SHAPing the Supermarket Rental Landscape with Explainable AI and Geodemographics*

Mitchell D. Lobbes[2627692]

Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam

Abstract. This research anticipates supermarket revenue potential by predicting rental prices for supermarkets. It adopts a unique perspective from a commercial real estate investment trust and incorporates Explainable AI techniques. SHAP values are used in combination with the interpretable machine learning models: decision tree, random forest, lightGBM and XGBoost to study the effects of geo-demographic features on supermarket rental prices. Findings show an increase in rental predictions for higher WOZ values and a higher income. Similarly, accessibility indicators such as a low average distance to a transfer station also show higher rental predictions. The models show varying performance, yielding an average accuracy of 83% across the four models but negative R^2 values. Opportunities for future research are focused on expanding datasets, possible adaptation of deep learning models and advanced hyperparameter optimization.

Keywords: Explainable AI · Tree Models · Shapley Values · SHAP · Ensemble Models · Supermarket · XGBoost · Rent Prediction.

* Supported by Software Company Clappform.

1 Introduction

Anticipating a supermarket's revenue potential by predicting how much rent to inquire, holds substantial importance for commercial real estate investment trusts (CREITs). Grocery-anchored real estate contracts often extend over a decade, where the CREIT's primary source of income hinges on rental earnings. Supermarkets are required to pay rent equivalent to 3% of the revenue they generate [1]. Given the prolonged nature of these contracts, the supermarket becomes tied to a particular location for an extended period of time. This emphasizes the importance of the surrounding environment. Thus, exploring the possibilities to accurately and comprehensively predict appropriate rental prices using geodemographic data, stands to offer long-terms benefits for CREITs[1].

A supermarket's revenue potential is important for CREITs, due to the structure of the CREIT's business model. CREIT's primary source of income is the rental income they receive by renting spaces out to commercial tenants, such as supermarkets. The rental price corresponds to a mutually agreed-upon percentage, generally hovering around 3% to 4% of the total revenue generated by the supermarket [1]. According to regulations from the chamber of commerce in the Netherlands (KVK), all supermarkets are obligated to present an annual report containing their revenue numbers [2]. Nonetheless, supermarket chains are not particularly inclined to disclose revenue figures for individual stores. Supermarkets often tend to present their earnings on an aggregated or total basis. This cleverly conceals specifics and makes it challenging to discern the revenue generated by an individual store. Currently, CREITs determine the rental price through comparison with similar supermarkets, or other retail properties, located in comparable areas [3]. In these areas, both rental and revenue data are accessible or are already owned by the CREIT. However, a scenario that often arises involves supermarkets contesting the proposed rental price as being excessively high. The supermarkets argue that achieving such high earnings to justify the suggested rent price is unlikely, leading to CREITs potentially lowering the rental price. This causes a situation of asymmetric information, where one party, in this case the supermarket, has more information than the CREIT. The difficulty here lies in CREIT's ability to verify the supermarket's claim of an unrealistic rental price. This highlights the need to develop a reliable and explainable approach for predicting these rental prices.

Another way for CREITs to generate income is through real estate appreciation. This is as straightforward as selling the real estate property for a price higher than its initial value.

[1] The study stems from an internship project undertaken at software company Clappform. The interest in this research follows from a client's request for seeking an analysis about the feasibility to predict the potential revenue of Dutch supermarkets. Due to a confidentiality agreement, the client will be referred to as Stealth Enterprises throughout the entirety of this thesis.

The formula used to calculate the market value of an income-producing asset is shown in equation 1.

$$\frac{\text{annual net operating income (NOI)}}{\text{cap rate}} = \text{asset's market value} \tag{1}$$

NOI stands for the income generated by the real estate asset, otherwise known as the annual net operating income. This equals the total years' rent for commercial real estate properties. This, once again, highlights the importance of a fitting rental price corresponding to a supermarket's revenue potential at a set location. The higher the amount of yearly rent collected, the higher the asset's market value will be. Another term that influences an asset's market value is the *cap rate*, which stands for capitalization rate.

The capitalization rate, or *cap rate*, provides investors with insights into the anticipated returns on an investment. A higher value of the cap rate means a better potential return on investment. However, this also implies that the investment does carry an additional risk, which results in a lower asset market value.

The *cap rate* shows an investor the return they can expect from an investment and how long it will take for an asset to pay for itself. As a general rule of thumb, a higher cap rate implies that an investment property offers a higher return than a similar investment. However, it also often suggests this investment carries more risk, resulting in a lower asset market value. The *cap rate* is not set by a single entity or organization. Instead, it is influenced by various factors, such as market conditions, state of the property and most importantly geo-demographic factors of the surrounding the real estate asset. [4][2]. The importance of these environmental factors indicate that the better the surrounding neighbourhood of a supermarket is, the better its potential revenue. Therefore, it's essential to recognize that leveraging geo-demographic data from the surrounding neighbourhood can enhance the ability to predict rental prices.

Most previous research regarding rental price prediction, primarily relied on internal store data or information related to competitors' gross profit [5, 6]. There were some studies that did pursue the inclusion of various economic features such as consumers' disposable income [7] or the WOZ value of houses surrounding a retail store [8, 9]. These studies researched the influence of these features using an ordinary least squares method. However, further advanced machine learning methods were not implemented, and therefore possibly failing to fully capture the explanatory power of these features. It was not until recent years that the real significance of adding geographical features was researched [10]. This study focused on predictive modelling by using a small geo-demographic dataset. Features such as neighbourhood points of interest, population numbers and distance to nearest competitor were included. However, a thorough investigation into the extent of the explanatory capabilities of geo-demographic features was not pur-

[2] It is important to note that while market forces heavily influence cap rates, real estate professionals, appraisers, and investors play a significant role in analyzing and interpreting these rates.

sued. Remarkably, there exists a considerable research gap in this domain. Despite, various studies relying on geo-demographic variables, there has been a lack of comprehensive research that exploits a plethora of geo-demographic features for retail rent prediction.
There is exists an similar scarcity of research utilizing explainable AI (XAI) techniques to understand the retail rent landscape. XAI refers to the set of techniques, methods, and approaches used in AI and machine learning to make machine learning models and their predictions more understandable and interpretable [11]. One of the XAI techniques that is used in this research specifies which features have the most influence on a model's prediction. A XAI technique called SHapley Additive exPlanations (SHAP), will leverage 25 geo-demographic features, trying to gain insights into the supermarket rent prediction landscape [12]. To facilitate this explainable endeavor, models known for their interpretability will be included. The models are: decision tree, random forest, lightGBM, and XGBoost. Exploiting the current gap in the existing body of knowledge, this research presents a unique opportunity and places itself within a realm of nicheness. It aims to explore the realm of model interpretability, AI explainability and overall predictive performance.

2 Related Work

To further elaborate on research that support including geo-demographic features, exploring a different sector can shed new light on how these factors can enhance the precision of forecasts. In [14], they emphasize the use of demographic and socioeconomic features in predicting maternal postpartum rehospitalization. Features such as *race*, *smoking habits*, and *income* are used in models such as random forest and decision trees. Furthermore, in Japan an interesting case study explored the effect of meteorological information that included 6 aspects: *solar radiation*, *rainfall precipitation*, *relative humidity*, *temperature*, *north wind velocity*, *east wind velocity* on specific products such as drinks and beverages [15]. While the focus is on different areas, the inclusion of geo-demographic data adopted in the methodology only highlights the variability and wide spectrum of influence geo-demographic data can have.

In a study by Brooks & Tsolacos the use of geo-demographic data was discussed in the context of retail rent predictions in the United Kingdom [7]. Variables like *consumer expenditure*, *retail sales*, *disposable income*, and *gross domestic product* were researched for their potential effect on rent prediction. However, their research results did not support effectiveness of these features. On the contrary, in 2008, a use-case was presented regarding spatial data mining for aggregate sales forecasting in retail location planning [13]. This case study was based on data from the spatial data warehouse of a big European food retailing company and emphasized the use of a variety of spatially aggregated geo-demographic features for sales forecasting. These authors expanded the input demand by including features like *population density*, *urbanity* and *tourism potential* along 49 other explanatory features. Consequently leading to results sup-

porting the importance of geo-demographic features for sales prediction. While [7] did also acknowledge the importance of geo-demographic features, their research did not seem to fully support the idea that their selection of features could provide accurate predictions. On the other hand, [13] not only supports the use of these features but also expands on them, using a comprehensive set of attributes for sales forecasting in retail location planning. Their approach appears to be more data-driven and detailed in its use of these features. In conclusion, while both papers recognize the significance of geo-demographic data, [13] seems to provide a more robust and supportive stance on their usefulness, especially when combining a plethora of features.

An important recent research is the study conducted in 2022 by Ting and Yu Jie [10]. Their research primarily focuses on the importance of geo-demographic data for retail site selections. To capture the importance and effects of these features a random forest and XGBoost model are used. Their research concludes that the XGBoost had the highest accuracy with 94 %, outperforming the random forest. Ting and Yu Jie suggest for future research to include a broad plethora of related features. These authors stated that *"the retail market is seemingly more complex as it tends to be affected by multiple factors"*[10]. This suggests that there is room for further exploration beyond the features the authors investigated. Features like related to nearby transportation, education and economic features are mentioned as possible suggestions for future research. Hence, this body of work not only highlights the potential of geo-demographic data but also provides a solid foundation for our work. By incorporating a wide range of relevant geo-demographic features, this research aims to build upon this foundation, further trying to enhance the understanding of the intricate relationships between geo-demographic data and supermarket rent predictions.

