Random Forest Regression

Random Forest Regression (RFR) benefits include fast training speed, high prediction accuracy, and the ability to process high-dimensional data ([39]) easily. Additionally, as previously discussed, Mohd et al. ([28]) compared random forest, decision tree, ridge, linear, and lasso regression for predicting house prices in Malaysia. Random forest was preferred for its overall accuracy, as evaluated by the root mean squared error (RMSE). Random Forest is an ensemble learning method for classification and regression tasks that uses multiple Decision Trees to make decisions. Unlike a single deep Decision Tree, a random forest aggregates the outputs from many shallow trees, improving overall performance. It is useful to first look at Decision Trees to understand Random Forests. Decision Trees split data into similar segments, working well with high-dimensional data. Decision Trees are created based on the sum of squared residuals. However, Decision Trees can easily overfit the training data, leading to high variance and low bias. Methods to avoid overfitting include setting a minimum split size for leaves, which prevents overly small segments, and pruning, which removes some leaves and replaces them with averages of larger instances. Cost complexity pruning, such as weakest link pruning, calculates the sum of the SSR and a tree complexity penalty, improving generalization on test datasets. Decision Trees are easy to create, use and interpret. According to Hastie et al. ([14]), the primary drawback of Decision Trees, which prevents them from being the ideal tool for predictive learning, is their inaccuracy and lack of flexibility. Random Forests are used to get around these drawbacks. Random Forest have the simplicity of a Decision Tree but with flexibility, resulting in better accuracy. Random Forest uses bagging, see Figure 5.3. Bagging starts

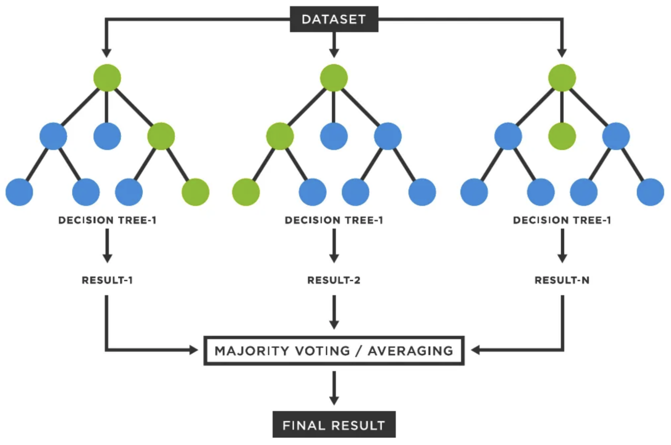


Figure 5.3: Structure of a Random Forest([13])

by selecting a bootstrapped sample of the original data with the same size as the original dataset where instances are allowed to occur multiple times. For each bootstrapped dataset, a new Decision Tree is created with only a subset of the features for each node. This is often done about 100 times. Finally, the average of all these trees is calculated in the case of a regression task. This method reduces variance, improves estimation accuracy, and handles large datasets efficiently. Random Forests can also estimate the importance of variables and perform well even with missing data.

5.3.3 Extra Trees Regression

Extra Trees is similar to Random Forest, it uses multiple Decision Trees just like Random Forest. The Extra Trees model is only even more randomised than the Random Forest model. The values on features the splits in the Decision Trees are random in this algorithm whereas in Random Forest, the optimal value. This makes the model faster and this allows it to work better with noisy data.

5.3.4 LightGBM and Catboost

Despite the recent popularity of artificial neural networks (ANN), boosting algorithms remain preferred for medium or larger-sized datasets due to their relatively fast training times and minimal parameter tuning requirements ([10]). Gradient boosting combines numerous small Decision Trees to improve prediction accuracy. It uses boosting to introduce variety among Decision Trees. Bagging and boosting are the two main methods of ensemble learning. Boosting iteratively adds weak learners to the ensemble model. This is done in a targeted manner, focusing on parts of the data that are not well understood. Each iteration weights the data points, guiding new weak learners to improve on the previously mispredicted points. This process results in an ensemble model that integrates all the intermediate weak models, enhancing overall performance ([22]). Daoud ([10]) investigated the efficiency of three gradient boosting methods: XGBoost, CatBoost, and LightGBM, using the Home Credit dataset with 219 features and 356,251 records. The findings indicate that, for the dataset used, LightGBM is faster and more accurate than both CatBoost and XGBoost across different numbers of features and records. While other studies comparing those methods showed also that LightGBM outperformed CatBoost and XGBoost ([15]), ([35]), ([11]) it can't be concluded that this is always the case. For example, for a dataset with more categorical variables, CatBoost could potentially be more accurate ([10]). Many algorithms implement gradient boosting with slight variations. AdaBoost was the first

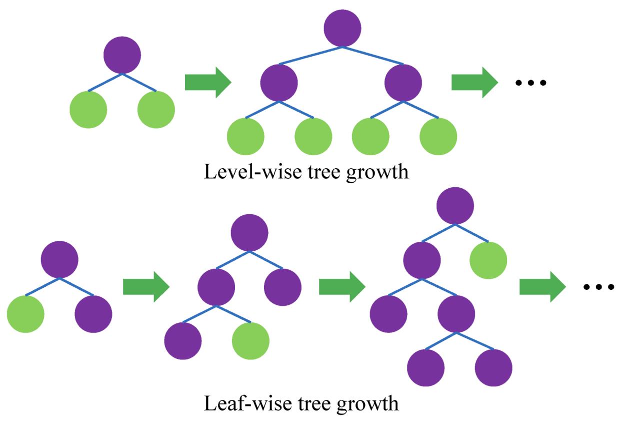


Figure 5.4: Difference between level-wise and leaf-wise growth ([26])

algorithm to adapt to weak learners. Newer gradient boosting methods include XGBoost, LightGBM, and CatBoost. XGBoost short for eXtreme Gradient Boosting, is popular in data science competitions for its accuracy and computational efficiency. XGBoost handles missing data by fitting all non-missing values first and then determining the best branch for missing values to minimize prediction error. This results in compact trees and an effective imputation strategy ([7]). Despite XGBoost's success, LightGBM emerged 2 years later by Microsoft as a high-performance alternative. LightGBM is designed to distribute and handle large datasets quickly. LightGBM employs a depth-first search (DFS) strategy, resulting in more complex trees, and higher prediction accuracy, but a higher risk of overfitting. LightGBM is particularly effective on large datasets with over 10,000 examples. It benefits from parallelization and GPU utilization, making it scalable for even larger problems. It is memory efficient, using histogram-based algorithms to discretize continuous variables into bins ([7]). Close after the published LightGBM, Yandex made another interesting GBM algorithm called CatBoost public. Its strongest point is the capability of handling categorical variables by adopting a mixed strategy of one-hot encoding and target encoding. Encoding categorical variables in other models is often done in the feature engineering part instead of the modelling. While it seems a smart feature engineering tick, it has downsides, target encoding can lead to overfitting because you are taking information from the target into predictors ([7]). The primary difference between the algorithms lies in their approach to identifying the best splits within weak learners. All methods avoid the time-consuming process of evaluating all possible splits by using alternative techniques. XGBoost employs histogram-based splitting, where histograms are built for each variable to determine the best split per variable, which is then retained as the best overall

split. LightGBM uses Gradient-Based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) to enhance efficiency. GOSS filters out data points with low gradients, focusing on those that need more learning. EFB speeds up the process by handling many correlated variables. CatBoost employs an approach where the tree splits on the same condition at each tree level. During training, all possible values of features are divided into buckets, known as feature-split pairs. Minimal Variance Sampling (MVS) is used to maximize the accuracy of these splits. The feature-split pair that results in the lowest loss is selected and applied to all nodes at the same level, leading to symmetric trees. In Figure 5.4 the difference between level-wise and leaf-wise splitting is shown. XGBoost and CatBoost grow trees level-wise, but CatBoost creates symmetric trees, unlike XGBoost. In contrast, LightGBM grows trees leaf-wise, which theoretically offers higher accuracy but carries a higher risk of overfitting, especially with small datasets ([22]).
As discussed in the literature review, a study showed that XGBoost faces challenges when the dataset size increases, whereas LightGBM excels in processing vast amounts of data quickly while maintaining high accuracy ([17]). Additionally, CatBoost is highly effective at handling categorical features efficiently. Therefore, this study will explore LightGBM and CatBoost.

5.4 Experimental setup

For the first part of this research, the goal is to predict the final position of an applicant. Initially, the effectiveness of including the actual number of responses is tested. Subsequently, the relation between the final position and the chance of allocation is shown. Then, the effectiveness of different machine learning approaches is compared to predict these responses. For the first approach for predicting an applicant's final position, the unique sorting text from the validation set is integrated into the training applicant listings. Subsequently, for each validation sorting text, the average position among those applicant listings is calculated. This process is repeated on the test dataset. For the second approach for predicting an applicant's final position, the sorting ratios are calculated using the training dataset and applied to predict positions for both the validation and test data. Because the dataset encompasses 281 features, for predicting the number of responses using machine learning models, the performance metrics are compared across different sets of features. Initially, the default parameters are used. The feature set demonstrating the best performance is chosen for hyperparameter optimization. As tree-based models are not sensitive to high mutual correlation among features, retaining these columns remains an option. To understand feature importance and instance contributions, SHAP values were calculated. SHAP (SHapley Additive exPlanations) values represent the average contribution of each feature across all possible combinations of variables. This gives insight into how each feature influences the model's predictions. Finally, the performance metrics of the position prediction models, with the incorporated predicted number of responses, are calculated.

