Norwegian School of Economics
Bergen, Spring 2021

Interpreting machine learning models: An overview with applications to German real estate data

Andreas Grimen
Huseyn Ibrahimli

Supervisor: Mr. Geir Drage Berentsen

Master thesis, MSc in Economics and Business Administration, Major in Business Analytics

NORWEGIAN SCHOOL OF ECONOMICS

Abstract

Machine learning models have demonstrated huge improvement in examining complex patterns, which allow them to make predictions about the unobserved data. While the accuracy of these models increases over time, so does complexity, which makes them extremely difficult to interpret.

There are many problems where accuracy is the main focus of machine learning applications, but some cases also require model interpretability. This thesis seeks to present and apply some of the most prominent methods in the relatively new field of interpretable machine learning. In our application, we use these methods to interpret a random forest model which is predicting the monthly rent in a dataset about German real estate. Through this interpretation, we discovered that methods such as Permutation Feature Importance, Partial Dependence Plots, and ALE Plots visualize the mechanisms of the random forest in an easily understandable way. We also analyzed individual predictions with the LIME algorithm and Shapley Values and found that they can provide interpretable explanations of how those predictions were produced. However, while experimenting with the LIME model, we have noticed slightly unstable results produced by this algorithm. So, we offer our solution to this problem by using K-Nearest Neighbours as a sampling method for LIME instead of its own random perturbation technique for sampling observations.

In summary, based on our findings, we conclude that the interpretable machine learning methods can provide comprehensible explanations of model mechanisms, but they still have some limitations when it comes to explaining the more complicated processes in the model.

Acknowledgments

NHH's Business Analytics profile has been a great arena to practice our analytical skills. We have learned a lot about Big Data, Data Analytics, Machine Learning and how these powerful methods can be used to gain insight and improve decision-making. This thesis has been a great opportunity to apply these tools in a practical way, and in the process, we have improved our competence in the field of business analytics.

We would like to say a special thank you to our supervisor, Geir Drage Berentsen. His support, guidance, and overall insights in this field have made this an inspiring experience for us. Finally, we would like to thank our family and friends for supporting us during the compilation of this thesis paper.

Table of Contents

List of Figures

1. Introduction

In the age of big data, Machine Learning (ML) has been flourishing. Sophisticated models that are being trained on large datasets are often able to predict unknown variables with impressive accuracy. Therefore, they are being applied to solve problems such as image recognition, fraud detection, content recommendations, and health diagnosis, just to name a few.

The task of improving model accuracy increases model complexity in the form of additional model parameters, which makes it harder for humans to understand how the model transforms input to output. This problem is known as the accuracy-interpretability trade-off. The trade-off can be problematic in cases where both interpretability and high accuracy are necessary.

For instance, in the healthcare sector, a lack of interpretability in ML models can potentially have life-threatening consequences or cause injuries. Examples are when a model recommends the wrong drug for a patient or fails to notice a tumour on a radiological scan (Nicholson Price II, 2019). To prevent such scenarios, there are regulations like The European Union's General Data Protection Regulation (GDPR) that requires organizations that use patient data for ML modelling to provide on-demand explanations of model outcomes (Ahmad, Ecker, Mckelvey and Teredesai, 2018). Similarly, in the banking industry, there are laws governing decisions that are based on ML. In the United States, if an institution is using a model to determine whether to accept credit applicants, it must be able to provide a clear explanation for refusals (Babel, Buehler, Pivonka, Richardson, and Waldron, 2019).

The cases discussed above are just a few of the many scenarios where interpretability is required in ML projects. This thesis will present some methods that can provide interpretability to ML models and demonstrate their usefulness on a dataset about German Real Estate. We start by introducing the concepts and terminology that are necessary to understand the Interpretable machine learning methods and review some of the literature on the topic. Next, Chapter. 2 will provide detailed explanations of our chosen methods. Then we apply these methods to German Real Estate Data in Chapter. 3. Finally, we end the thesis by discussing our findings.

1.1 Terminology

This section will define the Interpretable Machine learning concepts that are central in this thesis.

Black Box Model is referred to a machine learning model for which we can only understand the inputs and outputs but not the internal mechanisms. Examples are random forest, deep neural networks, etc.

Interpretable Machine Learning (IML) are methods and models that make the mechanisms of an ML model comprehensible for humans. Elshawi, Al-Mallah, and Sakr (2019) defined machine learning interpretability "… as the degree to which machine learning user can understand and interpret the prediction made by a machine learning model".

Inherently Interpretable Models are models that can be interpreted by looking at parameters or feature summary statistics. Examples are Linear regression and simple decision trees.

Model-agnostic methods are tools that can be used to understand any ML model, and they are applied after the model has been trained. All methods discussed in chapter 2 and applied in chapter 3 are model agnostic.

Surrogate models are defined as follows: if the outcome of interest is hard to measure in terms of expensiveness or time, as an alternative, a cheap and fast surrogate model of the outcome can be applied instead. The main goal of the surrogate models is to approximate the predictions of the underlying machine learning model as precisely as possible while being interpretable (Molnar, 2021).

Global Surrogate Methods interpret the entire model behaviour. Because of humans limited working memory, we can never fully understand complicated models such as deep neural networks or Random Forests. But global methods can give us insight into which features are most important in the model, the general effect of each of them, and how they interact.

Local Surrogate Methods interpret the predictions of a single observation or a group of similar observations. Models that are very complicated at the global level might be easier to understand at a local level.

1.2 Literature Review

Some of the models that are inherently interpretable such as linear regression have been used by academics since the beginning of the 19th century, but research in the IML field really started growing in popularity around 2015. After this, many model-agnostic methods have been developed, and open-source software implementations such as *iml* and *Dalex* for R and *InterpretML* for Python have been introduced (Bischl, Casalicchio, and Molnar, 2020).

Guestrin, Ribeiro, and Singh (2016) argue that an advantage of using model-agnostic methods instead of just using interpretable models is that it separates the model from the explanation. Separating interpretability from the model allows the model to be as flexible as needed for the problem at hand, enabling the use of any machine learning method such as neural networks or random forests. This also makes it easier to switch between models without having to learn a new form of explanation. Moving from one interpretable model to another, for instance, from linear regression to a simple decision tree, requires changing the explanations. Model-agnostic methods avoid this problem because the way the explanations are presented is the same for all models.

Guestrin et al. (2016) further argue that there are still some challenges for model-agnostic explanations. For instance, getting a comprehensive global understanding may be very challenging if the model is very complex. In some cases, strictly accurate explanations are needed, and using a black-box model may be inadequate or even illegal. Interpretable models might also be preferable when interpretability is more important than accuracy or when black-box models are not more accurate than interpretable ones.

Model-agnostic methods have been applied to solve problems and gain insight into many fields. Arash Khoda Bakshi and Mohammed M. Ahmed (2020) used random forest and IML methods such as Partial Dependence Plots (PDP) and Accumulated local effects (ALE), among others, to

study real-time traffic-related crash contributing factors. By using these tools, they discovered the causal effects of significant predictors of crash risk, demonstrating how the accuracy-interpretability trade-off can be alleviated when modelling for active traffic management.

Eui-Jin Kim, Youngseo Kim, and Dong-Kyu Kim (2020) used a machine learning approach to predict trip purposes of transit passengers by using spatio-temporal features extracted from smart card data and geographic information data. IML methods such as feature importance, feature interactions, and ALE were used to understand the decision-making process of the model. They revealed that temporal features of travel, like the length of the activity, the trip sequence, and the departure time were the most important factors in predicting trip purpose. Spatial features mainly affected prediction through interaction effects with temporal features. The authors suggested that their findings could be used by transit authorities to determine what type of data should be collected by smart card systems.

Ramirez, Villanueva, and Blazquez (2020) state that Alzheimer's Disease is the most common form of dementia which is very common among the elderly. Moreover, mild cognitive impairment (MCI) is an intermediate stage between the expected decline of normal aging and the pathological decline caused by dementia. Therefore, it is very crucial to identify risk elements of MCI as early as possible. In order to clarify which features are most important, authors build a random forest that computes feature importances based on Gini impurity. But the authors argue that Gini importances are biased as they weigh continuous and high-cardinality categorical variables higher. In order to solve this problem, the authors implement the permutation-based feature importance method. They reveal that the most important risk factors are subjective cognitive decline (SCD), diet features (sweets and white fish), hours of sleep during the day, and APOE (risk gene called apolipoprotein E). In the next step, PDP plots are used to further examine the effect of each of the four features on the predictions. Finally, the authors argue that their study aims to solve two problems: first, reducing complexity in order to gain interpretability, and second, to provide a methodology that can be used as a prognosis support tool that will help practitioners to identify elderly who has a high risk of developing MCI in 5 years.

Katuwal and Chen (2016) argued that the adoption of complex machine learning models in healthcare had been delayed because of less interpretability, leading to less trust among clinicians. In their research, they used the random forest model to predict the Intensive Care Unit (ICU) mortality rate and interpreted the relative effect of features on the individual predictions, using Local Interpretable Model-agnostic Explanations (LIME). The random forest model yielded 80% balanced accuracy on the test data. The authors revealed that for the randomly selected test subjects, the four most important features were temperature, total CO2, atrial fibrillation, and lactate level to predict ICU mortality rate. As these results were consistent with the modern medical understanding, authors suggest that simplifying complex models by interpretable models is one of the ways to overcome the black-box problem in healthcare machine learning studies.

In their case study of individuals with a risk of hypertension, Elshawi, Al-Mallah, and Sakr (2019) revealed that the LIME technique suffered from the instability of results. Although the LIME algorithm fits a local regression model, which was easy to interpret, explanations provided by the model were not stable because of the random perturbation during the sampling process.

Ariza-Garzon, Arroyo, Caparrini, and Segovia-Vargas (2020) argued that it is possible to have a machine learning credit scoring model to be both accurate and transparent. They built a black box machine learning model which assesses the credit risk of a customer based on nine features. In the next step, Shapley Values are used to explain the findings of the machine learning model. Authors note that quantitative variables were more important than qualitative ones (70% and 30%). *FICO score* (information from the Fair Isaac Corporation credit bureau: defined between 300 and 850) and *loan amount* were the most important quantitative variables in defining the credit risk of customers. On the other hand, in the case of categorical variables, the *purpose* of the loan, specifically credit card loan, and *home ownership* were the most influential variables. The authors also reveal that the same feature values do not influence each individual observation in the same manner, emphasizing the importance of interpretability on a local scale. For example, home improvement as a *purpose* of the loan can either decrease or increase the default chance, while the *purpose:* small business always leads to an increase in the default probability. Finally, according to the authors, their methodology is suitable for credit risk models where interpretability and transparency are required. This is done by adjusting Shapley Values for categories of categorical

variables and quantitative variables in order to better account for each combination between feature values.

2. Methodology

In this chapter, we will explain and discuss the methods we have used in Chapter. 3 for analyzing German Real Estate Data. This includes the Random Forest model, methods for feature selection, and of course, the global and the local IML techniques. Many of our chosen methods are visual in their nature. Therefore, we will provide examples of plots using 50 observations simulated from the following model:

$$y = g(x_1, x_2, x_3) = 5.5x_1^2 + 30x_2 + x_3^{0.5} + x_1x_3 + \varepsilon$$

where $x_1 \sim N(10, 3^2)$, $x_2 \sim P(n)\begin{cases}0.5\ for\ n = 1\\0.5\ for\ n = 0\end{cases}$, $x_3 \sim N(50, 10^2)$ and $\varepsilon \sim N(200, 50^2)$. We fitted a random forest to this data, and we will demonstrate the IML methods applied to that model in this section. Predictions of the random forest will be denoted as $f(x_1, x_2, x_3)$.

2.1 Random Forest

In our analysis of IML methods, we will be using random forests as a reference black-box model. There are a couple of reasons why we want to further interpret random forests. First, random forest is one of the most widespread machine learning models used in real-world cases. Random forests offer increased flexibility as they can be used both in classification and regression problems with a high degree of accuracy. Moreover, random forests have low risks of overfitting as averaging through uncorrelated trees offers lower variance and prediction error. But along with their benefits, random forests bring in quite a few challenges, such as being computationally expensive in some cases and having less interpretability. (IBM CE,2020)

To describe random forests, we will briefly discuss decision trees as they are building blocks of the random forest model. Decision trees usually start with a binary question such as "Will it rain

today?" and from there, additional questions are asked to determine the features and divide the values into different decision nodes. Each question helps us to arrive at a certain final decision which is represented by a leaf node. At each decision node, the algorithm goes through all possible features and selects the one that generates the best split to subset the data. (IBM CE,2020).

