AirBnB Dublin

Explainable Price Prediction using Deep Learning Models

Lars Knieper, Dominik Becker
Chair of Statistics, University of Göttingen

March 5, 2022

Abstract

This paper introduces different models to explain the price composition of *AirBnB* accommodations in Dublin using structured tabular data and unstructured image and text data. Interpretability of the models is achieved using SHAP Values, weight constraints and extensive data preprocessing. XGBoost and TabNet models are trained for the tabular and text data, and ResNet-based approaches are introduced for processing the image data. The best performance was achieved by an XGBoost model using the tabular and processed text data. In addition, an ensemble of a TabNet trained on tabular and text data and a ResNet approach trained on images is presented.

Keywords: Interpretability, TabNet, SHAP, PCA, Price, Prediction

Contents

List of Figures

List of Tables

1 Introduction

This chapter is divided into two parts. In the first part, the given *AirBnB* data on the city of Dublin are briefly presented. In the second part, the individual variable categories are examined in more detail and their processing is described further. This also includes variable generation via processing of existing variables whose format is not suitable for the immediate analysis.

1.1 Dataset Introduction

The available data covers listings on the *AirBnB* website for various cities and regions in the world and are publicly accessible via "http://insideairbnb.com/". This paper deals with the corresponding data for the city of Dublin in Ireland. The website basically offers four different types of data - general listing information (listings data in the following), reviews, calendar and neighborhood data. The listing data basically describes the data that can be accessed directly via the listing on "https://Airbnb.com/". This includes, for example, the price, information about the maximum number of guests (accommodates), the amenities of the listing, review scores and descriptions of the host, the accommodation or the neighborhood. In total, the dataset for Dublin includes 6976 listings with 74 different variables. As an example, the histograms for the variables "accommodates" and "room type" are shown in the upper section of Figure 6. It can be seen that listings with accommodation for two guests represent the modal class (total of 3213), while accommodation for large groups of people (>9) is less represented (total of 76). For the type of accommodation, there are about as many private rooms as there are entire homes or apartments offered. Only a few of the listings are shared rooms (total of 138) or hotel rooms (total of 54). The lower part of Figure 6 shows the histograms of the price in euros (left) and the logarithmized price in euros. If one considers the prices in euros, it is noticeable that these vary strongly. While the average value is about 107 euros (red dashed line), there are also some listings that cost more than 400 euros per night. By logarithmizing, this strong scatter can be corrected and a distribution is obtained that can be approximated by a normal distribution with an expected value of about 4.46 log euros.

The review data includes all text reviews that were submitted by guests for the respective listing.

Reviews and listings data are linked by a listings ID. Thus, the data for each individual listing can be aggregated and evaluated from the listings and review data. The calendar data and neighborhood data are not considered for further analysis.

1.2 Data Preprocessing

Before variables can be selected and models trained, it is necessary to bring all variables into a consistent form. For the individual listings, this means excluding such observations that do not belong to the population of apartments for rent on *AirBnB* in Dublin. These include hotel and hostel rooms. Moreover, there are some variables that have a lot of Not Available (NA) values. The NA values per variable are shown in Table 8.

To circumvent computational problems with these NA values, different imputation methods were used. The variables with more than 4000 NA values were dropped. Especially for text variables, texts from variables holding a similar content were used for imputation, like imputed "neighbourhood_cleansed" in "neighborhood_overview". For the different review scores random numbers were drawn from a manipulated halfnormal distribution and for "beds" and "bedrooms" simple linear regressions were used utilizing the variable "accommodates" as independent variable. Further NA values arose by the data generation processes and were imputed as well. An overview of the different imputation methods can be found in the Table 11.

Once all NA values are cleaned up, the variable processing can be carried out. At first, some variables such as "host_id" or "scrape_id" have no meaningful relationship to the price of the accommodation and will therefore be excluded from the beginning. The remaining variables or in other words the available information can be divided into five broad categories:

1. Metric variables (for example, various review scores)
2. Categorical variables (for example, the type of accommodation)
3. Text variables (for example, the description of the accommodation and the reviews)
4. Spatial (accessible via latitude, longitude)
5. Images (accessible via the URL of the listing)

Metric and categorical variables The metric variables are already in an appropriate format to be used in a price estimation model. It should be mentioned that the date variables, such as the time of the last review, have been transformed into metric variables taking the time difference to the scraping date of the data. Categorical variables, on the other hand, cannot be used for every pricing model without further preparation. Many of the variables can be translated into quantitative variables via simple one-hot encoding, where a new variable is generated for each category of a variable, indicating the presence of the category for the subject of the sample with a 1 and the absence with a 0. An example of a categorical variable that shouldn't be represented using one-hot encoding alone is "bathrooms_text". This variable has 29 different values, each indicating how many bathrooms the accommodation has (including half bathrooms) and whether these are shared with other people, private or conventional bathrooms (presumably accessible via a corridor only for the tenants). Thus, there are expressions like "1.5 shared baths" or "shared halfbath". There are two types of information in this variable - the number of bathrooms and the privacy of their use. Therefore, the variable was divided in two separate variables. One that indicates the number of baths (0, 1, 2, 3, more than 3) and one that describes the privacy (shared, private, normal). Further manual variable transformations are summarized in Table 7.

The data also contains the "amenities" of the listing. This column lists different items which the accommodation has at its disposal. These items have a high level of detail, so that even the manufacturer of a shampoo, a soap or a household appliance is distinguished.

In total, the list of all amenities includes 520 different items. Heo and Hyun (2015) found that at least for hotels certain amenities increase the willingness of customers to pay a premium which underpins the potential explanatory power of these variables. To make these variables usable, they have been aggregated into several distinguishable groups to avoid the inclusion of a large number of one-hot-encoded variables. These groups describe, for example, whether the accommodation has a TV or a pool. But also more abstract variables like the availability of items for babies (crib, high chair, etc.) or the number of luxury items (sauna, bicycles, pool table, etc.) were generated. Variables that cannot be combined in a meaningful way are included as simple one-hot-encoded variables. 121 observations remained with missing values for several amenity variables, so random guesses from a binomial distribution with the frequency of each variable as the probability were drawn.

Text variables Several variables containing verbal information need further analysis in order to be used as a variable in the model. Those variables are "name", "host_name", "description", "neighborhood_overview", "host_about" and the "reviews" of each listing. Except for "host_name" the length of these variables, excluding spaces, were taken as variables as it might be a measure of effort put in the offered listing. The language of the reviews, "name", "description", "neighborhood_overview" and "host_about" was detected and results in a dummy variable "eng_host", which describes, whether the host is writing in English (1) or does not (0), and in the variable "prop_of_eng_reviews" (Proportion of English written reviews).

Inspired by several studies like Hinz and Auspurg (2017), that find serious evidence of discrimination based on the names in the housing market of Germany, a variable was used to control for such an effect. Here, the discrimination would come from less demand and would have a negative effect on the price for listings, where the name of the host is not common in the culture of the city, here Dublin. To determine this, the dataset Name-Census (2021) published on *kaggle.com* and a list of traditional irish names (Burch, 2021)[1] was used. The *Name Census* dataset contains the most popular 100 names of 226 countries given the country and continent, which were then sorted into cultural units according to Table 9. By this dummy variables with the prefix "host_name_sounds_" and ending with the corresponding cultural group were generated (frequencies in Table 10).

Especially for the reviews but also for the texts within "description", "neighborhood _overview" and "host_about" a Sentiment Analysis was conducted, which is part of the toolkit of Natural Language Processing. After inspecting the possible outcome scores of the packages Transformers, TextBlob and NLTK, the latter was chosen due to the supply of different scores for "negativity", "positivity", "neutrality" and a "compound". The NLTK-Package describes the "positive and negative opinions, emotions and evaluations" (Wilson et al., 2005, p. 1) of the analyzed text by a rule-based model called Valence Aware Dictionary for sEntiment Reasoning (VADER) (Hutto and Gilbert, 2014). The VADER method is based on a mixture of qualitative and quantitative methods outperforming several Machine Learning algorithms. The positive review scores (range from 0 to 5, Figure 7) are

[1]Examples for traditional irish names: Abban, Blinne, Conn, Echna, Life, Niall, Searc, Uaine

confirmed by the compound scores (range from -1 to 1, figure 9) as well as by the wordcloud (Figure 10). Large absolute compound values arise for reviews longer than one sentence, hence those reviews explain extensively about the stay. For illustration, exemplary excerpts from an extremely positive and extremely negative review can be seen in the Chapter A.1.1.

214934 reviews were analyzed by detecting the language, the length and the sentiments, which get grouped by the listings ID. Hence, this results in variables for the fraction of English reviews, a mean length and mean values of the different sentiment scores. Additionally, the fraction of negative compound scores per listing and the most negative as well as most positive compound score per listing were added as variables to potentially control for those outliers within each listing.

The same Sentiment Analyzer was used for the other text variables, except for the *name* variable, where the texts were too short for the analyzer to result in a sensible value. In the case of those other text variables, the value is representing mainly to what extent a positive mood was created in the description. For example a neutral descriptions like "the house is next to an old burial ground the suicide plot which was the inspiration for Bram Stoker's Dracula." receives a negative compound value. The same holds for both other variables "host_about" and "description". The distribution of those scores can be seen in Figure 11.

Spatial In addition to the neighborhood variable, the longitude and latitude are supplied as spatial variables. As a first naive approach a Gaussian Process Regression with a Radial Basis Function (RBF) kernel, a product of a RBF and dot product kernel and constant kernel were fitted and optimized using the Python module Scikit Learn (Schulz et al., 2018). This regression didn't show any effect of the distance to the city center of Dublin on the logarithmized price per accommodate (Figure 13). Hence, Open Street Map data was chosen to describe a good location using the OSMnx module (Boeing, 2917).

OpenStreetMap is a collaborative project since 2004 containing freely available geo data, which, among other things, can be used to generate commercial maps (FOSSGIS e.V., 2004). Figure 15 confirms that there is no clearly discernible effect of distance to city center on the log-price per accommodate. Additionally, it shows, that there are still listings within a 25km radius around the city center. For the area under consideration, a wide variety of objects and their coordinates were then searched for, which OSM can identify. Corresponding frequencies can be seen in Figure 16. Then, for each listing and after adjusting the coordinates for the curvature of the earth's surface, the number of objects within a 1km radius were counted and included as a new variables. In addition, the distances to each of Tripadvisor's Top10[2] attractions[3] and to the travel options[4] were measured. Here, however, a pre-selection was made directly by including only the distances to the three closest sights, the distance to the closest travel option, the mean distance to all sights and the mean distance to the travel options as variables (Frequencies in Figure 17).

[2] `https://www.tripadvisor.com/Attractions-g186605-Activities-Dublin_County_Dublin.html`

[3] The distance to the famous *Temple Bar*, which is suprisingly not one of Tripadvisor's Top10, got included as well.

[4] airport, train station and port

Images The variable generation process for 10 further variables of the image data via the URL of the listings is described in detail in Chapters 2 and 3, since a separate model was set up to extract these variables. The results of this model are the number of images for specific room types and technical measures like the brightness and contrast of the images.

Data Split The remaining 6633 observations were then divided by the common 80:20 ratio into training and test data, with the training data divided again into training and validation data by the same ratio (Table 1).

Training	**Validation**	**Test**
4250	1062	1321

Table 1: Number of Observations after split into Training, Validation and Test Data.

2 Methods

This chapter describes the theoretical methods and their embedding in the models used. First, methods are presented that aim to select a pool of variables that can be ranked by their importance for the model using SHapley Additive exPlanations (SHAP values) (Lundberg and Lee, 2017). Then, the models themselves and their modular components, especially for neural networks, are firmly introduced.

2.1 Variable Selection Methods

By introducing many one-hot-encoded variables, the number of variables has greatly increased. Each of these binary variables can be described as the presence or absence of a category. In order to keep only those variables for which a significant difference in the price level can be assumed, a Welch test (Welch, 1947) was performed for all binary variables. This excluded eight variables[5] due to non-significant results (5% significance level). Additionally, variables with a variance close to zero got dropped. Hence, 92 metric and 57 binary potential features, including image variables, are obtained.

SHAP Values To estimate the importance of the variables for the later models, an approximation of Shapley Values (Nowak and Radzik, 1994) is used. The idea behind Shapley Values originates from game theory and aims to reward all participating parties involved in a project in the fairest possible way based on their contribution to the project outcome. It is particularly important that Shapley Values take into account cooperations among the individual parties and that values of individual parties can be interpreted additively. If one is interested in the influence of parties A and B, the Shapley Values of these parties can simply be added together. Also, those parties that have no part in the success of the project

[5] "host_is_superhost", "Exta_pillows_and_blankets", "Luggage_dropoff_allowed", "Free_parking", "host_name_sounds_west", "host_name_sounds_rare", "host_location_country_Ireland", "neighbourhood_cleansed_Dn Laoghaire-Rathdown"

are assigned a Shapley Value of 0 (null player) while parties contributing in the same way get the exact same values. In addition, it is ensured that the sum of all Shapley Values at the end is exactly the profit of the project (efficiency) (Hart, 1989). In the case of a price prediction model, the price prediction for a single listing can be seen as a joint project of all available variables. The goal is to assign a high value to those variables that played a major role in making the predictions, while assigning a low value to more uninvolved variables. Shapley Values take into account the number of available variables as well as all possible combinations of them.

$$\phi_{ij} = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!} \cdot (f(h(z')_i) - f(h(z' \setminus j)_i)) \tag{1}$$

Equation 1 gives a formal definition for an approximation of the Shapley Values for predictive models introduced by Lundberg and Lee (2017) which will be refered to as SHAP Values in the following. Suppose there is an input vector x_i for a listing i with $1, ..., M$ variables and the goal is to identify the importance of the variable j for the model prediction of the listing $\hat{y}_i$. To do this, first a binary form of the input vector x' where a 0 indicates the absence and a 1 indicates the presence of the variable at the index is created. The function $h(x')_i = x_i$ maps the binary vector to its actual values, replacing all values at indices in z' with zero entries by their mean value across all observations. For all subsets $z' \subseteq x'$, i.e. all possible combinations of the indices in x'_i that are 1, the model prediction with $f(h(z')_i)$ and without $f(h(z' \setminus j)_i)$ the variable j is computed. The difference between these two predictions is now weighted using the possible combinations of variables in z' that are not equal to 0. The result ϕ_{ij} describes the deviation from the expected prediction for i if none of the variables for the observation i were given. In the context of the listings' log prices, this would be the mean of the log price. SHAP Values thus enable an interpretation of the variables for complex non-linear models, similar to the interpretation in simple linear models. The SHAP Values are also additive and can be aggregated, which will be of particular importance later on. In order to compute the SHAP Values for the predictive models the SHAP Python module (Lundberg et al., 2020) is utilized which uses computational more efficient approximations than the one stated in Equation 1.