Since geo-demographic data, is typically represented as tabular data, tree models and ensemble techniques reign supreme in this domain [16, 17]. Tree models such as random forest, gradient boosting and XGBoost are great at dealing with uninformative features, which are likely be present in a plethora of geo-demographic features [18]. The research showed that tree-based models beat neural networks across a wide range of hyperparameter choices and that the best methods on tabular data share two attributes: they are ensemble methods, that represent bagging models like random forest or boosting models like gradient boosting or XGBoost. Tree and ensemble models tend to have another advantage, which is interpretability. This research strives to be transparent in its attempt to make accurate supermarket rental predictions. Moreover, people are less likely to accept models that they cannot understand [22]. Interpretable machine learning models address these issues. They are defined as models that can be clearly visualized or explained using plain text to the end-user [22, 23]. Applying these models is important to enhance transparency and understandability in AI, both of which are fundamental aspects of XAI [26, 27]. In contrast to interpretable models, black-box models are models that do not disclose any meaningful information about their outputs or their internal structure [24]. Hence, in this research, the use of models such as artificial neural networks is

not pursued. Instead, the inclusion of more transparent and interpretable models like tree models is preferred. Furthermore, even without the inclusion of geo-demographic, do tree models such as random forest and XGBoost achieve relatively low errors of 0.06 and 0.04 respectively [19]. Even in other areas like rock pillar stability prediction, do the XGBoost and gradient boosting model have an accuracy for 83% and 82% respectively. Whereas, for stock price prediction, these scores were even higher with 90% accuracy reached [20, 21]. In summary, these models demonstrate their versatility and wide-ranging applications, establishing a strong basis for their inclusion into this research. This allows us to leverage their proven capabilities to improve the interpretability of our retail rent predictions.

In recent years, XAI has seen an increasing interest [11, 25]. However, XAI is not a new concept. The earliest work on XAI can be found dated back to 40-year old literature [26, 27]. The upcoming surge of ChatGPT has raised the question how exactly do this machine learning model work and how is its output exactly generated [28]. Hence, a framework that facilitates the interpretation of predicted output values is adopted. This framework is called SHapley Additive exPlanations (SHAP). As can be seen in figure 1, SHAP assigns each feature an importance value for a individual prediction. SHAP is based on the shapley value, which was introduced by Lloyd Shapley in 1953 [29]. In game theory, the shapley value is used to determine the fair distribution of the total value among players who collaborate in a cooperative game. It calculates how much each player contributes to the overall outcome, considering all possible permutations of player combinations. In machine learning it helps in understanding and attributing contributions of individual features, representing the players in game theory, to the prediction made by a machine learning model.

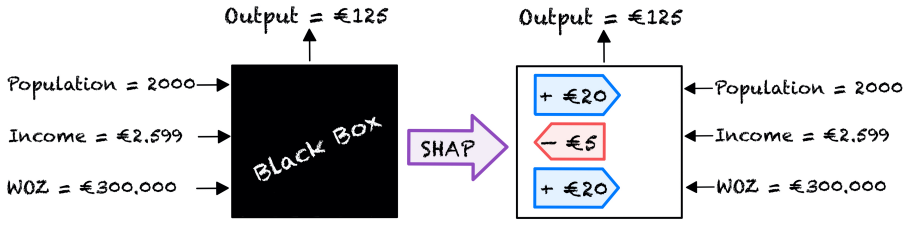


Fig. 1. Simplified intuition of how XAI makes models more explainable using SHAP

SHAP is known to be a unique method that satisfies the properties: local accuracy, missingness and consistency. The desirability of these properties and their uniqueness make a strong case for using shapley values [30]. Previous research showed that SHAP has shown significant strides in terms of explainability [31]. One specific research about detecting traffic accidents showed a great result of an XGBoost model, that can detect accidents with an accuracy, detection rate, and false alarm rate of 99%, 79%, and 0.16%, respectively. Using SHAP, the authors

revealed that the most influential factor on accident detection was the variance in speed before and after an incident occurred at a given location. [32]. This creates a dependable and transparent basis for a crucial prediction task such as accident detection. Therefore, including SHAP to explain which features influence predictions establishes a trustworthy, transparent, and robust foundation for the utilization of our machine learning models.

3 Methodology

The primary dataset used in this research contains information merged from two distinct datasets. Each dataset originates from separate providers. The initial dataset contains data received from Stealth Enterprises and is shown in table 1.

Table 1. Confidential supermarket dataset containing the target value

Variable	Description
Name	Name of the grocery store chain
Zip Code	Zip Code of the supermarket
GLA	Gross Leasable Area
Rental Price	Rental price / m^2 (per year)

Stealth Enterprises' supermarket dataset contains 4 features about 303 dutch supermarkets[3]. table 1 shows the target variable *Rental Price*, which is the supermarket's rental price per m^2. Furthermore, the feature GLA, stands for the grocery store's Gross Leasable Area. This feature is one of the explanatory features. Finally, the *Zip Code* will serve as a key connection point, linking each supermarket to its respective neighbuorhood and associating them with the neighborhood's geo-demographic features. The secondary dataset is based on publicly available geo-demographic data, which is accessible from the central bureau of statistics (CBS) in the Netherlands [33]. The data is aggregated on neighbourhood level and each instance equates to a unique neighbourhood with 25 hand-picked geo-demographic features. These two datasets are merged according to the supermarket's *Zip Code* and their corresponding neighbourhood. The resulting dataset is shown in table 4. The dataset contains 303 supermarkets and 25 explanatory geo-demographic features.

3.1 Data Preparation

Tree-based models inherently address missing values within the tree construction process. Further details regarding the tree construction process will be provided later. For now it is important to know that this built-in capability of tree-based

[3] To ensure confidentiality, all references to specific supermarket chains have been anonymized throughout this research.

models removes the explicit need for imputing missing values. Moreover, imputation can introduce bias or noise into the data, potentially affecting the models in a negative way [34]. Nonetheless, it is advisable to still try and minimize the occurrence of missing values. This is especially the case since the given dataset only contains 303 instances. The initial step in data preprocessing uses a specific approach for handling missing values. The most recently available dataset from CBS serves as the initial dataset. When an instance is missing within this dataset, it is substituted with the corresponding instance from the most recent year for which data is available. This iterative way of replacing dates back to 2018. The remaining missing values by feature and by store are summarized in table 2 and table 3. Only the features and stores that have non-zero missing values are shown. table 2 displays that out of the 25 features, only 12 have missing values. The feature with the highest percentage of missing values is absent in just 22 out of the 303 supermarkets, which is less than 10%. Other features have even lower missing percentages, mostly below 4%. However, given the relatively small dataset, it is advisable to retain as much data as possible and consider using an imputation technique. Examining missing data store-wise in table 2, it becomes apparent that imputation may be beneficial, especially when there are 10 supermarkets with over 30% missing data across 25 features. Despite tree and ensemble models' ability to handle missing data, the objective remains to maximize data quality within the limited size of the dataset. Therefore, the *Scikit-Learn* iterative imputer with *ExtraTreesRegressor* is implemented that predicts missing values based on other features [35][36].

Table 2. Non-zero missings by feature

Feature	**%**	**n**
avg_income	7.26	22
k40_low_income	3.96	12
k20_high_income	3.96	12
grocery_store_1km	3.30	10
d_school	3.30	10
d_transfer	3.30	10
d_mainstreet	3.30	10
n_earners	2.97	9
WOZ	0.99	3
%_single_family_homes	0.66	2
n_owner_occupied	0.66	2
d_supermarket	0.33	1

Table 3. Non-zero missings by store

Store	**%**	**n**
Store 174	42.31	11
Store 293	30.77	8
Store 298	30.77	8
Store 181	30.77	8
Store 296	30.77	8
Store 297	30.77	8
Store 180	30.77	8
Store 301	30.77	8
Store 4	30.77	8
Store 5	30.77	8
Store 207	23.08	6
Store 201	11.54	3

After the data has been imputed, the next step is to perform an explanatory data analysis. In table 4 a descriptive summary shown regarding all the explanatory features. The *population* feature contains a minimum value of only 5 residents. This indicates the presence of a sparsely populated neighbourhood. A neighborhood with such a low population is unlikely to generate substantial

supermarket revenue. Examining the binned variant of the population distribution in figure 2, we observe that the majority falls within the range of 0 to 5000. However, within the neighborhoods with the lowest populations, there are two instances with populations of 5 and 10. Specifically, one of these neighborhoods corresponds to *Store 174* as listed in table 3, which also happens to have the most missing values. The other instance corresponds to *Store 207*. Given their extremely low populations and the presence of missing values, these two stores are unlikely to contribute meaningfully to the prediction models. Therefore, these two instances have been removed from the dataset, leaving us with 301 supermarkets.

Table 4. Descriptive statistics

	mean	**std**	**min**	**max**	**description**
avg_income	28.45	6.25	16.1	61.4	Avg. income per person (x €1 000 euro)
d_mainstreet	1.65	0.87	0.3	5.6	Avg. distance to the nearest mainstreet (km)
d_school	0.53	0.26	0.1	2.4	Avg. distance to the nearest school or university (km)
d_supermarket	0.48	0.22	0.1	2.2	Avg. distance to the nearest supermarket (km)
d_transfer	9.98	8.37	0.3	52.5	Avg. distance to the nearest transfer station (km)
household_size	1.96	0.34	1.3	3.1	Avg. household size
households_with_children	454.49	497.01	0.0	4560.0	Number of households that have children
GLA	1484.34	645.22	150.0	4533.0	Gross Leasable Area (€ / m2)
k0_k15	473.48	532.34	0.0	4995.0	Residents within the range of 0 to 15 years
k15_k25	419.98	444.73	0.0	3135.0	Residents within the range of 15 to 25 years
k25_k45	924.16	956.45	0.0	8005.0	Residents within the range of 25 to 45 years
k45_k65	861.12	818.80	0.0	6795.0	Residents within the range of 45 to 65 years
k65+	732.31	741.76	0.0	7870.0	Residents within the range of 65+ years
k20_high_income	18.19	8.90	2.8	67.2	% of residents in the national top 20% earners
k40_low_income	41.21	7.84	9.9	69.3	% of residents in the national bot 40% earners
n_business	456.60	431.14	30.0	2550.0	Number of business establishments
n_earners	2749.32	2559.09	0.0	22000.0	Number of residents that earn taxable income
n_owner_occupied	49.25	19.69	0.0	96.0	Number of owner occupied houses
n_res_benefits	958.09	998.47	0.0	9670.0	Residents that have benefits (AO, AOW, Bijstand, WW)
population	3410.91	3239.92	5.0	28890.0	Population of the neighbourhood
rental_price	164.07	36.66	85.0	317.0	The yearly rental price (€ / m2)
single_households	825.25	867.32	0.0	6405.0	Number of single-person homes
supermarket_1km	3.43	2.44	0.0	16.3	Avg. supermarkets within a radius of 1km
urbanity_score	2.23	1.06	1.0	5.0	The urbanity score of the neighbourhood 1 - Very Urban (>= 2 500 adresses/km2) 2 - Slightly Urban (1 500 - 2 500 adresses/km2) 3 - Urban (1 000 - 1 500 adresses/km2) 4 - Not really Urban (500 - 1 000 adresses/km2) 5 - Not Urban (<500 adresses/km2)
WOZ	292.77	107.22	121.0	883.0	Avg. house WOZ-value (x €1 000 euro)
%_single_family_homes	49.09	30.45	0.0	96.0	% of single-family homes