5.4.1 Hyperparameter tuning

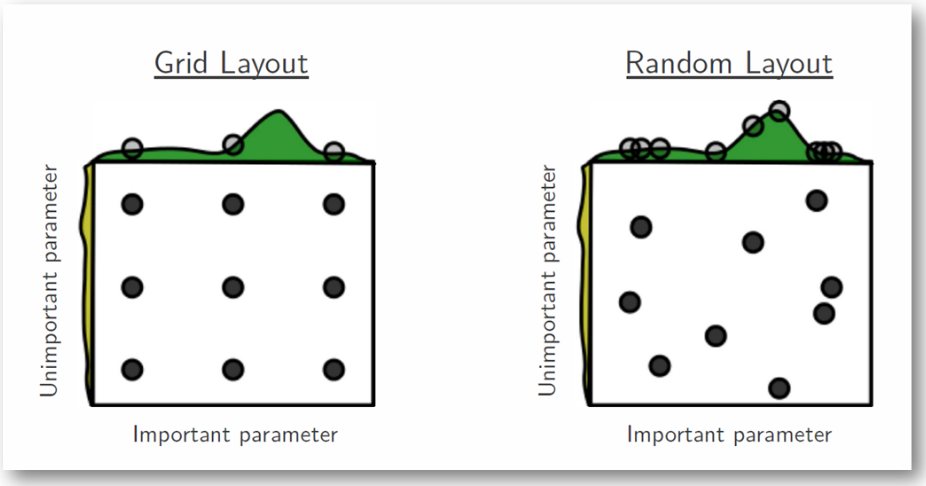


Figure 5.5: Comparison between grid search and random search ([5])

While grid search and manual search are used widely for hyperparameter tuning, Bergstra and Bengio ([5]) demonstrated that random search is more efficient for hyperparameter optimization, both empirically and theoretically. Random search is more efficient because it can find parameters that are as good or better than those found by grid search in a fraction of the time. Additionally, the most important hyperparameters vary for each problem. This variability makes grid search inefficient, as it allocates too many trials to unimportant dimensions and does not adequately cover the important ones, see Figure 5.5. Therefore, in this research, a random search will be used for hyperparameter tuning. For the random search, k-fold cross-validation with k = 5 was used. The RandomizedSearchCV function from the sklearn library's model_selection module was utilized in this research. The hyperparameters and their ranges utilized for model tuning are detailed below.

Support Vector Regression (from Scikit-learn)

- **C**: Controls the regularisation of the model; it adds a penalty for each misclassified data point. The higher the value of C the more accurate the model performs on the training data but the greater the chance of overfitting. Therefore a wide range is chosen: from 0.1 to 100.
- **epsilon**: The size of the insensitive region, errors within this threshold are ignored. Smaller epsilon values result in a model sensitive to small deviations in the data, while larger values are less sensitive. Again a wide range is chosen from 0.001 to 1.
- **kernel**: The function that computes the inner product in the feature space. The kernels explored are linear, polynomial, radial basis function and sigmoid.
- **degree**: the degree when the polynomial is used. A range of 2 to 6 is implemented.
- **gamma**: How much influence a single point has, where auto is a simpler heuristic than scale.

Random Forest and Extra Trees (from Scikit-learn)

- **n_estimators**: Determines the number of decision trees in the random forest. A higher number of trees generally results in better model performance, but also to higher computational costs. A range from 100 to 1050 with a step of 50 was selected to balance performance and efficiency.
- **max_features**: Defines the maximum number of features considered for splitting a node. Options include using all features (none), the square root of the number of features (sqrt) and the logarithm of the number of features (log2). Overfitting can be avoided by not including all features, as this reduces complexity.
- **max_depth**: Specifies the maximum depth of the trees. Limiting the depth helps avoid overfitting. A range of 10 to 30 with a step of 2 was chosen to examine shallow to moderately deep trees.
- **min_samples_split**: Specifies the minimum number of samples needed to split an internal node. Setting this range from 10 to 20 with a step of 2 was chosen to balance model complexity and generalisation.
- **min_samples_leaf**: Specifies the minimum number of samples required at a node in the leaves. A range of 5 to 19 with a step of 2 was chosen to ensure that each node in the leaves has at least 5 samples, reducing the chance of overfitting.

LightGBM (from Microsoft)

- **num_leaves**: Defines the complexity of an individual tree in the Light GBM model. Lower values prevent overfitting by regeralizing more. A range from 20 till 70 was chosen.

- **learning_rate**: Determines how fast the model converges when training. A wide range was chosen, to cover very slow to relatively fast learning rates.
- **n_estimators**: Determines the boosting iterations in the model. Generally, a higher number results in better model performance, but also in higher computational costs. A range from 100 to 1050 with a step of 50 was selected to balance performance and efficiency.
- **feature_fraction**: Indicates the proportion of features chosen before training each tree. The algorithm will randomly select this fraction of features in each iteration. This approach can speed up training and help mitigate overfitting.
- **subsample**: Adds randomness by sampling a random sample containing a fraction of the training data.

CatBoost (from Yandex)

- **one_hot_max_size**: The number of values that can be used for one-hot encoding for the categorical features. Categorical features with more levels than specified will be transformed. Values between 10 and 30 seem a reasonable amount for this dataset.
- **iterations**: The number of iterations in the algorithm. Ranging from 100 till 2000.
- **od_wait**: The number of boosting iterations to continue the training after the optimal metric value is found. Stopping earlier can help to prevent overfitting and computation costs. Ranging from 20 to 40.
- **learning_rate**: Determines how fast the model converges when training. A wide range was chosen, to cover very slow to relatively fast learning rates.
- **depth**: Controls the maximum depth of the trees. Ranging from 4 till 10, to balance complexity and generalization.
- **l2_leaf_reg**: Specifies the coefficient of the L2 regularization applied to the leaves to avoid complexity and overfitting. Values from 5 till 9.
- **random_strength**: Defines the amount of randomness to use for coring splits after the tree structure is selected. A range from 0.4 to 1 is selected.
- **bagging_temperature**: Defines the random weights settings of the Bayesian bootstrap. How much more aggressive the bagging is. Values from 0.5 till 1.5 are selected.

5.4.2 Performance metrics

To evaluate the performance of the models, two metrics will be compared across all models. The coefficient of determination (R-squared) will be calculated for the predictions versus the true values. Additionally, the Root Mean Squared Error (RMSE) will be assessed to compare the performance of the algorithms. The R-squared measures how well the estimated regression line of the model fits the distribution of the data and how many of variance in the results can be explained by the model. The RMSE measures the standard deviation of the residuals. It represents the spread of true values around the prediction. An advantage of the RMSE is that it has the same scale as the target variable. RMSE gives a higher weight to large errors. Before taking the mean, the errors are squared, so larger errors affect the RMSE relatively more than smaller errors. The formulas of the R-squared and the RMSE are as follows:

R-squared

$$R^2 = 1 - \sum_i^n \frac{(\hat{y}_i - y)^2}{(\hat{y}_i - \bar{y})^2}$$

RMSE

$$RMSE = \sqrt{\frac{1}{n}\sum_i^n (y_i - \bar{y})^2}$$

6

Results

6.1 Predicting final position

In this section, the results of the mean and sorting ratio approaches are presented. Table 6.1 shows the results for the mean approach without incorporating the number of responses, alongside the results for the scaled mean approach and the sorting ratio approach, both incorporating the number of responses. The second column presents the performance measures of the mean approach without scaling, based on a dataset of 1,206 listings, which were divided into training, validation, and test subsets. The model shows a low R-squared score and a high RMSE, indicating a poor model. The third column presents the results of the scaled model. The model shows high scores on the training, validation and testing datasets. Despite the illustration of the effectiveness of the approach, the actual performance will likely be lower due to the need to predict the number of responses. Moreover, these are only the sorting results for only one sorting rule. This makes the model less scalable. The last column illustrates the results of the sorting ratio approach, applied to all 5,577 listings. Similar to the mean approach incorporating the number of responses, again this model shows promising performance. Although the R squared and RMSE are worse than those of the mean approach, this method is scalable and applies to all listings in the train, validation and test data.

6.2 Determining chance

The final positions were divided into groups based on their likelihood of receiving an offer, to link the allocation probability to the final position of applicants. The left segment of Figure 6.1 shows the distribution of the final positions of allocated applicants on the training data. It can be observed that applicants with a final position of 1, are allocated frequently. After final position 1, the number of allocations decreases sharply. The right segment of Figure 6.1 shows the distribution of the chance of receiving an offer by the final position of an applicant. Applicants with a final position of 1, have a 100% chance of receiving an offer, due to the assumption that applicants receive the offer before declining or getting declined. The chance of receiving an offer smoothly decreases with the final position (given the highest position is 1). The probability of receiving an offer in the "very high" class (80% - 100%) is observed only for applicants with a final position of 1, whereas a final position of 13 or lower corresponds to a "very low" chance (0% - 20%).

Table 6.1: Performance of models for predicting the final position of applicants

		Mean	Scaled Mean	Sorting Ratio
Train	RMSE	267.99	26.68	49.87
	R-squared	0.1632	0.9917	0.9703
Validation	RMSE	220.18	27.89	51.65
	R-squared	0.2451	0.9879	0.9675
Test	RMSE	299.08	32.14	54.69
	R-squared	0.1190	0.9898	0.9655

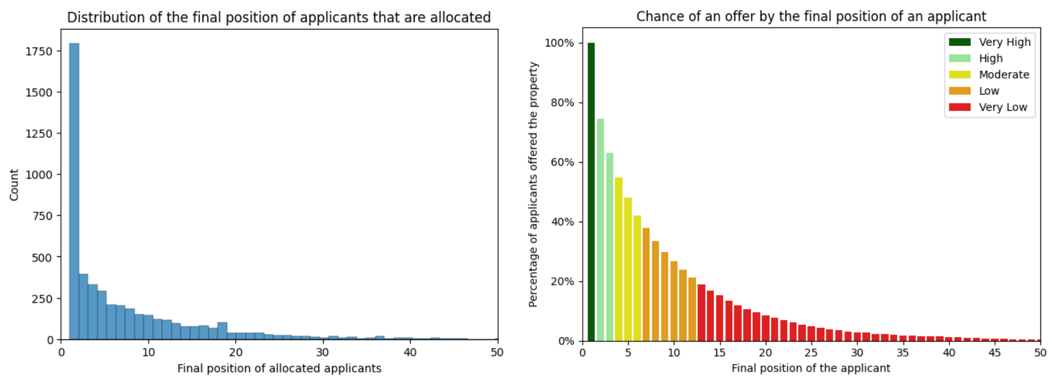


Figure 6.1: Insight into the final positions up to 50 of allocated applicants.