The SHAP Values are subject to the basic assumption of independent variables. It is therefore necessary to test all remaining variables for strong correlation and resolve those correlations.

Correlation Adjustment The correlation of the numerical variables can be easily determined using the well-known Pearson correlation coefficient (Freedman et al., 2007). The Jaccard coefficient (Jaccard, 1912) is used to determine the relationship between two binary variables and the point-biserial coefficient (Tate, 1954) for the correlation between binary and numeric variables. The coefficients are explained in Section A.2.1. Since the direction of the correlation is not decisive for the analysis and a uniform measure for comparability of the correlations is needed, the absolute value of the respective coefficients is taken into account.

The individual coefficients can be represented as a symmetrical $M \times M$ matrix, with each row or column specifying the coefficients for one variable. This matrix is shown in Figure 18. It can be observed that several variables have a high correlation, e.g. the binary variables "Neighbourhood_cleansed_Dublin_city" and "in_city". In order to identify groups of variables that correlate strongly, the variables are clustered using Ward's linkage (Ward Jr, 1963). Ward's linkage method starts with one cluster for each variable. Individual clusters are then merged by a distance measure between the individual clusters which essentially computes by how much the error sum of squares (ESS) increase by combining two clusters X and Y compared to the ESS of those individual clusters.

$$D(X, Y) = ESS(XY) - (ESS(X) + ESS(Y))$$

For multidimensional vectors like in the present case (i.e. the correlation coefficients of each variable) the euclidean distance is used to compute the ESS for the cluster. The generated dendrogram is shown in Figure 1. In order to reduce the number of correlated variables a threshold was set at $D = 0.5$ (indicated by the red line). All clusters below that line had to be dissolved. There are two options for dissolving a cluster. One possibility is to keep only one variable from that cluster. As a result, some information would be lost and probably lead to a significantly poorer performance of the final model. The second option is to identify variables within one cluster that can be summarized to one broader topic. For instance the variables "availability_30", "availability_60", "availability_90" and "availability_365" which share the same cluster and all relate to the occupancy for the listing. These variables can be summarized into n_C independent variables using Principal Component Analysis (PCA) (Pearson (1901) and Hotelling (1933), see Section A.2.1 for Theory of PCA). The number of components n_C kept for each broad topic depends on the share of explained variance by the components. A threshold of 90% explained variance by each PCA was set. The respective number of components were included as new variables, while the variables used for the PCAs were dropped accordingly. In order to ensure a explanatory power later, in certain cases variables have been added to a PCA, which may have an $D > 0.5$ because of their thematic context. The performance of the different PCAs is described in Figure 19, while Table 12 gives an overview which variables have been combined on how many components n_C of a PCA. Due to the additivity of the SHAP Values, these can be summed up later for the different PCA variables and an interpretability is further given by the topic of the PCAs. However, this interpretability is restrictive in the sense that no statement can be made about the direction of dependence, i.e. whether a large feature value leads to a higher price. The manipulated PCAs only allow a statement as to whether the variable has an influence and how strong this influence is. The interpretation of the SHAP Values for the PCA variables will be discussed in more detail in Chapter 3

Having resolved strong correlations in the variables (see Figure 20 for updated dendrogram), 69 features remained for which SHAP Values can be calculated to identify the most important ones for the following models.

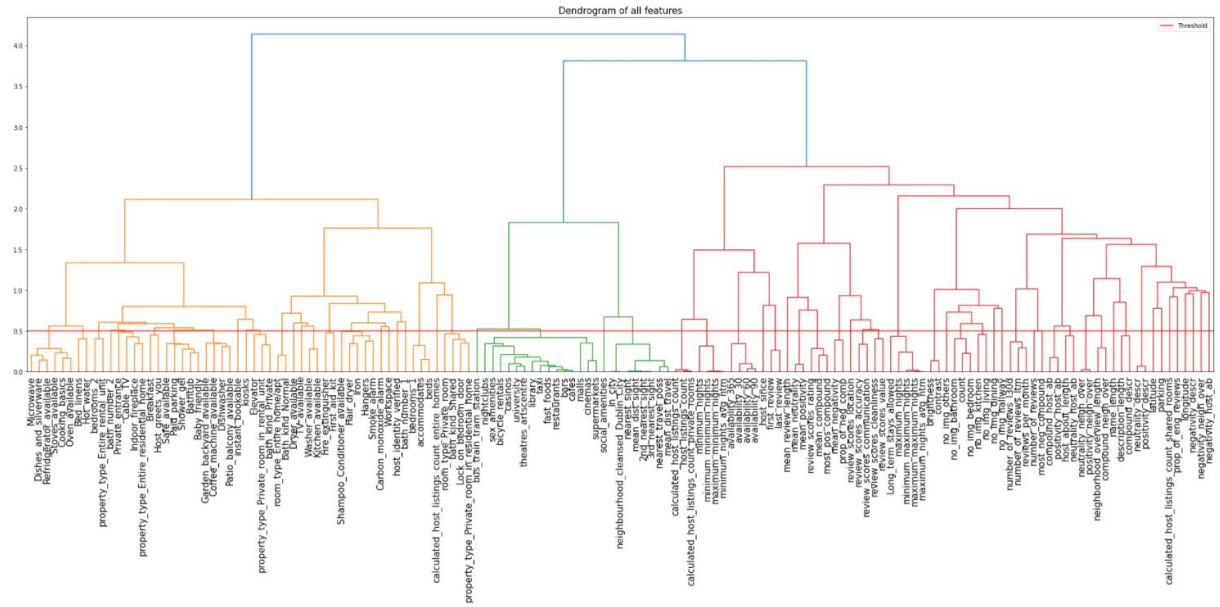


Figure 1: Dendrogram of Variable Correlation Clusters using Ward's Linkage

2.2 Models

In this section a methodological introduction to the models used for price prediction is given. The first model described, however, is the model used to generate the image variables mentioned in Section 1.2.

2.2.1 Room Classification

Most of the listings on *AirBnB* have various images showing the accommodation itself and the surrounding area. The images of the listings potentially also contain information about the price of a listing, which is why these were scraped with the Selenium (Huggins, 2008) and Beautiful Soup (Richardson, 2007) modules. An average listing in Dublin contains 12 images, the maximum number of images was found to be 86 and the minimum number 0. Since a correlation of the brightness and the contrast in images with the click behavior of consumers was found by Goswami et al. (2011), corresponding variables were included. The brightness was determined by the average pixel value after conversion to a grayscale image and the contrast by the Michaelson contrast (Michelson, 1927)[6].

The images also contain semantic information. In particular, the type of room or objects shown in the images of a listing could be a decisive factor for customers. Since the images do not have unique labels, recording the number of specific rooms in the images per listing is not readily possible. One way to label some of the images is to use the descriptions of the images to infer which room is shown in the image. To do this it was sought for specific keywords and word combinations in the descriptions which made an assignment of unique labels feasible. In addition phrases like "the view from the bedroom" were removed beforehand to avoid images of a landscape being labeled as "bedroom". Especially those

[6]quotient of maximum minus minimum pixel value and maximum plus minimum pixel value

images which would be assigned into two or more categories were problematic. To avoid a number of images to be labeled incorrectly, only those images that were uniquely assigned to one category were used. Six different categories for the rooms were chosen - bathroom, bedroom, kitchen, living room, dining room and hallway. In addition, a category "others" was introduced in which images that do not depict a room but, for example, facades, logos, parks or cityscapes were collected. This way a dataset with about 20000 labeled images, with 6237 bedrooms, 3269 kitchen, 1199 dining rooms, 2541 living rooms, 3330 bathrooms, 811 hallways and 4367 images in "others" was set up. Unfortunately, this way of labeling the data is error prone and there is an unobserved number of images which might not be labeled correctly. Nevertheless, on the first glance most of the images show the expected room. In the next step, this dataset was used to train a model which in turn labels the remaining images that could not be assigned to a category. First, a ResNet50 (He et al. (2016)) with pretrained weights on the ImageNet dataset (Deng et al., 2009) is used to obtain latent features of each labeled image. Then, using these features, a simple multi-layer perceptron (MLP) network (Rosenblatt, 1961) with two fully connected (FC) hidden layers, ReLU activation functions (Nair and Hinton, 2010), and an FC output layer reflecting the seven possible image categories was trained using a softmax cross entropy loss function. The regularizing components of the network are of special importance as the labels are error prone. Therefore the model needs to avoid a high degree of overfitting. Specifically, this can mean that the model doesn't make a prediction based on a single item/feature which it has been seen in a training image. For example in an overfitted model a training image of a bathroom showing a yellow duck that was incorrectly labeled by the image description as living room could result in a prediction of a living room for test images showing a yellow duck. Common generalization methods are batch normalization layer (Ioffe and Szegedy, 2015) and dropout layer (Srivastava et al., 2014). The motivation for batch normalization is to standardize the individual feature representations in the network. In doing so, the batch normalization layers parameterize estimates for the expectation and variance of the individual n_{BN} (mini-) batches. Using moving averages, these two parameters are trained over the model iterations. For $n_{BN} \to \infty$ a good approximation for the expectation and the variance of the feature representations is obtained. Although Ioffe and Szegedy (2015) claim that dropout layers would no longer be necessary when using batch normalization, it was used anyway to reduce the focus of the net on individual features of the ResNet50. The dropout layers result in a fraction p of the weights of the FC layers not being trained during the training process. Which weights are affected is determined by a random variable r which follows a Bernoulli distribution with its parameter $\pi = p$. For each weight w_{lj} and each (mini-) batch a value for r_{lj} is sampled. The optimization via backpropagation and stochastic gradient descent is then performed without the "switched off" weights.

2.2.2 Price Prediction

XGBoost Model XGBoost is a tree-based boosting algorithm proposed by Chen and Guestrin (2016). Just as in tree-based bagging approaches such as Random Forests (Ho, 1995), XGBoost is also based on individual decision trees. The difference, however, is that the individual decision trees in boosting don't predict the target value $\hat{y}$ itself, but use the

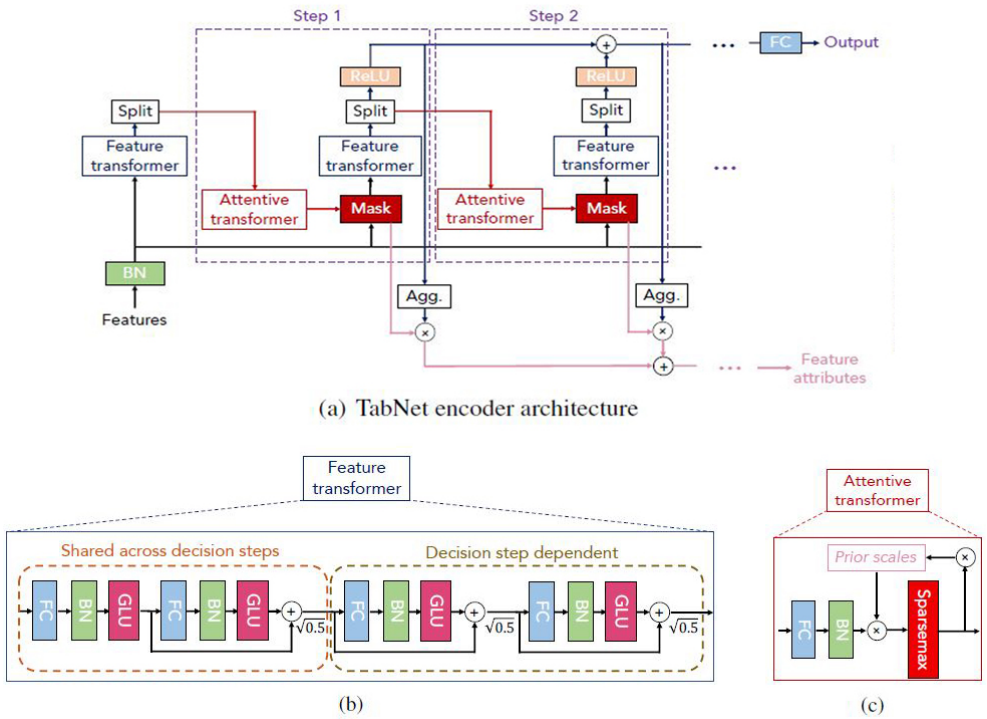


Figure 2: Architecture of the TabNet introduced by Sercan and Pfister (2019).

residuals from the previous decision tree $y - \hat{y}_n$. A firm introduction into the methodology of XGBoost is given in the Appendix A.2.2.

XGBoost models have already won numerous Kaggle competitions, which is why it is primarily considered here as a benchmark model for the neural network presented below.

TabNet Model Researchers, like Shwartz-Ziv and Armon (2021) and Borisov et al. (2021), are currently discussing the observation that neural networks are outperformed by algorithms such as XGBoost when analyzing tabular data. Therefore, developing Deep Learning approaches, that deal with this kind of data is a current field of research. The model called *TabNet* by Sercan and Pfister (2019) is mentioned frequently in this context because it brings the special features of simple data handling, possible interpretability and a working integration in Tensorflow (Abadi et al., 2015; Majumdar et al., 2019) as well as in PyTorch (Fischmann et al., 2020). The architecture of the TabNet can be seen in Figure 2. The features are batch normalized after their input, then get passed through a Feature Transformer and successive steps, which contain an Attentive Transformer, a mask and again a Feature Transformer. Finally the extracted features are obtained in a FC layer before the final dense layer for the output prediction.

The Feature Transformer, as described in (b) of Figure 2, consists of 4 layers, two of which are specific to the step, while the other two are shared with all decision steps. Each of those layers consists of a FC layer, a batch normalization, gated linear unit (GLU) nonlinearity and a normalized residual connection, which ensures that the variance does

not vary too much within the network. The GLU is inspired by Language Modeling: While controlling for the amount of information flown through, a higher accuracy and a faster convergence is achieved by them (Dauphlin et al., 2016). After the features D are processed through the first Feature Transformer, they are passed into an Attentive Transformer (c) before getting masked. The decision-step-specific mask $\mathbf{M}[\mathbf{i}] \in R^{BxD}$ softly selects features, where B is the batch size. This masks learns throughout the training and since the i-th decision step knows about the $(i-1)$-th step by the Attentive Transformer the model is more parameter efficient. Within the Attentive Transformer a FC layer is followed by batch normalization with the aggregated information about the previous usage of the features by a prior scale: $\mathbf{P}[\mathbf{i}] = \prod_{j=1}^{i}(\gamma - \mathbf{M}[\mathbf{j}])$, while $\mathbf{P}[\mathbf{0}]$ is initialized by 1s and not considered features have corresponding entries of 0. γ is a relaxation parameter, which controls for in how many steps a feature shall be considered. By a *sparsemax* activation function, which is similar to a *softmax* one but with more sparse probabilities (Martins and Astudillo, 2016), a sparsely selection of salient features is performed. Additionally, for further control of the sparsity a sparsity coefficient λ_{sparse} is added to the overall loss, which is helpful especially for datasets with many redundant variables. The splits seen in (a) are for splitting information for the output $\mathbf{d}[\mathbf{i}]$ and information to be processed further $\mathbf{a}[\mathbf{i}]$ in the upcoming step.