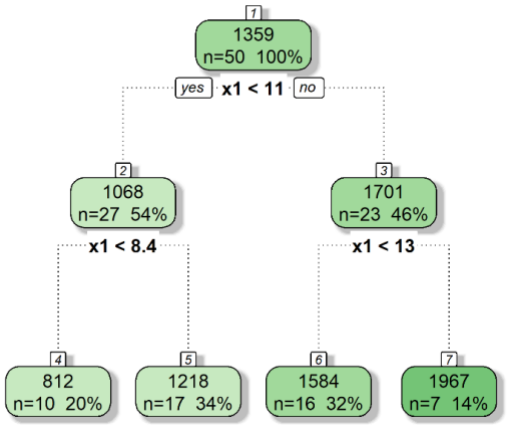


Figure 0.1: Regression tree for predicting y using the simulated dataset

The point along the tree where predictor space is split are referred to as internal or decision nodes. For example, on the left-hand side of Figure 2.1, the split is done based on whether the feature x1 is less than 8.4. This point is called the decision node, and there are a total of 3 decision nodes. Furthermore, nodes with no further split are called leaf or terminal nodes. The number in each leaf indicates the mean of the response of observations that belong there. There are a total of 4 leaves in this decision tree.

Decision trees can be used both for regression and classification problems. The process of constructing a regression tree can be summed up in two steps:

1. The predictor space, which is the set of possible values for the features $X_1, X_2, \ldots, X_p$ is divided into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.
2. For every observation that is within the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

For example, let's say that after the first division, we end up with regions R_1 and R_2, and the response mean of training observations in the first region is 5, while the response mean of training observations in R_2 is 10. Then for a given observation $X = x, if\ x \epsilon R_1$, we will predict a value of 5, and if $x \epsilon R_2$, we will predict a value of 10. An important question is how to construct regions $R_1, \ldots, R_J$? Theoretically, regions can be any kind of shape. But to ensure more interpretability and less complexity, predictor space is divided into high-dimensional rectangles or "boxes". The goal is to find boxes $R_1, \ldots, R_j$ that minimize the RSS, given by

$$\sum_{j=1}^{J} \sum_{i \epsilon Rj} \left(y_i - \hat{y}_{R_j}\right)^2 \quad (2.1)$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box. (Pal,2007) However, it is computationally impossible to consider every single piece of feature space in the J boxes. A *top-down greedy* method is to use as *recursive binary splitting.* This approach is called top-down because we start from the top of the tree where only one region exists and then subsequently split the predictive space into two branches each time. On the other hand, this approach is called greedy because, at a particular step, the best split is done rather than looking forward and coming up with a split that will lead to better results in the future. (Yiu, 2019)

Random forests consist of multiple decision trees, which together create an ensemble. Each individual tree comes up with a prediction, and the final prediction of the random forest is decided through either averaging for regression or by the majority rule for classification. An ensemble of uncorrelated trees will always outperform any individual decision tree. The reason for this is as follows: the ensemble of trees protects each other from their individual errors (Yiu, 2019). But there are two prerequisites for the high performance of random forest models: 1) there should be predictive power of features so that the model built using those features perform better than random guessing, 2) the predictions of individual trees should be uncorrelated from each other (at least very low correlation). (Yiu, 2019) So the question arising here is how the algorithm ensures that trees have a low correlation between them? The answer is the usage of the following two methods: Bagging (bootstrap aggregation) and feature randomness. The bagging method was introduced in 1996 by Leo Breiman (Breiman cited in Yiu, 2019); random samples

with replacement are generated from the dataset for each tree. This means that each tree will be estimated on a different subset of data. The second method for obtaining low correlation is the feature randomness of random forests. As explained above, in the decision trees, when a decision node is split, the algorithm goes through all features and picks the best one that offers the most separation. However, in random forests, each individual tree can select from only a random subset of features at each split, forced by the algorithm, which ensures further variation and ultimately lowers correlation amongst the trees (Yiu, 2019).

2.2 Feature Selection by Recursive Feature Elimination

Performing feature selection before fitting the machine learning model has several advantages. It can improve model performance by removing redundant variables that might add noise to the data and increase the risk of overfitting. This is also beneficial for the IML methods since we do not want to study features that are not associated with the target variable. Removing redundant variables also reduces the computational costs, which is very helpful when working with computationally expensive methods like Shapley Values or the H-Statistic that are described in sections 2.4.7 Shapley Values and 2.4.2 H-Statistic, respectively.

The Recursive Feature Elimination (RFE) is a backward feature selection algorithm that fits a machine learning model to different subsets of predictors (Kuhn, 2019). First, it fits a model to all the predictors and ranks them by their importance to the model. Let S be a sequence of ordered numbers where each number is a candidate value for numbers of features to retain ($S_1 > S_2 > \cdots$). For each iteration, the S_i most important features are retained for refitting the model, and the error is measured. The algorithm avoids the problem of overfitting by implementing a resampling method, which in our case is Cross-Validation. The method can be summarized as follows:

Algorithm 2.1 Recursive Feature Elimination

1. For each fold in the Cross-Validation:
    - Train the model on the training set using all p features
    - Predict the held-back samples
    - Calculate feature importance and rank them

      For each subset size S_i, $S(i = 1,\ 2, \ldots p)$:
        - Keep the S_i most important features
        - Train the model on the training set using S_i features
        - Predict the held-back samples

      End

    End
2. Calculate the performance profile over the S_i using the held-back samples
3. Determine the appropriate number of features
4. Estimate the final list of features to keep in the final mode

2.3 Multicollinearity tests

If two or more of the variables that are chosen by RFE are highly correlated, then some of the variables might be redundant if the consequence of removing it from the model is only a negligible increase in error. Multicollinearity tests can be used to test for such situations, and performing such tests is also useful for the IML analysis because some of the IML methods are only reliable when the features are uncorrelated.

When checking for multicollinearity, there are three different cases that require different methods, depending on whether we are investigating the degree of association between pairs of continuous variables, pairs of categorical variables, and pairs with one categorical and one continuous variable.

Variable Pair	Dependence Test
Continuous-Continuous	Pearson's r
Categorical-Categorical	Cramér's V
Continuous-Categorical	ANOVA Test

Table 0.1: The dependence tests used for each variable pair.

Pearsons'r

Pearson's r measures the linear correlations between two continuous variables x, and y (James, Witten, Hastie & Tibshirani, 2021). It is the covariance of the variables divided by the product of their standard deviations. It will be a number between -1 (perfectly negatively correlated) and 1 (perfectly positively correlated).

$$r_{xy} = \frac{cov(X,Y)}{\sigma_x \sigma_y} \quad (2.1)$$

Cramer's V

Cramér's V measures the degree of association between two categorical variables (Medium, 2018). It uses the chi-square statistic: Let a sample size of n of the simultaneously distributed categorical variables A and B for $i = 1, \dots, r: j = 1, \dots, k$ be given by the frequencies n_{ij} = the number of times the values (A_i, B_j) were observed. Then, the chi-square statistic is computed the following way:

$$\chi^2 = \sum_{i,j} \frac{\left(n_{ij} - \frac{n_i n_j}{n}\right)^2}{\frac{n_i n_j}{n}} \quad (2.2)$$

Where, r is the number of rows and k is the number of columns. The next step is using the chi-square statistic to compute Cramér's V:

$$V = \sqrt{\frac{\chi^2/n}{\min(k-1, r-1)}} \quad (2.3)$$

It ranges from 0 (no association) to 1 (complete association).

ANOVA

Analysis of variance (ANOVA) is a statistical test used to check whether the means of two or more independent groups are significantly different from each other (Kim, 2017). We use ANOVA when we have a single independent variable (categorical), and our goal is to check if variations or different levels of the categorical variable have a significant effect on a dependent variable (continuous). ANOVA analyses the level of variance within the independent groups through samples taken from them. If the variance is high within the groups, then it means that the mean of the sample chosen from the data will be different due to chance. In ANOVA, a Null and Alternative hypothesis is formulated to check these conditions. Under the null hypothesis, all the

sample means are equal, and they do not possess any significant differences. In contrast, the alternative hypothesis is true when at least one sample mean is different from the others (Kim, 2017). These hypotheses mathematically expressed as follows:

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_L \qquad Null\ hypothesis$$
$$H_1 : \mu_l \neq \mu_m \qquad Alternative\ hypothesis$$

where, $\mu_l\ and\ \mu_m$ belong to any two sample means out of all samples included in the test. If the null hypothesis cannot be rejected, then the categorical variable did not have any impact on the dependent continuous variable. On the other hand, if the null hypothesis is rejected, then at least one of the sample means is different from the other (Kim, 2017).

As a measure of the strength of the association in ANOVA, we have used the statistical term, η^2 $(Eta\ squared)$. η^2 represents the percentage of variance in the dependent variable accounted for by the independent variable (Richardson, 2011). The formula for η^2 is:

$$\eta^2 = SS_{effect}/SS_{total} \quad (2.4)$$

where, SS_{effect} is the sum of squares of the effect we are investigating, and SS_{total} is total sums of squares for all effects, errors, and interactions in the ANOVA (Richardson, 2011).

2.4 Interpretable Machine Learning Methods

2.4.1 Partial Dependence Plots

Partial dependence plots (PDP) are one of the global methods that describe the marginal effect that features have on the prediction by a machine learning model (Boehmke & Greenwell, 2021). The plot shows how predictions change as the value of features changes while averaging the effect of all the other features. In this way, we can learn about the degree of association between the feature and the target variable and the structure of the association, e.g., if it is linear or more complex.

Algorithm 2.2 Estimating PDP

For a selected feature x

1. Create a grid of j evenly spaced values over the distribution of x: $(x_1, x_2, \dots, x_j)$
2. For *i* in (1, 2, …, *j*):
    - Copy the training data, but replace the original values of x with the constant x_i
    - Apply an ML model to obtain a vector with predictions of every target variable in the training data
    - Average the predictions

    End
3. Plot the averaged predictions against $x_1, x_2, \dots, x_j$

A partial dependence function can also be expressed mathematically:

$$PD_{x_s}(x_s) = \frac{1}{n}\sum_{i=1}^{n} f\left(x_s, x_c^{(i)}\right) \quad (2.5)$$

Where x_s are any given value for the feature of interest, PD_{x_s} is the partial dependence function for this constant, and f is the ML model that the PDP is estimated for. *n* is the number of observations in the training data, i is a single observation, and $x_c^{(i)}$ is a vector of the real values of the other features in the training data. The function measures the average effect of any given values of x_s on the outcome of the prediction.

The formula and algorithm above are for estimating the marginal effects of one feature, but PDPs can also be estimated for multiple features. In practice, a maximum of two features are used for the plots because visualizing plots with more features are very impractical. A PDP showing the effects of two features is a two-way PDP.

PDPs should be used with great caution. They assume that the features in x_c are uncorrelated with the feature of interest. When this assumption is violated, we can get extremely unlikely data points in the estimation of the PDP. For instance, in our application in section 3.4.3, an apartment of 50 $M^2$and 10 rooms.

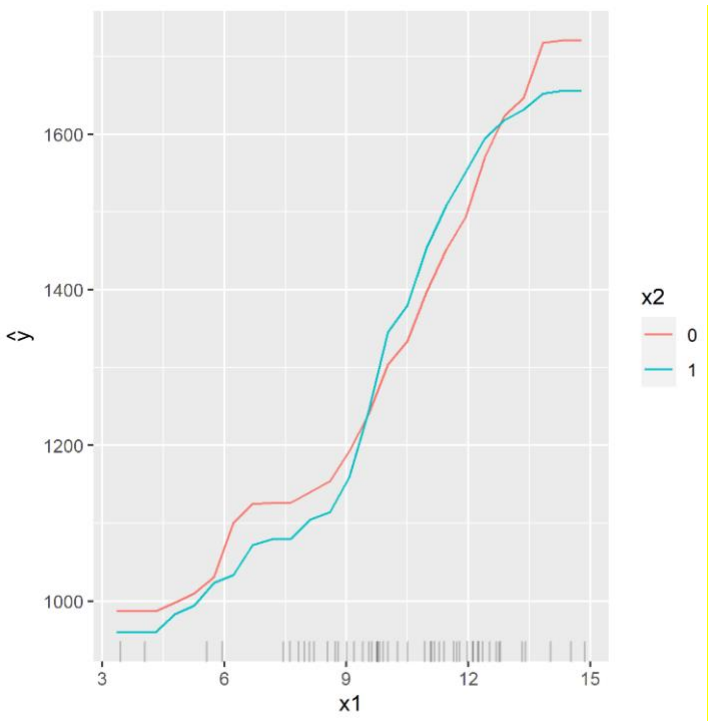


Figure 0.2: Two-way PDP of x_1 and x_2 effects on y, using 25 equally spaced values of x_1.