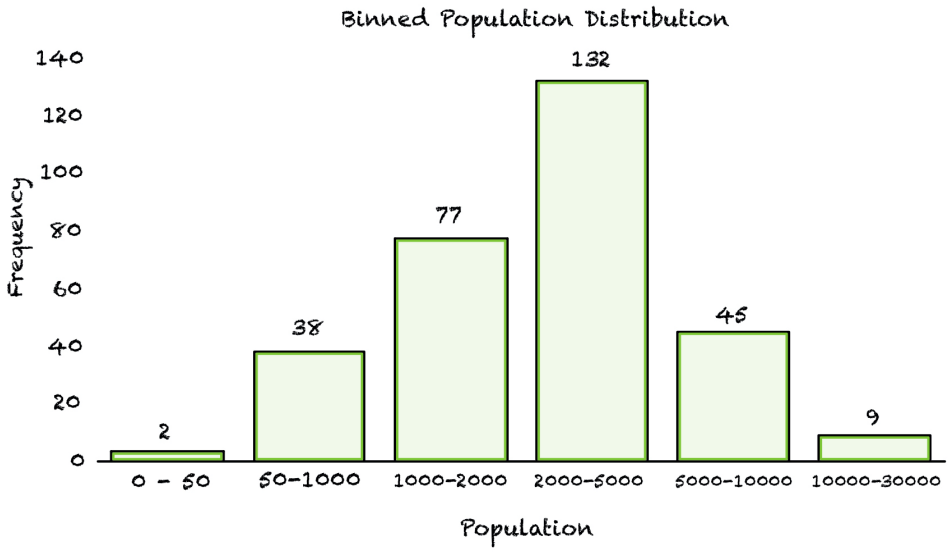


Fig. 2. Binned population distribution reveals the presence of two sparsely populated neighborhoods.

To dive further into the underlying relations between the explanatory features a correlation matrix is created. While the primary intention is not to remove any features at this stage, since our features are specifically hand-picked, conducting a correlation analysis is a valuable endeavor. It allows us to gain insights into how different features are interrelated with each other. In Figure 31 in the Appendix, some high correlation can be found in the correlation matrix of the explanatory variables. High correlations between variables can indicate strong relationships, which may seem redundant for predictive modeling. However, it is important to consider that correlation does not imply causation, and there could be nuances in the data that make including these variables valuable for predicting supermarket rental prices, despite their high correlations. If we look at the correlation between *population* and $k0\_k15$ for example. Even though these variables are highly correlated, keeping them all can be important. Residents that fall in a specific age group are intuitively related to the total population, but a supermarket's performance on different types of customers. Therefore, it could depend on the distribution of age groups in a neighborhood as well as the neighbourhoods total population. For example, areas with more young families (0-15 years) might have a higher demand than single elderly people have. Another correlation pair is *population* with *n_earners*. The *n_earners* feature represents the number of people with a taxable income which reflects the economic capacity of a neighborhood. There are neighbourhoods that consist for the majority out of students, retirees, and individuals who are not employed. As a result, there is a relatively lower number of people who have a taxable income. Even though there might be a high population, the lower number of individuals with taxable

income can impact supermarket revenue potential. Nonetheless, this does raise doubts about the significance of the *population* feature. However, in this context, it plays an important supportive role for the *n_earners* feature. A neighborhood with a high absolute number of *n_earners* may still be less appealing than one with a lower absolute number. This is due to the relative proportion to a neighborhood's total *population*. Therefore, justifying the inclusion of the *population* feature.

The last pair we look at is *avg_income* and *WOZ*. These variables may exhibit a correlation because neighborhoods with higher property values (WOZ value) often have residents with higher incomes. However, it is essential to retain both because they reflect different economic aspects. Average income is a measure of individual or household earnings, while WOZ value pertains to property values. There are often residents that earn good incomes, but choose to live in properties with lower assessed values. This can be due to factors such as housing affordability, personal preferences, or the availability of different types of housing. Therefore, supermarket rental prices can be influenced by both income levels and property values in an area.

Previous research has highlighted the importance of the following features: *population*, *supermarkets_in_1km*, *avg_income* and *WOZ* [7, 8, 9, 10]. Therefore, scatterplots have been created for these features. These scatterplots serve as visual aid to illustrate the connections between our explanatory features and the target variable. At a first glance, when looking at the scatterplots in figure 3, they show no clear-cut trends. They show that the data points are relatively dispersed for all 4 scatterplots. However, among 3 out of the four scatterplots: *supermarkets_in_1km*, *avg_income*, and *WOZ*, display a slight upward trend. This indicates a potential existence of a positive linear relationship. Nonetheless, it is important to state that the dataset's size plays a crucial role here. With only 301 instances, the ability to make definitive statements about these relationships is limited. Therefore, these trends are approached with caution, recognizing the need for further analysis to substantiate these potential associations.

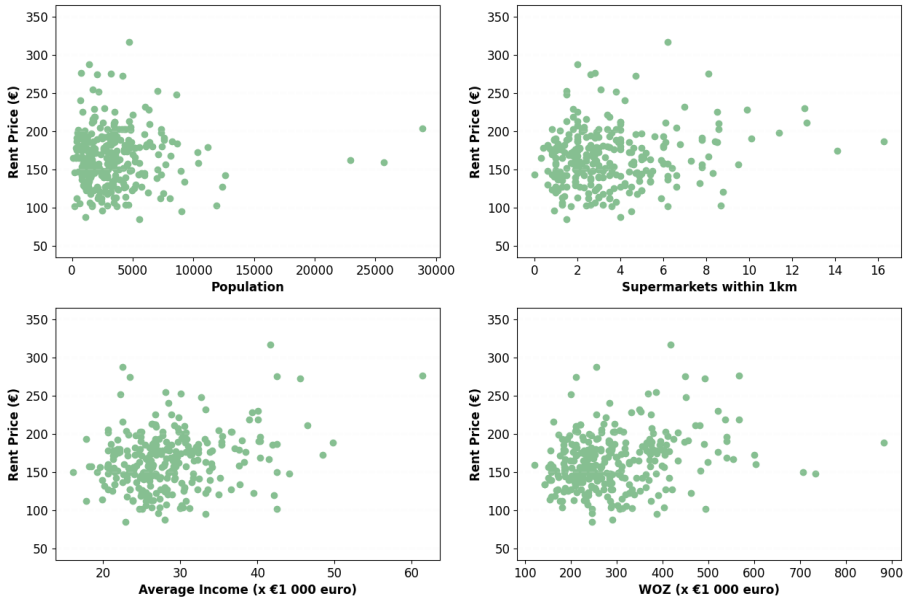


Fig. 3. Scatterplots of important features indicate slight positive trends.

3.2 Decision trees & random forest

The first model is a random forest model, which is a collection of decision trees. A decision tree is a straightforward yet robust machine learning algorithm utilized for both classification and regression tasks. However, before delving into how random forests work, we delve into the mathematics behind a single tree. Given that this research revolves around a regression problem, our focus is on a regression tree. The first regression tree algorithm introduced in the literature was automatic interaction detection (AID), which functions by recursively dividing the data into subsets based on the features [37]. The goal is to create subsets that are as pure as possible in terms of the target variable. An example of the architecture of a regression tree can be seen in figure 4.
However, it is important to understand why a decision tree decides to split features and how these subsets are formed. Building a regression tree consists of the following 2 steps:

1. Dividing the predictor space, which includes all the possible values of the explanatory features $X_1, X_2, ..., X_p$ into J separate and non-overlapping regions known as, $R_1, R_2, ..., R_J$
2. For every supermarket that falls into the region R_j, we make the same prediction. This prediction is equal to the mean of all target values, in this research the rental price, from all supermarkets that fall into the region R_j

Dividing the predictor space into boxes simplifies the model. Each box corresponds to a specific set of conditions on the features, and within each box, the

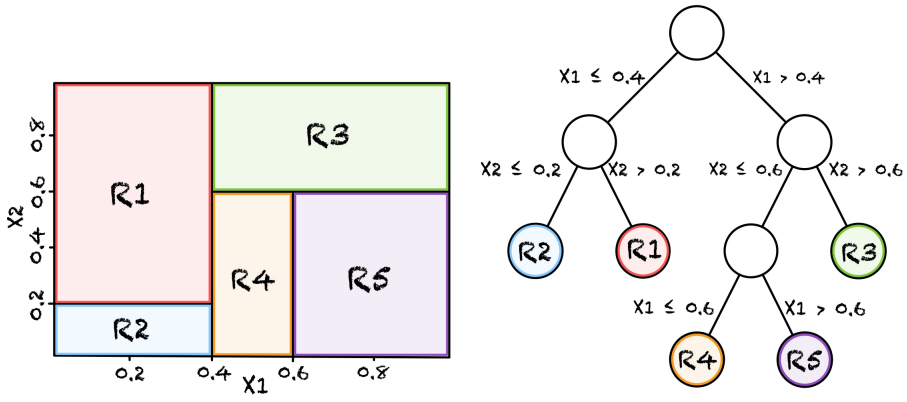


Fig. 4. A visual representation of a regression tree and the corresponding terminal regions that represent the tree's leafs.

prediction is straightforward. Additionally, it is easier to interpret the model when it is divided into boxes. You can see which combinations of feature values lead to higher or lower predictions, which can be valuable for understanding the factors influencing the target value. The goal is to find boxes $R_1, R_2, ..., R_J$ that minimize the RSS, given by equation 2:

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_i)^2 \tag{2}$$

RSS, which stands for the Sum of Squared Residuals, serves as a metric to measure how effectively a model fits the data. When constructing a decision tree to learn the relationship between features and rental prices, the primary objective is to minimize the difference between the real values y_i and the predicted values $\hat{y}_i$. In equation 2 $\hat{y}_i$ stands for the average response value for all individual training instances that are in the jth box. Now to build a regression tree, an initial predictor X_j and a specific cutoff point s is selected. This selection is made so that it maximizes the reduction in RSS by splitting the predictor space into two regions $X|X_j < s$ and $X|X_j \geq s$. This step is part of an iterative process that continues to search for the best predictor and corresponding cutpoint that further minimize the RSS. This process continues until a stopping criterion is reached. The iteration carries on, until a certain condition is met that terminates the proces. For example, when there are fewer than five observations within any given region.