In the end of this process, which in fact is inspired by decision trees, the overall decision embedding is constructed by aggregation with help of the $ReLU$ activation function, $\mathbf{d_{out}} = \sum_{i=1}^{N_{steps}} ReLU(\mathbf{d}[\mathbf{i}])$. Additionally, the TabNet gives the possibility to receive explainability out of the model by the masks. But, since the SHAP values are the focus of this paper concerning explainability, those will be used for the purpose of better comparison with the benchmark model.

The authors claim to have built a model, that shall perform at least as good as decision trees algorithms like XGBoost.

Image Model In addition to analyzing prices using the preprocessed variables, another approach was followed to develop a purely image-based model in parallel. The most promising approach is discussed in the following, although a second approach can be found in the Appendix Section A.4.

The basic idea of this model is when estimating the price from pictures, the room in the picture is crucial. For example, a high-quality living room can be identified as high-quality through other properties than a bathroom. For this purpose, the room classification model presented in Section 2.2.1 was used to identify the individual rooms shown in the images for a listing. If there are no images for the respective room category for a listing, black dummy images are added. The architecture of the network is visualized in Figure 21. Initially, similar to the room classification model, 2048 features of each image are extracted using a ResNet50. The extracted features for all images within a room category per listing are then stacked and only the largest value of each feature is retained via max pooling. This procedure is based on the Multiple Instance Learning (MIL) method proposed by Ilse et al. (2018), while using the maximum value represents the simplest case of MIL. With MIL, not every individual image receives a label and is processed individually instead a "bag" of images is treated as single input. Which leads to bags of all bathrooms, bedrooms,

dining rooms, living rooms, kitchens and halls being processed together through the model using only one label which is obviously the log price of the listing. The model creates a feature vector aggregated with max pooling for each of the six room categories. These six feature vectors are now independently routed through a room-specific FC layer (Dense Layer 1) with ReLU activation. In the end, each room category gets a dense layer with one node (Dense Layer 2) and ReLU activation, which represents a room-specific estimate of the log price. The ReLU activations for the last layers with one node ensure reasonable price predictions above one euro. In order to enable a generalization of the model, batch normalization layer, dropout and L2 regularization are also used. The final layer of the model aggregates the six room-specific predictions of the log price into a room-spanning estimate (Dense Layer 3). The six final weights are regularized in such a way that they are greater than zero and add up to one. The individual final weights can thus be interpreted as the contribution of each room category to the overall log price prediction.

Compound Model Due to the implementation of the image model as well as the TabNet in Tensorflow 2, these two models can be combined. This procedure would make it possible to include the additional explanatory power contained in the images but not captured by any of the structured variables used in the TabNet Model. For this purpose, the constraint for the weights in the last layer was retained, where two weights are obtainable for the Compound Model, namely one for the price prediction of the Image Model and one for the TabNet. Thus, the contribution of each of them to the final log price prediction can be assigned respectively.

3 Results

The following chapter presents the results of the models presented in Chapter 2. For the training of the individual models, also within the cross validation folds, it was ensured that all transformations such as standardizations and PCAs are fitted solely on the basis of the training data in order to avoid a data leak into the test and validation data. To be able to analyze all numerical variables on the same scale, the usual standardization (z-score) for each numerical variable to a mean of 0 and a standard deviation of 1 was used. Data normalization is common practice in machine learning and usually leads to better performance of the respective models (Jayalakshmi and Santhakumaran, 2011; Singh and Singh, 2020).

3.1 Room Classification

The room classification network described in Section 2.2.1 was trained for 10 epochs with a grid of different values for the hyperparameters learning rate, dropout rate and L2 regularization penalty (Table 13). The best performing parameters were identified using the validation data. Due to the use of batch normalization and dropout the L2 regularization turned out to be obsolet while the dropout rate was chosen to be 0.2.

The model was trained with the software library Tensorflow using a softmax loss, an ADAM optimizer (Kingma and Ba, 2014), a learning rate of 0.0001, a learning rate scheduler that reduces the learning rate by a factor of 0.1 when the validation loss plateaus with patience 5 (Bengio, 2012), and early stopping with a patience of 10 epochs (Prechelt, 1998).

The model achieves 85.98% accuracy on the test data and 90.19% on the training data. Figure 22a shows the training curves for the training and validation set. It can be observed that the overfitting of the model is minimal and a good generalization of the model predictions can be assumed. Figure 22b shows the confusion matrix of the trained model for the test dataset.

The model struggles the most with the prediction for dining rooms. This is not surprising as in particular the difference between dining rooms, kitchens and living rooms is in most cases not clear and can be challenging even for humans. Also unsurprising is the good classification of bathrooms, as they usually contain distinctive items that do not appear in other rooms. This assumption is confirmed by Figure 3 which shows some examples of the test dataset. Rectangles colored in red indicate a high SHAP Value for the area in the picture and the corresponding room which can be interpreted as an increase in the probability for predicting the room. Blue rectangles on the other hand indicate a decrease in the probability for the corresponding room. In the bathroom, the sink, toilet brush and the shower faucet are recognized as indicators of a bathroom while the plushy towel hanging over the bathtub is rather seen as a bedroom item which can be seen by the red rectangles for the bedroom probability. For the bedroom in the second image, the upper part of the bed with prepared pillows plays a decisive role. In the example picture for the kitchen, it can be seen that upper cabinets and the flooring have a high share in the kitchen classification. An interesting pattern also emerges for the hallway in the fifth image. Presumably, the perspective of the long narrowing hallway with several doors is decisive for the model here. But one have to bear in mind that the labels were generated using the image descriptions alone and may therefore also contain errors. Some of the, according to the generated labels, misclassified images from the test data are shown in Figure 23. The images (a) - (d) give examples where the model predicts a more plausible room type than the generated label. However, some images (e-h) also show the limitations of the model. If the image shows two different room types (e) or smooth transitions between the areas (f, g) then a clear assignment of the rooms is difficult. Also unconventional room designs like brick walls and wooden flooring in the bathroom (h) are not always correctly recognized by the model. Nevertheless, the model delivers good results on average, especially considering the missing ground true labels, and is therefore used to categorize the images. Since dropout and batch normalization was used the model doesn't suffer from severe overfitting. This fact was used to re-label the training data to identify and correct severely mislabeled images.

3.2 Price Prediction

XGBoost Model The XGBoost Model presented in Section 2.2.2 was trained and evaluated using the Python module "xgboost" (Chen and Guestrin, 2016). First, the hyperparameters of the number of weak learners (n_estimators), the proportion of random features

Figure 3: Examples for the Room Prediction with SHAP Values. Left Room Category corresponds to predicted Category by the Model

for each weak learner (colsample_bytree), the maximum depth of each weak learner decision tree (max_depth), the proportion of subsamples from the data (subsample = 0.9) and the regularization parameters γ and λ were optimized using exhaustive grid search, five-fold cross validation, and R^2 as the optimization metric on the validation data [7]. The parameter combinations are summarized in Table 14.

The best model was obtained for n_estimators = 500, colsample_bytree = 0.9, subsample = 0.9, max_depth = 3, gamma = 1 and lambda = 10 and achieved an average R^2 score of 0.66 on the validation data across all folds. A predictive model was then set up using the best of the five folds with the optimal parameterization. Table 2 shows the validation results for all five folds. The fifth fold was selected as the best one due to a R^2 value of around 0.65. The necessity to choose one fold over the others arises from the fact that the predictive model can not be trained on the training and validation data as it would be common practice, but only on the training data to allow a comparison to the TabNet Model. The TabNet Model uses the R^2 score of the validation data as final stopping criterion to avoid overfitting.

Table 2: MAE, MSE and R^2 on the Validation (above) and Test (below) Data for the XGBoost Model with 5-fold Cross Validation and best Hyperparameter Set

Fold	MAE	MSE	R^2
1	0.298	0.155	0.640
2	0.289	0.152	0.639
3	0.307	0.171	0.621
4	0.304	0.168	0.625
5	**0.286**	**0.151**	**0.646**

Test Results on 5th Fold:

MAE	MSE	R^2
0.291	0.152	0.618

As indicated in Table 2 a R^2 score of 0.62 was achieved for the test data with the predictive model.

Of particular interest is the influence of the individual variables on the prediction of the model. For this purpose, the SHAP Values presented in Section 2.1 were utilized. Since the SHAP Values are additive, all SHAP Values of the individual PCA categories can be summed up to determine the influence of the respective "PCA topic" on the prediction for a listing. In the following the interpretation using the example of the PCA for accommodation size will be presented. This PCA was calculated over 15 different variables with a total of seven PCA components being used as variables. The difficulty in the interpretation lies here in the fact that a positive value in the individual PCA components does not necessarily correspond to a larger accommodation. Moreover, the components cannot be aggregated to form a single variable. However, the influence of accommodation size can

[7] For a short explanation of the parameters see Section A.2.2

be determined by adding the SHAP Values of the individual components. This limits the interpretation of the effect of the individual PCA components, but allows to conclude whether the accommodation size in general effects the prediction of a listing in a positive or negative way.

Two example listings (A and B) and their SHAP Values are shown in Figure 24. The x-axis indicates the log price. The respective values for the variables/PCA topics along the y-axis located at the arrows indicate how much the estimate of the log price for the listing changes based on them. Next to the variables on the left side of the graphs, are the respective variable values of the listing, where "nan" values account for PCA blocks. Thus in "accommodation_size" the SHAP Values of all PCA components are aggregated. Below the x-axis, the expected value excluding all variables from the prediction ($E[f(x)]$) is shown, which is basically the mean value of all listings in the test dataset, and in the upper area, the value estimated with the help of the variables ($f(x)$) is displayed.

Figure 24a shows listing A where the size of the apartment has a negative influence on the price prediction, starting from the mean value. Figure 24b, on the other hand, shows listing B, where the opposite case can be observed. If the actual values of the variables that ultimately led to the respective PCA component values for both listings (Table 15) are considered, it can be obtained that A is an accommodation for a single person consisting of a room with a private bathroom, while B is an entire house for a maximum of six people. Thus, the price deviations of each listing can be indirectly traced back to the variables used for the PCA calculation. However, since the influence of the respective variables per listing are not the main focus, a global measure of the importance of the variables is needed. Herefore, all SHAP Values of the individual variables over all listings can be considered simultaneously. Lundberg and Lee (2017) suggest to calculate the mean value of the absolute SHAP Values as importance measure. Thus, it can be determined which variables are responsible for a particularly strong deviation of the prediction from the mean value. Figure 4 illustrates this by showing the respective variables in descending order of their SHAP importance measure. Each point represents a SHAP Value of the variable in a listing. The strength of the SHAP Values is plotted on the x-axis. The color of the dots indicates whether the value of the respective variable is large (red) or small (blue) for the corresponding listing. If the dots are gray, then these are aggregated PCA component blocks to which, as described above, no value can be assigned reasonably.

For example, a larger value for the variable "parking" (amount of parking lots around apartment) the variable leads to a rather large predicted price. Individual outliers can also be seen, for example a low value for the variable "host_since" has a very negative effect on the price prediction for a listing. Such outliers could hold interesting information, but will not be pursued further here. Overall, the most important variables are the PCA components on accommodation size, number of host listings, availability of accommodation, and physical proximity to cafes, restaurants, or the like. Interestingly, the availability of a TV seems to be a good indicator for higher or lower prices. Also, longitude and latitude still have a high relevance despite the many spatial variables generated with OSM, suggesting that there seem to be further spatial effects on price.

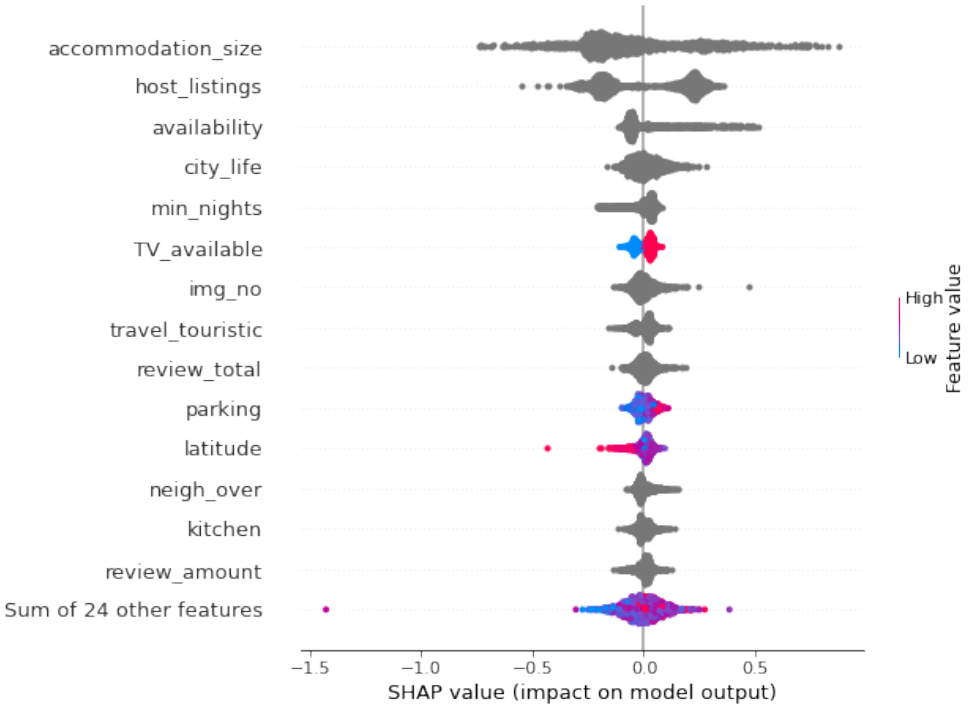


Figure 4: SHAP Values for top 15 Variables and PCA Blocks across all Listings ordered by their Absolute Mean SHAP Value for XGBoost Model. See Figure 25 for the complete Plot.

The same analysis including grid search was also performed on the 149 correlated variables obtained before decorrelation and without PCA. Due to the strong correlation between the variables the SHAP Values can't be interpreted reasonably (Lundberg and Lee, 2017). The purpose of this analysis was to demonstrate that the exclusion of certain variables and the reduction with the help of PCAs does not result in a serious loss of performance. On the validation data, an MSE of 0.1361, an MAE of 0.2740, and an R^2 of 0.6885 were obtained, which represents a minimal improvement over the decorrelated model. Slightly better values were also obtained on the test data with 0.1395, 0.2776 and 0.6485. To include the reduction in complexity of the model, the adjusted R^2 was also calculated on the test data for both models. The decorrelated model yields with a value of 0.5968 almost the same score as the correlated model with 0.6038. Thus the reduction from 149 to 69 variables did not lead to a significant loss of information. The loss obtained can be seen as the price one has to pay to use SHAP Values for the explainability of the models.