6.3 Predicting number of responses

6.3.1 Support Vector Regression

Table 6.2 shows the results of models using different feature sets. Default parameters were employed for selecting a final feature set. The model trained solely on the listing dataset

Table 6.2: SVR Comparison with best kernel

Model	Number of Features	RMSE Train	R^2 Train	RMSE Validation	R^2 Validation
Listing only	60	205.34	0.4511	206.63	0.4493
Listing + Allocation Rules	91	193.05	0.5149	194.78	0.5107
Listing + CBS	220	197.71	0.4912	200.03	0.4839
Listing + Webscraped	72	203.40	0.4615	204.48	0.4607
Listing + Feature engineering	78	196.91	0.4953	198.35	0.4926
Full Data	281	181.47	0.5713	183.61	0.5652
Full Data Transformed Target	281	106.56	0.8522	124.41	0.8004
Full Transformed - Knowledge-based	181	103.29	0.8611	121.47	0.8097
Full Transformed - Knowledge and Mutual Correlation based	107	110.91	0.8399	132.02	0.7752
Full Transformed - Knowledge, Mutual and Low Correlation based	77	109.43	0.8441	127.24	0.7912
Full Transformed - Knowledge and Low Correlation based	**151**	**101.25**	**0.8666**	**117.64**	**0.8215**
Best 50 features	50	197.14	0.4941	205.96	0.4529
Best 25 features	25	207.38	0.4402	214.39	0.4072

showed poor performance on both training and validation data based on performance metrics. Incorporating additional datasets (Allocation Rules, CBS, Webscraped) all improved model performance, with Allocation Rules showing the most influential enhancement. The full dataset, consisting of 281 features, demonstrated improved performance after transforming the target variable. For comparison, after modelling, both model predictions and target values were scaled back to their original form before computing performance metrics. By removing features based on knowledge, the performance remained approximately the same while reducing the feature count by 100, covering mostly CBS features. Conversely, removing highly correlated features resulted in decreased performance, whereas retaining features based on domain knowledge and low correlation, yielding a subset of 151 features, proved beneficial. A model trained only with the top 50 or 25 features showed worse performance, indicating that a broader set of features is needed to capture the target accurately. A random search was performed on the 151 features subsequently for hyperparameter tuning, the results are shown in Table 8.7 of the Appendix. After tuning the hyperparameters, a small improvement in the performance on both the training and validation set is observed (R-squared 0.028 higher and RMSE 9.61 lower). Although, the model overfits more on the training data than without tuning. The scores on the validation dataset and the test set are comparable, indicating stability.

Table 6.3: Performance scores of the final model with tuning

Dataset	Root Mean Squared Error (RMSE)	R^2 Score
Train	67.33	0.9410
Validation	108.03	0.8495
Test	114.81	0.8400

6.3.2 Random Forest

Table 6.4: Random Forest Performance Comparison

Feature set	**Number of Features**	**RMSE Train**	**R^2 Train**	**RMSE Validation**	**R^2 Validation**
Listing	60	50.63	0.9666	143.53	0.7343
Listing + Allocation Rules	91	42.00	0.9770	108.19	0.8490
Listing + CBS	220	48.84	0.9690	137.72	0.7554
Listing + Webscraped	72	48.99	0.9688	138.50	0.7525
Listing + feature engineering	78	46.46	0.9719	132.99	0.7720
All	281	40.00	0.9792	106.37	0.8541
All Transformed Target	281	42.71	0.9763	107.24	0.8517
All - Knowledge-based	**181**	**39.56**	**0.9796**	**105.53**	**0.8564**
All - Knowledge and Mutual Correlation based	107	39.67	0.9795	106.16	0.8547
All - Knowledge, Mutual and Low Correlation based	77	39.70	0.9795	108.08	0.8493
Best 50	50	68.06	0.9397	170.11	0.6268
Best 25	25	116.96	0.8219	184.58	0.5606

Again, several feature sets were evaluated for the Random Forest. Table 6.4 summarizes the performance differences across these feature sets. Adding the allocation rules features was most beneficial. Transforming the target did not increase the performance, so the original scale was retained. Removing features based on domain knowledge was effective, reducing the number of features by 100 while slightly improving the R-squared and lowering the RMSE. Removing highly mutually correlated or low correlation features yielded similar performance. Selecting only the best 50 or best 25 features resulted in decreased performance, indicating that a wider set of features is necessary to capture the complexity of the data. The feature set containing 181 features was selected, excluding columns based on prior knowledge. The best-fit parameters identified by the random search for the RandomForestRegressor model are shown in Table 8.8 of the Appendix.

Table 6.5: Performance scores of the final model with tuning full

Dataset	**Root Mean Squared Error (RMSE)**	**R^2 Score**
Train	62.48	0.9492
Validation	102.63	0.8642
Test	108.84	0.8562

The performance scores of the final model with tuning on the training, validation, and test datasets are presented in Table 6.5. The training set performance decreased after tuning (RMSE increased by 22.92 and R-squared decreased by 0.0304), while the validation set performance improved slightly (RMSE decreased by 2.90 and R-squared increased by 0.0078). The decrease in performance on the training set suggests that the model generalizes better on unseen data when using the tuned hyperparameters. The performance on the test dataset does not differ much from the performance on the validation set, indicating a relatively stable model. Figure 6.2 displays the feature importance and the contribution of the features on instances using SHAP values. The left segment of Figure 6.2 shows the features that had the most impact on the Random Forest model. The feature "Select min age" had the highest impact, with a mean SHAP value of 84.21. This boolean value indicates whether there was a minimum age selection criterion for the listings, primarily for senior homes. The figure also illustrates the substantial combined contribution of the 172 other features. The right segment of Figure 6.2 shows the impact of each instance in the training dataset on the outcome of the Random Forest model. The colour of each instance represents the value, with red representing a high value and blue representing a low value. The x-axis shows whether the impact of the SHAP value on the outcome is positive or negative. From this figure, we can conclude that a high value (1) for "Select min age" negatively affects the number of responses. This makes sense, as there are typically fewer responses for senior housing.

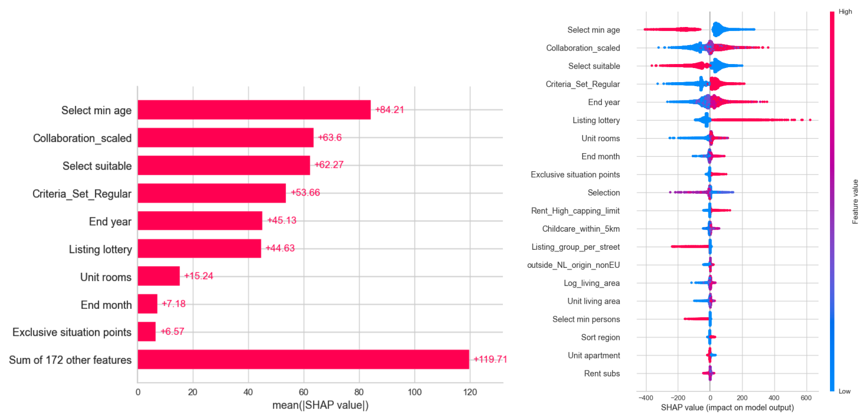


Figure 6.2: Feature importance of Random Forest Regression

6.3.3 Extra Trees

The performance of the Extra Trees model with different feature sets shown in Table 6.6, demonstrate that the most effective approach is to create a model using all features except the knowledge-based features. Additionally, tuning the hyperparameters yielded the same values as those for the Random Forest model shown in Table 8.8 of the Appendix. The performance of the model using the tuned hyperparameters and 181 features shown in Table 6.7. After tuning the hyperparameters, the performance on the training set decreased, with RMSE increasing by 52.05 and R-squared decreasing by 0.0406. In contrast, the performance on the validation set remained similar, with RMSE decreasing by 0.96 and R-squared increasing by 0.0025. This decrease in training set performance suggests that the model generalizes better to unseen data when using the tuned hyperparameters. Although the test set scores are slightly lower than the validation set scores, the model is stable due to the small difference. The R-squared score of approximately 0.85 on the test dataset indicates that 85% of the variation in responses on a listing can be explained by the input features. Figure 6.3 illustrates the importance of features in the Extra Trees model. Both segments of the figure are similar to that of the Random Forest model due to the similarities between the models. The feature "Select min age" has the highest impact on the model. The figure also indicates that although the top 9 features are influential, the combined contribution of 172 other features is also substantial, with a mean SHAP value of roughly 135. The scaled collaboration region and the end month of the listing features have a lower impact than in the Random Forest model, while the region sorting criteria feature has more impact.

Table 6.6: Extra Trees Performance Comparison

Features	Number of Features	RMSE Train	R^2 Train	RMSE Validation	R^2 Validation
Listing	60	50.63	0.9666	143.57	0.7343
Listing + Allocation Rules	91	42.00	0.9770	108.19	0.8490
Listing + CBS	220	48.84	0.9690	137.72	0.7554
Listing + Webscraped	72	48.99	0.9688	138.50	0.7525
Listing + feature engineering	78	46.46	0.9719	132.99	0.7720
All	281	3.93	0.9998	104.89	0.8581
All Transformed Target	281	3.99	0.9998	107.67	0.8505
All - Knowledge-based	**181**	**3.93**	**0.9998**	**104.20**	**0.8600**
All - Knowledge and Mutual Correlation based	107	4.05	0.9998	104.79	0.8584
All - Knowledge, Mutual and Low Correlation based	77	5.62	0.9996	107.21	0.8518
Best 50	50	25.18	0.9917	149.53	0.7116
Best 25	25	108.87	0.8456	189.97	0.5346

Table 6.7: Performance scores of the final model with tuning ET

Dataset	Root Mean Squared Error (RMSE)	R^2 Score
Train	55.98	0.9592
Validation	103.24	0.8625
Test	111.76	0.8484

6.3.4 LightGBM

The performance of the LightGBM model with default parameters on different feature sets in Table 6.8 follows a similar pattern to that of the CatBoost and Extra Trees models.