Figure 2.2 shows that the curve of x_1 is similar for both values of x_2. Increasing x_1 results in a higher random forest prediction of y.

2.4.2 H-Statistic

Another way to study global model behaviour is to look at how feature interaction affects predictions (Molnar, 2021). The H-Statistic uses partial dependence to measure the interaction strength between two features or between one feature and all other features. Specifically, it estimates how much of the variation in the predictions can be explained by the interaction of two features.

When there is no interaction between two features, the two-way partial dependence function, $PD_{jk}(x_j, x_k)$, can be expressed as follows:

$$PD_{jk}(x_j, x_k) = PD_j(x_j) + PD_k(x_k) \quad (2.6)$$

where $PD_j(x_j)$ and $PD_k(x_k)$ are the partial dependence function for the single features. A lack of interaction between one feature and all other features means that the machine learning model with all the features can be expressed similarly:

$$f(x) = PD_j(x_j) + PD_{-j}(x_{-j}) \quad (2.7)$$

where $f(x)$ is the machine learning model, and $PD_{-j}(x_{-j})$ is the partial dependence function for all features except *j*.

If there was any interaction between two features or between one feature and all other features, the two-way partial function and the entire function could not be expressed as the sum of the individual parts. The H-Statistic measures the degree of interaction by comparing the two-way dependence function with the same function under the assumption of no interaction:

$$H_{jk}^2 = \sum_{i=1}^{n} \left[PD_{jk}\left(x_j^{(i)}, x_k^{(i)}\right) - PD_j\left(x_j^{(i)}\right) - PD_k\left(x_k^{(i)}\right) \right]^2 / \sum_{i=1}^{n} PD_{jk}^2\left(x_j^{(i)}, x_k^{(i)}\right) \quad (2.8)$$

To capture interactions between one feature and all other features, the H-statistic is expanded as follows:

$$H_{jk}^2 = \sum_{i=1}^{n} \left[f\left(x^{(i)}\right) - PD_j\left(x_j^{(i)}\right) - PD_{-j}\left(x_{-j}^{(i)}\right) \right]^2 / \sum_{i=1}^{n} f^2\left(x^{(i)}\right) \quad (2.9)$$

The former is called the two-way H-Statistic, and the latter is the H-Statistic for total interaction. If there is no interaction effect, the statistic is 0, and when all variation can be explained by the interaction effect, the statistic is 1.

2.4.3 Accumulated Local Effects

Like the PDP, accumulated local effects (ALE) plots show how one or two features influence the predictions of a machine learning model (Molnar, 2021). But instead of looking at the average prediction in the data for any given value of the relevant feature, ALE measures how small changes in the feature lead to a difference in prediction. By just slightly changing the feature values, ALE avoids the type of problems featured in PDPs, with data points having unlikely or even impossible combinations of feature values.

Algorithm 2.3 Estimating ALE plots

For a selected feature x

1. Divide the feature distribution into m intervals that are defined by quantiles
2. For each interval z in (1, 2, …, m):
   - Calculate the difference in prediction for all observations in z by replacing the feature with the upper and lower limits of the interval without changing the values of the other features
   - Average the differences in prediction to obtain the uncentered effect: $\tilde{f}_{z,ALE}(x)$
   - Calculate the average difference in prediction for all observations in the training data, and subtract this from $\tilde{f}_{z,ALE}(x)$ to get the centered effect: $\hat{f}_{z,ALE}(x)$
3. Accumulate $\hat{f}_{z,ALE}(x)$ across the entire distribution to obtain a plot

This way, $\hat{f}_{z,ALE}(x)$ can be interpreted as the effect of the feature at a certain value compared to the average prediction in the data.

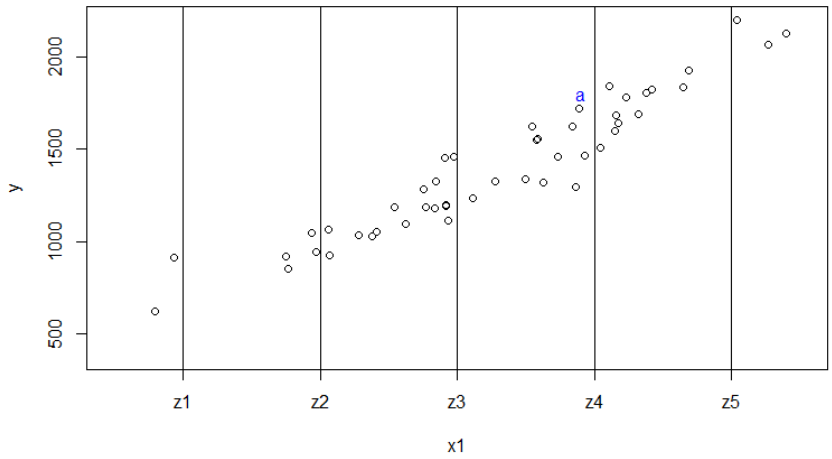


Figure 0.3: The intuition of ALE. The response variable y is plotted against the feature x1. The difference in prediction for observation "a" with respect to x1 is: f (z4, x2, x3) – f(z3, x2, x3).

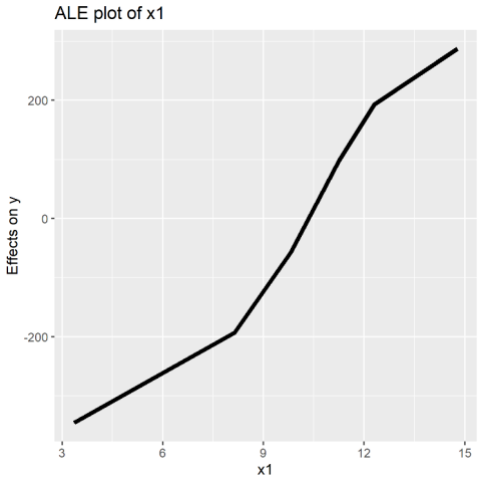


Figure 0.4: Example of an ALE plot on our simulated data. The local effects of x1 on y are accumulated across the entire distribution, drawing a plot. The plot was estimated using five intervals.

Figure 2.4 is similar to the PDP in Figure 2.2; increasing x_1 results in a higher random forest prediction of y.

ALE for categorical features

Since the ALE method accumulates effects in a certain direction, the feature values need to have an order. Because categorical features do not have a natural order, we must find one. This is done by comparing the different categories based on their similarity with respect to the other features.

For instance, let us say that in our application in section 3.4.2 Ale Plots, we only have two features, *regio2* (geographical location) with two locations and *livingSpace* (area of the apartment in square meters). Then we can compute the cumulative distribution of *livingSpace* in both categories of *regio2* and obtain a feature-wise distance based on how similar the distributions are. That is how feature-wise distance is estimated with respect to numerical features, but for categorical features, the relative frequency table is used to compare similarity. In practice, there are often more than two features, and in this case, the feature-wise distance would be the sum of the differences between both categories across all other features. When the distances between all categories have been calculated, multi-dimensional scaling is used to reduce the distance matrix to a one-dimensional distance measure. Then we get a similarity-based order of the categories.

2.4.5 Permutation Feature Importance

The idea behind permutation feature importance is simple: the importance of the feature is measured by finding the increase in the model's prediction error after we randomly permute the values of the feature. It works the following way: if we randomly permutate the values of an important feature, then the predictive power of this particular important feature will decrease. Also, the feature is unimportant if randomly permuting its values does not change much of the model's error because the model did not rely heavily on this feature in the initial prediction process (Boehmke & Greenwell, 2020). The permutation feature importance was first used by Breiman (2001) for random forests. After this idea was proposed, Fisher, Rudin, and Dominici

(2018) introduced a new notion called model reliance which is a model agnostic version of feature importance. The algorithm proposed by Fisher, Rudin, and Dominici (2018) is as follows:

Algorithm 2.4 Permutation Feature Importance

Input: Trained model f, feature matrix x consisting of the p different observed features, the corresponding target vector y, and an error term $L(y, f(x))$.

1. Calculate the initial model error $e^{orig} = L(y, f(x))$.
2. For each feature *j* in (1, 2, …, *p*):
    - Create feature matrix x^{perm} by permuting feature j in the data x. By doing so, the dependency between feature j and true outcome y is broken
    - Calculate error $e^{perm} = L(y, f(x^{perm}))$ based on predictions of permuted values
    - Estimate the permutation feature importance $FI^j = e^{perm}/e^{orig}$

    End
3. Based on FI values, sort the features in descending order.

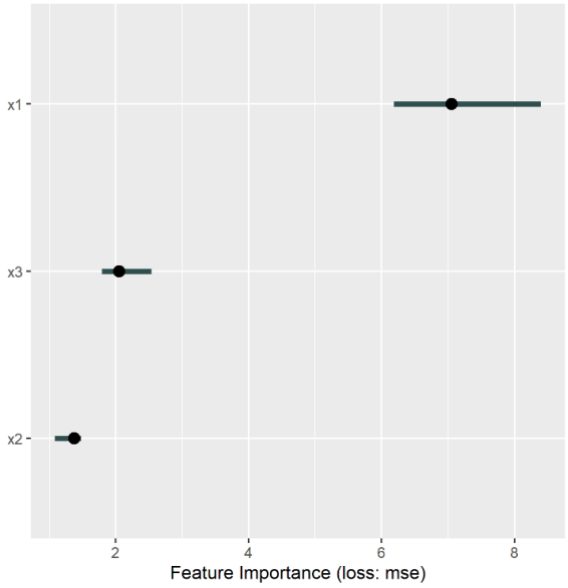


Figure 0.5: Permutation Feature Importance applied to our simulated data

Figure 2.5 displays the estimated permutation feature importance for the simulated model. Here, the permuting process was repeated three times, and the loss function is the mean squared error. x1 is the most important feature as permuting it increases the error by about 6 to 8 times.

2.4.6 Local Interpretable Model-Agnostic Explanations (LIME)

As opposed to global surrogate models, which explain the global and general behavior of the black-box model, local interpretable model-agnostic explanation (LIME) is an algorithm that explains individual predictions and was introduced by Riberio, Singh, and Guestrin (2016). The LIME algorithm assumes that every complex machine learning model is linear in a small neighborhood around the target observation (local scale). Under this assumption, it is possible to fit a simple model around a target individual observation, which will represent how the global model behaves at that local point (Riberio, Singh, and Guestrin, 2016). To achieve this, the LIME algorithm samples through the training data to come up with observations that are similar to the target individual observation. Then it tests how the predictions change when the ML model is given different variations of the training data. Afterward, a new dataset is created, which consists of permuted samples and corresponding predictions by the black-box model. On this dataset, the LIME algorithm trains an interpretable model, typically a regression model using the least absolute shrinkage and selection operator (LASSO) method, which is weighted by the proximity of the sampled instances to the instance of interest (Boehmke & Greenwell, 2020). The requirement is that the new interpretable model should be a good enough approximation of the black-box model on a local scale, but the same performance is not expected on a global scale. This notion is called local fidelity. Local surrogate models can be mathematically expressed as follows:

$$explanation(x) = arg \min_{g \epsilon G} L(f, g, \pi_x) + \Omega(g) \quad (5)$$

where x is the target individual observation, g is the interpretable model, L is a loss function (e.g., mean squared error), f is the original black box model, π_x is proximity measure which defines how large the neighborhood around target x should be, and $\Omega(g)$ is model complexity (Biecek, Burzykowski, 2020). The user should define complexity, for example, the maximum number of features that the linear model may use.

Algorithm 2.5 LIME

1. Permute the training data to come up with the replicated datasets.
2. Estimate the proximity measure between target observation and each permuted observation.
3. Use the machine learning model to come up with predictions based on the permuted data.
4. Choose the m number of the features that have the most predictive power.
5. Fit a new simpler interpretable model to the permuted data, explaining the complex model's predictions with m features from the permuted data weighted by its similarity to the original observation.
6. Use the new feature weights to explain local behavior. (Boehmke, Greenwell, 2020)

One question that may arise here is how we create variations of the data? In the case of tabular data, new variations are created by perturbing each feature individually based on the mean and standard deviation of the feature in the normal distribution.

One of the difficulties of the LIME algorithm is defining a meaningful neighborhood around target observation (proximity measure). The LIME algorithm uses an exponential smoothing kernel to define this neighborhood (Molnar,2021). For instance, small kernel width means that the neighboring instance must be very close to affect the local model. The best way to come up with a reasonable kernel value is to try out different values reasonable for the application and test if the explanations make sense.

2.4.7 K-Nearest Neighbours

The first step of the LIME algorithm is permuting the observations of interest. This introduces an element of randomness that can create different results when running the algorithm multiple times. The instability of the results is a potential weakness in the LIME method.