TabNet Model The TabNet's performance was validated by datasets with a size of at least 10k observations but mostly up to several hundreds of thousands in the original paper of Sercan and Pfister (2019), which makes the comparison to XGBoost challenging here. Due to the implementation of the image models in Tensorflow the Tensorflow implementation of the TabNet by Majumdar et al. (2019) was used to keep consistency. In contrast to the XGBoost hyperparameter grid search, the grid search for the TabNet was

conducted stepwise because of infrastructural constraints. According to the documentation of the TabNet package the largest performance gain can be achieved by varying the feature dimension N_a and output dimension N_d. N_a is the dimension within the Feature Transformation block, that is transferred between the layers and N_d is the output dimension of each decision step, which gets mapped to the final regression output later. Furthermore, a similar number of dimension is reasonable, but $N_a > N_d$ is necessary. Some arbitrary choice for the number of decision steps N_{steps} indicated that two steps seemed most appropriate for the dataset. This is consistent with the documentation, which states: "Most datasets yield the best results for $N_{steps} \in [3, 10]$. Typically, larger datasets and more complex tasks require a larger N_{steps}." (Majumdar et al., 2019). As the dataset used here is far smaller than the ones used in the paper, $N_{steps} = 2$ seemed appropriate in the first place but was tested in the grid search in the second step as well. A grid search for the relaxation coefficient γ and the sparsity coefficient λ_{sparse} was conducted accordingly. An overview of the tested hyperparameters over 50 epochs can be seen in the Table 16. Again, a five-fold cross validation with R^2 as the optimization metric on the validation data was used. The results of the stepwise grid search is presented in Figure 26.

The best performance got achieved with the following hyperparamters: output dimension $N_d = 120$, feature dimension $N_a = 125$, number of decision steps $N_{steps} = 2$, relaxation factor $\gamma = 1, 5$ and sparsity coefficient $\lambda_{sparse} = 1e{-}05$. After training models of 100 epochs as a five-fold using the outlined hyperparameters, the best fold concerning the validation set yielded a R^2 on the test data of 0.535. For the results of the different folds see Table 3 and the history of the best model concerning the MAE, the MSE and the R^2 is visualized in Figure 27.

Table 3: MAE, MSE and R^2 on the Validation (above) and Test (below) Data for the TabNet Model with 5-fold Cross Validation and best Hyperparameter Set

Fold	MAE	MSE	R^2
1	0.37	0.24	0.47
2	0.32	0.19	0.57
3	0.32	0.18	0.56
4	**0.32**	**0.18**	**0.58**
5	0.33	0.19	0.56

Test Results on 4th Fold:

MAE	MSE	R^2
0.327	0.185	0.535

The same grid search procedure including the five-fold with the best hyperparameters with all features generated, without any selection or building of the PCAs, the TabNet achieved only slightly better results on the test data with a R^2 of 0.567 (see table 30 for results concerning the grid search and the history of the best fold). For a proper comparison of both models, the adjusted R^2 on the test data is used again, that amounts to 0.509 for

the network with uncorrelated features and 0.512 for the network including correlated features. Hence, only a small loss in performance can be obtained due to the exclusion and grouping of variables using PCAs but therefore a better explainability by the SHAP Values is secured.

Accordingly to the XGBoost Model, the interest of this paper lies on the SHAP Values. Figure 28 shows again listings A (28a) and B (28b). Compared to the XGBoost model, a significantly different prediction can be seen. For listing A, the PCA "availability" again has the strongest positive influence. However, this is almost three times as large as in the XGBoost Model. Likewise, the PCA "host_listings" weighs significantly more negatively, while the PCA "accommodation_size" exerts an equally large negative influence on the price of this listing. For Listing B, the positive influence of "accommodation_size" is slightly higher than for the XGBoost Model. It is noticeable here that the next strongest features in terms of absolute SHAP Values are all related to the reviews. The timing of the last review has a positive influence, but both the PCA "review_total" and review scores for location and cleanliness and also communication have a negative influence on the price. In addition, with the PCA "city_life", a negative rating of the listing in combination with a location worse than the average seems to have a negative impact on the price. The TabNet seems to be more sensitive to reviews, which is confirmed by the aggregated SHAP Values (Figure 5). The PCA "review_total" is found here in the fifth most important position, calculated by the sum of absolute SHAP Values, while it was in the ninth position for the XGBoost Model. Common to both models is that the PCA "accommodation_size" is recognized as the most important feature, which corresponds to the intuitive perception. According to both models, availability and the PCA "city_life" also play a major role. With regard to the other variables, the specific ranking is more varied, but apart from "TV_available" in the XGBoost and "last_review" in the TabNet, the same features are found among the more important ones.

Image Model The image-based price prediction model presented in Section 2.2.2 was trained with 3070 different listings and 40309 individual images. In order to limit the amount of imputed black images, only listings with images in at least four of the six room categories were considered. In total 5583 black images had to be added to the 56704 images in train, validation and test dataset. The hyperparameters dropout rate, L2 regularization parameter and the number of nodes in Dense Layer 2 (see Figure 21) were optimized using a small exhaustive grid search due to the computational intensity of the model. The chosen hyperparameter values can be found in Table 17. The results of the grid search are shown in the Figure 18.

The best results in terms of R^2 value and MSE on the validation data were achieved using a dropout rate of 0.2, an L2 regularization parameter of 0 and 512 nodes for the Dense Layer 2. However the smaller architecture with 64 nodes was used due to less overfitting and almost the same R^2 score on the validation set. The model with the best parameterization was trained over 300 epochs with an exponential learning rate decay with stepsize of 100 iterations and a rate of 0.9 and a batchsize of 128 listings per epoch again for all five folds. The epoch with the best R^2 value on the validation data was then chosen as the final predictive model. Table 4 shows the results for each fold on the the validation

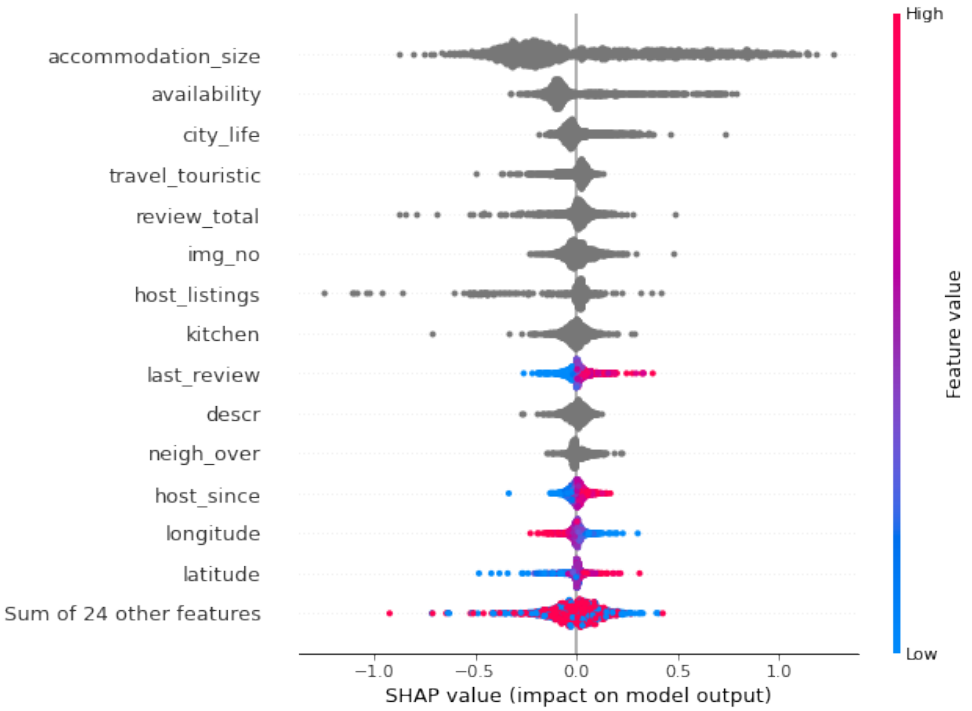


Figure 5: SHAP Values for top 15 Variables and PCA Blocks across all Listings ordered by their Absolute Mean SHAP Value for TabNet Model. See Figure 29 for the complete Plot.

data. As indicated, the best results were achieved by the model trained on the first fold which will be used as predictive model.

Table 4: MAE, MSE and R^2 on the Validation (above) and Test (below) Data for the Image Model with 5-fold Cross Validation and best Hyperparameter Set

Fold	MAE	MSE	R^2
1	**0.4151**	**0.2741**	**0.3235**
2	0.4082	0.2832	0.3246
3	0.4517	0.3377	0.1861
4	0.4320	0.3058	0.3174
5	0.4314	0.3073	0.2144

Test Results on 1st Fold

Fold	MAE	MSE	R^2
1	0.4159	0.2967	0.2439

The model achieves an R^2 score of 0.324 on the validation data, and 0.244 on the test data set. However, the high variance between the individual folds should also be emphasized. The corresponding weights for the last layer are listed in Table 5. It turns out that images of living rooms with a contribution of 0.377 have the strongest impact on

the final price prediction. Kitchens, bathrooms and bedrooms follow with 0.153, 0.141 and 0.130. The images of halls have the smalles impact on the final prediction at all. However, the results should be interpreted with caution, as strongly fluctuating weights were observed depending on the fold used. This uncertainty might be a result of the two main model's weaknesses. The first is the underlying model used to categorize the room in the images in the first place. Although the image classification model offers solid results on average as described in Section 3.1, it can be a decisive factor for this model via misclassifications for listings with only a few images. An example would be images of a neighboring restaurant being classified as the apartment's kitchen or dining area. Better training data for the room categorization model or even a more advanced unsupervised model approach might provide better results and improve the image-based price prediction model. The second weakness lies in the large number of black dummy images that had to be added to train the model. It might be reasonable to use *AirBnB*'s worldwide database and filter out only those listings that have pictures for all categories. Training the model on this data would likely lead to a more reliable result in regards of the weights.

Table 5: Contributions of Room Categories to final log Price Prediction

	weight
bath	0.140740
bed	0.130050
dining	0.100526
hall	0.099289
kitchen	0.153168
living	0.376227

Compound Model Unfortunately, an extensive hyperparameter optimization via cross validation and grid search was computational not feasible. This is mainly due to the high number of parameters that would have to be optimized simultaneously for TabNet and Image Model and the limited access to excessive computational resources. Nevertheless, a first model was trained with the parameters identified as optimal from the Image and TabNet Model. An ADAM optimizer with exponential learning rate decay of 0.95 in every 100th iteration was used again. The initial learning rate was set to 0.01 and the model was trained for 300 epochs. Moreover the weights of the fitted TabNet and Image Model described above were used. Before model training those weights were frozen and only the final weights of the Compound Model were trained from scratch.

The model achieved an R^2 of 0.465 an MSE of 0.245 and an MAE of 0.372 on the validation data and values of 0.442, 0.237 and 0.374 on the test dataset. Inspecting the weights of the final layer, it was observable that 100% of the final prediction can be accounted to the TabNet branch. Interestingly the optimum found by the model performed significantly worse for the validation data compared to a TabNet-only model. The reason for this is that the weights of last layer of the TabNet which basically connect the TabNet output features

to a single node dense layer were not frozen to allow the TabNet branch to adapt more flexibly. These observations indicate that further fine tuning of the model might also lead to better results. However it also indicates that the images might not give further relevant information about the price of the apartment. It should also be mentioned that the frozen weights were set to be trainable and the model was again trained with a smaller learning rate of 0.0001. Unfortunately this neither improved the model performance significantly (R^2 on validation data of 0.456) nor changed the contributional weights of the final layer.

3.3 Munich Data

To test the generalization of models described above, the dataset of the city of Munich was used. Most of the variables were transformed with the fitted transformations (i.e. PCAs, standardization) of the Dublin dataset. Some variables had to be adjusted manually. For example, Tripadvisor's top 10 sights in Dublin have been replaced by those in Munich. Log prices were also transformed to have the same mean and standard deviation as in the Dublin data. The results can be seen in Table 6. It quickly becomes evident that the models trained on Dublin are not able to achieve a performance close to the Dublin test data. For the TabNet and XGBoost models one crucial issue is that the spatial variables of the two cities are simply not comparable. It is not only the location of the sights or the city structure in which the two cities differ, it is also the dataset itself: Figure 32 visualizes that for Munich no listings are located outside the city area, but all listings are located extremely centrally, which was the opposite case for Dublin (Figure 15). In addition, the standardization of the variables as well as the PCAs cannot be generalized to another city. This becomes apparent when looking at the SHAP Values for the variable longitude in Figure 31 which was also standardized on the Dublin training set. In Munich all values for longitude are strongly positive, which means that the model is confronted with high values that it did not encounter in the training. However, the results indicate that the XGBoost Model is much more robust to this issues while the TabNet with a negative R^2 seems quite sensitive to it. Rather surprisingly, the generalization problem also applies to the Image Model which achieves a negative R^2 of -0.025. The spatial structure of the apartments could also be decisive for the images. While in Ireland many rural accommodations and entire houses are offered, which are some kilometers away from the city center, the accommodations from Munich are for the most part city apartments which could also be reflected accordingly in the images.

Table 6: Model Performance on the Munich Dataset

Model	MAE	MSE	R^2
XGBoost	0.4628	0.3286	0.2515
TabNet	0.5737	0.499	-0.150,
Image Model	0.5174	0.4322	-0.0251

4 Conclusion

In order to find a transparent model predicting the charged prices of an *AirBnB* listing three Deep Learning models were introduced besides the XGBoost algorithm. The results of the TabNet Model were extensively compared to the XGBoost results while the explainability was approximately ensured by the SHAP Values. In order to interpret the SHAP Values correctly, PCAs were used to ensure no significant correlations between the features. This extensive selection did not lose significant information and explanatory power was also maintained by the PCAs. Nevertheless, at least for this dataset, the XGBoost algorithm outperforms the Deep Learning methods both in terms of explanatory power and computational efficiency. However, those computational terms are a big constraint for the Deep Learning approaches as there was no access to an adequate computational power, which might result in slightly better performances by a far more detailed grid search.