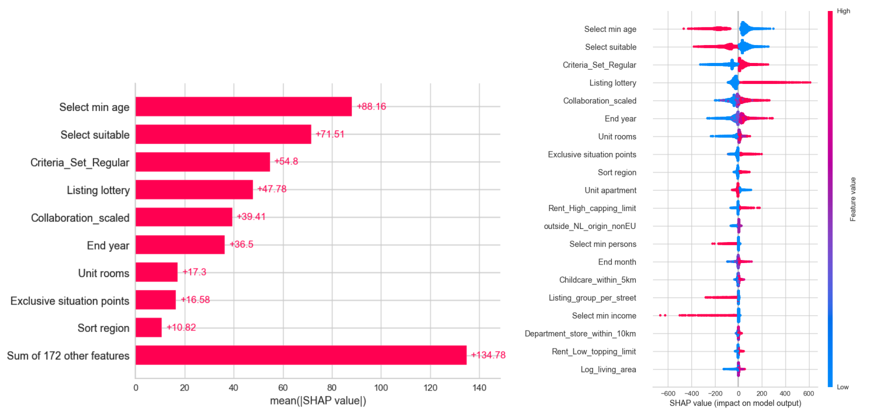


Figure 6.3: Feature importance of Extra Trees Regression

Although the model with all features yields a slightly higher R-squared than the model without features based on knowledge, it is preferable to use the smaller feature set. The difference is only 0.001 in R-squared and 0.02 in RMSE. Additionally, the model with the reduced feature set has 100 fewer features. Random Search resulted in the hyperparameters shown in Table 8.9 in the Appendix. The performance of the LightGBM model with tuned hyperparameters is shown in Table 6.9. The results suggest a slight reduction in generalization after tuning. After tuning, the model demonstrated improved performance on the training dataset and the metrics on the validation set were comparable, with the RMSE decreasing by only 2.69 units and the R-squared showing a marginal increase of 0.007. The test set achieved an R-squared of 0.88 and an RMSE of 100, indicating reasonable predictive accuracy, especially given the wide range of the target value in the dataset. In Figure 6.4 the feature importance of the LightGBM regression model shows that the year of the listing's end date is the most influential feature, followed by the minimum age selection criteria feature, which was the most important feature in the Random Forest and Extra Trees models. Notably, the combined influence of the 172 other features is substantially greater than in the other models. The SHAP values in the right segment of the figure indicate that a higher end year positively impacts the model's output. Indicating that the number of responses in 2023 on listings were higher than in 2020.

6.3.5 CatBoost

The results of the CatBoost model are shown in 6.10. CatBoost can handle categorical features without the necessity of creating dummy features, this reduces the overall number of features. Using listing features only yielded better performance (R-squared of 0.80 and

Table 6.8: Light GBM Performance Comparison

Model	Number of Features	RMSE Train	R^2 Train	RMSE Validation	R^2 Validation
Listing only	60	105.33	0.8556	143.18	0.7356
Listing + Allocation Rules	91	86.92	0.9017	106.38	0.8540
Listing + CBS	220	85.05	0.9058	131.65	0.7765
Listing + Webscraped	72	99.28	0.8717	139.41	0.7494
Listing + feature engineering	78	89.04	0.8968	131.58	0.7767
All	**281**	**66.78**	**0.9419**	**98.98**	**0.8737**
All Transformed Target	281	78.63	0.9195	99.19	0.8731
All - Knowledge-based	**181**	**67.97**	**0.9399**	**99.00**	**0.8736**
All - Knowledge and Mutual Correlation based	107	71.18	0.9340	102.43	0.8584
All - Knowledge, Mutual and Low Correlation based	77	74.55	0.9276	104.06	0.8603
Best 50	77	104.87	0.8568	141.74	0.7409
Best 25	77	153.38	0.6938	180.98	0.5776

Table 6.9: Performance scores of the LightGBM model with tuning, removed features knowledge

Dataset	Root Mean Squared Error (RMSE)	R^2 Score
Train	54.39	0.9615
Validation	96.29	0.8804
Test	99.86	0.8790

RMSE of 124.62) compared to all other models which all achieved an R-squared of around 0.73 and an RMSE of around 143. The addition of feature sets improved model performance, except for adding CBS features. Transforming the data yielded slightly better performance metrics. The highest performance metrics were observed when knowledge-based features only were removed, resulting in 65 features. Hyperparameter tuning with random search resulted in the parameters shown in Table 8.10 in the Appendix. The performance metrics of the tuned model, as shown in Table 6.11, demonstrate that parameter tuning did not result in better performance. The validation and test metrics of the tuned model are comparable. Note that all performance metrics for models predicting the transformed target were rescaled to the original scale before calculation. The R-squared value on the test dataset suggests that 89% of the variance in the data is explained by the CatBoost model, indicating good performance. Notably, the model performed well with the addition of only allocation rule features. The feature importance of the CatBoost model is presented in Figure 6.5, the SHAP values show the impact on the model output before rescaling the predictions. Again the 'Select min age' feature has the most impact on the model output. Similarly, as observed in the feature importance of the LightGBM model, the sum of other features is considerable. The feature-engineered variables 'Age_range' and 'Nearest' did not rank among the top 9 features in terms of importance in the other models. Analysis of the SHAP values from the right segment of the figure indicates that

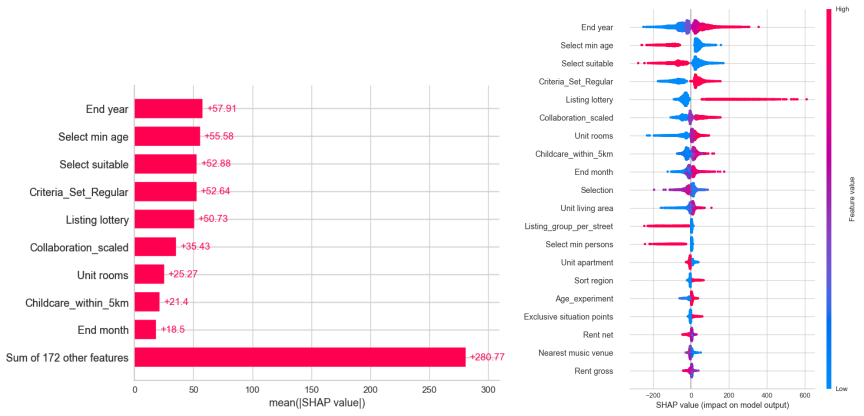


Figure 6.4: Feature importance of LightGBM Regression

a higher value of 'Nearest' has a negative impact on the model output, whereas a higher value of 'Age_range' has a positive impact. The remaining SHAP values are consistent with those from other models, and categorical variables are depicted in grey in the SHAP value visualization.

Table 6.10: CatBoost Performance Comparison

Model	Number of Features	RMSE Train	R^2 Train	RMSE Validation	R^2 Validation
Listing only	28	98.26	0.8743	124.62	0.7997
Listing + Allocation Rules	59	78.91	0.9189	96.78	0.8792
Listing + CBS	188	87.43	0.9005	124.78	0.7992
Listing + Webscraped	37	93.12	0.8871	121.93	0.8082
Listing + feature engineering	43	86.97	0.9015	120.05	0.8141
All Data - CBS	83	71.98	0.9326	96.45	0.8800
Transformed All Data - CBS	83	77.03	0.9228	95.03	0.8835
Transformed All - CBS and Knowledge based	**65**	**77.39**	**0.9221**	**94.36**	**0.8852**
Transformed All - CBS, Knowledge and Mutual Correlation based	50	79.28	0.9182	97.27	0.8780
Transformed All - CBS, Knowledge, Mutual and Low Correlation based	45	82.49	0.9114	99.92	0.8712

Table 6.11: Performance scores of the CatBoost model with tuning

Dataset	Root Mean Squared Error (RMSE)	R^2 Score
Train	53.77	0.9624
Validation	95.08	0.8834
Test	96.77	0.8864

6.3.6 Residuals

The left segment of Figure 8.1 in the Appendix illustrates the distribution of residuals for the five tuned models, focusing on the range from -150 to 150. CatBoost shows the highest concentration of residuals near zero, which partly explains the high performance. It is

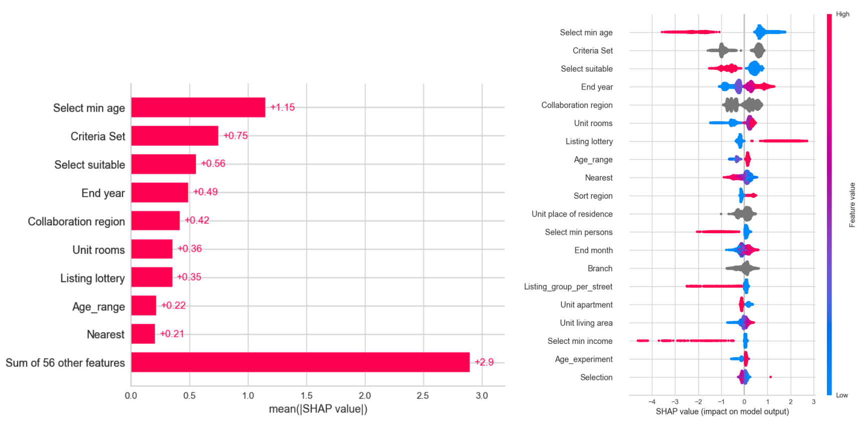


Figure 6.5: Feature importance of CatBoost Regression (Transformed Target)

notable that while the LightGBM model had similar performance, its residual distribution curve more closely resembles those of the other models than that of CatBoost. The right segment of Figure 8.1 in the appendix illustrates the residuals of the Support Vector Regression Model. Figure 8.2 in the Appendix displays the residuals of the Random Forest and the Extra Trees models and 8.3 in the Appendix shows the residuals of the Random Forest and the Extra Trees models. Analyzing the residuals of all models indicates that, although the residuals generally resemble random noise, the range of positive residuals is larger than the range of negative residuals across all models. This suggests that the largest residuals are more often due to under-prediction rather than over-prediction. The CatBoost model shows the most balanced distribution around zero, although its largest residuals are bigger than those of the LightGBM model. To investigate the source of these large residuals, the listings associated with the residuals were examined on WoningNet's website. Remarkably, these are primarily listings of houses in Hilversum or Amersfoort that lack interior photos.

6.4 Full model

The same train, validation, and test sets were utilized for both position prediction and the number of responses prediction, allowing the models to be evaluated together. The position prediction data specifically covers the collaboration region 'Gooi en Vechtstreek.' Due to the random split of the data, the execution remains valid. It is important to note that the scaled mean approach was tested on a smaller subset of listings, which might result in an optimistic bias in the results. The performance metrics for the scaled mean approach in Table 6.12, illustrate that the R-squared values on the test set range from 0.79 to 0.86. This represents a satisfactory level of performance. The RMSE values range between 126 and 156, which is acceptable if the largest squared errors are for applicants with an observed low or poor position. However, it would not be ideal if the largest squared errors originate from applicants with a high or good position. The sorting ratio approach, detailed in Table 6.13, shows R-squared values on the test set ranging from 0.75 to 0.81. These values are lower compared to the scaled mean approach, this method is applied to all listings and is scalable. The RMSE values for this approach are between 130 and 151. It stands out that the best RMSE observed in the sorting ratio approach (130 for the LightGBM model) is lower than the RMSE values of all other models using the scaled mean approach.