As an attempt to deal with this problem, we will use the K-Nearest Neighbours (KNN) method to find the K observations from the dataset that are most similar to our observations of interest and fit a simple model to this data. By using real observations from the dataset rather than permuted data, we remove the element of randomness.

The KNN method is mostly used as a prediction method (James, Witten, Hastie & Tibshirani, 2021). To predict a test set observation x_0 in a regression problem, KNN finds the K most similar observations from the training set, and the average outcome of those observations is the

prediction for x_0. However, we will only be using KNN to identify the nearest neighbors, not for prediction.

There are several distance metrics that KNN can use to identify nearest neighbours, but we will use Euclidean distance. The Euclidean distance between two observations is the sum of differences between all the features (Fiori, 2020).

$$d(x, y) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

Where x and y are the observations to measure the distance between, i is any given feature, and m is the total number of features.

2.4.8 Shapley Values

It is normal that when a model produces predictions, not all the features play an equal role in them. The influence of each feature can be measured by estimating the prediction error of the model by removing a particular feature and calculating its importance based on its absence from the model. However, estimating a single feature influence one at a time means that dependencies between features are not considered, which could lead to some inaccuracies. Therefore, to observe all dependencies between features, we use Shapley Values which is also a method to explain individual predictions from the black-box model (Boehmke & Greenwell, 2020). The concept of Shapley values is that the feature values of an individual observation work together to influence the model's predictions with respect to the model's expected output, and it divides this total change in prediction among the features in a way that is fair to their contributions across all possible subsets of features. To achieve this, Shapley Values go through each combination of features to assess their predictive power. For example, to calculate the importance of feature x, the model will consider the accuracy of all combinations of features excluding x and then analyze how accuracy changes by adding x to all these combinations. Therefore, calculating Shapley Values is computationally expensive (Boehmke & Greenwell, 2020).

Algorithm 2.6 Shapley Values

Here we first denote the following required variables: number of iterations M, an instance of interest x; feature index j; data matrix X; and machine learning model f.

For each feature *j* in (1, 2, …, *m*):

1. Draw a random instance z from the data matrix X
2. Select a random permutation o of the feature values
3. Order instance x: $x_0 = (x_{(1)}, \dots, x_{(j)}, \dots, x_{(p)})$
4. Order instance z: $z_0 = (z_{(1)}, \dots, z_{(j)}, \dots, z_{(p)})$
5. Create two new instances
    - With feature j: $x_{+j} = (x_{(1)}, \dots, x_{(j+1)}, x_{(j)}, z_{(j+1)}, \dots, z_{(p)})$
    - Without feature j: $x_{-j} = (x_{(1)}, \dots, x_{(j-1)}, z_{(j)}, z_{(j+1)}, \dots, z_{(p)})$
6. Calculate marginal contribution: $\phi_j^m = \hat{f}(x_{+j}) - \hat{f}(x_{-j})$
7. Compute Shapley value as the average: $\phi_j(x) = \frac{1}{M}\sum_{m=1}^{M} \phi_j^m$ (Molnar, 2021).

3. Applications to German Real Estate Data

In this chapter we apply the IML methods to a dataset about German Real Estate to show how they can reveal some of the mechanisms of a black box model that predicts monthly rent. First, the dataset is presented along with the pre-processing steps. Then, feature selection with RFE and multicollinearity tests are conducted to find the features that are most important for predicting rent. After finding the best features, we estimate a random forest model, and finally, we interpret this model with the IML methods.

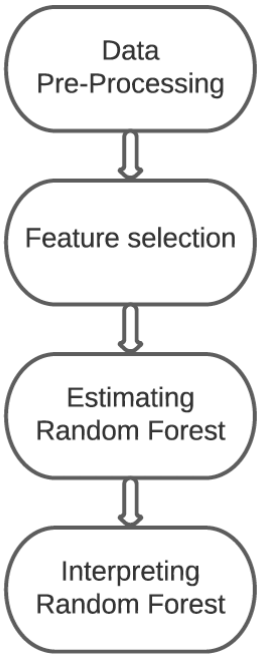


Figure 0.1:The process of our application.

3.1 Dataset and Pre-processing

Our dataset was scraped from Germany's leading real estate platform *Immobilienscout24* on the dates 2018-09-22, 2019-05-10, and 2019-10-08. It contains information about offers on rental properties in all German regions. The variables are related to the monthly rent of the offer,

physical characteristics of the apartment, geographical information, and text descriptions. The raw data consists of 268.850 observations and 49 variables.

The dataset did not need much pre-processing, as most of the variables were suitable for machine learning in their raw form. However, some modifications to the dataset were needed. Some variables had to be excluded for various reasons. A few of them were unsuitable because of a large majority of missing values (NAs). We also left out text descriptions as textual analysis is outside the scope of this thesis.

There were also some problems with a few very unlikely or impossible data points. For instance, zero square meters of living space, zero monthly rent, or apartments with 2.5 rooms. Data points with zero square meters or no monthly rent were simply removed, while non-integer rooms were rounded. After the pre-processing, we were left with 55,810 observations.

3.2 Feature Selection

The Recursive Feature Elimination found that the best Random Forest model contained the following features:

Variable Name	Variable Type	Description
livingSpace	Numerical	Area of apartment in square meters
Regio2	Categorical	The city where the apartment is located
interiorQual	Categorical	Label of interior quality
yearConstructed	Numerical	The year of construction
hasKitchen	Categorical	If the apartment has a kitchen
lift	Categorical	If the building has a lift
balcony	Categorical	If the apartment has a balcony
condition	Categorical	The condition of the apartment
noRooms	Numerical	Number of bedrooms in the apartment
noRoomsRange	Numerical	The range of the number of bedrooms
numberOfFloors	Numerical	Number of floors in the building
heatingType	Categorical	Heating type in the apartment
newlyConst	Categorical	If the apartment is newly constructed
telekomUploadSpeed	Numerical	Internet upload speed
picturecount	Numerical	Number of pictures on the apartment listing

Table 0.1: Variables selected by Recursive Feature Elimination.

As argued in 2.3 Multicollinearity Tests, some of these features might still be redundant if they are highly correlated with other features, and removing them from the model only causes a negligible increase in prediction error.

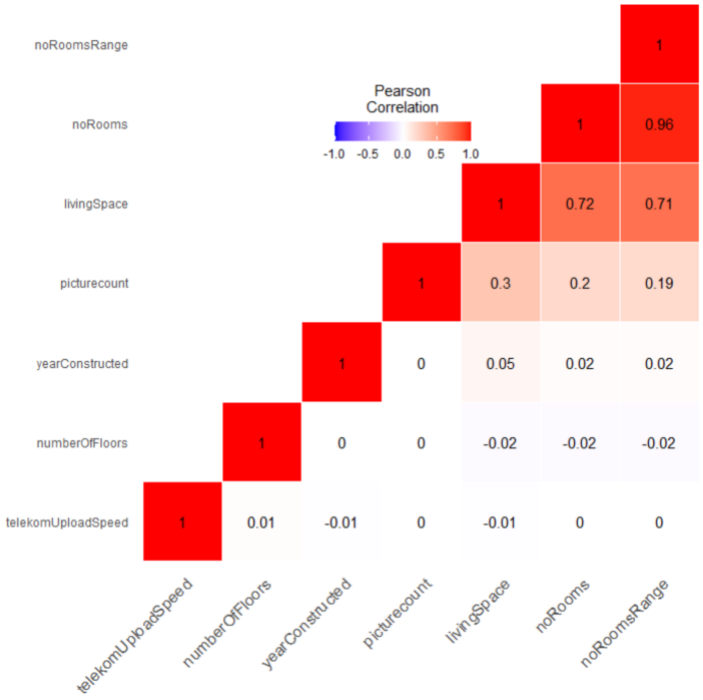


Figure 0.2: Pearson correlation measuring the strength of association between numerical features.

Unsurprisingly, the Pearson Correlation plot shows an almost perfect correlation between *noRooms* and *noRoomsRange*.

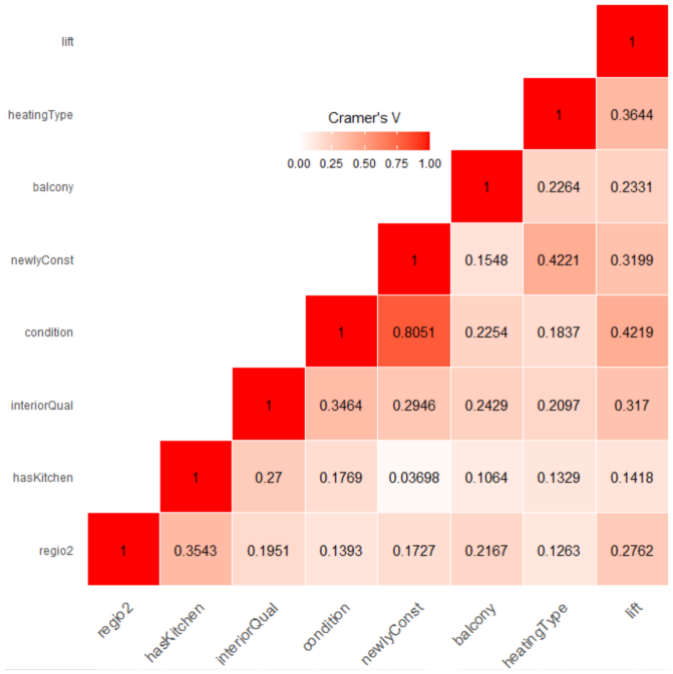


Figure 0.3:Cramér's V measuring the strength of association between categorical variables.

The Cramér's V plot indicates that the categorical variables *condition* and *newlyConst* are very strongly associated and can potentially be eliminated from the model without causing a problematic increase in error.

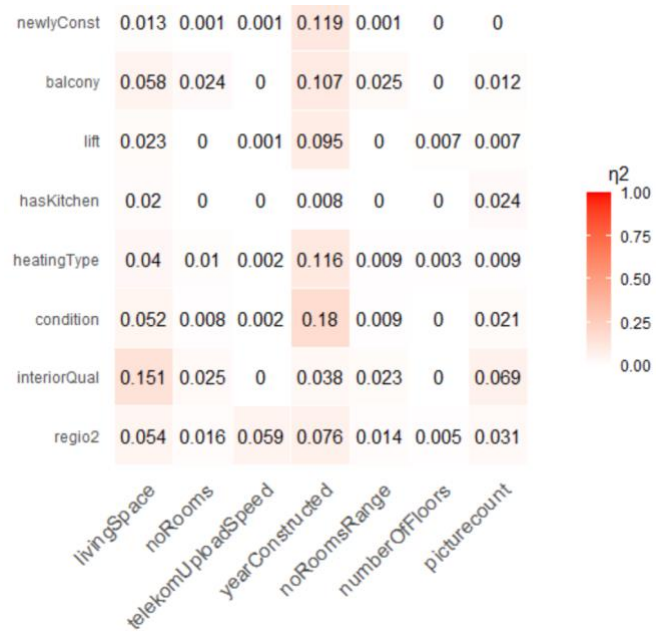


Figure 0.4: η^2 measuring the strength of association between pairs of continuous and categorical variables.

The pairs of categorical and continuous variables are very weakly associated such that no feature will be removed from the dataset because of the results from the ANOVA test.
This leaves *noRooms*, *noRoomsRange*, *newlyConst*, and *condition* as potentially redundant variables. We estimated a Random Forest with all features included as the base model. Then four other models were estimated, where each excluded one of the features mentioned above. We compare the models using mean absolute percentage error (MAPE).

Model	MAPE
All features included	13.64%
noRooms excluded	13.66%
noRoomsRange excluded	13.68%
newlyConst excluded	13.65%
condition excluded	13.81%

Table 0.2: MAPE of random forests with highly associated features removed.

Removing *newlyConst* or *noRooms* results in the smallest increase in error compared with the base model. Because *newlyConst* and *condition* measure different properties of the apartment and *noRooms* and *noRoomsRange* are just different measures of the same characteristic, we reason that it is only reasonable to remove *noRooms*. Thus, the final model will include all features from table 3.1 except *noRooms*.

3.3 Random Forest Estimation

Before estimating the random forest model, we have divided the dataset into training and test sets to assess the model accuracy on unseen data. Training set consisted 80% of all observations (44,648 observations) and test set included 20% (11,162 observations).
For a model to achieve high accuracy, it must capture the relationship between features and outcome to a large degree. Such a model would therefore be more interesting to interpret compared to a less accurate one. To achieve high accuracy, we searched through different random forest hyperparameters to find the best performing model. The following hyperparameters were tested:

- Nodesize, which controls the minimum number of observations in the terminal nodes. Decreasing this number creates a deeper and more complex tree.
- Sample size used to estimate each tree.
- Mtry, which is the number of features to consider at each split.
- Ntree, number of individual decision trees in the model.