3.3 Baggers can not be Boosters

Now that we have a grasp of how an individual decision tree operates, let's delve into the workings of the random forest algorithm. The way that a random forest

interacts with a decision tree in machine learning, is based on the ensemble learning technique. This is a technique that involves combining multiple models to make predictions. Instead of relying on a single model, ensemble learning utilizes a collection of models. Ensemble learning uses two types of methods:

1. **Bagging** 5 – This technique involves creating various training subsets by sampling from the training data with replacement. The final output is determined through majority voting.
2. **Boosting** 6 – This method assembles weak learners into strong learners by constructing sequential models in such a way that the final model achieves the highest possible accuracy.

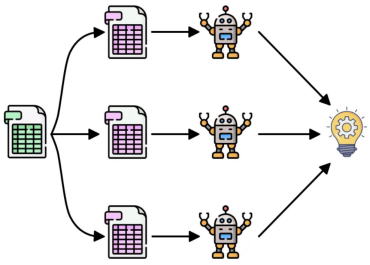


Fig. 5. Bagging

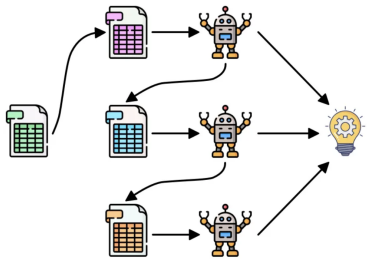


Fig. 6. Boosting

Bagging, the ensemble technique used by random forest, is also known as bootstrap aggregation. Bagging involves selecting random subsets, or samples, from the entire dataset. Each model is created from these samples, using a process known as bootstrap sampling, where data is randomly selected with replacement. This method allows for the creation of independent models, each generating its own results. The final output is determined through majority voting after merging the results from all the models. This aggregation step, where results are combined and the output is determined by majority vote, characterizes the essence of the bagging technique. In summary, the random forest algorithm constructs a group of decision trees with controlled randomness and combines their predictions, ensuring accurate and reliable results through a majority voting system.

3.4 The math behind boosting

On the other hand, boosting is a technique that employs the concept of additive modeling. The concept is straightforward: construct a better performing model by aggregating simpler components. In gradient boosting, multiple simpler models are combined to create a more robust final model. Specifically, gradient boosting learns by forming a weighted sum of a number of base learners, as will be explained in more detail.

Step 1 The gradient boosting process begins by initializing a model $F_0(x)$, which function as the base model, to predict the target y-values. This is done by initializing the model with a constant value γ. The goal is now to find the optimal value of γ, specified by $\gamma_{optimal}$, so that it minimizes the loss between the real y-values and the predicted $\hat{y}$-values. In this initial base model $\hat{y}$ is γ for all observations. Now, $\gamma_{optimal}$ is determined by solving the optimization problem show in equation 3.

$$F_0(x) = \gamma_{optimal} = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma) \tag{3}$$

If the mean squared error (MSE) is used as the loss function, a useful property emerges. When trying to minimize the MSE by calculating its derivative with respect to γ and setting it equal to zero, the optimal value, denoted as $\gamma_{optimal}$, always ends up being the average of the actual target values.

Step 2 Right now our full model $F_M(x) = F_0(x)$. However, it will now be iteratively updated until all the M base learners are added to it with suitable weights. Hence, this step is to be followed for each base learner from $m = 1$ to $m = M$. We now use $F_M(x) = F_0(x)$ where $F_0(x)$ uses $\gamma_{optimal}$, the sample mean, to make predictions. Next, we proceed to compute the pseudo residuals which can be calculated according to equation 4:

$$r_{im} = -\frac{\partial L(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)} \tag{4}$$

Here, m is the m^{th} decision tree and $F_{m-1}(x_i)$ are the predictions made by previous tree $F_0(x)$. So, now we are again taking a derivative of the loss function w.r.t. to the predictions made by previous tree. In the particular case where the loss function is the MSE, we end up with the difference between the actual and predicted values, as shown in equation 5:

$$r_{im} = (y_i - \hat{y}_i) \tag{5}$$

Step 3 Now we fit a new base learner $f_m(x)$ to the pseudo residuals. This means that we build a model $f_m(x)$ on these pseudo residuals and make new predictions. This step is essential because our objective is to reduce these residuals. Through the minimization of the residuals, we enhance the overall accuracy and predictive capability of the model. It is important to note that in this context, the predictions represent the error values rather than the predicted values.

Step 4 Just like in equation 3, the goal is to find the value of $\gamma_{optimal}$ that minimizes the loss function. However, there is one difference in the loss function. Now its includes the previous predictions in equation 6, against when earlier there was no previous prediction.

$$\gamma_{optimal} = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma f_m(x_i)) \tag{6}$$

At this point the value of $\gamma_{optimal}$ is obtained for every leaf in the newly created tree. The model $F_m(x)$, thus is obtained as shown in equation 7.

$$F_m(x) = F_{m-1}(x) + \nu_m f_m(x) \tag{7}$$

Here $F_{m-1}(x)$, the previous prediction, is added to the newly created tree h_m multiplied by the learning rate ν. It reduces the effect each tree has on the final prediction.

3.5 Pre-Extreme Intuition

The regular gradient boosting algorithm tries to find the best model $h_m(x)$ that minimizes the loss function in equation 7. But to do so, a simple regression model is fitted to the residuals with the MSE as objective function. This causes an indirect minimization, where the residuals and the MSE are used as a proxy to do so. Therefore, the algorithm may not find the perfect or most ideal model that perfectly minimizes the loss function. However, the method is effective and functional. This highlights the trade-off between perfection and practicality. Achieving a reasonably good result is often more valuable than pursuing a perfect solution. This is very similar to traditional gradient descent, where the goal is to minimize a loss function $f(x)$ by following the gradient of the function $\nabla f(x)$ at every step shown in equation 8.

$$x^{i+1} = x^i - \nabla f(x^i) \tag{8}$$

While it does not achieve the optimal minimum in a single step, each iteration does bring you a step closer to that minimum. However, this is under the assumption that the used loss function is convex. Despite being an approximation, this method tends to work effectively and is the conventional method to perform logistic regressions. To summarize, it is not the case that for every split created, gradient boosting computes the loss function. It uses the particular loss function of the weak learner to fit the residuals.

3.6 XGBoost

This is where XGBoost introduces a more detailed objective loss function that includes a regularization term:

$$\mathcal{L}_m = \sum_{i=1}^{n} l(y_i, F_{m-1}(x_i) + f_m(x_i)) + \Omega(f_m) \tag{9}$$

Where the regularization term is defined as:

$$\Omega(f_m) = \gamma T + \frac{1}{2}\lambda f_m(x_i)^2 + \alpha|f_m(x_i)| \tag{10}$$

As we can see in equation 9, in regularization, information is added to the objective function. Regularization is used to add some bias into the model to prevent it overfitting to the training data. Here,γT represents the regularization parameter for the minimum loss reduction required to make a further split on a leaf node of the tree. $\lambda f_m(x_i)^2$ stands for the L2-regularization term. It tries to reduce overfitting by forcing the influence of feature to be small, but never zero. This means the less significant features would still have some influence over the final prediction, however small it may. On the other hand $\alpha|f_m(x_i)|$ stand for the L1-regularization term. L1-regularization can reduce the influence of features towards 0, hence making some features obsolete.

A useful comparison to gradient descent, in trying to explain and understand XGBoost, is *Newton's Methods* [38]. *Newton's Methods* calculates the second order derivative in combination with the gradient. In the specific case of gradient descent, at each iteration, we adjust the point x^i as shown in equation 8. And because the gradient $\nabla f(x^i)$ represents the direction of steepest ascent for the function f, the negative counterpart should represent the direction of the steepest descent. In *Newton's Methods*, x^i is updated according to equation 11:

$$x^{i+1} = x^i - \frac{\nabla f(x^i)}{\text{Hess} f(x^i)} \tag{11}$$

Here, the second order derivative is represented by $\text{Hess} f(x)$. Considering the second-order derivative, the direction now moves away from being the one of steepest decrease. However, it now shows a more precise direction towards the x^{i+1} such that $f(x^{i+1}) = 0$. Knowing this, XGBoost's loss function can now be explained further using *Taylor's expansion* [39]. Using the *Taylor's expansion* to optimize the objective in equation (9), we get equation 12.

$$\mathcal{L}_m \approx \sum_{i=1}^{n}[l(y_i, F_{m-1}(x_i)) + g_i f_m(x_i) + \frac{1}{2}h_i f_i^2(x_i)] + \Omega(f_m) \tag{12}$$

Where $g_i = \partial_{F_{m-1}(x_i)} l(y_i, F_{m-1}(x_i))$ and $h_i = \partial^2_{F_{m-1}(x_i)} l(y_i, F_{m-1}(x_i))$ are the first and second order derivatives of the given loss function. By eliminating the constants from the equation we get approximation 13:

$$\mathcal{L}_m \approx \sum_{i=1}^{n}[g_i f_m(x_i) + \frac{1}{2}h_i f_i^2(x_i)] + \Omega(f_m) \tag{13}$$

This tailored approximation is effective when trying to find the values of f that are close to x, which represents the core objective of boosting.

3.7 LightGBM

For this research a gradient boosted Decision is implemented using the lightGBM framework.

Figure 7 shows that lightGBM uses a leaf-wise strategy for tree splitting. This distinguishing lightGBM from other boosting algorithms that utilize a level-wise approach. During the tree building process, both the leaf-wise and depth-level-wise approaches ultimately create identical trees. However, the key difference here lies in the order in which the tree is build. Trees are often restricted from reaching their maximum depth. This is because, to prevent the model from memorizing the entire train set, and not being able to handle unseen data, which is known as overfitting. Therefore making the order of the tree building process an important aspect. The application of early stopping can therefore result in varying tree structures. Leaf-wise splitting in lightGBM choose the splits based on their impact on the overall loss, as opposed to solely focusing on branch-specific loss. This results in a leaf-wise based model, finding a lower-error tree faster than a level-wise based model. Moreover, lightGBM significantly accelerates the training of traditional gradient-boosted trees. LightGBM achieves nearly identical accuracy levels, and manages to achieve a speed improvement of over 20 times the normal computation time, as reported in the original lightGBM paper [40]. In this original lightGBM paper, two new novel techniques called gradient based one-side sampling (GOSS) and exclusive feature bundling are proposed.