Still, both models, the TabNet as well as the XGBoost, identified mostly the same variables as the most important ones, measured by the SHAP Values. These variables, which according to the models have the greatest influence on price, also correspond predominantly to the expectations beforehand. Especially, the PCA for accommodation has high explanatory power and the variables taken into account give great information about what kind of accommodation is considered, hence, it was expected to be identified as a very important variable. Nevertheless, it must be taken into account that several variables are considered in the PCAs and their SHAP Values are summed up afterwards. Thus, those values are hard to compare with SHAP Values of individual variables to a certain extent. A mitigation to this would be to expand the PCAs by building PCAs from the very beginning on, that are split by topics. Moreover, the models are trained with 69 features of which a lot do not contribute any significant explanatory power to the models, according the the SHAP Values. By forward or backward selection, the evaluation with the adjusted R^2 would be a consistent method to arrive at a more sparse and therefore computationally more efficient productive model.

The idea of the image model seems appropriate and the underlying room classification model in some cases worked better than the corresponding descriptions. However, this is not entirely quantifiable, since rooms cannot always be clearly separated and supervision would be extremely time-consuming. Additionally, to avoid the imputation of black images and increase the size of the dataset further cities could be included since the effect of images might be similar across cities.

The final idea of a compound model from the tabular and image data showed moderate results in the first attempt, as not even the performance of the TabNet was achieved. Again, with more computing power and time a grid search might produce better results, but this is beyond the scope of this paper.

Both models for tabular data, the XGBoost and the TabNet were not able to generalize enough to predict the prices of Munich. Reasons for this are the specific variables and the corresponding standardization especially for the spatial variables, which clearly follow other dependencies in Munich than in Dublin. Same holds for the PCAs, which carry a crucial part of the explanatory power.

A Appendix

A.1 Intro

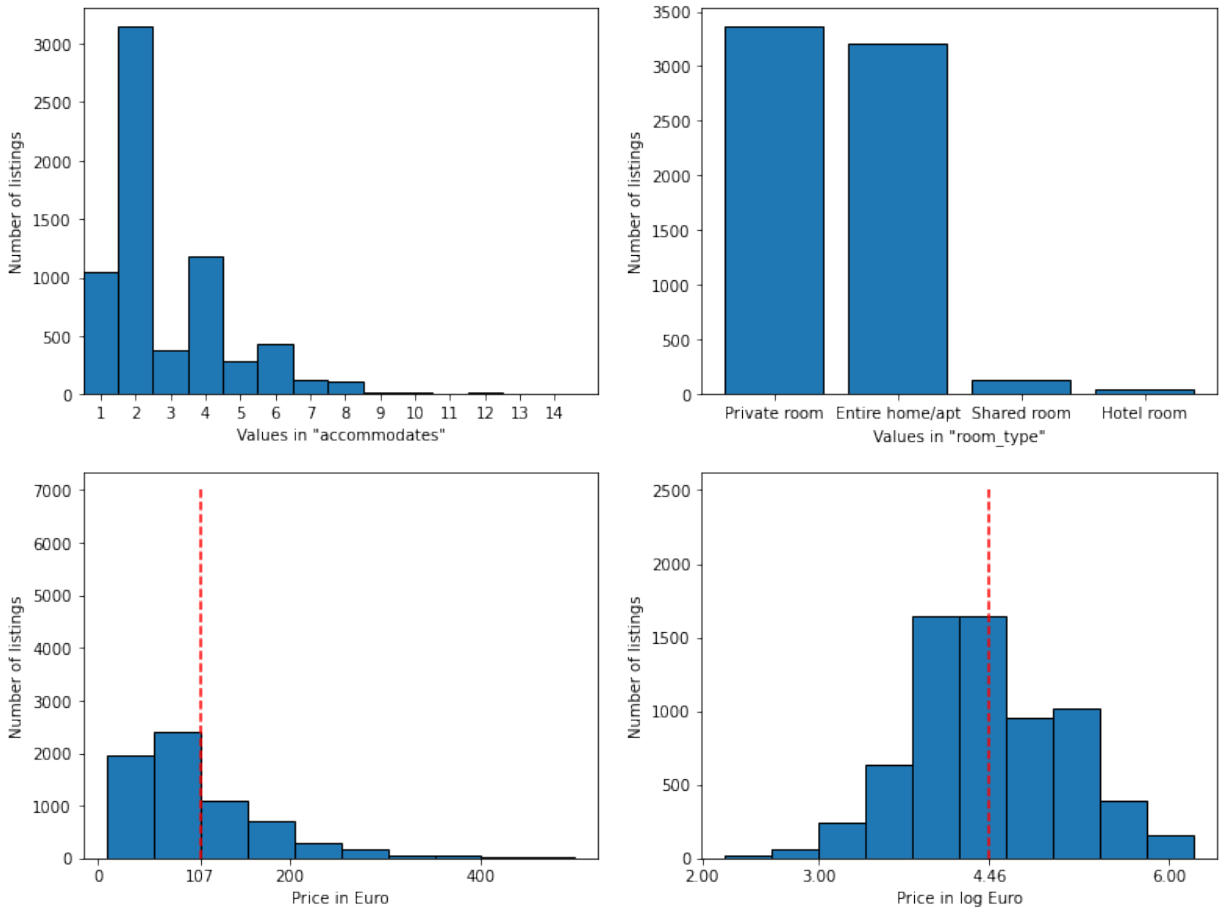


Figure 6: Histograms for the Variables "accommodates", "room_type" at the Top and the Price and log Price at the Bottom. Extreme Prices above 500 Euro have been excluded

Table 7: Manual Variable Transformations. _X stands for the individual Categories of the Variable

Variable	Transformation	Transformed Variable
bathrooms_text	Split into categories and number of bathrooms	bath_kind_X, bath_number
host_location	Replace by country of host's residence	host_location_country
bath_number	Less frequent categories grouped into "Others"	bath_number_X
host_location_country	Less frequent categories grouped into "Others"	host_location_country_X
bedrooms	Less frequent categories grouped into "Others"	bedrooms_X
property_type	Less frequent categories grouped into "Others"	property_type_X

Variable	NAs
neighbourhood_group_cleansed	6976
calendar_updated	6976
license	6976
bathrooms	6976
host_response_rate	4509
host_response_time	4509
host_acceptance_rate	4328
host_about	3466
neighbourhood	3028
neighborhood_overview	3028
host_neighbourhood	2173
review_scores_value	1582
review_scores_location	1580
review_scores_checkin	1580
review_scores_accuracy	1574
review_scores_communication	1574
review_scores_cleanliness	1573
review_scores_rating	1429
last_review	1429
first_review	1429
beds	242
bedrooms	236
description	206
host_location	32
host_name	11
host_since	11
host_identity_verified	11
host_listings_count	11
host_total_listings_count	11
host_has_profile_pic	11
bathrooms_text	6
name	1

Table 8: NA Values for each Variable (excluding zero NAs)

Table 9: Culture Units built for the Variable *host_name*

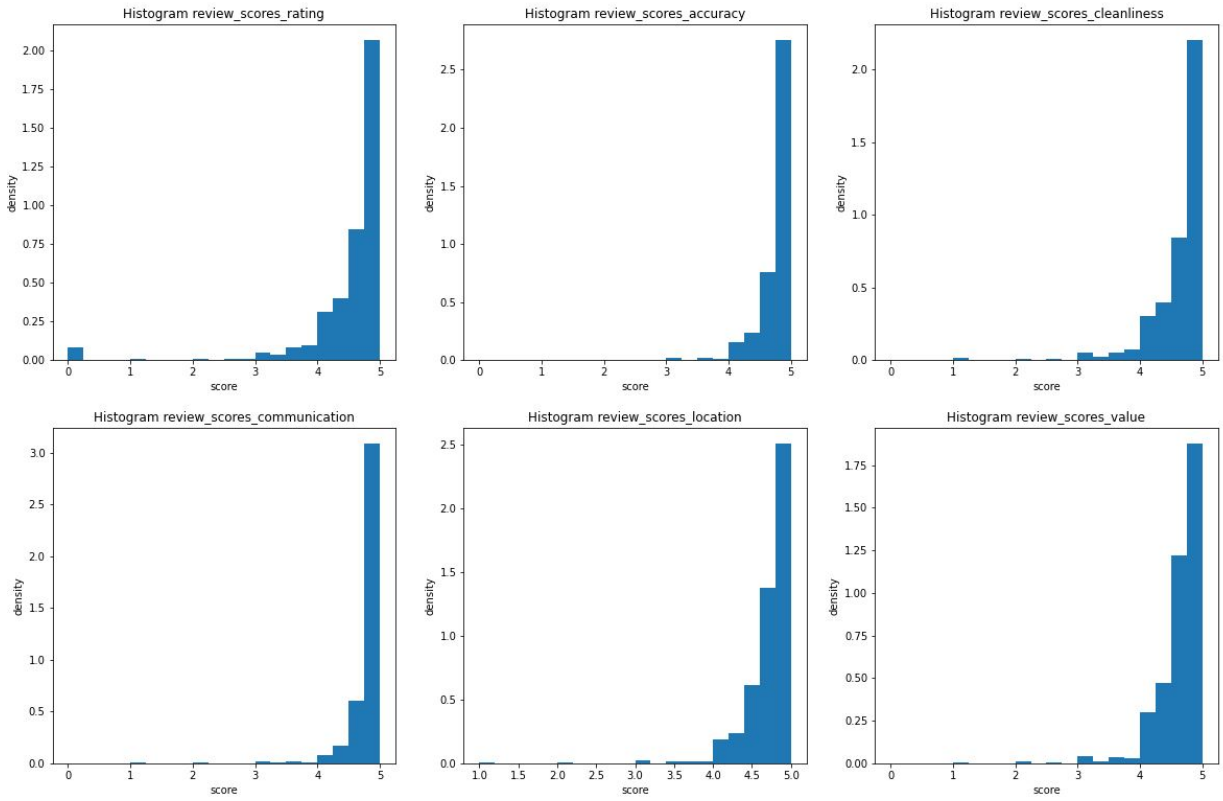
Cultural Unit	**Included Countries**	**Source**
arabic	Algeria, Egypt, Iraq, Jordan, Kuwait, Lebanon, Morocco, Palestine, Qatar, Saudi Arabia, Sudan, Syria, United Arab Emirates, Tunisia	https://en.wikipedia.org/wiki/Arab_world
africa	Angola, Burkina Faso, Cameroon, Ghana, Ivory Coast, Kenya, Liberia, Mayotte, Nigeria, Senegal, Somalia, Tanzania, Uganda, Zambia Zimbabwe	https://en.wikipedia.org/wiki/Africa
asia	Afghanistan, Azerbaijan, Bangladesh, China, Georgia, Hong Kong, India, Indonesia, Israel, Iran, Japan, Kazakhstan, Malaysia, Nepal, Pakistan, Philippines, Singapore, South Korea, Taiwan, Thailand, Turkey, Vietnam	https://en.wikipedia.org/wiki/Asia
eastern Europe	Albania, Belarus, Bosnia and Herzegovina, Kosovo, Macedonia, Russia, Serbia, Ukraine	https://en.wikipedia.org/wiki/Eastern_Orthodoxy_by_country
latin america	Argentina, Bolivia, Brazil, Chile, Colombia, Costa Rica, Dominican Republic, Ecuador, El Salvador, Guatemala, Honduras, Jamaica, Mexiko, Nicaragua, Panama, Paraguay, Peru, Panama, Panama, Uruguay, Venezuela	https://en.wikipedia.org/wiki/Latin_America
west	Australia, Austria, Belgium, Bermuda, Bulgaria, Canada, Croatia, Cyprus, Czech Republic, Denmark, Finland, France, Germany, Greece, Hungary, Ireland, Italy, Jersey, Latvia, Lithuania, Luxembourg, Netherlands, New Zealand, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, United Kingdom, United States	https://en.wikipedia.org/wiki/Western_worldModern_definitions

Cultural Groups	**Frequency**
afria	141
arabic	109
eastern europe	23
latin america	42
rare	1536
traditional irish	545
west	4271

Table 10: Frequencies of Cultural Groups

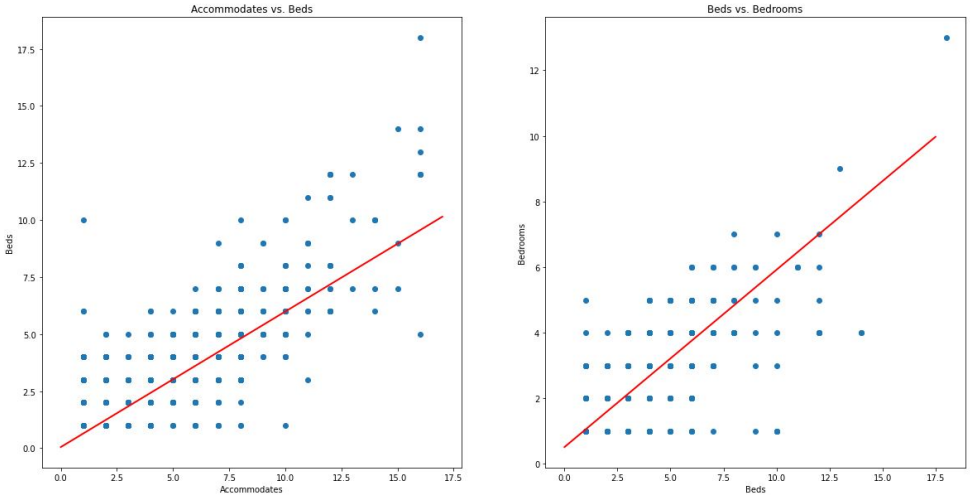
Table 11: Different Variables, their Methods of Imputation and Corresponding Notes.

Variable	**Imputation method**	**Note**
name	room_type	
description	room_type	NLTK-Sentiment Analyzer gives 1 for neutral, else 0 values.
neighborhood_overview	neighbourhood_cleansed	NLTK-Sentiment Analyzer gives 1 for neutral, else 0 values.
host_about	" "	NLTK-Sentiment Analyzer gives values of 0.
first_review	last_scraped	No review yet.
last_review	last_scraped	No review yet.
reviews_per_month	number_of_reviews	No review yet.
host_location_country	"Ireland"	Most likely case.
host_listings_count	Counting.	No. of listings with same host id.
host_name	Web-Scraping	Read field of host name, if nothing there, set "Anonymous".
host_since	first_review	
beds	OLS Regression	See Figure 8
bedrooms	OLS Regression	See Figure 8
all review_scores	Draws from manipulated Halfnormal distribution.	See Figure 7
several amenity-dummies	Draws from Binomial distribution.	Relative frequency as probability.
grouped review variables	Mean	including length, sentiments, porportion of eng. reviews, etc.
no of images	0	Rooms and in total.
Brightness	Mean	
Contrast	Mean	



If $X \sim N(0, \sigma^2)$, then $Y = |X|$ follows a half normal distribution. Let y be a realisation of Y, we calculated $y * (-1) + 5$ for imputations.