We tested a total of 120 combinations of parameters. Table 3.3 shows the optimal parameters.

Parameter	Value
Nodesize	5
Sample Size	16000
Mtry	8
Ntree	500

Table 0.3: Optimal parameter values of the Random Forest.

After fitting the model with these parameters, the error on the test set measured by mean absolute percentage (MAPE) was 13.26%.

3.4 Interpreting the Model Using IML Methods

As previously discussed in the thesis, ML models can be interpreted at the global scale and the local scale. We start by interpreting global model behavior with permutation feature importance, ALE plots, the H-Statistic, and PDPs. Then, the model is studied at the local scale with implementation of LIME and Shapley Values.

3.4.1 Permutation Feature Importance

As a first step in interpreting the model, we used permutation feature importance. As discussed before, PFI measures the importance of features by permuting values of the particular feature. If the training performance of the model decreases after randomly permuting the values of a certain feature, then that feature is important. The results of PFI are shown in Figure 3.5. As we can see, the most important feature is *livingSpace*, followed by *regio2*, which represents the city where

the apartment is located. Interestingly, features such as *condition, hasKitchen, balcony* have almost no power in explaining *total rent*, according to the permutation feature importance.

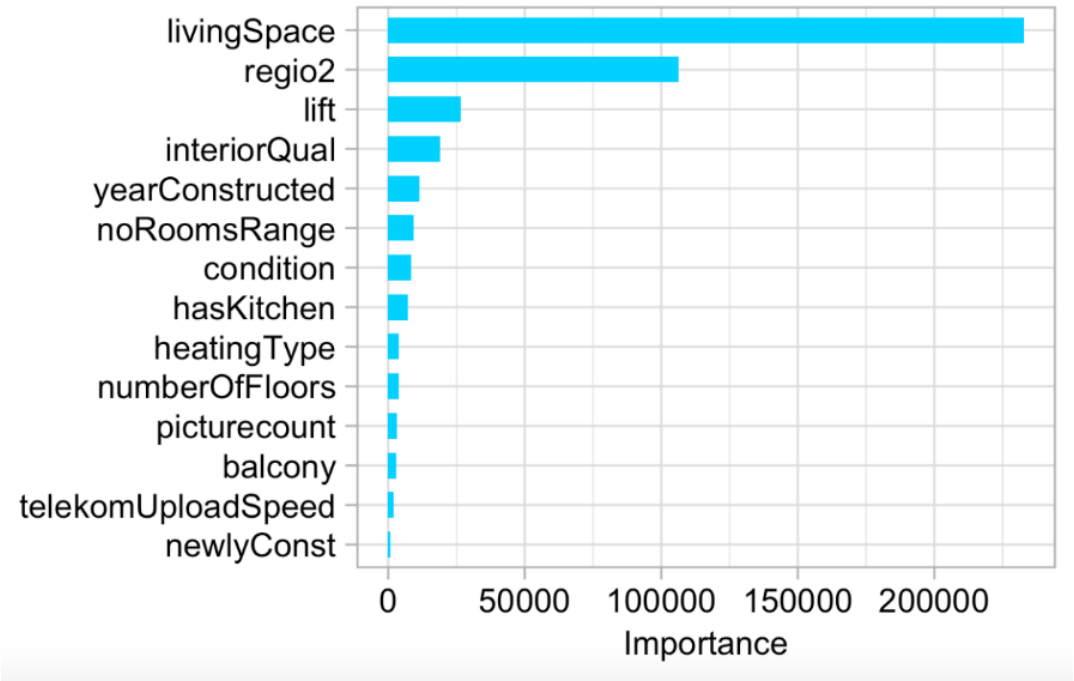
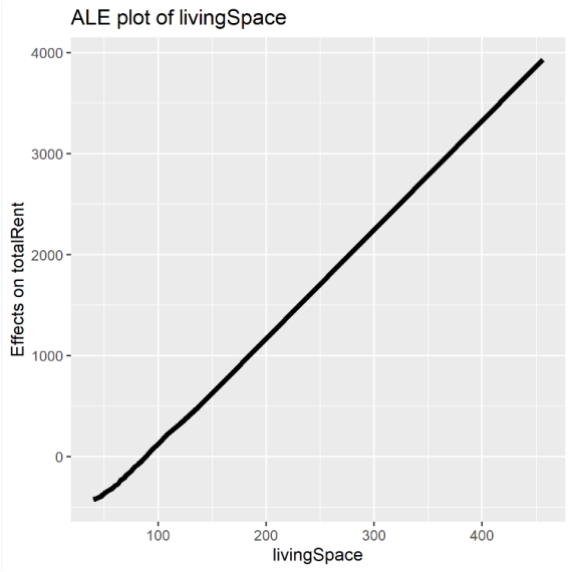


Figure 0.5: Permutation Feature Importance

3.4.2 ALE Plots

The feature-based importance method showed that *livingSpace* and *regio2* are the most important features. Therefore, it would be interesting to look closer at the effects of these features on *totalRent*.



.

Figure 0.6: The local effects of livingSpace on totalRent. 20 quantiles were chosen as the number of intervals.

As seen from Figure 3.6, the effect of *livingSpace* corresponds to an almost perfectly linear increase of *totalRent*. The effect of having less than 100 m2 of *livingSpace* results in a smaller than average prediction of *totalRent*.

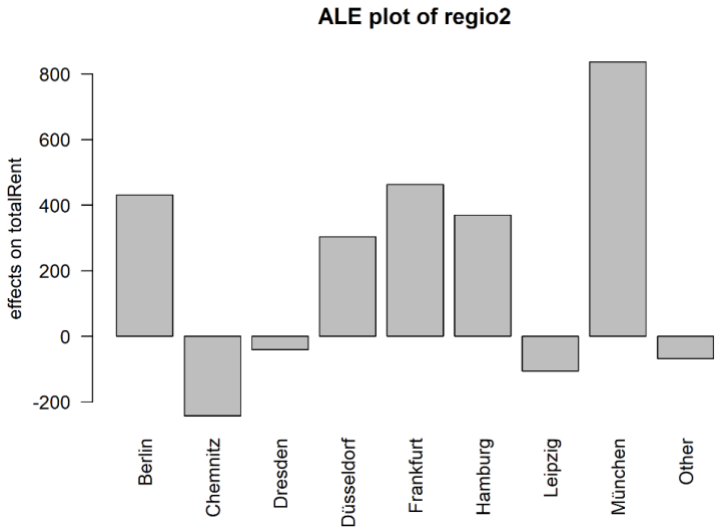


Figure 0.7: The local effects of the biggest cities on totalRent. We choose the nine most frequent cities from the dataset as it would be impractical to visualize all 51 categories.

Among the biggest cities, there are big differences in how they affect the prediction of *totalRent*. München has the strongest positive effect, but other large cities like Berlin and Frankfurt are also

associated with *totalRent* that is higher than average. Chemnitz has the strongest negative effect. The "Other" category, which captures apartments outside the biggest cities, is associated with a prediction that is slightly below average.

3.4.3 H-Statistic and PDP

The H-Statistic for total interaction shows to what extent a feature interacts in the model with all other features.

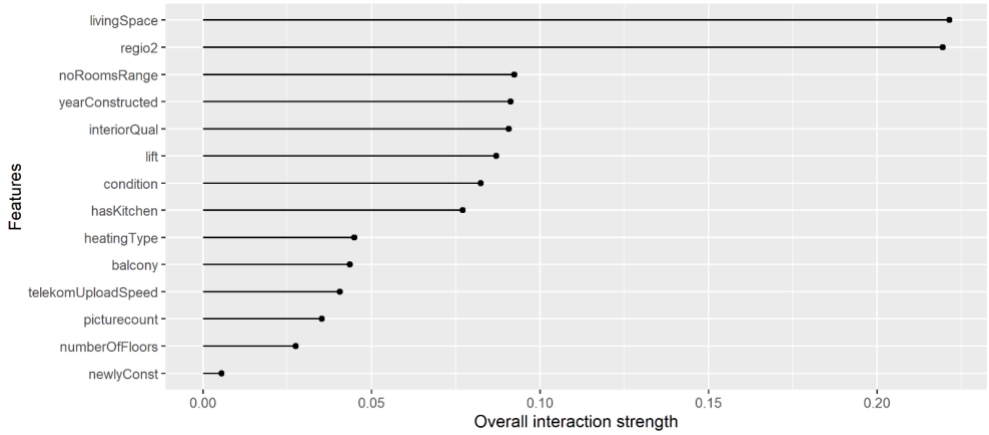


Figure 0.8: The H-Statistic for total interaction for all features.

The most important variables from the feature-based importance method are also those who interact the most with other variables. The H-Statistics of *livingSpace* and *regio2* are respectively 0.222 and 0.220, which means that for both features, approximately 22% of the variation in the predicted outcome can be explained by their interaction with other features.

To investigate these interaction effects further, we will look at how *livingSpace* and *regio2* interact with all other features by studying their two-way interaction effect, equation (4.3) in 2.4.2 H-Statistic.

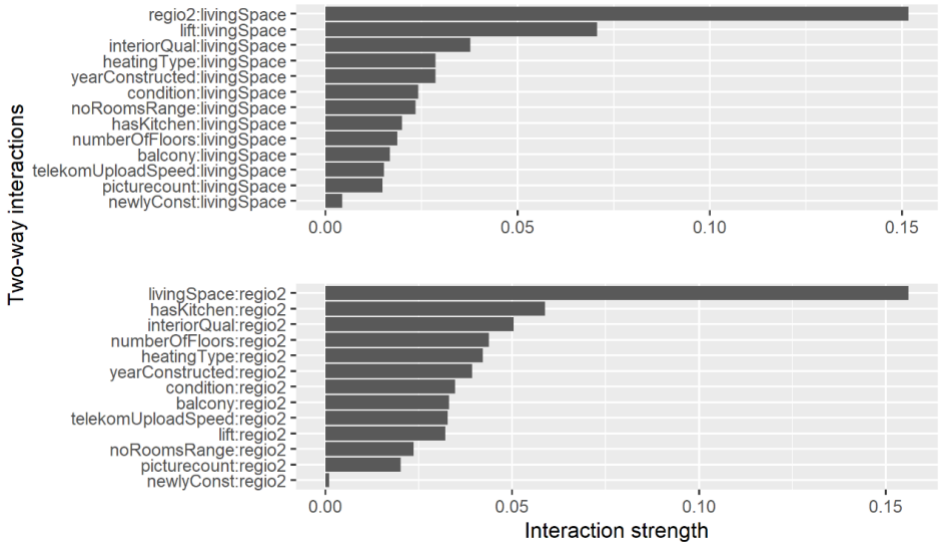


Figure 0.9: The two-way H-Statistics of livingSpace and regio2.

The strongest interaction is between *livingSpace* and *regio2*. Since the multicollinearity tests revealed a week degree of association between these variables, we can safely study their interaction by using a two-way PDP.

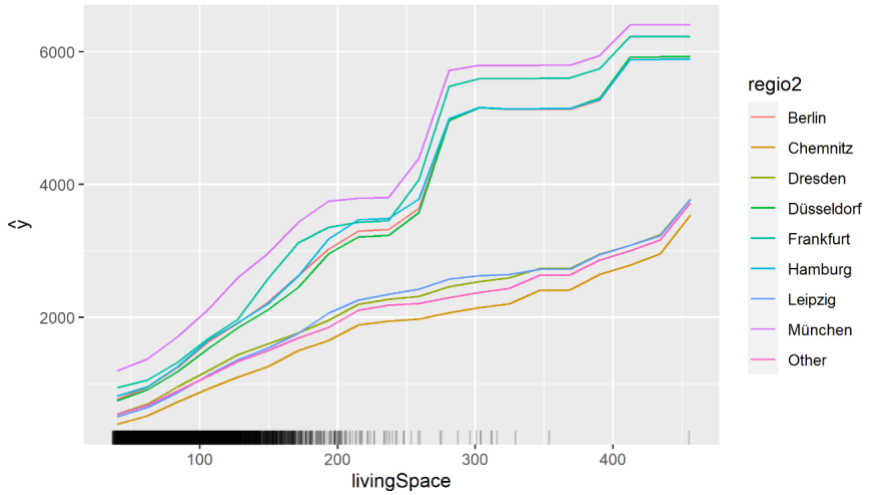


Figure 0.10: Two-Way PDP of regio2 and livingSpace's effect on totalRent. We chose the nine most frequent cities as it would be impractical to visualize all 51 categories. 20 Quantiles were chosen as the number of intervals for livingSpace.