The pseudo-residuals or gradients in Equation 4, can be used for effective data sampling. Whenever a data point has a small gradient, it indicates that the model as already trained well on this data point. This gives a possibility to discarding instances with small gradients. However, this would disrupt the data distribution and model accuracy. To adress this issue, GOSS was proposed. Initially, GOSS sorts the data instances based on the absolute values of their gradients. The top $a \cdot *100\%$ instances are selected and it randomly samples $b \cdot 100\%$ instances from the remaining instances. After that, GOSS amplifies the samples data by a constant $\frac{1-a}{b}$. This strategy allows GOSS to put more focus on the under-trained instances without significantly altering the data distribution [40].

LightGBM identifies sets of features that are mutually exclusive. In other words, these feature sets rarely have nonzero values together. LightGBM bundles these exclusive features together into what it calls exclusive feature bundles. Instead of considering each exclusive feature individually during the training process, lightGBM combines them into a single feature bundle. This cause the feature feature dimensionality of the dataset to decrease. In this way, the complexity changes from $O(\#\text{data} \cdot \#\text{feature})$ to $O(\#\text{data} \cdot \#\text{bundle})$, while $\#\text{bundle} < \#\text{feature}$. Which significantly speeds up the training of a gradient boosted decision tree without decreasing the accuracy.

3.8 TNT - Training & Tuning

Before proceeding with model training, splitting the data according to a train/test split is an essential step during training preparation. The test set serves as an, for the model, unseen dataset. It plays a vital role in assessing how well the model can generalize their learned pattern to new, unseen data. Therefore, using an unseen test set, can reliably estimate a models' performance on real-world data.

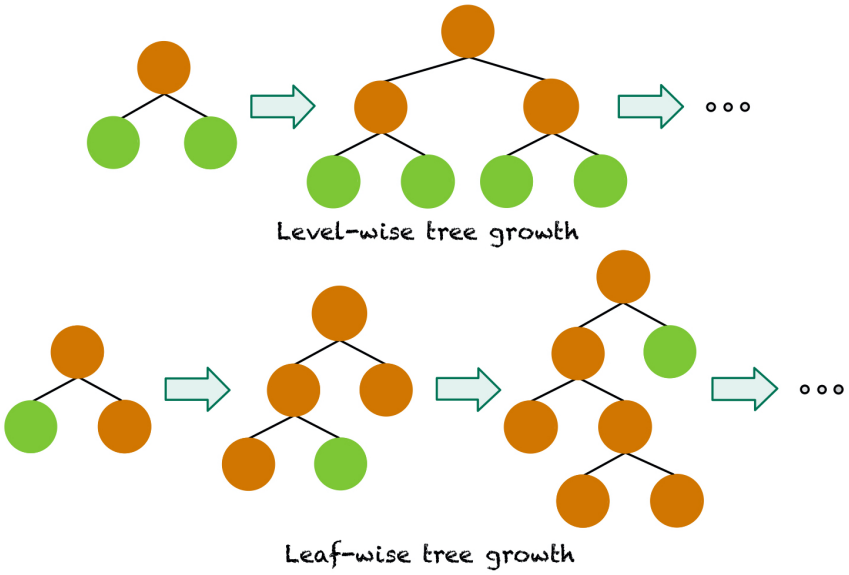


Fig. 7. Level-wise tree growth used in XGBoost & leaf-wise tree growth used in light-GBM

This ensures that the model is not entirely memorizing the training data. Given the relatively small size of the dataset, which consists of only 301 supermarkets, previous research suggest to divide the dataset in such a way that it optimally utilizes the limited number of available instances [41]. Therefore, a 90/10 split is used to use most of the data to train the model with and leave sufficient instances left to test on. This means that we will have 270 training instances and 31 test instance for our experiment.

Before training, the model specific hyperparameters must be specified. Hyperparameters are tweakable configurations that can enhance model performance. All models have default values set for these hyperparameters. However, they are not guarenteed to be optimal, as there is no such one size fits all situation when it comes to parameter tuning. To enhance the model's robustness, a technique known as cross-validation (CV) is used. An example of this is the method called k-Fold CV. It divides the data into training and testing sets and with K-Fold CV, it takes the training set and divides it into K subsets, referred to as folds.

Subsequently, K iterations of model training are done. Each time it uses K-1 of the folds for training and the remaining fold, known as the validation data, for evaluation.This process is shown in Figure 6 for a clearer understanding.

Fig. 8. A visual representation of K-fold cross-validation for K=5

With *Scikit-Learn's RandomizedSearchCV* method, you can define a range of hyperparameter values within a grid. *RandomizedSearchCV* then conducts K-Fold cross-validation by randomly sampling from this grid, evaluating the model's performance for each combination of hyperparameter values.

Table 5. Random forest / decsion tree - hyperparameters

Hyperparameter	Default Value
n_estimators	10
max_features	auto
*max_depth**	None
*min_samples_split**	2
*min_samples_leaf**	1
bootstrap	True

Table 6. XGBoost / lightGBM - hyperparameters

Hyperparameter	Default Value
n_estimators	100
learning_rate	0.3
max_depth	6
min_child_weight	1
gamma	0
colsample_bytree	1
reg_alpha	0
reg_lambda	0

In Table 5 the hyperparameters of a decision tree[4] and random forest are shown with their default values. The *n_estimators* parameter specifies the number of trees in the forest. More trees reduce overfitting and provide a more robust prediction. The maximum number of features each tree considers when splitting a node is specified through *max_features*. A lower value reduces the randomness in each tree and makes them more diverse. Higher values make the trees more

[4] The hyperparameters marked by * are used for the single decision tree algorithm

similar. The minimum threshold for the number of samples needed to initiate a node split is specified *min_samples_split*. This parameter may reduce overfitting by setting its value higher. However, it also risks underfitting the data where it does not learn the patterns between the features and the target variable. The parameter *min_samples_leaf* specifies the minimum samples needed for a leaf node to form. Higher samples can make the tree more robust, but also risks underfitting, similar to *min_samples_split* does. Finally, the *bootstrap* parameter determines if during tree building, random sampling with replacement is used. Setting this value to true, introduces randomness, which reduces overfitting.

For the XGBoost model and the lightGBM model, the hyperparameters and their initial value are shown in table 6. The *learning_rate* parameter controls the step size at each iteration whole moving toward a minimum of the loss function. Specifically it reduces the effect each tree has on the final prediction. Therefore a smaller learning rate prevents the model from overshooting towards the minimum but makes the training process slower, which is not suitable with a small dataset. The *min_child_weight* hyperparameter specifies the minimum sum of instance weights required for a child node to be considered during the splitting process. If the sum of instance weights for a potential child node is below the *min_child_weight* threshold, XGBoost will stop further splitting, and the node will become a leaf node. Increasing this value makes the model more conservative. Hence, it prevents the model from overfitting. The parameter *Gamma* ensures that node splitting does not occur when the gain from the parent node to the children node is minimal. This done through a specification of the minimal loss that is required for a further split to happen. *Gamma* is a regularization term that, with higher values, prevents overfitting. The hyperparameter *colsample_by_tree* defines the subset of features that will be randomly selected for constructing each individual tree. More features make trees more similar, risking overfitting. *reg_alpha* and *reg_lambda* are two regularization terms that add L1 and L2 regularization to the objective function respectively.

3.9 Evaluation Metrics

Mean Absolute Error (MAE) The formula for the mean absolute error (MAE) is shown in equation 14. It provides an average magnitude of errors in target value predictions. It is easy to interpret in the context of retail rents and can help to understand how much the predictions deviate from the actual rents on average. However, MAE doesn't take into account the relative magnitude of errors and therefore does not give more weight to larger errors. However, for retail rents, understanding the average absolute error could still be informative.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{14}$$

Root Mean Squared Error (RMSE) Compared to the MAE, the root mean squared error (RMSE), does penalize larger errors. The larger errors receive more

weight compared to smaller errors. This is done due to the squaring operation in its calculation which is shown in equation 15. However, this also means that the RMSE is sensitive to outliers. Therefore, we need to be mindful of extreme rent values that could disproportionately influence the RMSE.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{15}$$

Mean Absolute Percentage Error (MAPE) MAPE would gives the average percentage difference between the predictions and actual rents, which might be meaningful for stakeholders who are interested in relative accuracy. The formula is shown in equation 16

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100 \tag{16}$$

R-squared (R^2) The R^2 metric can provide insights into how well the geodemographic features explain the variance in retail rents. If the geodemographic features are meaningful predictors of rents, a higher R2 value would indicate a better fit. While R^2 is a valuable metric for assessing the goodness of fit, it does not provide information about the model's accuracy in predicting individual data points. The formula is shown in equation 17

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{17}$$

3.10 Explainable AI with SHAP

In the research of Lundberg and Lee, they explain that SHAP uses shapley values, which are a concept from game theory, to compute the contribution of each input feature to the prediction [12]. By comparing the model with and without feature X_1, SHAP values determine the importance of feature X_1. Hence, for observation i corresponding to feature X_1, a SHAP value of 10% indicates that adding feature X_1 to the model improves the model's prediction by 10%. As the impact of excluding a particular feature can be influenced by the presence or absence of other features in the model, the differences are calculated for all subsets denoted as $S \subseteq F$ where F stands for all the features. Subsequently, the Shapley values are calculated and used as feature attributions. These shapley values represent a weighted average of all the differences, as shown in equation 18.

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} * [f_S \cup \{i\}(x_S \cup \{i\} - f_S(x_S)] \tag{18}$$

SHAP values are unique as they have three desired attributes: local accuracy, missingness and consistency.