Figure 7: Distribution of the different Review Scores.



In both cases the coefficients were highly significant. The predictions for missing values got rounded to be an integer again.

Figure 8: OLS Regression for Beds and Bedrooms as Imputation.

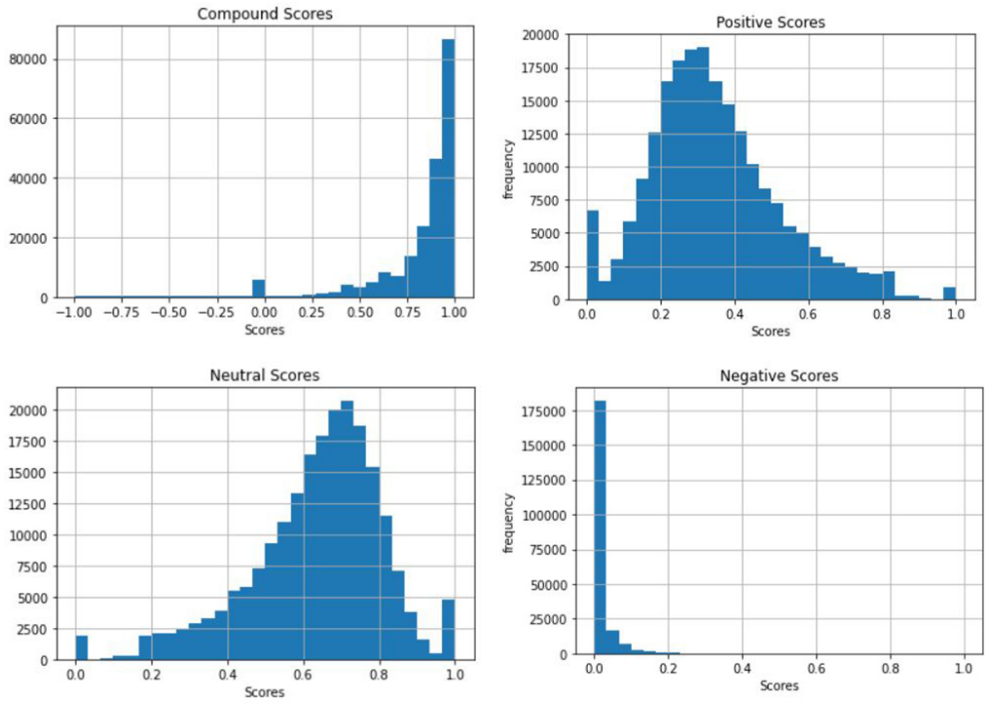


Figure 9: Sentiment Scores of the Reviews

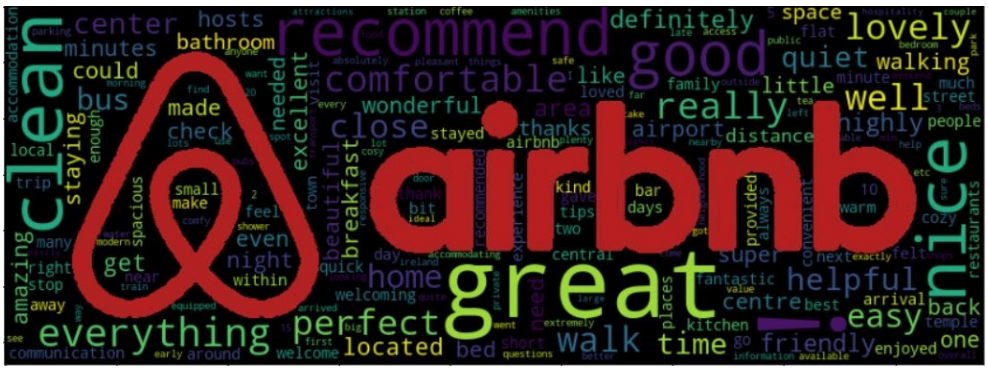


Figure 10: Illustration of Word Frequencies of the Reviews of Dublin

A.1.1 Data Preprocessing

Sentiment Analysis - Examples Example with a compound value of -0.98 and an example with a compound value of 0.98:

> `[...] In one of bathrooms under a shower it is impossible to wash as very weak pressure of water because of the strange pump which is installed there. [...] But the hugest minus is Rasplozheny apartments directly over a local pub. All who go are obliged to know that thursday through sunday you won't sleep to 2 o'clock in the morning. It is`

impossible because of noise!!! The owner of the apartment hasn't warned about it. [...] We didn't get enough sleep and the child has been forced to sleep in kitchen.

Wow, we only skim the surface! [...] I might add the landscaped garden and its beautiful apple trees are great to see as you drink your morning coffee and there s a great local country pub only a few hundred meters away which was a great bonus for a quick nightcap. The apartment itself is new, super clean, modern, warm and cosy and great value for money.

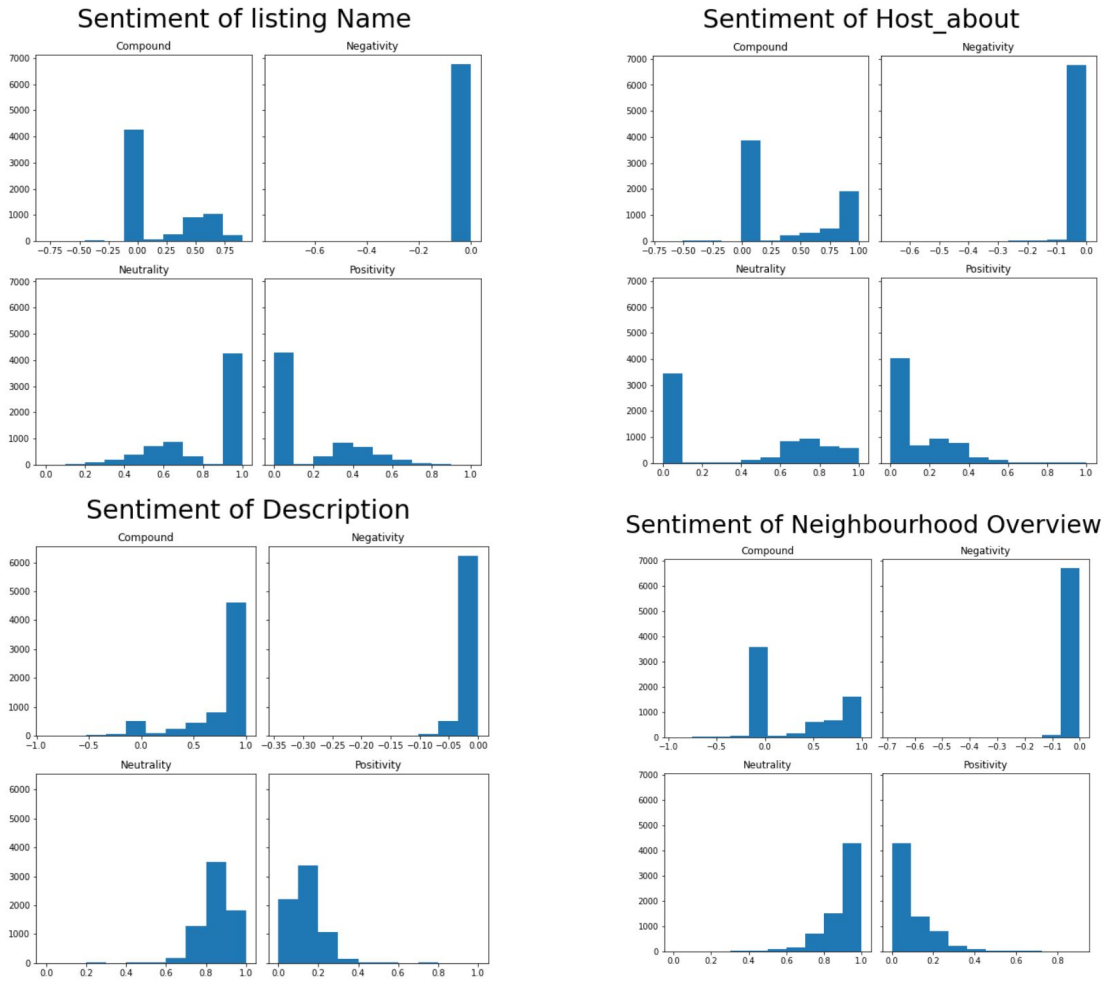


Figure 11: Distribution of Sentiment Scores of further Text Variables

(The Sentiments for *name* were not used as Variables.)

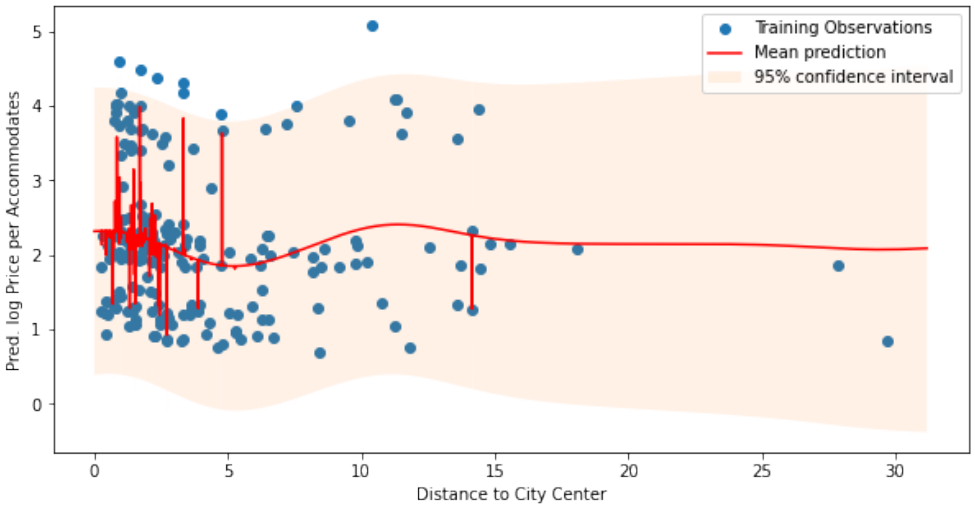


Figure 13: Gaussian Process Regression

(city Center-Distance on log-price per Accommodate using a RBF kernel, a dot Product and RBF Kernel and a Constant Kernel)

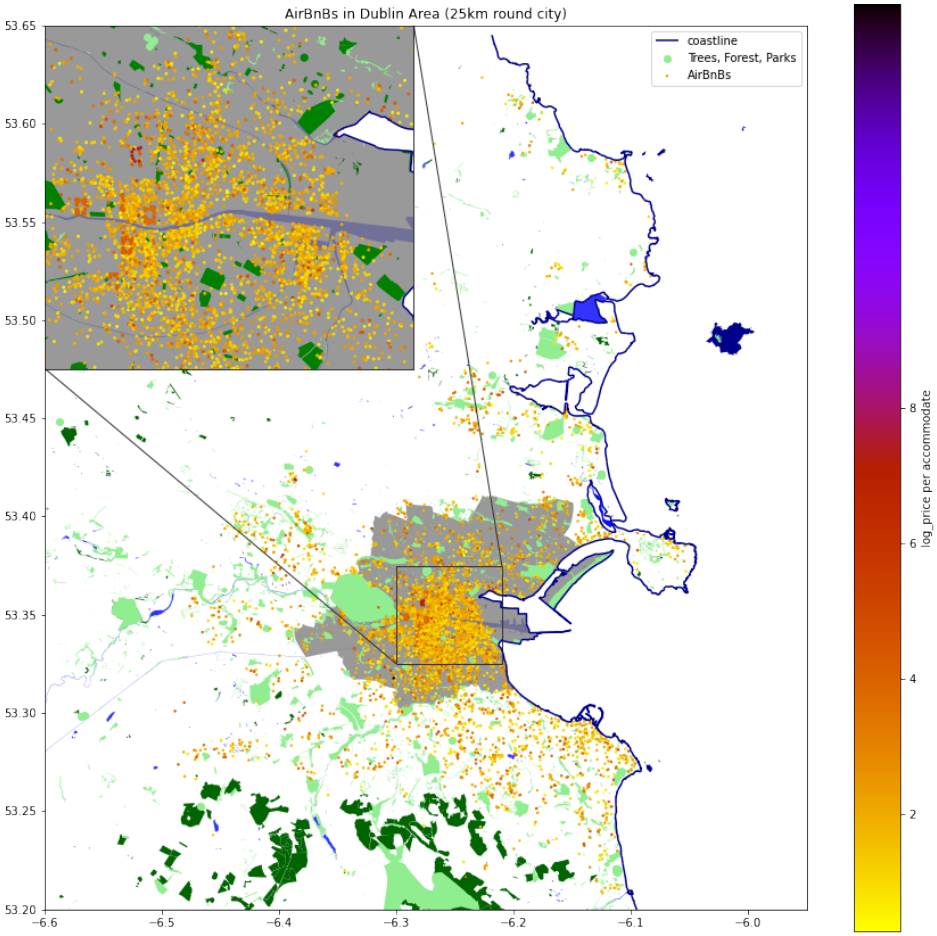
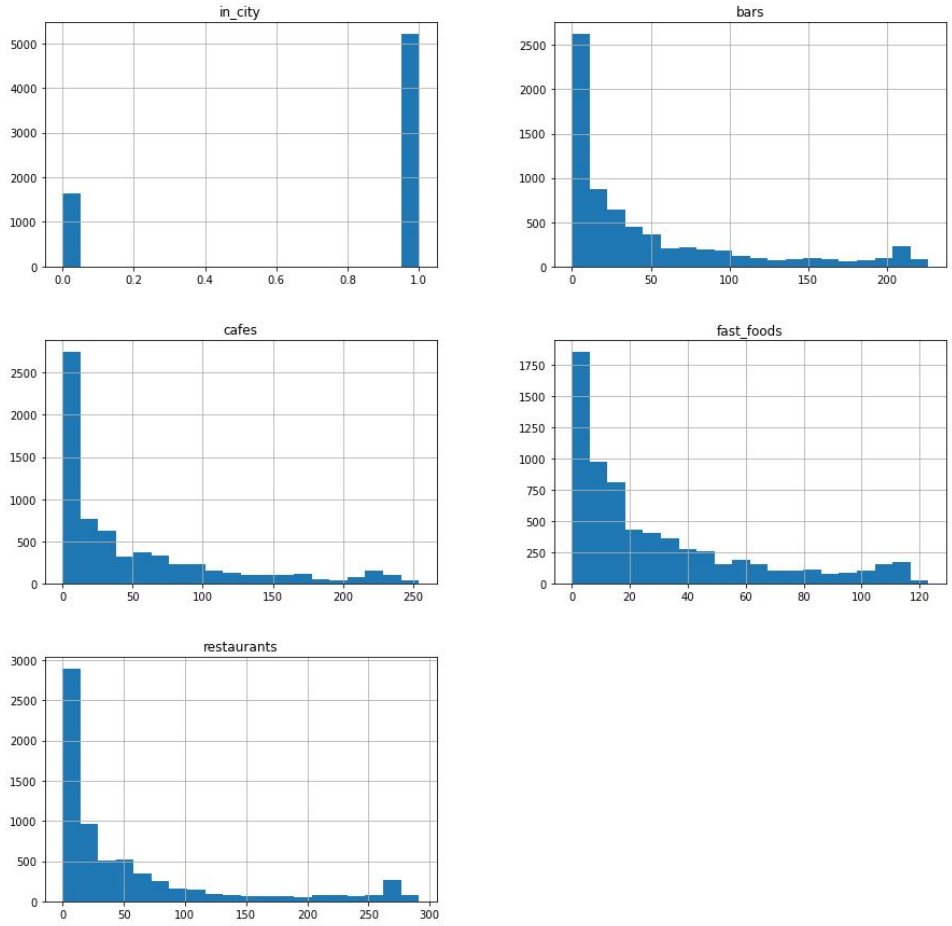
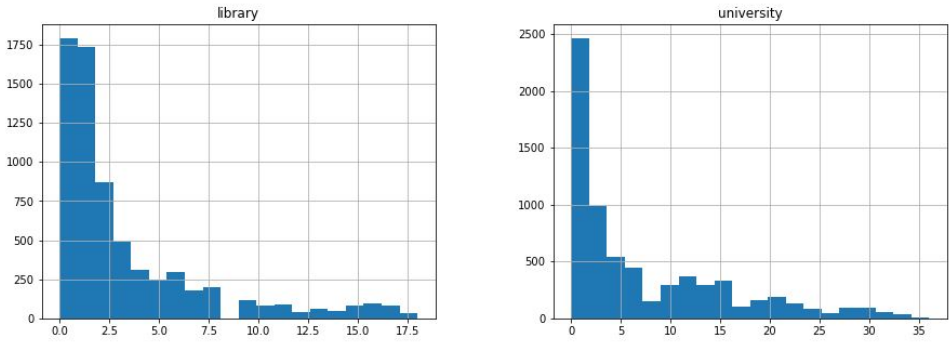