The effects of *livingSpace* in Berlin, Düsseldorf, München, Hamburg, and Frankfürt are very similar, and they are not linear. They are close to linear up to *livingSpace* of approximately 250 M^2. From this point, the effect of increasing *livingSpace* is a lot stronger at about 300 M^2. However, we should be careful in interpreting the effects from this range because there are a lot

fewer observations in it compared to the rest of the distribution. In the other cities, the effects are simpler. They are close to linear across the entire distribution of *livingSpace*. The model clearly predicts a higher *totalRent* for big apartments in the biggest cities compared to big apartments in smaller cities.

Other strong interactions are between *livingSpace* and *lift* and between *livingSpace* and *interiorQual*. The ANOVA tests showed that *livingSpace* is only weakly associated with *lift* and *interiorQual*, which means that we can use PDP to study these interactions as well.

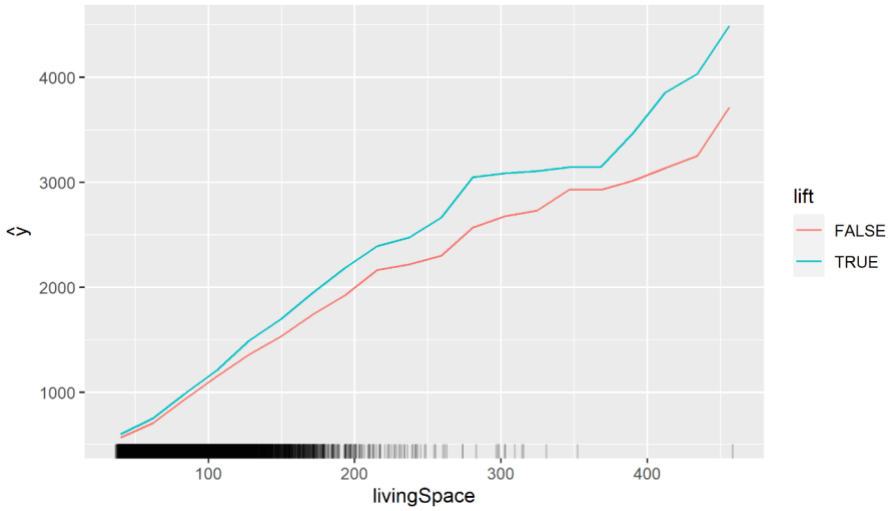


Figure 0.11: Two-way PDP of lift and livingSpace effect on totalRent. 20 Quantiles were chosen as the number of intervals for livingSpace.

The model predicts a higher *totalRent* for apartments in a building with a *lift*, and this difference increases with the size of *livingSpace*. A possible explanation for this is that big apartments in a building that are tall enough to have a lift might be luxurious penthouses at the top of tall buildings.

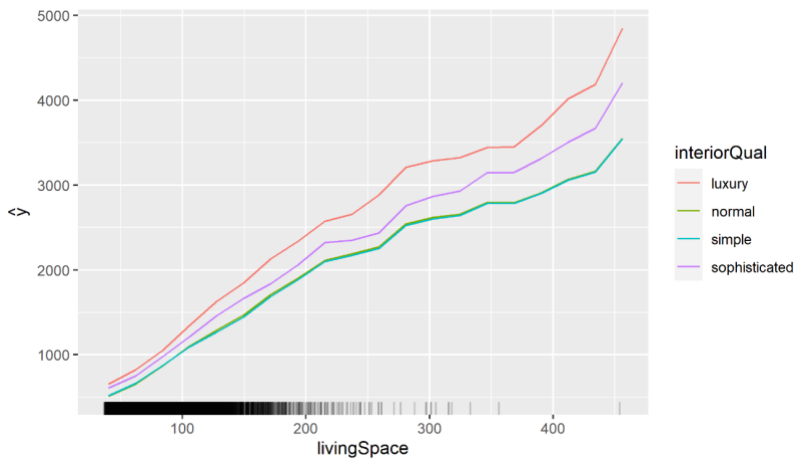


Figure 0.12: Two-way PDP of interiorQual and livingSpace's effect on totalRent. 20 Quantiles were chosen as the number of intervals for livingSpace.

totalRent as a function of *livingSpace* is similar for all the categories of *interiorQual*. However, apartments with the interior quality described as "luxury" have the most expensive rents followed by "sophisticated". The descriptions "simple" and "normal" have a very similar effect on *totalRent*.

3.4.4 LIME

The methods discussed above are good to derive conclusions from the global interpretability perspective. Global interpretability helps us to understand how the features influence the target variable, what kind of potential interactions exist between features, etc. But in the further analysis, we intended to explain the predictions of some interesting individual observations. In such cases, we use local surrogate models such as LIME or Shapley values.

In order to analyze the LIME model, we first run predictions on the test set, using random forest. After we get predictions, we select the highest and lowest predictions for analysis. To build the LIME model, we specify the data to use (training data), the model to use (random forest), and the number of bins to classify continuous variables. Also, this part required us to do a lot of testing with different parameters. The most important two parameters are kernel width and distance function. As default, the LIME algorithm uses 0.75 as kernel width and Gower's distance as a distance function to calculate the distance to the permutation. Gower's distance is calculated as

the mean of the partial dissimilarities across individuals, and the general equation of the coefficient is:

$$D_{Gower}(x_1, x_2) = 1 - (\frac{1}{p}\sum_{j=1}^{p} s_j(x_1 x_2))$$

where, $s_j(x_1 x_2)$ is the partial similarity function (Anand, 2020). Another distance function is the Manhattan distance which calculates the distance between two real-valued vectors by the sum of the absolute differences between the two vectors (Craw,2011).

So, in order to achieve the best possible fit, we have tested different hyperparameters of LIME, including kernel widths, distance functions, and type of model to fit locally. After a couple of tests, distance function= Manhattan, kernel width=0.75, and local model= LASSO were determined as the best parameter choices.

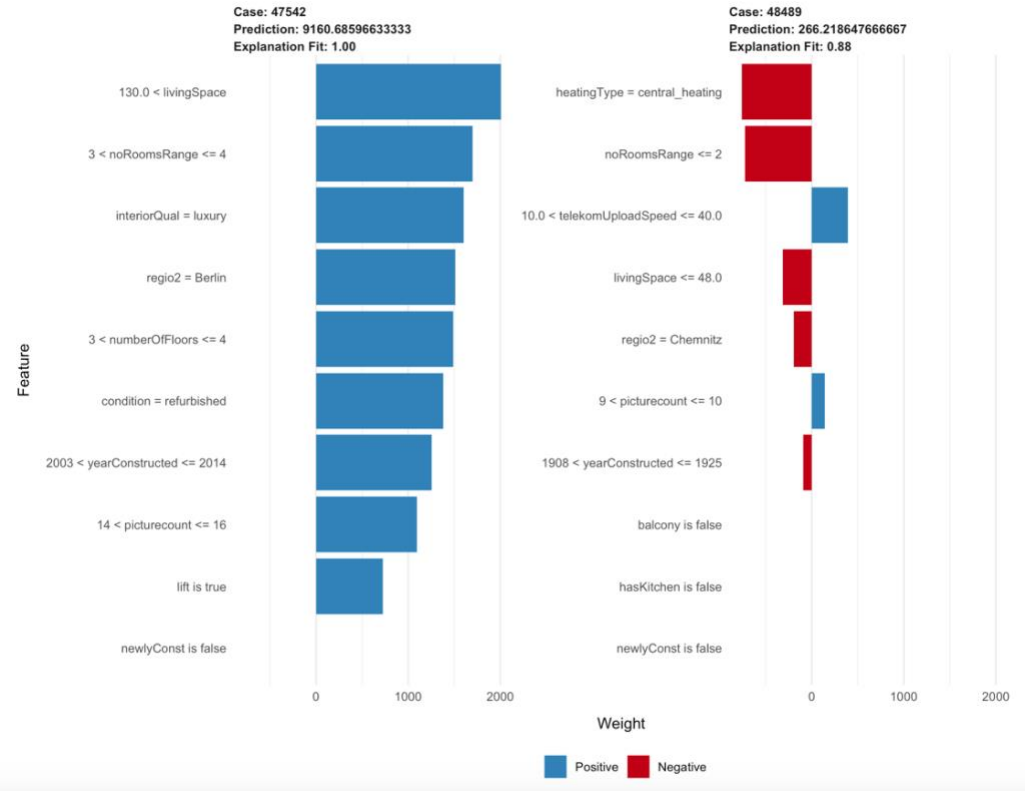
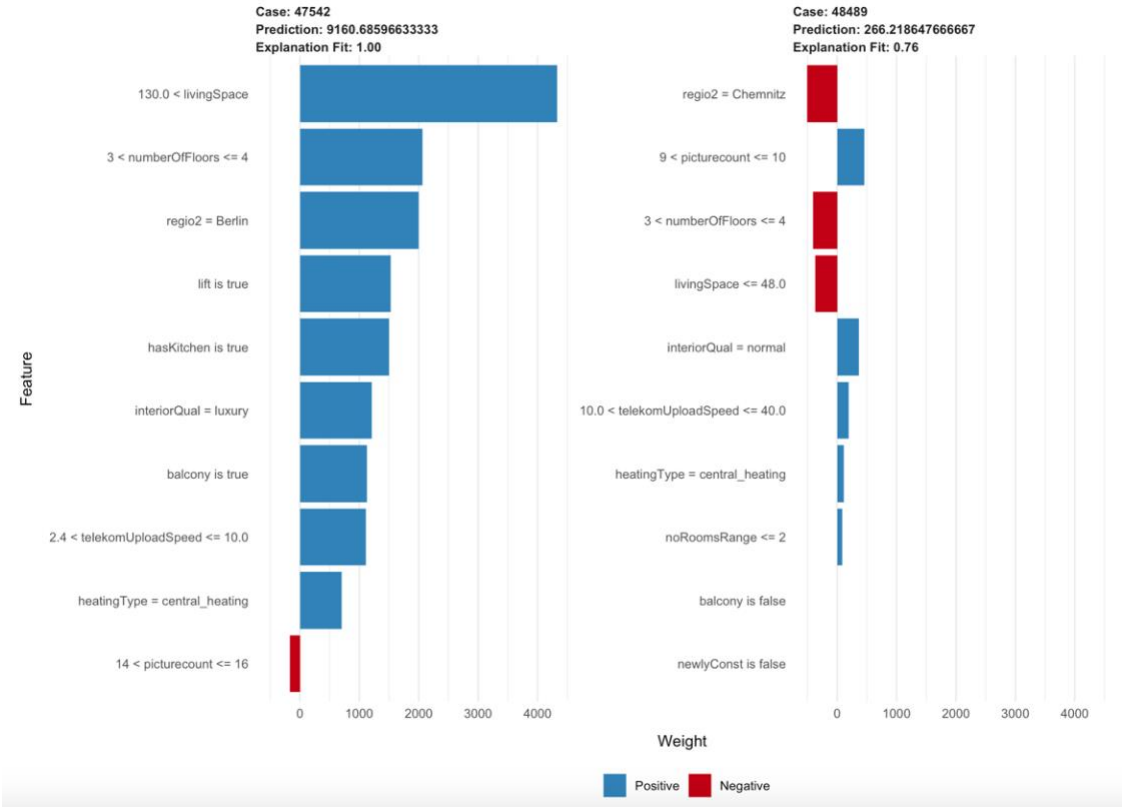


Figure 0.13: LIME estimates with distance function=Manhattan and kernel width=0.75

Results of this run are shown in Figure 3.13; it explains why these observations were predicted so far from the mean prediction. As we can see, the explanation fits for both cases are nearly perfect. Now we can clearly observe which factors influenced the target variable and in which direction. Also, we achieved some interesting results, which differ from those we got in global methods. For instance, *condition=refurbished* and *yearConstructed* are actually important features in explaining the highest observation, but those features had very little predictive power according to permutation feature importance. Moreover, *heatingType* and *noRoomsRange* are the best explanatory features for the lowest observation according to the LIME method. Those features also did not have any predictive power according to permutation feature importance.

Although the LIME algorithm gives a pretty good understanding of important features on a local scale, explanations generated by LIME are not stable. Each time the algorithm was run, we got slightly different results in terms of feature importances. For example, Figure 3.15 demonstrates this instability.