Local Accuracy When trying to approximate model f, the input is simplified to x' for a specific input x. To ensure local accuracy holds, the model should produce an output that is consistent with the output of the original model f for the input x. A simplified input x' mostly represents a feature vector turning into a binary vector, where features are either included or excluded. Local accuracy is shown in equation (19)

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i \tag{19}$$

Missingness Missingness states that if a feature is excluded from the model, its attribution must be zero. Meaning that the only thing that can effect the output of the explanation model is the inclusion of feature, not the exclusion.

$$x'_i \implies \phi_i = 0 \tag{20}$$

Consistency Consistency means that if the original model changes so that a particulars features contribution changes, the attribution and the explanatory model can not change in the opposite direction. So for example if we have a new model where a specific feature has a more positive contribution than the original, the attribution in our new explanatory model can not decrease. Now let $f_x(z') = f(h_x(z'))$ and $\frac{z'}{i}$ denote setting $z'_i = 0$. For any two models f and f', if:

$$f'_x(z') - f'_x(\frac{z'}{i}) \geq f_x(z') - f_x(\frac{z'}{i}) \tag{21}$$

In summary, SHAP values assign a value to each feature, which indicates the change in the expected prediction when that feature is included in the model. The SHAP values provide an explanation of how the base value $E[f(z)]$, which is the prediction without any feature knowledge, changes to the actual predicted output $f(x)$ that is based on different feature attributions. Figure 9 shows an abstract example for a single prediction.

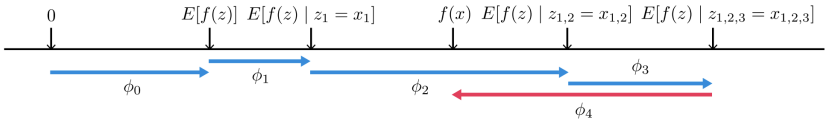


Fig. 9. A SHAP forceplot, showing the effects of different explanatory features on a single prediction.

4 Results

In table 7 we can see the best results for all of the models after 10-fold cross validation. The table shows that the decison tree model shows relatively high values for the RMSE and MAE, alongside a negative R^2 value. The negative R^2 value indicates that the model fails to capture the variance in the data effectively. The predictions based on the decision tree do not explain the variability in the target variable very well using the explanatory geo-demographic variables. However, the 1-MAPE score of 0.84 suggests that the model has a good overall accuracy in predicting rental prices. Random forest, while achieving a relatively low RMSE and MAE, also has a negative R^2 value. Similar to the decision tree, random forest also attains an 1-MAPE score of 0.84. XGBoost, like the decision tree and random forest, encountered challenges in explaining the variance in the data with a negative R^2 value. Compared to the random forest model, XGBoost also has a slightly higher RMSE and MAE. However, based on the 1-MAPE score of 0.83, XGBoost provides accurate predictions, although not as good as the decision tree and random forest. Furthermore, lightGBM offers a competitive performance with an RMSE and MAE close to that of the random forest model. However, similar to the other 3 models, it has a negative R^2 value.The 1-MAPE score of 0.84 shows that lightGBM also delivers accurate predictions.

Table 7. Performance measures along for the best performing models after 10 fold cross validation

	RMSE	**R2**	**MAE**	**1-MAPE**
Decision tree	33.92	-0.09	24.43	0.84
Random forest	32.57	-0.01	23.39	0.84
XGBoost	36.20	-0.24	25.87	0.83
LightGBM	33.56	-0.07	23.42	0.84

If we look at table 10, we can dive deeper into the values for the hyperparameters for the 4 best performing variants of each model. Random forest and XGBoost went for a similar approach and created 50 trees. LightGBM on the other hand only created 20 trees. However, lightGBM did have a deeper architecture with a *max_depth* of 12, compared to 10 from both random forest and XGBoost and only 3 from the decision tree. Noticeable is the high L2 *reg_lambda* hyperparameter of 5 for lightGBM compared to XGBoost's 0.5. Meaning a higher punishment against overfitting.

To assess the performance of all 4 models, a scatterplot is generated comparing the predicted rent values against the actual rent values. From figure 11, distinctive patterns can be seen from the outputs generated by the decision tree. These patterns shows that for a broad range of actual rental prices, the decision tree predicts nearly the same value. This behavior is due to the nature of decision trees. As already explained, they tend to create relatively simple, piecewise

Table 8. Hyperparameter values for the best performing models

Hyperparameter	Decision tree	Random forest	LightGBM	XGBoost
n_estimators	-	50	20	50
max_depth	3	10	12	10
min_samples_split	2	5	-	-
min_samples_leaf	1	4	-	-
max_features	-	sqrt	-	-
bootstrap	-	True	-	-
learning_rate	-	-	0.1	0.1
min_child_weight	-	-	3	5
gamma	-	-	-	-
colsample_bytree	-	-	0.4	0.5
reg_alpha	-	-	0.5	0.5
reg_lambda	-	-	5	0.5

constant models. They split the feature space into regions and predict the same value for all instances that fall within the same region 10. Furthermore, in the scatterplots in figure 12 and figure 13 , the majority of data points for random forest and lightGBM are closer to the regression line. This indicates a comparably better performance against the XGBoost model which is shown in figure 14. This is particularly evident in the central cluster of points. However, a noteworthy observation across all four scatterplots is the presence of a few data points that appear as outliers. These outliers exhibit disproportionately large errors compared to the majority of the data. These deviations indicate that there are certain supermarkets for which the models' predictions significantly differ from the actual rent values. We are going to perform a more in-depth analysis to see which feature impacted these outlying predictions.

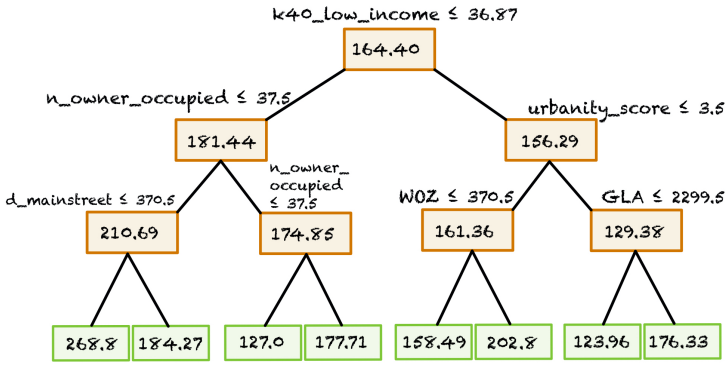


Fig. 10. The final decision tree shows the split criteria and terminal regions.

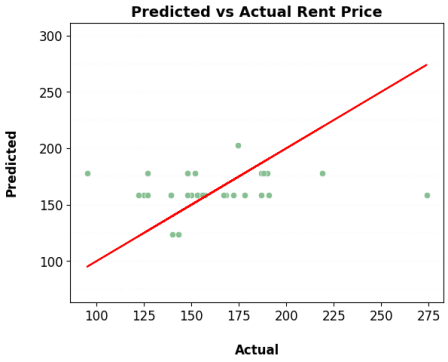


Fig. 11. Decision tree

Fig. 12. Random forest

Fig. 13. LightGBM

Fig. 14. XGBoost

4.1 SHAP Force Plots

In the SHAP force plots, each shapley value is represented as an arrow, exerting a force to either increase or decrease the predicted value. These forces collectively balance each other, ultimately converging at the actual predicted value. Figures 15, 16, 17, and 18 depict the force plots for the initial outlier. This particular outlier, which had an actual rental price of €95/m^2, was predicted to be €177.70, €190.02, €193.52, and €197.68 by the decision tree, random forest, lightGBM, and XGBoost, respectively. These prediction turned out to be excessively high. When looking at the force plots, it becomes evident that the most influential features are *WOZ*, *k20_high_income* and *k40_low_income*. An interesting observation here is that the decision tree uses less features to impact its prediction

value and mostly depends on the *WOZ* feature to generate the predicted target value. For random forest, lightgBM and XGBoost the combined effect of *WOZ* and *k40_low_income*, increases the predicted value by approximately €15. To be more precise, a *WOZ* value of 388 and a value of 34.9 for *k40_low_income* achieve this effect. However, when referring back to table 4, these values are not disproportionally high or low when comparing it to all instances. Therefore, it seems unexpected that they have such a high impact on the predicted value.

Fig. 15. Force plot decision tree - outlier 1

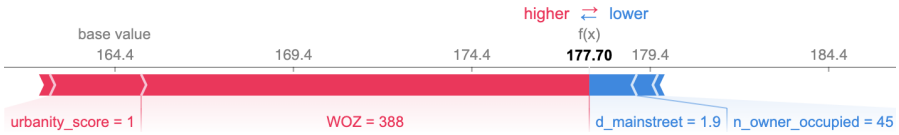


Fig. 16. Force plot random forest - outlier 1

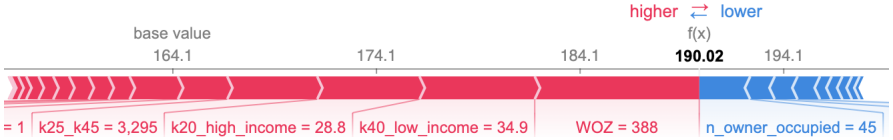


Fig. 17. Force plot lightGBM - outlier 1

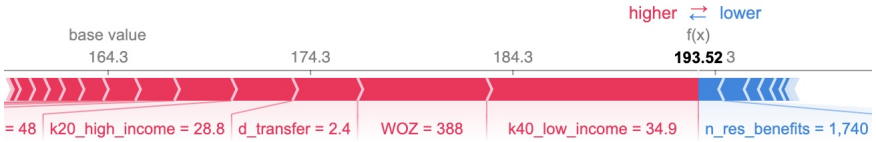
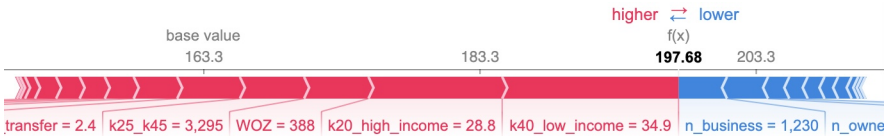


Fig. 18. Force plot XGBoost - outlier 1



Outlier 2 had an actual value of €274/m^2. Predictions by the decision tree, random forest, lightGBM, and XGBoost for this outlier were €158.48, €184.32, €168.89, and €180.85, respectively. In this case, the prediction values were notably lower compared to Outlier 1. The corresponding figures 19,20, 21, and 22 provide feature attributions for this outlier. Again can be seen that the predicted output by the decison tree model is based off less feature when compared to the other models. For both random forest and lightGBM, the lower value of 212 for the *WOZ* feature contributes to a decrease in the prediction value. Similarly, in

the case of lightGBM and XGBoost, the higher value of 50.9 for *k40_low_income* leads to a reduction in the prediction. The *n_owner_occupied* feature consistently increases the predicted value by approximately €10 to €15 across all models except the decision tree. Whereas, the decision tree assigns an high importance to the *k40_low_income* feature, which brings down the prediction by around €8. Furthermore, this particular instance has an *urbanity_score* of 1, which means that we are dealing with an urban neighbourhood. Intuitively, it is to expect that urban areas are better suited for supermarkets, leading to higher rental prices. However, this assumption is contradicted by the force plots as they only attribute an increase of around €1 to €2 across all models.