Table 6.12: Combined model performance of the scaled mean approach

		SVR	**RF**	**ET**	**LightGBM**	**CatBoost**
Train	RMSE	88.68	75.61	66.50	65.43	70.08
	R-squared	0.9012	0.9282	0.9445	0.9462	0.9383
Validation	RMSE	101	89.12	92.67	90.76	95.68
	R-squared	0.8474	0.88095	0.8713	0.8766	0.8628
Test	RMSE	155.83	147.15	149.37	126.63	131.50
	R-squared	0.7868	0.80987	0.8041	0.8592	0.84815

Table 6.13: Combined model performance of the sorting ratio approach

		SVR	**RF**	**ET**	**LightGBM**	**CatBoost**
Train	RMSE	98.76	90.40	81.86	78.44	84.25
	R-squared	0.8816	0.9008	0.9187	0.9253	0.9138
Validation	RMSE	129.51	124.98	123.30	116.51	121.42
	R-squared	0.7971	0.8110	0.8161	0.8358	0.82165
Test	RMSE	150.76	146.61	144.35	130.49	136,78
	R-squared	0.7477	0.7614	0.7687	0.8109	0.79235

7 Discussion

This chapter first discusses the limitations of the research and offers suggestions for possible directions in future research. The results seem promising, but it is important to recognize that the study has limitations before drawing conclusions.

7.1 Limitations

This research had certain limitations that will be acknowledged. One of the limitations was the presence of noise in the data, which likely impacted the model's performance. This noise arises from various sources, such as data entry inconsistencies and listing type variations. Variations in listing types were not distinguishable in the data. Some listings were pre-listings, where users express interest in a property without actually securing it. This is often the case for new-build homes but can occur with other types as well. The inclusion of these pre-listings without clear differentiation from actual listings complicates data modelling. Furthermore, sorting criteria within the data lacked specification. For instance, the minimum and maximum family size was not specified in the selection criteria, which affects the group allowed to respond to a listing. Moreover, the criteria for sorting text and listings were not always consistent with those used on the website. This is also a primary complaint among Woningnet users, as incorrect criteria can affect the popularity of a listing. It should be noted, however, that Woningnet is actively addressing these issues, suggesting that the inconsistencies are likely attributable to the data currently being utilized. The availability of homes with zero responses in the listing data was another limitation. Although listings with no responses are probably rare, their absence can affect the model's performance and ability to predict the number of responses to a listing accurately. In Amsterdam, recent changes to housing application rules have introduced the concept of "search points", where applicants apply to listings not necessarily out of interest but to accumulate points. This creates uncertainty about the model's applicability when scaling to this region, as the behaviour driven by search points might not reflect genuine interest in the listings. Fortunately, other regions do not have plans to implement similar systems, limiting this issue's impact to the region of Amsterdam. Moreover, due to time constraints, the final positions were not converted into chance classes yet. Lastly, the mean approach for predicting the final positions of applicants was conducted on a set of 1,206 listings that follow the same allocation rule, whereas the median approach was applied to 5,577 listings with various allocation rules. This difference makes the comparison between the two approaches not entirely fair.

7.2 Future research

For future research, there are numerous opportunities to explore. Expanding the sorting ratio approach by grouping the target groups is one such opportunity. For instance, senior applicants might have a wider range of registration years compared to starters. Another interesting future research direction involves replacing the current position prediction approach with a machine learning model. This model could assign specific weights to each allocation rule, with some rules being more influential than others. For example, the first criterion might carry more weight than the fifth criterion. Combining this with the listing data could lead to a model that directly predicts the position. To ensure the model's accuracy, it is important to split the data based on listing ID and perhaps undersample listings with many applicants. This can help avoid overfitting on popular listings. Moreover, using classification techniques for analyzing positions and chances, rather than regression techniques, would be interesting. Given the class imbalance, special attention should be paid to the first 10 positions, as those are the most important. Currently, Woningnet is working on designing a new data warehouse. It would be beneficial for this new data warehouse to include specific allocation rules. In addition, implementing restrictions on criteria entry could enhance data reliability. This includes making the input criteria more robust to prevent minor spelling errors from creating new, unintended rules, aligning sorting text criteria with listing criteria, and restricting urgency dates or registration dates that are later than the actual registration date. A more challenging but valuable task would be standardizing the allocation rules across different regions to ensure consistency, leading to a more reliable grouping of allocation rules. Exploring these directions in future research could improve the prediction of allocation chances. It would be valuable to convert both the currently predicted final positions and the actual positions into chance classes and to construct a confusion matrix to assess the performance. For future research, it would also be interesting to research how the combined models using LightGBM or CatBoost perform when tuning the model with only the added allocation rules features, as this requires fewer data sources. Moreover, examining the allocation chance by looking at the number of refusals on the offer by listing would be interesting. Furthermore, exploring the influence of pictures on the website could provide valuable insights into applicant behaviour and preferences. In addition, using natural language processing to analyze the listing descriptions could be beneficial.

8

Conclusion

Throughout this study, this research aimed to address the following question:

> *"How does incorporating the predicted number of responses affect the likelihood or final position prediction of a prospective tenant securing a social housing unit?"*

The dataset was evaluated using different approaches to determine the effectiveness of incorporating the predicted number of responses. The mean approach without scaling demonstrated poor predictive performance, with an R-squared value of 0.12 and an RMSE of 299 on the test dataset. The low R-squared indicates that the model explained only 12% of the variance in the data, making it unreliable. Incorporating the number of responses significantly improved the performance of the mean approach, achieving a high R-squared value of 0.99 on the test set, suggesting high performance. However, this approach was limited to a single set of rules, restricting its applicability across different listings. The sorting criteria method also performed well, with an R-squared value of 0.97 on the test set. Unlike the scaled mean approach, the sorting criteria approach is scalable across different regions and sets of sorting criteria. However, its success heavily depends on the appropriate grouping of the sorting criteria, which could be seen as a potential weakness. Incorporating the mean significantly improved the performance of the mean approach and was essential for utilizing the sorting ratio approach. Traditional models demonstrated strong predictive capabilities without using machine learning, provided that the number of responses was known. However, in practical scenarios, the number of responses is typically unknown. Chance groups were defined based on the likelihood of receiving an offer. Only applicants with a final position of 1 were categorized into the 'very high' chance group, whereas those with a position of 13 or worse were classified into the 'very low' chance group. The performance of the five machine learning models was similar, with R-squared values ranging from 0.84 to 0.89 on the test set and RMSE values ranging from 92 to 115. This is a reasonable performance. CatBoost achieved the best performance, while Support Vector Regression had the poorest performance. The difference between the training and test performance metrics of all models indicates some overfitting, though it is not severe. However, there is room for improvement due to the presence of noise in the data. The minimum age criterion for selection was the most important feature across most tree-based models. Senior listings often have a minimum age requirement, and those typically receive fewer responses. For the LightGBM model, the end year was an important feature, suggesting growth in the number of responses to listings over time. The residuals of the models appear as random noise and are roughly equally distributed among negative and positive values. However, the positive residuals had a wider range than the negative residuals, indicating that the largest residuals are more often due to under-prediction rather than over-prediction. Remarkably, the listings primarily consisted of houses in Hilversum and Amersfoort that lacked interior photos. Including other location features might improve the model. When adding different sets of features, the allocation rules emerged as the best addition to the listing data. Incorporating CBS data might not be necessary as it only led to small improvements, although the feature-engineered value 'Distance' derived from this dataset was among the top 9 most important features in the CatBoost model. The webscraped values also only led to small improvements, although they were used for data cleaning in the allocation rules. Relying solely on the 25 or 50 most important variables resulted in poor model performance for the SVR, RF, ET, and LightGBM models. Based on the results of combining machine learning models that predict the number of responses with traditional position methods, several conclusions can be drawn. Both position prediction approaches perform worse when using the estimated length of the applicant list compared to the true length. The scaled mean approach shows higher R-squared values on the test set, ranging from 0.79 to 0.86, indicating a satisfactory level of performance. The RMSE values for the scaled mean approach range from 126 to 156, which are acceptable, provided that the largest errors are associated with applicants in lower or poorer positions rather than those in higher or better positions. However, this approach may carry an optimistic bias due to the smaller sample size. In contrast, the sorting ratio approach, which is scalable and applicable to all listings, yielded slightly lower R-squared values on the test set, ranging from 0.75 to 0.81. Notably, the best RMSE value observed with the sorting ratio approach (130 for the LightGBM model) is lower than the RMSE values for all other models using the scaled mean approach. It is noteworthy that the LightGBM model demonstrated slightly better performance metrics compared to the CatBoost model across both combined approaches. However, the CatBoost model previously showed slightly better performance specifically in predicting the number of responses.

To address the research question, *"How does incorporating the predicted number of responses affect the likelihood or final position prediction of a prospective tenant securing a social housing unit?"*, it can be concluded that incorporating predicted responses enhances the performance of predicting the likelihood. The traditional methods, using the actual number of responses, demonstrated strong predictive capabilities. Incorporating the predicted number of responses, while naturally lowering the performance metrics, still yielded reasonably good results. Specifically, the LightGBM model achieved the highest score with a test R-squared of 0.81 and an RMSE of 130 for the scalable sorting ratio approach. Further investigation is warranted to understand where the largest residuals occur, particularly in positions with fewer responses. From a practical perspective, while the results are promising and exceed initial expectations, the current models require further refinement before they can be implemented for direct use by Woningnet. Future research, as outlined in Chapter 7, aims to address these limitations and enhance the models further. Unfortunately, due to the current lack of comprehensive data, such as detailed allocation rules and high-quality web-scraped data, significant improvements remain challenging at this stage.