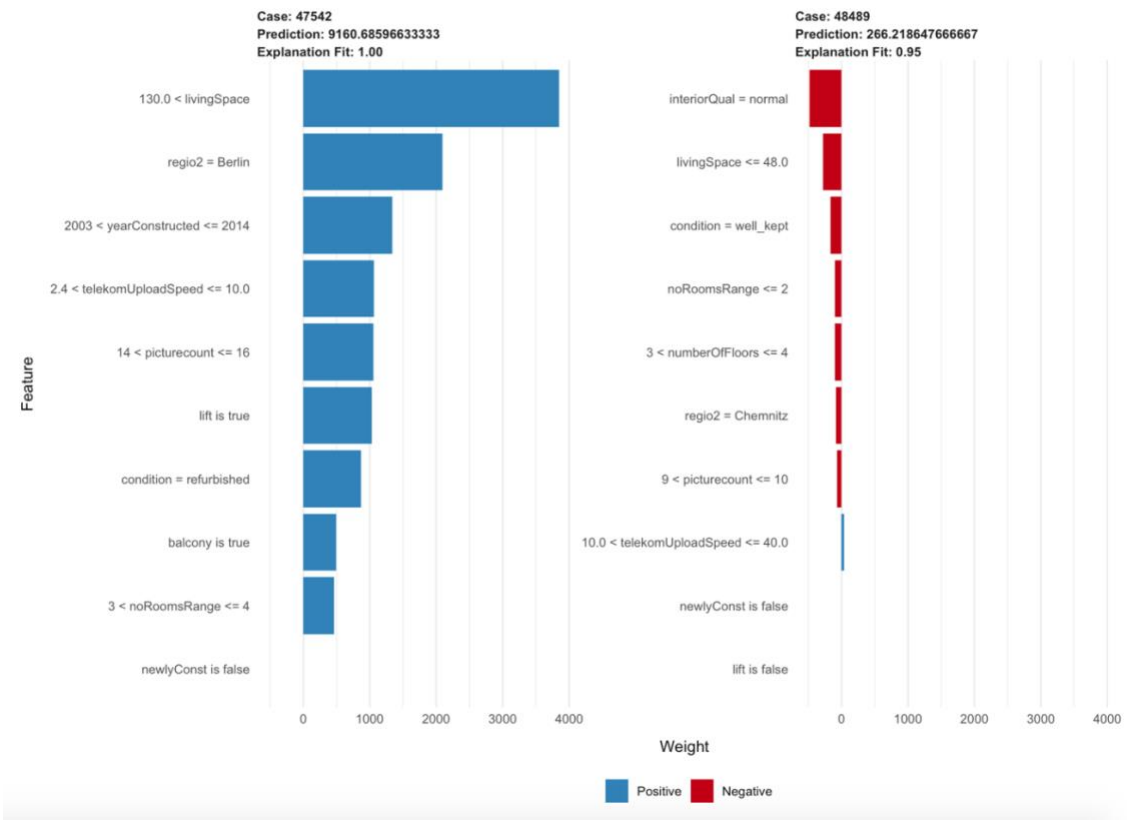


Figure 0.14: Two different LIME explanations for the same observations.

Both graphs are explanations generated by the LIME algorithm for the same two observations, but we can observe some features being placed differently in the importance list. This is caused by the nature of the LIME algorithm, as it uses random perturbing of observations each time. In the next section, we will try to overcome this problem by applying KNN instead of random permuting.

3.4.5 LIME with KNN

When using KNN to find observations similar to the minimum and maximum predictions from the random forest, we experimented with the values of K and found that using the 1000 nearest neighbours resulted in a good model fit for the LASSO in both cases. In Figures 3.16 and 3.17,

the contribution of each feature is the model coefficient multiplied by the feature value for this observation.

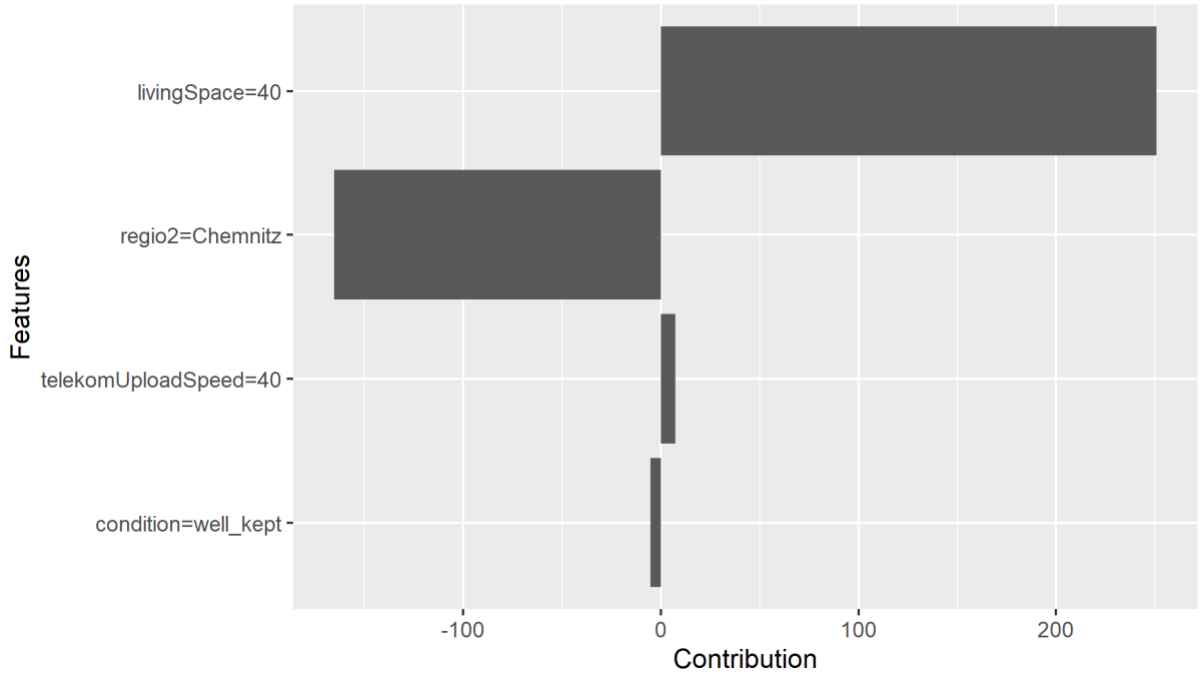


Figure 0.15: The features with the most significant contributions in the LASSO model's local approximation of the minimum prediction from the random forest.

The LASSO for the minimum prediction got an R^2 of 0.74 and predicted *totalRent* of 286, which is not far from the random forest prediction, which was 266. Therefore, the model fits the local data reasonably well and is a good approximation in this instance. *livingSpace=40* added approximately 250 to the prediction. On the other hand, *regio2=Chemnitz* had a significant negative impact of about 150. This effect matches Chemnitz effects on the global scale, as seen in Figure 3.6. *telekomUploadSpeed=40* and *condition=well_kept* had some minor contributions to the prediction.

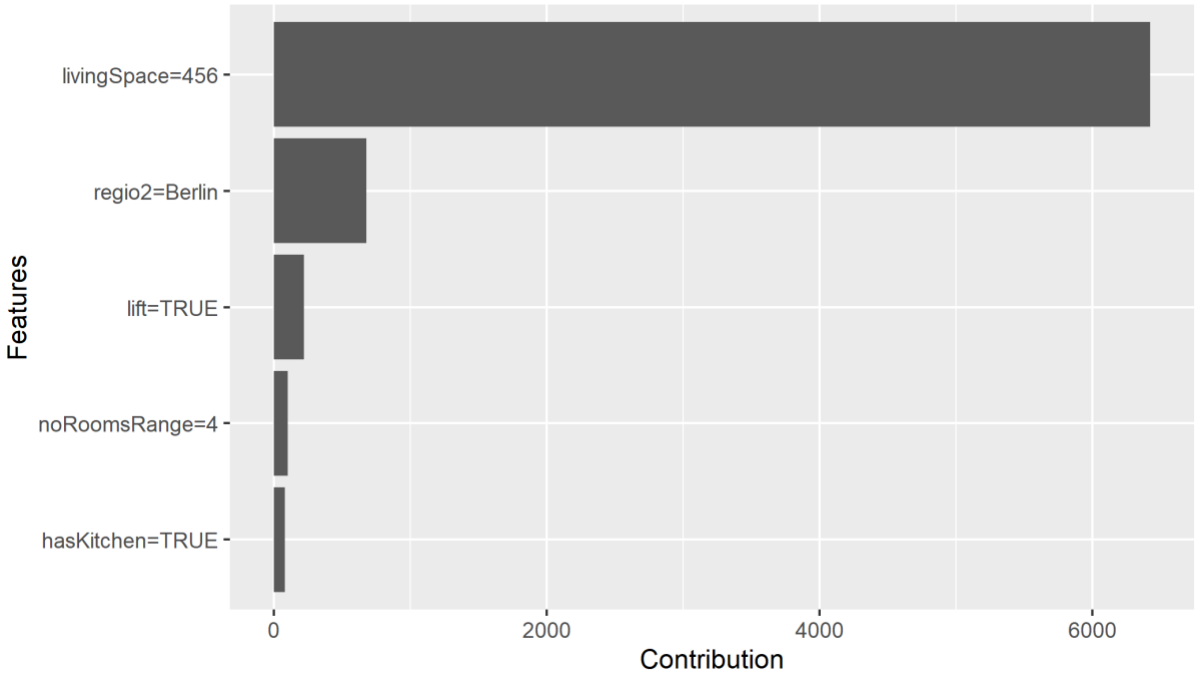


Figure 0.16: The features with the most significant contributions in the LASSO model's local approximation of the maximum prediction from the random forest.

The LASSO for the maximum prediction got an R^2 of 0.85 and predicted *totalRent* of 7411, which is a bit off from the random forest prediction, which was 9160. It fits the local data better than in the case above but is not as good at approximating the prediction of interest. *livingSpace=456* added more than 6000 to the prediction, and *regio2=Berlin* added 650. The local effects of Berlin also match the global effects in figure 3.6. *lift=True*, *noRoomsRange=4*, and *hasKitchen=TRUE* had relatively minor positive effects on the prediction.

3.4.6 SHAPLEY VALUES

Shapley Values is the next step to understanding local model behavior. We try to understand the influence of individual features and their coalitions on the final prediction. As a reference point, we have used the mean value of predictions which is *totalRent=921*, and try to analyze what are

the main drivers of deviation of lowest and highest predictions. As we can see from Figure 3.18, the most influential feature of the highest rent is the *living space*. The influence level significantly drops for *interiorQual=luxury* and *regio2=Berlin.* The last driver of the deviation is *lift=TRUE,* and the other features have little to no influence on the prediction.

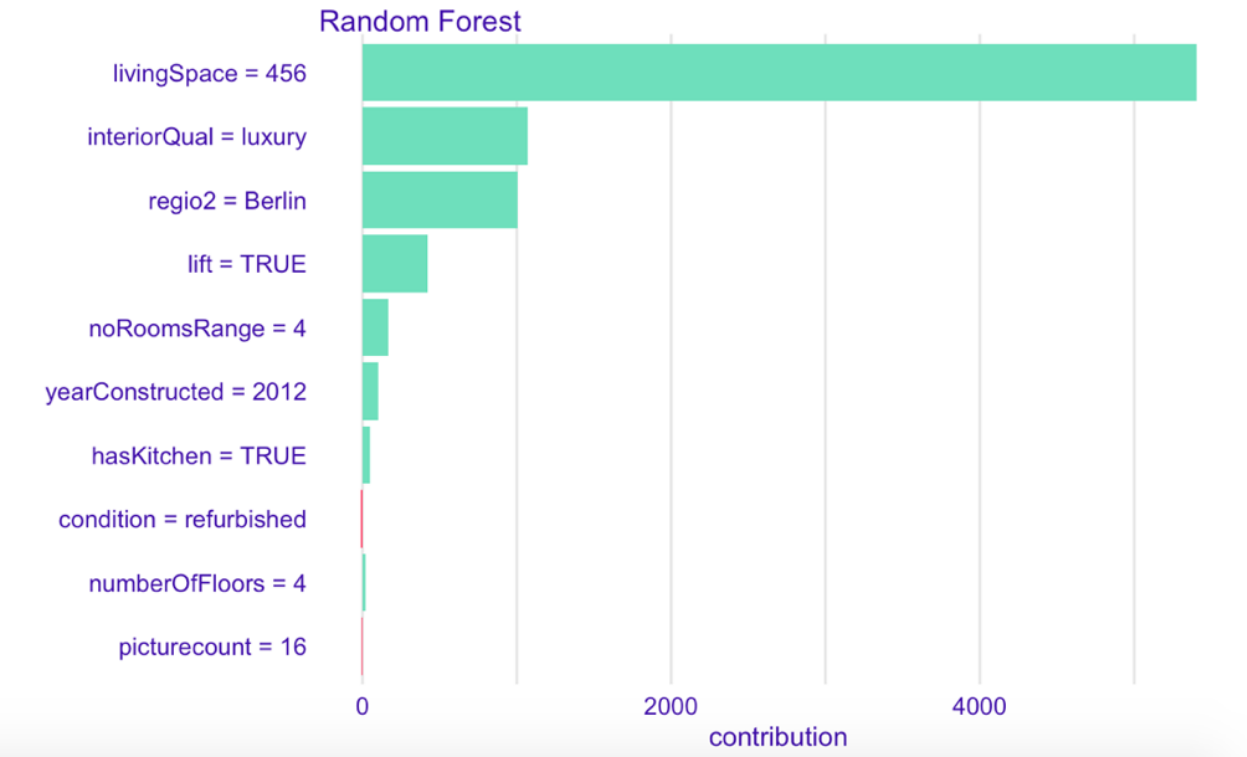


Figure 0.17: Shapley Values for highest prediction.