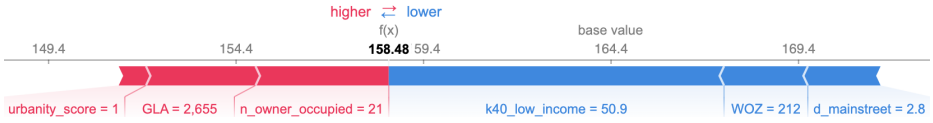


Fig. 19. Force plot decision tree - outlier 2

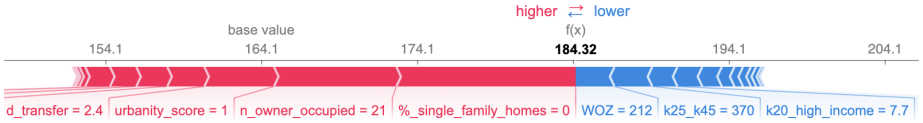


Fig. 20. Force plot random forest - outlier 2

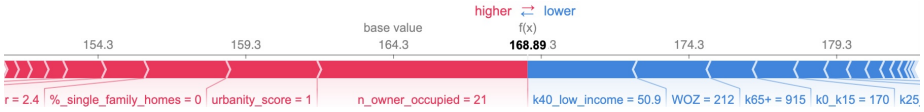


Fig. 21. Force plot lightGBM - outlier 2

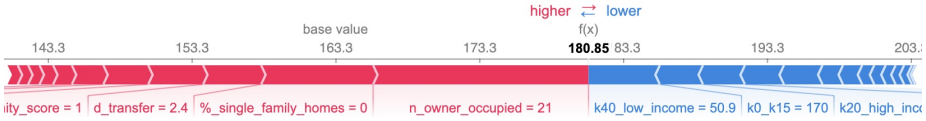


Fig. 22. Force plot XGboost - outlier 2

4.2 SHAP Summary Plot

The summary plot provides a comprehensive view by merging feature importance and feature effects. In this plot, each point represents a shapley value corresponding to a feature and a specific instance. The vertical position on the y-axis corresponds to the feature, while the horizontal position on the x-axis represents the shapley value. Furthermore, the color coding indicates the feature's value, ranging from low to high. To enhance visibility, overlapping points are slightly adjusted along the y-axis, allowing us to perceive the distribution

of shapley values for each feature. The features are thoughtfully arranged in descending order of average contribution to the predicted values, facilitating a clear understanding of their respective contributions.

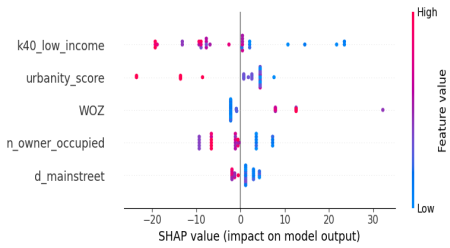


Fig. 23. Shap summary plot - decision tree

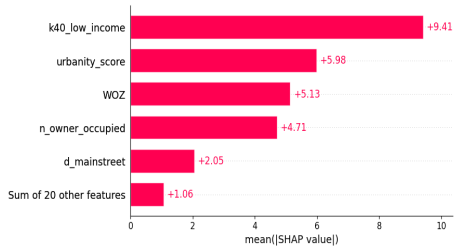


Fig. 24. Avg. feature contribution - decision tree

The summary plots of the decison tree model plots are shown in figure 23 and figure 24. The most important features in the decision tree model are: *k40_low_income*, *urbanity_score*, *WOZ*, *n_owner_occupied* and *d_mainstreet*. The most importance is given to the *k40_low_income* feature, which on average impacts the predicted value by €9.41. The left plot depicts that the higher the value for *k40_low_income*, the lower the predicted target value. This is the same for the other important features shown. However, for *WOZ* lower values mean a lower predicted value by the decision tree model.

Fig. 25. Shap summary plot - random forest

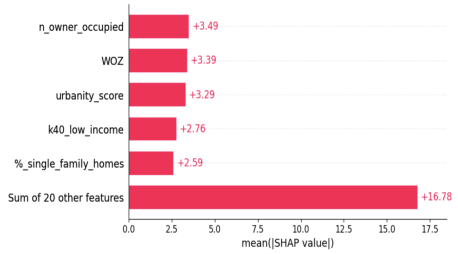


Fig. 26. Avg. feature contribution - random forest

In the summary plots presented in figure 25 and figure 26 for the random forest model, we can see the five most important features. These contributing features are: *n_owner_occupied*, *WOZ*, *urbanity_score*, *k40_low_income* and *%_single_family_homes*.The summary plot effectively illustrates that higher values of *WOZ* correspond to an increase in predicted rent values. In contrast, for the features *%_single_family_homes*, *n_owner_occupied*, *urbanity_score* and

k40_low_income, higher values indicate a decreasing effect on the prediction values.

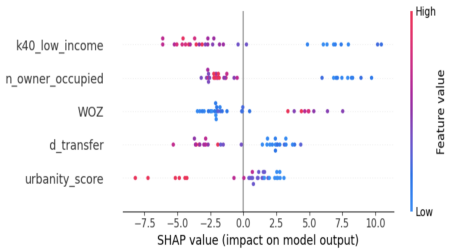


Fig. 27. Shap summary plot - lightGBM

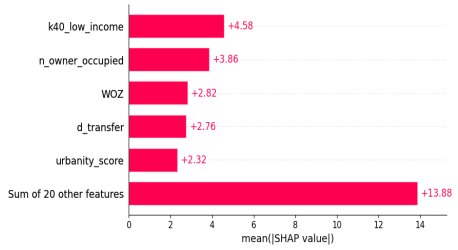


Fig. 28. Avg. feature contribution - lightGBM

In the subsequent summary plots in figure 27 and figure 28 ,regarding the lightGBM model, we can see that the 5 most important features here are are: *k40_low_income*, *n_owner_occupied*, *WOZ*, *d_transfer* and *urbanity_score*. This visualization reveals that higher *WOZ* values correspond to higher rental predictions. Conversely, for the features *k40_low_income*, *n_owner_occupied*, *d_transfer*, and *urbanity_score*, an increase in their values leads to a decrease in the predicted rental value.

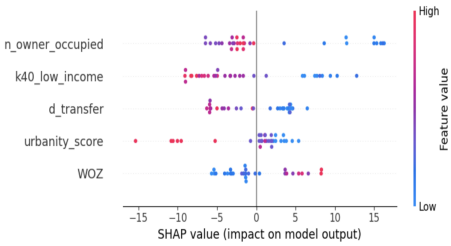


Fig. 29. Shap summary plot - XGBoost

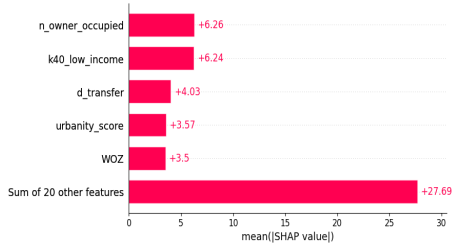


Fig. 30. Avg. feature contribution - XGBoost

In the final summary plots displayed in Figures 29 and 30, focusing on the XGBoost model, we identify the five most influential features: *n_owner_occupied*, *k40_low_income*, *d_transfer*, *urbanity_score*, and *WOZ*. Remarkably, these features align with those identified by lightGBM but with different levels of importance. Once again, we observe that higher *WOZ* values are associated with higher predicted rental prices, while higher values of the other features lead to reduced rental predictions.

5 Conclusion

This research aimed to anticipate supermarket revenue potential. This was done by trying to understand and predict rental prices for supermarkets, a task important to CREITs. Leveraging the power of advanced machine learning models and the insights of XAI, a dataset containing a plethora geo-demographic features was studied. While coming to a conclusion in this research, there are several key takeaways. However, it's essential to clarify that all observed effects reflect the behavior of the models and may not necessarily imply causation in the real world.

Higher values in features such as *WOZ* and *k20_high_income* were found to be associated with higher rental predictions. This aligns with our intuition. Higher property values often indicate a wealthier neighborhood. This can be confirmed by the higher values for the *k20_high_income* feature also effecting the rental predictions upwards. This feature, reflecting the proportion of top 20% national earners in a neighborhood, is indicative of increased spending capacity. Conversely, features like *d_transfer* and *urbanity_score* showed lower values causing higher rental predictions. The *urbanity_score*, which was defined on a scale from 1 to 5 with 1 being the most urban, logically led to increased rental predictions. As urban regions tend to have higher population, leading to more possible costumers. Similarly, a lower value in *d_transfer* implied shorter average distances to key transfer stations. This indicates efficient logistics for getting to supermarkets. Furthermore, the importance assigned to *k40_low_income* by the decison tree model was significantly higher than the other model. The *k40_low_income* feature impacted the predictions by an average of €9.41 as compared to €2.76 for random forest , €4.58 for the lightGBM model and €3.85 for XGBoost. The decision tree's heavy reliance on the *k40_low_income* feature can be explained due to its greedy nature. A single decision tree chooses the best feature at each node to minimize the loss, whereas ensemble models distribute this reliance across the combination of multiple trees. This could potential dilute the importance of *k40_low_income* in the predicted values of the ensemble models. A noteworthy discovery was the inverse relationship observed between higher values of *n_owner_occupied* and lower rental predictions. This suggests that when there are more owner-occupied houses in the neighbourhood, the rental prices tend to decrease. However, this finding contradicts intuition and raises questions about the models' ability to accurately capture the true impact of this feature on the target variable.