References

[1] Sociale huurwoningen gaan om meer dan de huurprijs. *Aedes*, 08 2022. URL https://aedes.nl/huurbeleid-en-betaalbaarheid/sociale-huurwoningen-gaan-om-meer-dan-de-huurprijs. 2

[2] M. Awad and R. Khanna. *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers.* Apress, USA, 1st edition, 2015. ISBN 1430259892. 34, 35

[3] A. Begum, N.J. Kheya, Z. Rahman, et al. Housing price prediction with machine learning. *International Journal of Innovative Technology and Exploring Engineering*, 11(3), 01 2022. 13

[4] R. Bentley, K. Baker, E. Simons, J. A. Simpson, and Blakely T. The impact of social housing on mental health: longitudinal analyses using marginal structural models and machine learning-generated weights. *International Journal of Epidemiology*, 47 (5):1414–1422, 10 2018. 11

[5] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(2):281–305, 2012. 40

[6] P. Boelhouwer. The housing market in the netherlands as a driver for social inequalities: proposals for reform. *International Journal of Housing Policy*, 20(3):447–456, 07 2020. 1

[7] A. Boschetti and L. Massaron. *Python Data Science Essentials.* Packt, UK, 3th edition, 2018. ISBN 978-1-78953-786-4. 38

[8] N. Chakrabarty, S. Chowdhury, and S. Rana. A statistical approach to graduate admissions' chance prediction. In *(eds) Innovations in Computer Science and Engineering. Lecture Notes in Networks and Systems*, page 333–340. Singapore: Springer Singapore, 03 2020. 11

[9] Y. Chen, R. Xue, and Y. Zhang. House price prediction based on machine learning and deep learning methods. In *2021 International Conference on Electronic Information Engineering and Computer Science (EIECS)*, pages 699–702. IEEE, 2021. 13, 34

[10] Essam Al Daoud. Comparison between xgboost, lightgbm and catboost using a home credit dataset. *International Journal of Computer and Information Engineering*, 13 (1):6–10, 2019. 37

[11] G.V. Dhruva Kumar, V. Deepa, N. Vineela, and G. Emmanuel. Detection of parkinson's disease using lightgbm classifier. In *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, pages 1292–1297. IEEE, 2022. 37

[12] I. El Guabassi, Z. Bousalem, R. Marah, and A. Qazdar. A recommender system for predicting students' admission to a graduate program using machine learning algorithms. *International Journal of Online and Biomedical Engineering (iJOE)*, 17(1): 135–147, 02 2021. 11

[13] D. Gunay. Random forest, 09 2023. URL `https://medium.com/@denizgunay/random-forest-af5bde5d7e1e`. 36

[14] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer New York, NY, 2 edition, 2009. ISBN 978-0-387-84858-7. 36

[15] S. Jaiswal and P. Gupta. Ensemble approach: Xgboost, catboost, and lightgbm for diabetes mellitus risk prediction. In *2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA)*, pages 1–6. IEEE, 2022. 37

[16] S.B. Jha, R.F. Babiceanu, V. Pandey, and R.K. Jha. Housing market prediction problem using different machine learning algorithms: A case study. *arXiv preprint arXiv:2006.10092*, 1(1):433–442, 06 2020. 13

[17] L.M. John, R. Shinde, S. Shaikh, and D. Ashar. Predicting house prices using machine learning and lightgbm. *7th International Conference on Innovations and Research in Technology and Engineering*, 1(1), 04 2022. 14, 39

[18] N. Joshi Padma, J. Sreekar, P. Chandana, Y. Sowmith Reddy, and Kavya G. University admission prediction using machine learning. *Global Journal of Research and Review*, 1(1):558–563, 05 2022. 11

[19] K.S. Kalaivani, C.S. Kanimozhiselvi, Z.H. Mohamed Bilal, G. Sukesh, and S. Yokeswaran. A comparative study of regression algorithms on house sales price prediction. *Institute of Electrical and Electronics Engineers*, 1(1):826–831, 09 2023. 13

[20] C. Kaynar and M. M. Özçiloğlu. Admission chance prediction models for the school of physical education and sports candidates. In *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–3. IEEE, 06 2021. 11, 12

[21] S. Knaven. Waarom je als woningcorporatie niet om artificial intelligence heen kunt., 12 2019. URL `https://www.sgnm.nl/artikelen/waarom-je-als-woningcorporatie-niet-om-artificial-intelligence-heen-kunt`. 11

[22] J. Korstanje. *Advanced Forecasting with Python: With State-of-the-Art-Models Including LSTMs, Facebook's Prophet, and Amazon's DeepAR*. Apress, France, 1st edition, 2021. ISBN 978-1-4842-7149-0. 37, 39

[23] L. Kraniotis. Schreeuwend tekort aan woningen, wat moet eraan gedaan worden? *NOS*, 02 2021. URL `https://nos.nl/artikel/2369109-schreeuwend-tekort-aan-woningen-wat-moet-eraan-gedaan-worden`. 1

[24] L. Kraniotis and W. De Jong. Sociale huurwoning? in zeker een kwart van de gemeenten wacht je meer dan 7 jaar. *NOS*, 04 2021. URL `https://nos.nl/op3/artikel/2377995-sociale-huurwoning-in-zeker-een-kwart-van-de-gemeenten-wacht-je-meer-dan-7-jaar`. 2

[25] I. Lacroix, O. Z. Güzelci, and J. P. Sousa. Evolutive dataset for social housing design projects through artificial intelligence: From pixel to bim through deep learning. In *Digital Design Reconsidered-Proceedings of the 41st Conference on Education and Research in Computer Aided Architectural Design in Europe*, page 629–639. eCAADe 2023, 09 2023. 11

[26] W. Liang, S. Luo, G. Zhao, and H. Wu. Predicting hard rock pillar stability using gbdt, xgboost, and lightgbm algorithms. *Mathematics*, 8(5), 2020. 38

[27] A. Maulana, T. R. Noviandy, N. R. Sasmita, M. Paristiowati, R. Suhendra, E. Yandri, J. Satrio, and R. Idroes. Optimizing university admissions: A machine learning perspective. *Journal of Educational Management and Learning*, 1(1):1–7, 08 2023. 11

[28] T. Mohd, S. Masrom, and N. Johari. Machine learning housing price prediction in petaling jaya, selangor, malaysia. *International Journal of Recent Technology and Engineering*, 8(2), 9 2019. 13, 36

[29] A. Morris, C. Robinson, and J. Idle. Dire consequences: waiting for social housing in three australian states. *Housing Studies*, 1(1):1–22, 05 2023. 3

[30] S. Mouissie and Kraniotis L. Schreeuwend tekort aan woningen en hoge huizenprijzen: hoe is het zo gekomen? *NOS*, 11 2023. URL `https://nos.nl/collectie/13960/artikel/2497415-schreeuwend-tekort-aan-woningen-en-hoge-huizenprijzen-hoe-is-het-zo-gekomen`. 1, 2

[31] M. Omaer Faruq Goni, A. Matin, T. Hasan, M. Abu Ismail Siddique, O. Jyoti, and F.M. Sifnatul Hasnain. Graduate admission chance prediction using deep neural network. In *2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE)*, pages 259–262. IEEE, 12 2020. 11

[32] N. Rasifaghihi. From theory to practice: Implementing support vector regression for predictions in python, 04 2023. URL `https://medium.com/@niousha.rf/support-vector-regressor-theory-and-coding-exercise-in-python-ca6a7dfda927`. 35

[33] A. Sivasangari, V. Shivani, Y. Bindhu, D. Deepa, and R. Vignesh. Prediction probability of getting an admission into a university using machine learning. In *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pages 1706–1709. IEEE, 04 2021. 11

[34] S. Sridhar, S. Mootha, and S. Kolagati. A university admission prediction system using stacked ensemble learning. In *2020 Advanced Computing and Communication Technologies for High Performance Applications (ACCTHPA)*, pages 162–167. IEEE, 10 2020. 11, 12

[35] R. Szczepanek. Daily streamflow forecasting in mountainous catchment using xgboost, lightgbm and catboost. *Hydrology*, 9(12), 2022. ISSN 2306-5338. 37

[36] Q. Truong, Dang H. Nguyen, M., and B. Mei. Housing price prediction via improved machine learning techniques. *Procedia Computer Science*, 174(1):433–442, 06 2020. 13

[37] T. Trzciński and P. Rokita. Predicting popularity of online videos using support vector regression. *IEEE Transactions on Multimedia*, 19(11):2561–2570, 2017. 35

[38] M. Van Den Eerenbeemt and X. Van Uffelen. Wie krijgt een sociale huurwoning? *de Volkskrant*, 12 2023. URL `https://www.volkskrant.nl/kijkverder/v/2023/voorrangsregels-bij-toekenning-sociale-huurwoningen~v951232/`. 1

[39] Qingdong Zhou, Penghui Zhu, Zhiming Huang, and Quanzhong Zhao. Pest bird density forecast of transmission lines by random forest regression model and line transect method. In *2020 7th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS)*, pages 527–530. IEEE, 2020. 13, 36

Appendix

Table 8.1: Listing dataset

Listing	Collaboration	Criteria Set	Branch	Responses	Start Listing	End Listing	End year	End month	Unit rooms	Unit address
31217	Woongaard	Regular	KleurrijkWonen	89	2019-12-24	2020-01-01	2020	1	4	Heivlinder 14, 4007JD Tiel
31330	Woongaard	Senior	Woonservice Meander	1	2019-12-24	2020-01-02	2020	1	2	Ridder van de Merwedestraat 40, 4268GR Meeuwen
31329	Woongaard	Regular	KleurrijkWonen	189	2019-12-24	2020-01-02	2020	1	3	Debussystraat 65, 4102AR Culemborg
31274	Woongaard	Regular	Woonservice Meander	77	2019-12-24	2020-01-02	2020	1	3	Wijnpeerhoef 4, 4266NB Eethen

Listing	Unit zipcode	Unit zipcode 4	Unit floor	Unit elevator	Unit independent	Unit living area	Listing drawing	Listing new build	Listing corporation	Rent class
31217	4007JD	4007	0	0	1	72	0	0	1	Maximum rental limit
31330	4268GR	4268	0	0	0	76	0	0	1	High capping limit
31329	4102AR	4102	1	0	0	53	0	0	1	Low capping limit
31274	4266NB	4266	1	0	0	66	0	0	1	High capping limit

Listing	Rent net	Rent gross	Rent subs	Rent service charge subs	Rent service charge	Personalized rent	Exclusive situation points	Reply term
31217	717.41	720.01	717.41	0.00	2.60	0	0	9
31330	636.87	636.87	636.87	0.00	0.00	0	0	10
31329	524.91	532.44	524.91	0.00	7.53	0	0	10
31274	606.75	626.75	622.75	16.00	4.00	0	0	10

Listing	Published units	Unit municipality	Unit residence	Unit accessibility	Unit home type detail	Listing contract form	Listing target group	Unit apartment
31217	1	Tiel	Tiel	House without special accessibility	Terraced house	Indefinite contract	Family	0
31330	1	Altena	Meeuwen	House without special accessibility	Terraced house	Indefinite contract	Seniors	0
31329	1	Culemborg	Culemborg	House without special accessibility	Gallery flat	Indefinite contract	Family	1
31274	1	Altena	Eethen	House without special accessibility	Upstairs apartment	Indefinite contract	Person	1