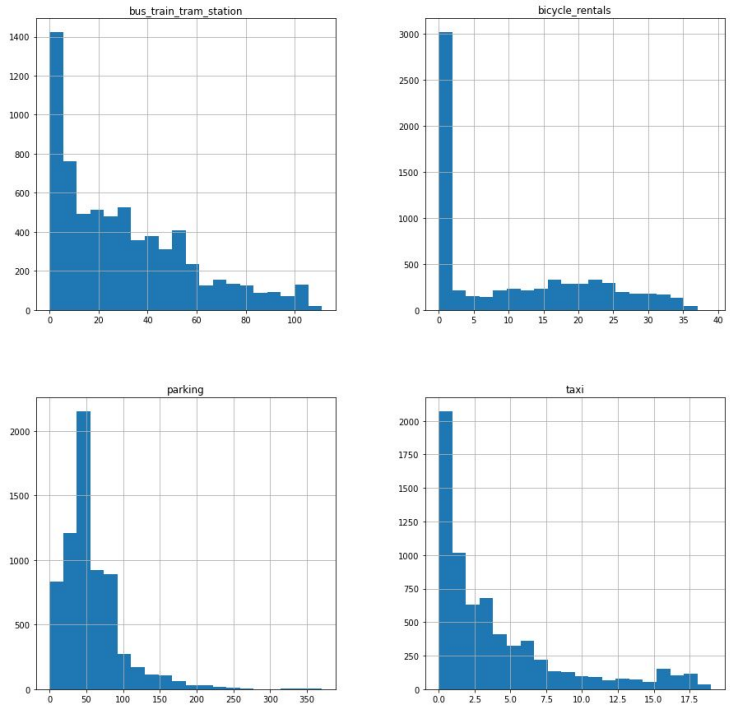
Figure 15: Spatially Distribution of the Dublin's *AirBnB* Listings with log-Price per Accommodate



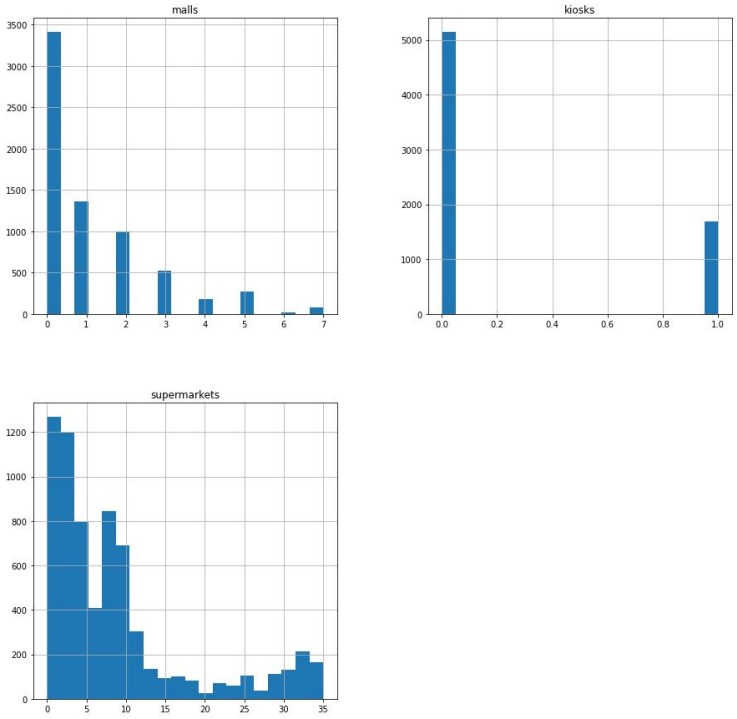
(a) Amount of Listings within the City, Bars, Cafés, Fast-Foods-Offers and Restaurants.



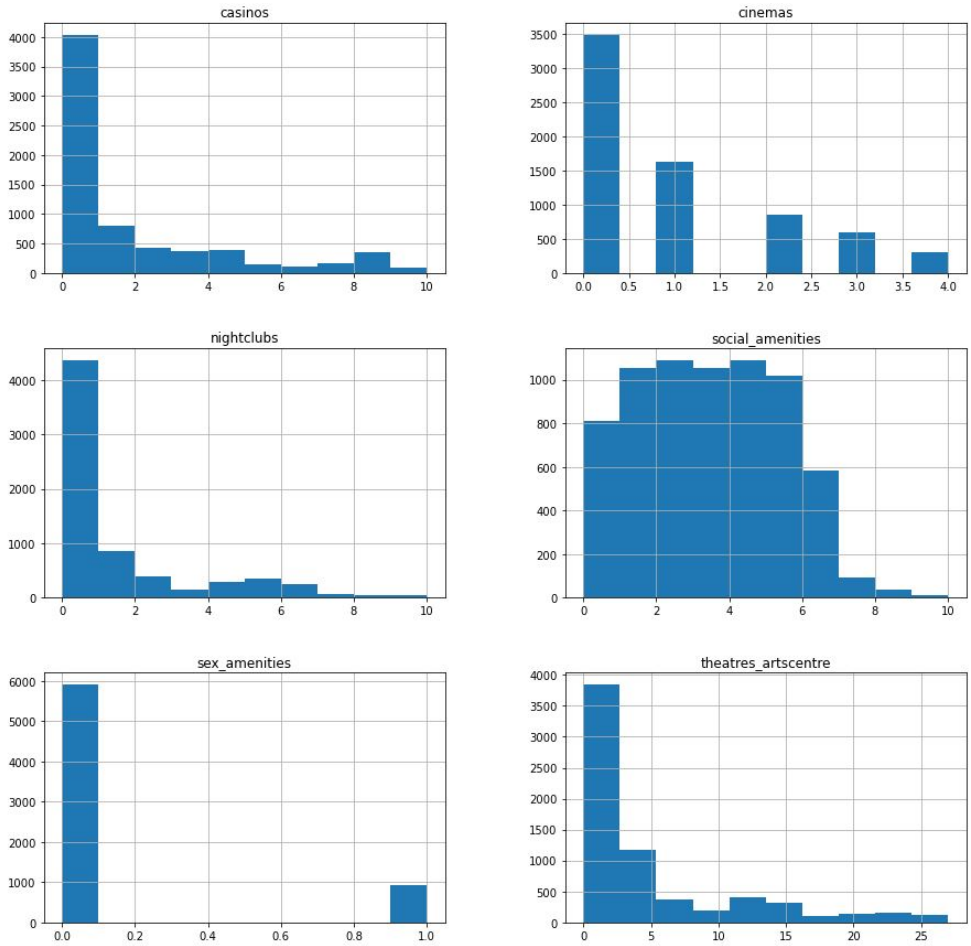
(b) Amount of libraries and university buildings.



(c) Amount of bus, train and tram stations, bicycle rents, parking possibilities and taxi stands.



(d) Amount of malls, kiosks and supermarkets.



(e) Amount of casinos, cinemas, nightclubs, social amenities, sex amenities and theatres or arts-centres.

Figure 16: Different Number of Objects Found in the Considered Area (25km Around Dublin's City Center)

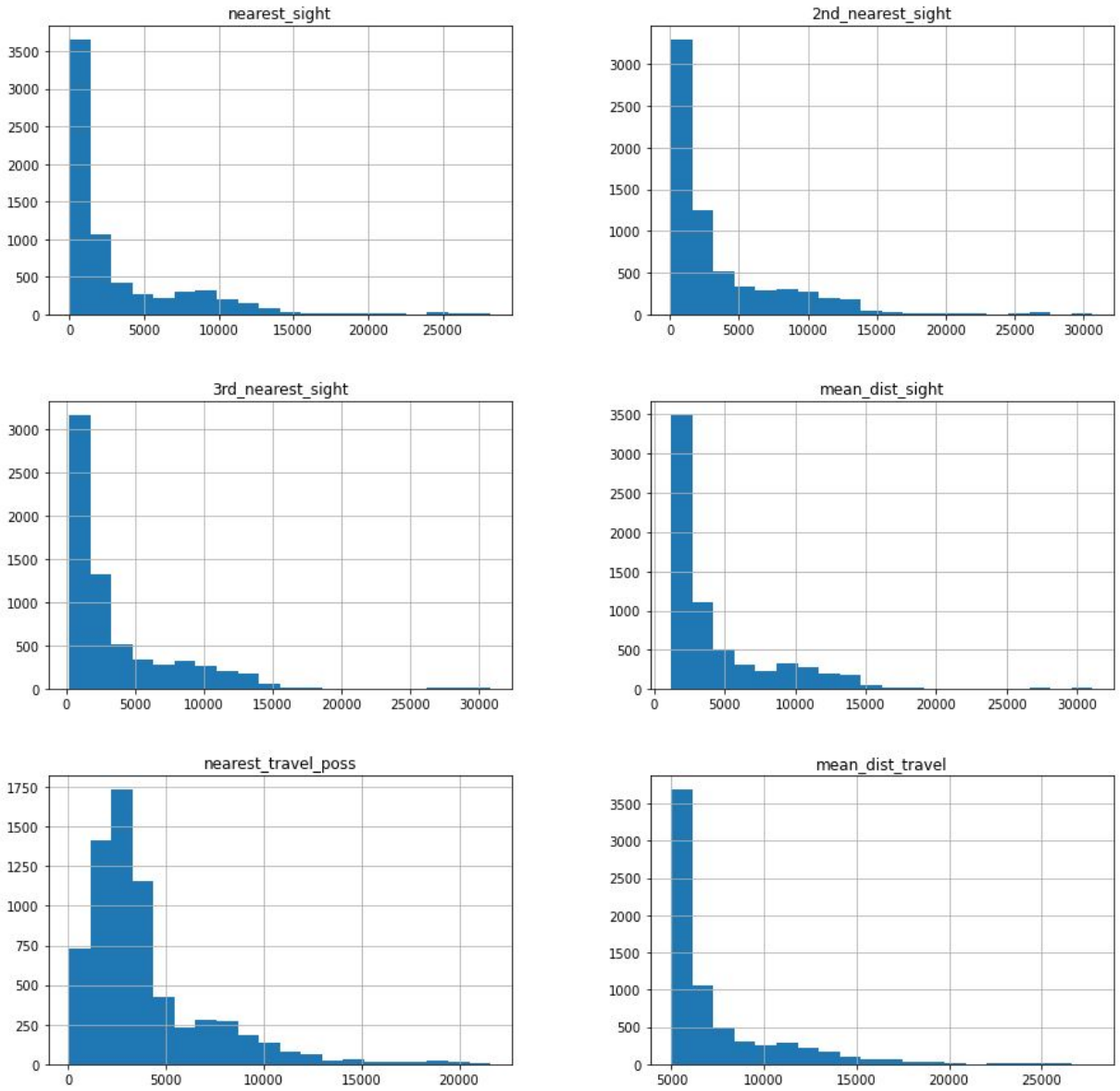


Figure 17: Frequencies of the generated Distance Variables

A.2 Methods

A.2.1 Variable Selection Methods

Correlation Coefficients The Jaccard coefficient is defined by:

$$r_J(j,k) = \frac{a}{a+b+c} \tag{2}$$

where a is the number of observations where the variable j and the variable k are both equal to 1, b is the number of observations with 0 in variable j and a value of 1 in variable k and c vice versa. By this definition the Jaccard coefficient is bounded between 0 and 1.

The point biserial correlation coefficient (Tate, 1954) is used to determine the relationship between a numeric variable x_j and a binary variable x_k. This can be described as the standardized difference in the means of x_j when divided into two groups using the binary variable x_k. This standardized difference is then additionally weighted using the group sizes. Formally, the point biserial correlation coefficient can be defined as:

$$r_{PB}(j,k) = \frac{\bar{x}_{j0} - \bar{x}_{j1}}{s_j} \cdot \sqrt{\frac{n_0 n_1}{n^2}} \tag{3}$$

where n_0 and n_1 are the observations of x_j in each group defined by x_k. The point biserial correlation coefficient is bounded between -1 and 1.