Expanding the view to analyze the research results, we can try to suggest neighbourhoods where supermarket are likely to excel. However, this is based off the behavior of the models and does not imply causation in the real world. The combination of important features such as *urbanity_score*, *d_transfer*, and *d_mainstreet* indicate city-like neighborhoods as promising locations for supermarkets. To be more precise, neighborhoods where a supermarket would be conveniently accessible due to a relatively short distance to a transfer station or main street. Additionally, individual features like *n_owner_occupied* and *WOZ* intuitively support this observation, as urban areas typically have a greater num-

ber of houses, including owner-occupied houses, and tend to have a higher average property value. Nonetheless, the significance of the *k40_low_income* feature should not be underestimated. It substantially influences the predicted values in all models. In city-like neighborhoods with a low spending capacity, supermarket performance may face difficulties.

The overall performance measures of the decision tree, random forest, lightGBM, and XGBoost, proved to be a challenging endeavor to interpret. The models performed decent in terms of MAPE, hovering around 83%. Even when considering the MAE, averaging at €23, these results do not indicate a relatively bad performance. However, if we do some calculation we can reveal a negative financial impact for the CREITs. For instance, with an error of €23 per square meter and an average gross leasable area of 1484 square meters, the cumulative rent potentially lost amounts to €34,132. Furthermore, the models exhibited negative R^2 scores, indicating their inability to capture the intricate relationships between explanatory variables. This finding shows the complexity of the task at hand and highlights the room for improvement in enhancing the models' explanatory power.

6 Discussion

This research, while providing valuable insights into the rental price prediction for supermarkets, also highlights the importance of model interpretability and model performance. Moving forward, there are several options for further exploration. The main suggestion for future research would be to acquire more data. With low volumes of data, machine learning models, whichever models they are, have too few training instances. This makes it all the more difficult to learn any intricate relations between explanatory values and target values. When more data is available, also more possibilities of using different machine learning and even deep learning model arise. However, deep learning models tend to be more black box which should be treated with caution. Another idea for future research involves adopting a different strategy for finding the optimal hyperparameters. In this research, a relatively straightforward randomized grid search was used for parameter optimization, leaving room for improvement. More sophisticated techniques such as Bayesian optimization or even exhaustive grid search, given enough computational resources and time, could potentially improve model performance. Furthermore, building upon the findings from this and prior research, a deeper investigation into the relationships among geo-demographic explanatory features could improve model performance. The retail landscape demonstrated to be very susceptibility to a plethora of influencing factors. Exploring specific features that exhibit similar characteristics may create more effective feature selection techniques for model predictions. By finding and leveraging these possibly high-impact features, future research could pave the way for more improved and transparent predictive accuracy in the retail real estate landscape.

References

1. Huisvestigingskosten detailhandel 2015, `https://shorturl.at/gjkBU`. Last accessed 21 Aug 2023
2. Jaarrekening wel of niet deponeren? 2023, `https://shorturl.at/bcgMP`. Last accessed 21 Aug 2023
3. O'Roarty, B., McGreal, S., Adair, A. & Patterson, D. Case-based reasoning and retail rent determination. *Journal Of Property Research*. **14**, 309-328 (1997)
4. Geltner, D., Miller, N., Clayton, D. & Eichholtz, P. Commercial real estate analysis and investments. (South-western Cincinnati, OH,2001)
5. Chu, C. & Zhang, G. A comparative study of linear and nonlinear models for aggregate retail sales forecasting. *International Journal Of Production Economics*. **86**, 217-231 (2003)
6. Penpece, D. & Elma, O. Predicting sales revenue by using artificial neural network in grocery retailing industry: a case study in Turkey. *International Journal Of Trade, Economics And Finance*. **5**, 435 (2014)
7. Brooks, C. & Tsolacos, S. Forecasting models of retail rents. *Environment And Planning A*. **32**, 1825-1839 (2000)
8. Ke, Q. & Wang, W. The factors that determine shopping centre rent in Wuhan, China. *Journal Of Property Investment & Finance*. **34**, 172-185 (2016)
9. Hardin III, W. & Wolverton, M. Micro-market determinants of neighborhood center rental rates. *Journal Of Real Estate Research*. **20**, 299-322 (2000)
10. Ting, C. & Jie, M. Location Profiling for Retail-Site Recommendation Using Machine Learning Approach. *International Conference On Computer, Information Technology And Intelligent Computing (CITIC 2022)*. pp. 48-67 (2022)
11. Doshi-Velez, F. & Kim, B. Towards A Rigorous Science of Interpretable Machine Learning. (2017)
12. Lundberg, S. & Lee, S. A unified approach to interpreting model predictions. *Advances In Neural Information Processing Systems*. **30** (2017)
13. Krause-Traudes, M., Scheider, S., Rüping, S. & Meßner, H. Spatial data mining for retail sales forecasting. *11th AGILE International Conference On Geographic Information Science*. pp. 1-11 (2008)
14. Wakefield, C. & Frasch, M. Demographic and socioeconomic factors predict maternal postpartum rehospitalization: a retrospective nuMoM2b dataset study. *MedRxiv*. pp. 2022-02 (2022)
15. Liu, X. & Ichise, R. Food sales prediction with meteorological data—a case study of a Japanese chain supermarket. *Data Mining And Big Data: Second International Conference, DMBD 2017, Fukuoka, Japan, July 27–August 1, 2017, Proceedings 2*. pp. 93-104 (2017)
16. Mohd, T., Harussani, M., Masrom, S., Johari, N. & Alfat, L. Office Rent Prediction based on the Influenced Features. *Environment-Behaviour Proceedings Journal*. **7**, 61-68 (2022)
17. Neloy, A., Haque, H. & Ul Islam, M. Ensemble learning based rental apartment price prediction model by categorical features factoring. *Proceedings Of The 2019 11th International Conference On Machine Learning And Computing*. pp. 350-356 (2019)
18. Grinsztajn, L., Oyallon, E. & Varoquaux, G. Why do tree-based models still outperform deep learning on typical tabular data?. *Advances In Neural Information Processing Systems*. **35** pp. 507-520 (2022)

19. Ming, Y., Zhang, J., Qi, J., Liao, T., Wang, M. & Zhang, L. Prediction and analysis of chengdu housing rent based on xgboost algorithm. *Proceedings Of The 3rd International Conference On Big Data Technologies*. pp. 1-5 (2020)
20. Liang, W., Luo, S., Zhao, G. & Wu, H. Predicting hard rock pillar stability using GBDT, XGBoost, and LightGBM algorithms. *Mathematics*. **8**, 765 (2020)
21. Ampomah, E., Qin, Z. & Nyame, G. Evaluation of tree-based ensemble machine learning models in predicting stock price direction of movement. *Information*. **11**, 332 (2020)
22. Lipton, Z. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.. *Queue*. **16**, 31-57 (2018)
23. Ribeiro, M., Singh, S. & Guestrin, C. " Why should i trust you?" Explaining the predictions of any classifier. *Proceedings Of The 22nd ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*. pp. 1135-1144 (2016)
24. Arrieta, A., Diaz-Rodriguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R. & Others Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*. **58** pp. 82-115 (2020)
25. Gade, K., Geyik, S., Kenthapadi, K., Mithal, V. & Taly, A. Explainable AI in industry. *Proceedings Of The 25th ACM SIGKDD International Conference On Knowledge Discovery & Data Mining*. pp. 3203-3204 (2019)
26. Scott, A., Clancey, W., Davis, R. & Shortliffe, E. Explanation capabilities of production-based consultation systems. *American Journal Of Computational Linguistics*. pp. 1-50 (1977)
27. Swartout, W. Explaining and justifying expert consulting programs. *Computer-assisted Medical Decision Making*. pp. 254-271 (1985)
28. Ayinde, L., Wibowo, M., Ravuri, B. & Emdad, F. ChatGPT as an important tool in organizational management: A review of the literature. *Business Information Review*. pp. 02663821231187991 (2023)
29. Shapley, L. & Others A value for n-person games. (Princeton University Press Princeton,1953)
30. Rozemberczki, B., Watson, L., Bayer, P., Yang, H., Kiss, O., Nilsson, S. & Sarkar, R. The shapley value in machine learning. *ArXiv Preprint ArXiv:2202.05594*. (2022)
31. Tallón-Ballesteros, A. & Chen, C. Explainable AI: Using Shapley value to explain complex anomaly detection ML-based systems. *Machine Learning And Artificial Intelligence*. **332** pp. 152 (2020)
32. Parsa, A., Movahedi, A., Taghipour, H., Derrible, S. & Mohammadian, A. Toward safer highways, application of XGBoost and SHAP for real-time accident detection and feature analysis. *Accident Analysis & Prevention*. **136** pp. 105405 (2020)
33. CBS Statline , `https://opendata.cbs.nl/statline/portal.html_la=nl&_catalog=CBS`. Last accessed 11 Aug 2023
34. Berrevoets, J., Imrie, F., Kyono, T., Jordon, J. & Schaar, M. To impute or not to impute? missing data in treatment effect estimation. *International Conference On Artificial Intelligence And Statistics*. pp. 3568-3590 (2023)
35. Deo, T. Data imputation and comparison of custom ensemble models with existing libraries like XGBoost, Scikit learn, etc. for Predictive Equipment failure. *ArXiv Preprint ArXiv:2111.10088*. (2021)
36. sklearn impute IterativeImputer, `https://shorturl.at/iuvBK`. Last accessed 11 Aug 2023
37. Morgan, J. & Sonquist, J. Problems in the analysis of survey data, and a proposal. *Journal Of The American Statistical Association*. **58**, 415-434 (1963)

38. Kou, J. The improvements of modified Newton's method. *Applied Mathematics And Computation*. **189**, 602-609 (2007)
39. Nielsen, D. Tree boosting with xgboost-why does xgboost win" every" machine learning competition?. (NTNU,2016)
40. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. & Liu, T. Lightgbm: A highly efficient gradient boosting decision tree. *Advances In Neural Information Processing Systems*. **30** (2017)
41. Uçar, M., Nour, M., Sindi, H., Polat, K. & Others The effect of training and testing process on machine learning in biomedical datasets. *Mathematical Problems In Engineering*. **2020** (2020)

7 Appendix

Fig. 31. Correlation matrix of the geo-demographic explanatory features