Table 8.2: Selection and Sorting Criteria dataset

Listing	Selection	Sorting	Select max income	Select appropriate	Select others	Select max persons	Select min age	Select no kids	Select max age high	Select min persons	Select min income	Select max age low	Select flexible	Select start study	Sort other
100056	2	5	1	0	0	0	1	0	0	0	0	0	0	0	5
100058	2	4	1	1	0	0	0	0	0	0	0	0	0	0	3
100060	1	3	1	0	0	0	0	0	0	0	0	0	0	0	2
100061	1	5	1	0	0	0	0	0	0	0	0	0	0	0	3

Listing	Sort primair	Sort senior low	Sort urgent	Sort regulation	Sort lotery	Sort flow	Sort income low	Sort middleage	Sort young	Sort max persons	Sort min persons	Sort region	Sort residence	Sort family	Sort senior high	Sort registration date
100056	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100058	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
100060	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100061	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0

Table 8.3: CBS 4-length zip code data (https://download.cbs.nl/postcode/2023-CBS_pc4_2022_v1.zip)

Zip code 4	Total persons	Male	Female	age_t_15	age_15_25	age_25_45	age_45_65	age_65p	NL_origin_NL	NL_origin_-EU	NL_origin_OutsideEU	OutsideNL_origin_EU	OutsideNL_origin_outsideEU
1012	8800	0.55	0.45	0.04	0.19	0.48	0.20	0.09	0.40	0.00	0.10	0.30	0.20
1013	21940	0.50	0.50	0.11	0.11	0.36	0.27	0.15	0.50	0.00	0.10	0.10	0.20
1014	3660	0.51	0.49	0.21	0.05	0.44	0.23	0.07	0.60	0.10	0.10	0.10	0.10
1015	14910	0.50	0.50	0.08	0.12	0.35	0.28	0.17	0.50	0.10	0.10	0.20	0.20

Zip code 4	Total families	Single	Multiple persons without children	Single-parent	Two-parent	Household size	Total houses	build_b_1945	build_1945_1965	build_1965_1975	build_1975_1985	build_1985_1995
1012	6275	0.68	0.25	0.03	0.04	1.40	6005	0.83	0.00	0.01	0.02	0.07
1013	13470	0.62	0.21	0.07	0.10	1.60	12560	0.54	0.02	0.01	0.13	0.12
1014	1705	0.39	0.29	0.04	0.28	2.10	1745	0.01	0.00	0.06	0.00	0.01
1015	9875	0.65	0.22	0.05	0.08	1.50	9965	0.74	0.01	0.02	0.08	0.11

Zip code 4	build_1995_2005	build_2005_2015	build_2015l	Multi-family	Owner-occupied home	Rental home	Rental corporation	Not inhabited	Mean woz	Persons on benefits	addresses/km2	category
1012	0.04	0.02	0.01	0.95	0.20	0.80	0.12	0.18	480.00	0.07	8578	1
1013	0.04	0.10	0.04	0.99	0.30	0.70	0.51	0.06	471.00	0.12	6581	1
1014	0.11	0.00	0.81	0.90	0.50	0.50	0.09	0.10	731.00	0.05	3309	1
1015	0.03	0.01	0.00	0.95	0.30	0.70	0.35	0.12	532.00	0.09	11080	1

Table 8.4: CBS full zip code data (https://download.cbs.nl/postcode/2023-cbs_pc6_2020_vol.zip)

Zip Code	Total	Male	Female	Age < 15	Age 15-25	Age 25-45	Age 45-65	Age > 65	NL Origin	Western Origin	Non-Western Origin	Total Households	Single Person Household
1011AC	20	0.50	0.50	0.17	0.17	0.25	0.17	0.25	0.60	0.20	0.20	10	0.50
1011AG	5	0.50	0.50	0.00	0.00	1.00	0.00	0.00	0.33	0.33	0.33	3	0.25
1011AH	30	0.67	0.33	0.08	0.08	0.67	0.08	0.08	0.40	0.30	0.30	15	0.33
1011AJ	20	0.75	0.25	0.08	0.08	0.50	0.25	0.08	0.60	0.20	0.20	10	0.50

Zip Code	Multi-Person Without Children Household	Single Parent Household	Two Parent Household	Household Size	Total Housing	Built Before 1945	Built 1945-1965	Built 1965-1975	Built 1975-1985	Built 1985-1995	Built 1995-2005	Built 2005-2015
1011AC	0.17	0.17	0.17	1.70	3	0.12	0.12	0.12	0.12	0.12	0.12	0.12
1011AG	0.25	0.25	0.25	1.70	3	0.12	0.12	0.12	0.12	0.12	0.12	0.12
1011AH	0.67	0.00	0.00	1.80	15	1.00	0.00	0.00	0.00	0.00	0.00	0.00
1011AJ	0.17	0.17	0.17	1.50	15	1.00	0.00	0.00	0.00	0.00	0.00	0.00

Zip Code	Built After 2015	Owner-Occupied House	Rented House	Property Value	Gas Consumption	Electricity Consumption	Nearest Supermarket	Supermarket Within 1km	Supermarket Within 3km	Supermarket Within 5km	Nearest Grocery Store	Grocery Store Within 1km
1011AC	0.12	0.50	0.50	517	1190.00	2830.00	0.50	1.00	43.00	132.00	0.70	9.00
1011AG	0.12	0.50	0.50	517	1190.00	2830.00	0.40	10.00	78.00	154.00	0.00	109.00
1011AH	0.00	0.20	0.80	304	860.00	1430.00	0.40	10.20	77.90	153.80	0.00	107.50
1011AJ	0.00	0.50	0.50	762	1160.00	2290.00	0.40	11.00	79.00	153.00	0.00	106.00

Zip Code	Grocery Store Within 3km	Grocery Store Within 5km	Nearest Department Store	Department Store Within 5km	Department Store Within 10km	Department Store Within 20km	Nearest Cafe	Cafe Within 1km	Cafe Within 3km	Cafe Within 5km	Nearest Snack Bar
1011AC	383.00	885.00	1.20	13.00	23.00	42.00	0.70	9.00	389.00	670.00	0.40
1011AG	582.00	966.00	0.60	13.00	23.00	45.00	0.10	161.00	533.00	695.00	0.00
1011AH	583.80	966.40	0.60	13.00	23.00	45.00	0.10	159.90	533.00	694.80	0.00
1011AJ	588.00	966.00	0.60	13.00	23.00	45.00	0.10	158.00	533.00	694.00	0.00

Zip Code	Snack Bar Within 1km	Snack Bar Within 3km	Snack Bar Within 5km	Nearest Hotel	Hotel Within 5km	Hotel Within 10km	Hotel Within 20km	Nearest Restaurant	Restaurant Within 1km	Restaurant Within 3km	Restaurant Within 5km	Nearest Daycare	Daycare Within 1km
1011AC	15.00	461.00	902.00	0.50	362.00	434.00	505.00	0.10	32.00	800.00	1717.00	1.10	0.00
1011AG	156.00	662.00	954.00	0.10	377.00	439.00	514.00	0.00	250.00	1252.00	1831.00	0.40	3.00
1011AH	155.50	662.20	953.50	0.10	377.00	439.00	514.20	0.00	248.20	1253.80	1833.50	0.40	3.00
1011AJ	157.00	666.00	952.00	0.20	377.00	439.00	515.00	0.00	244.00	1258.00	1835.00	0.40	3.00

Zip Code	Daycare Within 3km	Daycare Within 5km	Nearest Kindergarten	Kindergarten Within 1km	Kindergarten Within 3km	Kindergarten Within 5km	Nearest Fire Station	Nearest Highway	Nearest Train Transfer Station	Nearest Train Station	Nearest Attraction	Attraction Within 10km
1011AC	41.00	131.00	1.10	0.00	54.00	263.00	1.60	5.00	1.10	1.10	1.80	6.00
1011AG	53.00	161.00	0.40	6.00	105.00	326.00	0.90	4.40	0.50	0.50	1.10	6.00
1011AH	53.90	161.60	0.40	6.00	105.70	326.00	0.90	4.40	0.50	0.50	1.10	6.00
1011AJ	55.00	162.00	0.40	6.00	108.00	326.00	0.90	4.50	0.50	0.50	1.10	6.00

Zip Code	Attraction Within 20km	Attraction Within 50km	Nearest Cinema	Cinema Within 5km	Cinema Within 10km	Cinema Within 20km	Nearest Museum	Museum Within 5km	Museum Within 10km	Museum Within 20km	Nearest Performing Arts Venue	Performing Arts Venue Within 5km
1011AC	12.00	57.00	2.00	11.00	13.00	17.00	1.20	43.00	46.00	67.00	0.90	35.00
1011AG	13.00	57.00	1.30	12.00	13.00	19.00	0.50	44.00	47.00	67.00	0.50	37.00
1011AH	13.00	57.00	1.30	12.00	13.00	19.00	0.50	44.00	47.00	67.00	0.50	37.00
1011AJ	13.00	57.00	1.30	12.00	13.00	19.00	0.50	44.00	47.00	67.00	0.40	37.00

Zip Code	Performing Arts Venue Within 10km	Performing Arts Venue Within 20km	Nearest Library	Nearest Ice Rink	Nearest Music Venue	Nearest Sauna	Nearest Tanning Salon	Nearest Swimming Pool	Nearest School	School Within 1km	School Within 3km	School Within 5km
1011AC	48.00	59.00	0.40	5.50	3.10	1.40	2.30	2.70	1.10	0.00	19.00	79.00
1011AG	50.00	64.00	0.60	4.80	2.40	0.80	1.70	2.10	0.40	2.00	28.00	98.00
1011AH	50.00	64.00	0.60	4.80	2.40	0.80	1.80	2.10	0.40	2.00	28.60	98.00
1011AJ	50.00	64.00	0.60	4.80	2.40	0.80	1.80	2.10	0.40	2.00	29.00	98.00

Zip Code	Nearest HAVO/VWO School	HAVO/VWO School Within 3km	HAVO/VWO School Within 5km	HAVO/VWO School Within 10km	Nearest VMBO School	VMBO School Within 3km	VMBO School Within 5km	VMBO School Within 10km	Nearest Secondary School	Secondary School Within 3km	Secondary School Within 5km	Secondary School Within 10km	Nearest GP
1011AC	1.80	1.00	22.00	48.00	1.80	1.00	20.00	54.00	1.80	1.00	33.00	75.00	1.00
1011AG	1.20	9.00	27.00	48.00	1.20	5.00	29.00	55.00	1.20	9.00	45.00	76.00	0.30
1011AH	1.20	9.50	27.20	48.00	1.20	5.50	29.00	55.00	1.20	9.50	45.20	76.00	0.20
1011AJ	1.20	11.00	28.00	48.00	1.20	7.00	29.00	55.00	1.20	11.00	46.00	76.00	0.20