Moreover, Shapley Values for the lowest prediction are shown in Figure 3.19. The main driver of the difference between the lowest value and the mean value is *livingSpace* once more. But this time second most influential feature is *regio2=Chemnitz,* followed by *interiorQual=normal*. Other features have very low influence.

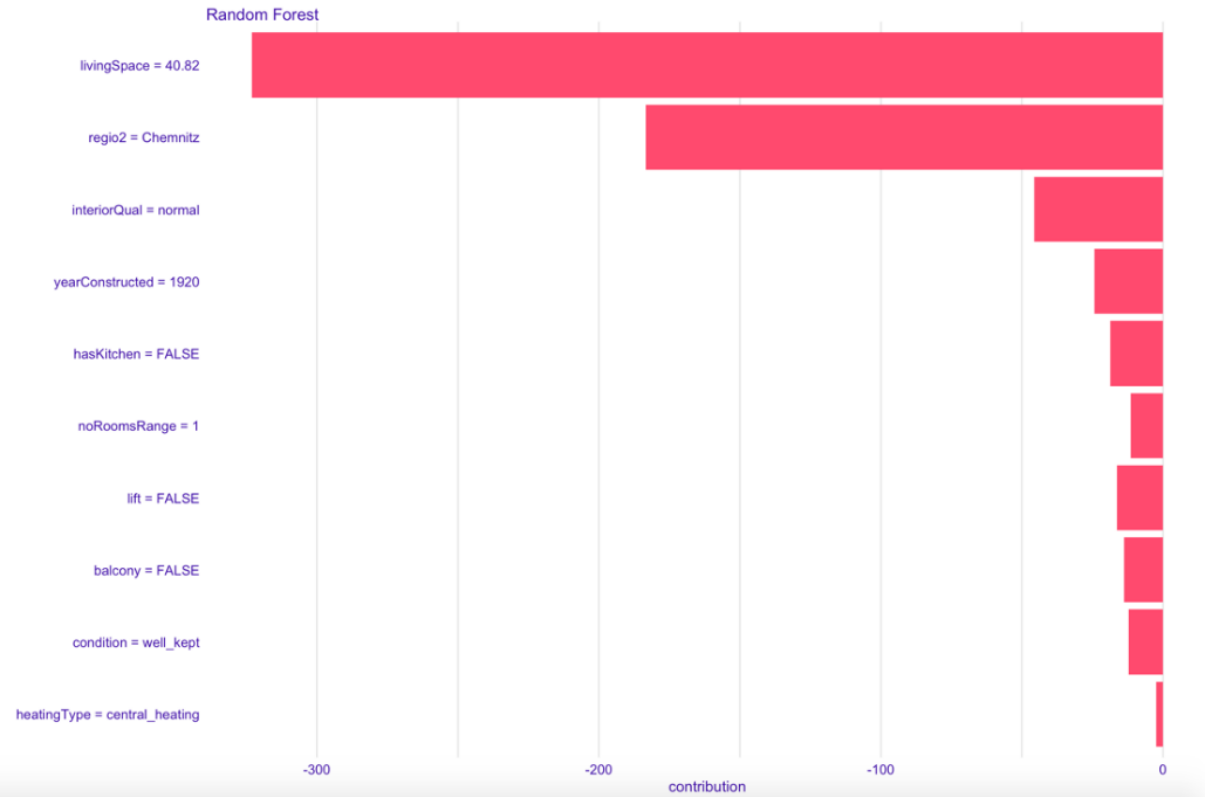


Figure 0.18: Shapley Values for lowest prediction.

As we can see, there are some similarities and differences between LIME and Shapley results. Some significant differences are, while the LIME algorithm values almost all features being important in the highest prediction case, Shapley results yield only four features being most influential. In the lowest prediction case, the LIME algorithm puts *heatingType=central_heating* as the most negative influencer, but the Shapley method puts this feature as the least important.

4. Discussion and Conclusion

The high accuracy achieved by black-box models comes at the cost of less interpretability. We cannot fully understand how a random forest consisting of hundreds of deep decision trees makes a prediction, but the methodology section showed that there are methods for gaining insight into the mechanisms of such a model to make it more interpretable. These methods present these insights in a visual and intuitive way, both at the global and the local level.

At the global level, they can show which features are most important in the model, the effect of each feature, and how different features interact.

In our application on the German Real Estate data, the global IML methods revealed some mechanisms that would be expected in a model that predicts monthly rent. Naturally, the prediction of rent increases with the size of the apartments. An apartment being in one of the biggest cities generally increases the prediction, especially for large apartments. Other mechanisms that were identified are that better interior quality is associated with higher predictions, and large apartments in buildings with lift generally result in higher predictions than large apartments in buildings without lifts.

After studying and applying these global methods, we are left with the impression that they have several good qualities. One of them is that they complement each other. For instance, the H-Statistic shows the interaction strength between two variables but not how the effect works. Introducing the Two-way PDP complements the H-Statistic by showing how this effect works, thereby telling a more complete story of the interaction. Another good quality of the methods is that they present the insights in an easily understandable way. This can be very useful when non-technical people require explanations of ML applications.

Even though the global IML techniques are useful in many ways, they still have a few limitations. The visual way of presenting the effects makes them easy to understand, but it also limits how interaction effects can be presented. For instance, we cannot use the methods to present interaction effects between three or more variables which makes it hard to understand the more complex mechanisms.

In the case of global methods, we have seen that some features such as apartment size, the region where the apartment is located, or interior quality are generally the main drivers of prediction. But in the analysis with local surrogate models, we have noticed these features are not necessarily the most important ones for each individual prediction. In the closer examination of lowest and highest rent apartments, we have seen features such as year of construction, condition, etc., are having stronger effects. This indicates that the general effects and the most important features on the global scale do not necessarily apply to the same degree on a local scale. When the global effects do not always generalize well to every prediction, these local methods are needed.

After implementing the LIME algorithm and seeing the instability of the explanations, we tried to solve this problem by sampling with the KNN method. This came at the cost of a slightly worse explanation fit; it was reduced from 1 to 0.85 for the maximum prediction and from 0.88 to 0.74 for the minimum prediction. In this alternative implementation of the LIME, the local effects were more similar to the global effects, as opposed to the normal implementation of LIME and the Shapley Values. But we have achieved stable results with LIME using KNN as opposed to results generated by LIME's own random perturbation.

Although the LIME method is a creative idea that can provide interesting explanations, we still think that Shapley Values fit the purpose of interpreting black box models better. This method does not suffer from any instability issues, which makes it a lot more reliable and trustworthy for those who require local explanations.

4.1 Suggestions for Further Research

As discussed above, the global methods are not practical for interaction effects between three or more features. Methods that can explain more complicated interactions would be very useful in the cases where simpler explanations are considered incomplete or insufficient. Consequently, we think that research into such methods could be beneficial for the field of interpretable machine learning.

This thesis has shown both the potential and the limitations of LIME. Solving the instability problem without losing accuracy would make LIME a trustworthy alternative to Shapley Values, which can be useful in the cases where they are too computationally expensive or time-consuming. Therefore, further developing the LIME algorithm could potentially be very productive.

Finally, there is a lot more to learn about ML models for real estate predictions. Potentially significant effects could be discovered by including more variables in the dataset. For instance, macroeconomic variables and more detailed geographic variables could have significant effects on prices that could be discovered by IML methods.

References

Ahmad, M. A., Eckert, C., Teredesai, A., & McKelvey, G. (2018) *Interpretable Machine Learning in Healthcare*. Intelligent Informatics Bulletin. http://www.comp.hkbu.edu.hk/~cib/2018/Aug/article1/iib_vol19no1_article1.pdf

Ariza-Garzón, M. J., Segovia-Vargas, M.-J., Caparrini, A., & Arroyo, J. (2020). *Explainability of a Machine Learning Granting Scoring Model in Peer-to-Peer Lending*. IEEE Xplore. https://ieeexplore.ieee.org/document/9050779?denied=

Anand, D. (2020, June 19). *Gower's Distance*. Medium. https://medium.com/analytics-vidhya/gowers-distance-899f9c4bd553.

Babel, B., Buehler, K., Pivonka, A., Richardson, B., & Waldron, D. (2019). *Derisking machine learning and artificial intelligence*. McKinsey & Company. https://www.mckinsey.com/business-functions/risk/our-insights/derisking-machine-learning-and-artificial-intelligence.

Bakhshi, A. K., & Ahmed, M. M. (2020). *Utilizing BlackBox Visualization Tools to Interpret Non-Parametric Real-Time Risk Assessment Models*. Transportmetrica A: Transport Science. https://doi.org/10.1080/23249935.2020.1810169

Breiman, L. (2001). *Random Forests*. Machine Learning. https://link.springer.com/article/10.1023/A:1010933404324

Burzykowski, T., & Biecek, P. (2020). *Explanatory Model Analysis*. 9 Local Interpretable Model-agnostic Explanations (LIME). https://ema.drwhy.ai/LIME.html

Craw, S. (2011). *Encyclopedia of Machine Learning*. https://doi.org/10.1007/978-0-387-30164-8_506.

Fisher, A., Rudin, C., & Dominici, F. (2019). *All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously*. arXiv.org. https://arxiv.org/abs/1801.01489.

Fiori, L. (2020). *Distance metrics and K-Nearest Neighbor (KNN)*. Medium. https://medium.com/@luigi.fiori.lf0303/distance-metrics-and-k-nearest-neighbor-knn-1b840969c0f4.

Gómez-Ramírez, J., Ávila-Villanueva, M., & Fernández-Blázquez, M. Á. (2020, November 26). *Selecting the most important self-assessed features for predicting conversion to mild*

cognitive impairment with random forest and permutation-based methods. Nature News. https://www.nature.com/articles/s41598-020-77296-4#citeas.

Greenwell, B. & Boehme, B. (2020). *Hands-On Machine Learning with R*. Github Sites. https://bradleyboehmke.github.io/HOML/

IBM Cloud Education. (n.d.). *What is Random Forest?* IBM. https://www.ibm.com/cloud/learn/random-forest

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning: with applications in R*. Springer.

Katuwal, G. J., & Chen, R. (1970). *Machine Learning Model Interpretability for Precision Medicine: Semantic Scholar*. https://www.semanticscholar.org/paper/Machine-Learning-Model-Interpretability-for-Katuwal Chen/707095416e8d814c08a2f33e7533ddaefdf4f338.

Kim, T. K. (2017). *Understanding one-way ANOVA using conceptual figures*. Korean journal of anesthesiology. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5296382/.

Kim, E. J., Kim, Y., & Kim, D. K. (2020) *Interpretable machine-learning models for estimating trip purpose in smart card data*. Proceedings of the Institution of Civil Engineers – Municipal Engineer. https://doi.org/10.1680/jmuen.20.00003

Kuhn, M. (2019). *The caret Package*. Github Sites. https://topepo.github.io/caret/

Medium. (2018). *An overview of correlation measures between categorical and continuous variables*. https://medium.com/@outside2SDs/an-overview-of-correlation-measures-between-categorical-and-continuous-variables-4c7f85610365.

Molnar, C. (2021). *Interpretable Machine Learning*. Github Sites. https://christophm.github.io/interpretable-ml-book/.

Molnar, C., Casalicchio, G., & Bischl, B. (2020). *Interpretable Machine Learning -- A Brief History, State-of-the-Art and Challenges*. Department of Statistics, LMU Munich. https://arxiv.org/abs/2010.09337.

Nicholson Price II, W. (2020). *Risks and remedies for artificial intelligence in health care*. Brookings. https://www.brookings.edu/research/risks-and-remedies-for-artificial-intelligence-in-health-care/.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). *Model-Agnostic Interpretability of Machine Learning*. University of Washington Seattle. https://arxiv.org/abs/1606.05386.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. "Why Should I Trust You?" | Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. https://dl.acm.org/doi/10.1145/2939672.2939778

Richardson, J. T. E. (2011). *Eta squared and partial eta squared as measures of effect size in educational research*. Educational Research Review. https://www.sciencedirect.com/science/article/abs/pii/S1747938X11000029

Yiu, T. (2019). *Understanding Random Forest*. Medium. https://towardsdatascience.com/understanding-random-forest-58381e0602d2