Zip Code	GP Within 1km	GP Within 3km	GP Within 5km	Nearest Hospital	Hospital Within 5km	Hospital Within 10km	Hospital Within 20km	Nearest Hospital Outpatient Clinic	Hospital Outpatient Clinic Within 5km	Hospital Outpatient Clinic Within 10km	Hospital Outpatient Clinic Within 20km	Nearest Pharmacy	Nearest GP Post
1011AC	1.00	31.00	111.00	3.30	1.00	5.00	10.00	2.10	2.00	6.00	14.00	0.80	3.30
1011AG	4.00	56.00	129.00	2.60	1.00	5.00	10.00	1.40	2.00	6.00	15.00	0.40	2.60
1011AH	4.00	56.00	129.60	2.60	1.00	5.00	10.00	1.50	2.00	6.00	15.00	0.40	2.60
1011AJ	4.00	56.00	130.00	2.60	1.00	5.00	10.00	1.40	2.00	6.00	15.00	0.40	2.60

Table 8.5: Webscraped data

Listing	number of images	construction year	target group	min persons	max persons	child	unknown persons	age min	age max	no kids
31409	0	1989	Regular	1	10	0	0	0	100	0
31412	0	1972	Regular	1	10	0	0	0	100	0
30539	3	1918	Regular	1	10	0	0	0	100	0
30596	4	1930	Regular	1	10	0	0	0	100	0

Deletion Reason	Features
Knowledge	Corporation publication, Flexible, Male, Female, NL origin EU, Semi-detached house, Attached house, Semi-detached two-under-one-roof house, Room, Detached house
	Temporary contract, Multi-family, Two-under-one-roof house, Seniors, Care, Youth, Start study, Middle age, Senior high, Registration date, Other
	Primary, Lottery, Female, Age 25-45, Age 45-65, Age 65+, Built before 1945
	Built 1945-1965, Built 1965-1975, Rental houses, Food stores within 3 km, Department store within 5 km, Café within 3 km
	Snack bar within 3 km, Snack bar within 5 km, Hotel within 5 km, Restaurant within 3 km, After-school care within 3 km, Daycare within 3 km, Cinema within 5 km
	Museums within 5 km, Performing arts within 5 km, School within 3 km, HAVO/VWO within 3 km, HAVO/VWO within 5 km
	VMBO within 3 km, VMBO within 5 km, VO within 3 km, VO within 5 km
	General practitioner within 3 km, Hospital within 5 km, Hospital outpatient clinic within 5 km, Max persons, Min persons, Unknown persons
	Urgent, Youth, Target group match youth, Max age high, No kids, Target group match old, Nearest attraction, Nearest HAVO/VWO
	Nearest sauna, Nearest tanning bed, Nearest swimming pool, Nearest train
	Nearest museums, Nearest department store, Nearest general practitioners' post, Nearest library
	Nearest fire brigade, Nearest hote, Nearest cinema, Nearest café, Nearest general practitioner, Nearest highway, Nearest after-school care
	Nearest restaurant, Nearest school, Nearest daycare, Nearest ice rink, Nearest food store, Nearest pharmacy, Nearest supermarket, Nearest snack bar
	Attraction within 20 km, After-school care within 1 km, Daycare within 1 km, School within 1 km, General practitioner within 1 km, Rental service costs, Household size
	Cooperation association Gooi and Vechtstreek, Cooperation association Eemvallei, Cooperation association Woongaard, Nearest hospital outpatient clinic
High mutual correlation	Excluded situation points, Living area, Gross rent, Rent subsidy, Log gross rent, Net rent, Senior rule set, Senior care target group, Youth students rule set, Max low age
	Senior low, Region, Total persons, NL origin NL, Outside NL origin outside EU, Total houses, Two-parent, Housing corporation, Owner-occupied house, Nearest performing arts
	Supermarket within 3 km, Supermarket within 5 km , Food stores within 5 km, Intermediate house, Nearest hospital, Restaurant within 5 km, Daycare within 5 km, School within 5 km
	General practitioner within 5 km, Total, Total household, Single-person household, Two-parent household, Supermarket within 1 km, Café within 5 km
	Café within 1 km, Snack bar within 1 km, Restaurant within 1 km, After-school care within 5 km, VMBO within 10 km, Museums within 10 km, HAVO/VWO within 10 km
	VO within 10 km, Hospital outpatient clinic within 10 km, Hospital outpatient clinic within 20 km, Department store within 10 km, Accessible ground-floor house, VMBO nearest
	VO nearest, Nearest pop podium, Hospital within 20 km, Museums within 20 km, Performing arts within 20 km, Cinema within 20 km, Hotel within 10 km
	Hospital within 10 km, Addresses per km^2, Category, Transfer station train, Year of construction, Regular target group match, Performing arts within 10 km
	Department store within 20 km, Age over 15, Food stores within 1 km, Hotel within 20 km, Age range, Max age, Min age, Number of persons, Regular rule set, Origin, NW origin
Low correlation	Age 15-25, Age up to 15, Built 2015 and later, Gas consumption, Unoccupied, Multi-person without children, Age 25-45, Built 1975-1985, Built 1975-1985, Built 2005-2015
	Multi-family, Built 1995-2005, Built 2015 and later, Electricity consumption, Other combination house type, Number of images, Built 1985-1995, Temporary contract Log net rent
	Regular special rule set, End house, Log living area, Child, Portico house, Corner house, Ground floor house, Starter target group, Multi-person without children household, Age 45-65, Sorting

Table 8.6: List of removing feature sets

Table 8.7: Tuned hyperparameters for the Support Vector Regression model

Parameter	Value
C	5
epsilon	0.1
kernel	rbf
degree (only polynomial kernel)	4
gamma	scale

Table 8.8: Tuned hyperparameters for the Random Forest and Extra Trees models

Parameter	Value
max_depth	18
max_features	None
min_samples_leaf	5
min_samples_split	10
n_estimators	550

Table 8.9: Tuned hyperparameters for the LightGBM model

Parameter	**Value**
num_leaves	20
learning_rate	0.05
n_estimators	750
feature_fraction	0.8
subsample	0.95

Table 8.10: Tuned hyperparameters for the CatBoost model

Parameter	**Value**
one_hot_max_size	30
iterations	1500
od_wait	40
learning_rate	0.1
depth	7
l2_leaf_reg	5
random_strength	0.4
bagging_temperature	1.5

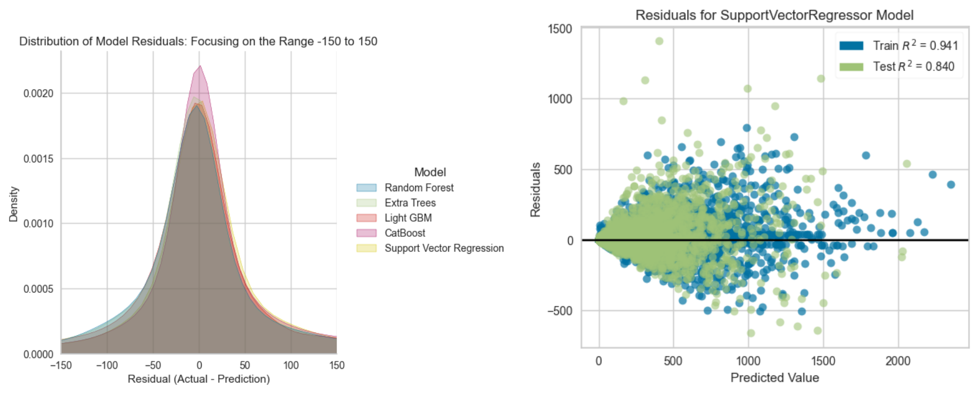


Figure 8.1: All model residuals and SVR model residuals

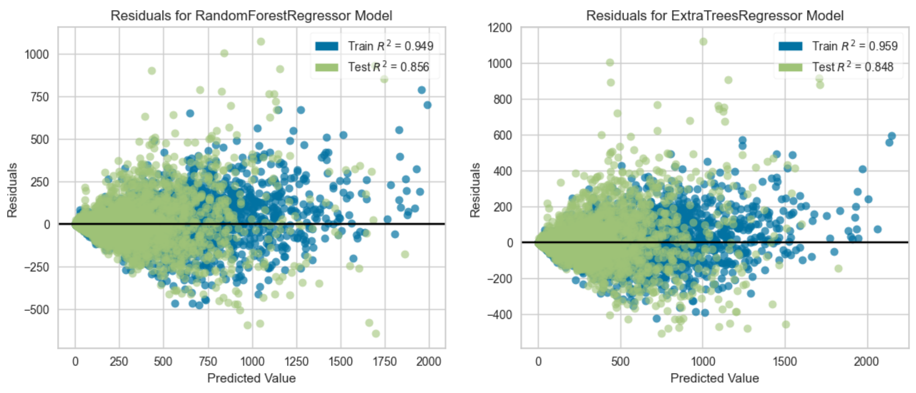


Figure 8.2: Random Forest model residuals and Extra Trees model residuals

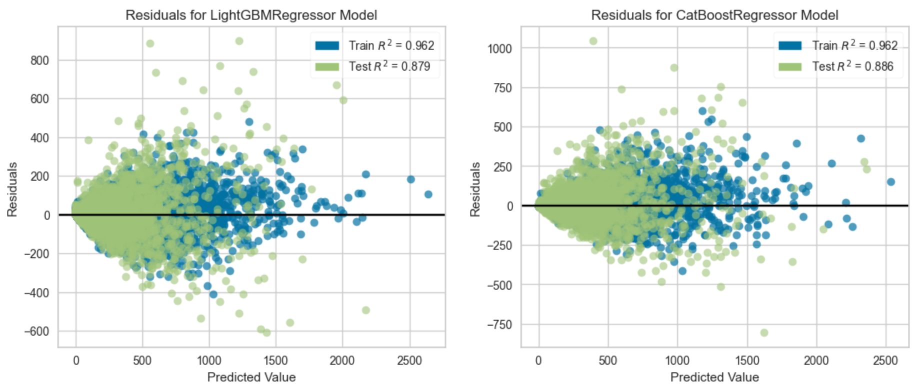


Figure 8.3: LightGBM model residuals and CatBoost model residuals