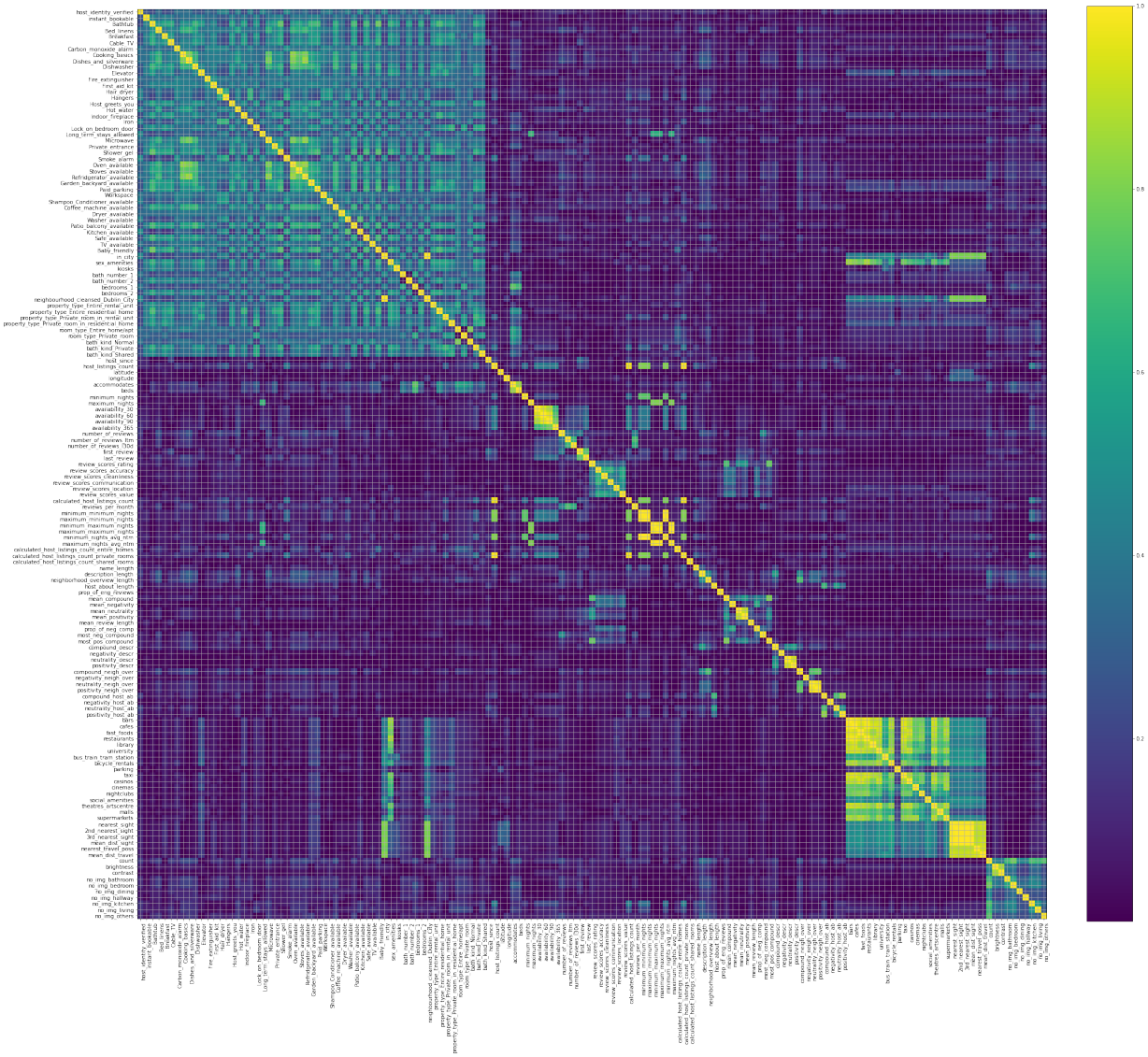


Figure 18: Heat Map of Absolute Values for Pearson, Point-Biserial and Jaccard Coefficients

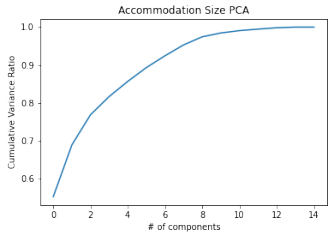
PCA Let the random vector $\mathbf{X}' = (X_1, \ldots, X_p)$ have the covariance matrix $\mathbf{\Sigma}$. Let $\mathbf{\Sigma}$ have the eigenvalue-eigenvector pairs $(\lambda_1, \mathbf{e}_1), \ldots, (\lambda_p, \mathbf{e}_p)$ where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p \geq 0$. Then the jth principal component is given by

$$Y_j = \mathbf{e}_j'\mathbf{X} = e_{j1}X_1 + \cdots e_{jp}X_p, \qquad j = 1, \ldots, p.$$

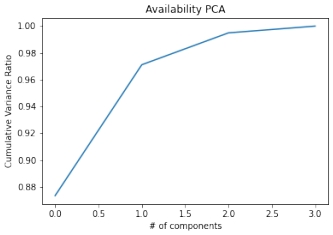
If some λ_j are equal, the corresponding eigenvectors and hence, Y_j are not unique. The signs within each eigenvector can be reversed.
The variance explained by p principal components equals the total population variance:

$$\sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{j=1}^{p} \sigma_{jj} = \sum_{j=1}^{p} \lambda_j = \sum_{j=1}^{p} (Y_j).$$

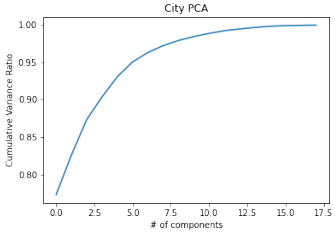
The proportion of variance explained by the kth PC: $\dfrac{\lambda_k}{\sum_{j=1}^{p} \lambda_j}$.
If a few PCs can explain most of the total population variance, then those components can considered instead of the many original variables without much loss of information.



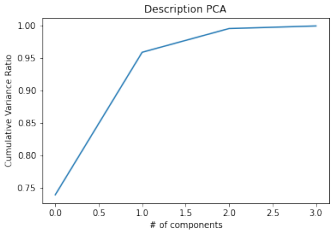
(a) Accommodation size PCA



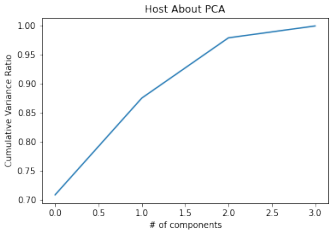
(b) Availability PCA



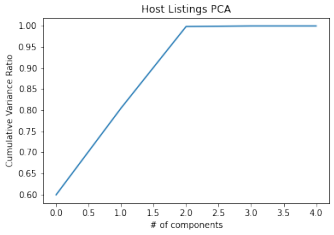
(c) City Life PCA



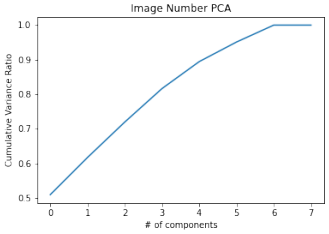
(d) Description size PCA



(e) Host About PCA

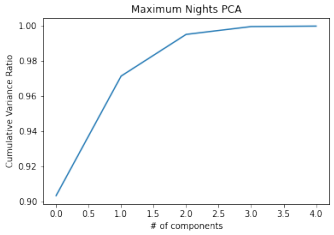


(f) Host listings PCA

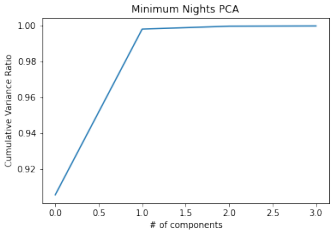


(g) Image Number PCA

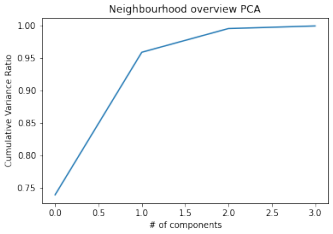
(h) Kitchen PCA



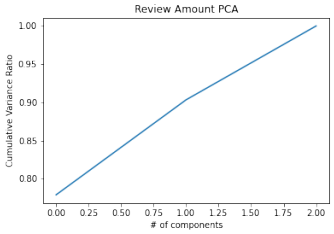
(i) Maximum Nights PCA



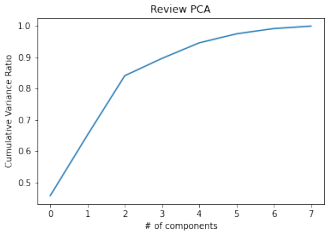
(j) Minimum Nights PCA



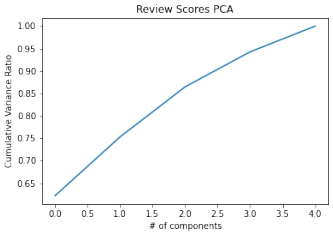
(k) Neighbourhood PCA



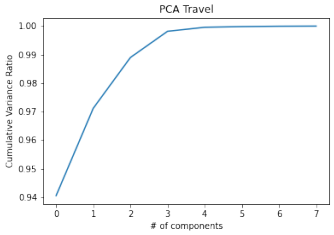
(l) Review amount PCA



(m) Review PCA



(n) Review Scores PCA



(o) Travel Touristic PCA

Figure 19: Cumulative explained Variance of PCAs (Variable overview: Table 12).

Table 12: Variables used for PCAs

PCA	n_C **used**	**Variables**
"accommodation_size"	7	"bedrooms_1", "bedrooms_2", "accommodates", "beds" "room_type_Entire_home/apt", "room_type_Private_room", "bath_number_1", "bath_number_2", "bath_kind_Shared", "bath_kind_Private", "bath_kind_Normal", "property_type_Entire_residential_home", "property_type_Entire_rental_unit", "property_type_Private_room_in_rental_unit" "property_type_Private_room_in_residential_home"
"availability"	1	"availability_365", "availability_30", "availability_60", "availability_90"
"city$_life$"	5	"nightclubs", "sex_amenities", "bicycle_rentals", "casinos", "university", "theatres_artscentre", "library", "taxi", "fast_foods", "restaurants", "bars", "cafes", "malls", "cinemas", "supermarkets", "kiosks", "bus_train_tram_station", "social_amenities"
"descr"	2	"compound_descr", "positivity_descr", "description_length", "neutrality_descr"
"host_about"	2	"compound_host_ab", "positivity_host_ab", "host_about_length", "neutrality_host_ab"
"host_listings"	3	"calculated_host_listings_count", "host_listings_count", "calculated_host_listings_count_private_rooms", "calculated_host_listings_count_entire_homes", "calculated_host_listings_count_shared_rooms"
"image_number"	5	"no_img_others", "no_img_hallway", "no_img_dining", "no_img_bathroom", "count", "no_img_bedroom", "no_img_kitchen", "no_img_living"
"kitchen"	6	"Microwave", "Dishes_and_silverware", "Refridgerator_available", "Dishwasher", "Stoves_available", "Cooking_basics", "Oven_available", "Kitchen_available", "Hot_water"
"max_nights"	1	"maximum_nights", "minimum_maximum_nights", "maximum_maximum_nights", "maximum_nights_avg_ntm", "Long_term_stays_allowed"

"min_nights"	1	"minimum_nights", "minimum_minimum_nights", "maximum_minimum_nights", "minimum_nights_avg_ntm"
"neigh_over"	2	"compound_neigh_over", "positivity_neigh_over", "neighborhood_overview_length", "neutrality_neigh_over"
"review_amount"	2	"number_of_reviews_l30d", "number_of_reviews_ltm", "reviews_per_month"
"review_total"	4	"review_scores_rating", "mean_compound", "most_pos_compound", "mean_positivity", "mean_neutrality", "mean_negativity", "most_neg_compound", "prop_of_neg_comp"
Review Scores	not used	"review_scores_location", "review_scores_accuracy", "review_scores_communication", "review_scores_cleanliness", "review_scores_value"
"travel_touristic"	1	"neighbourhood_cleansed_Dublin_City", "in_city", "nearest_sight", "mean_dist_sight", "2nd_nearest_sight", "3rd_nearest_sight", "nearest_travel_poss", "mean_dist_travel"

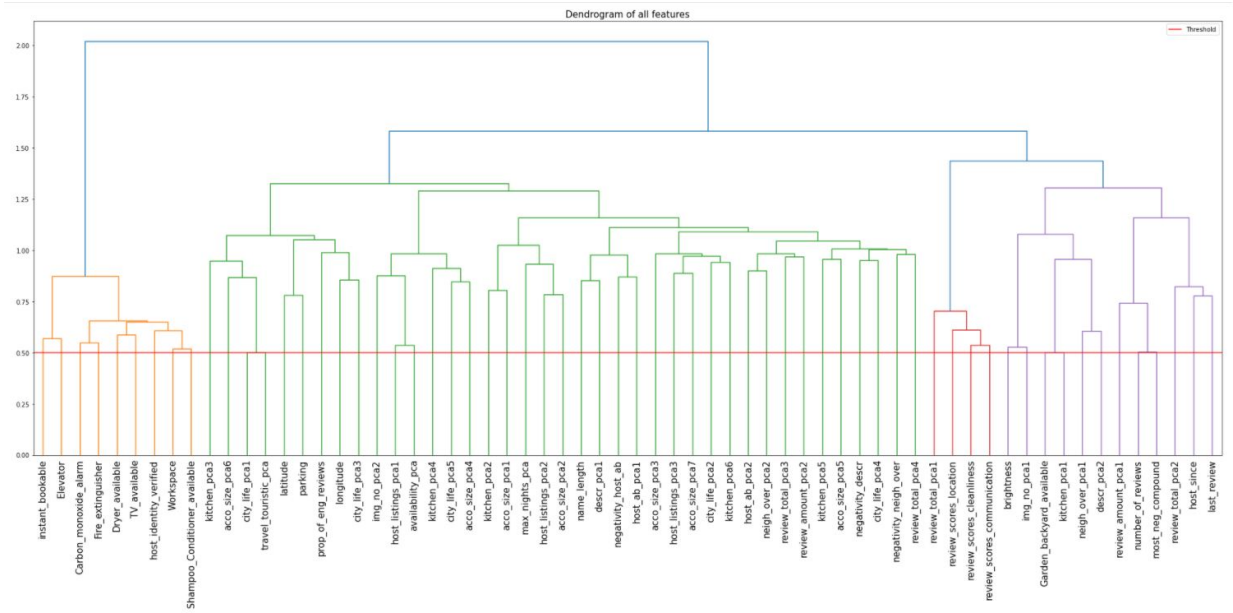


Figure 20: Dendrogram of Variable Correlation Clusters using Ward's Linkage after Decorrelation

A.2.2 Models

XGBoost The individual decision trees in the XGBoost algorithm that make up the boosting model itself are referred to as "weak learners". The individual predictions of the residuals $\tilde{y}_1, ..., \tilde{y}_N$ are summed up from weak learner to weak learner to form the overall prediction $\hat{y}$ where the learning rate η plays a crucial role. As can be seen in equation 4, after the n-th iteration only a portion η of the prediction of the n-th weak learner is added to the overall prediction $\hat{y}_n$ for the n-th iteration.

$$\hat{y}_n = \hat{y}_{n-1} + \eta * \tilde{y}_n \tag{4}$$

This prevents the model from overfitting after just a few iterations by allowing several weak learners to jointly explain the residuals. The loss function in 5 of the XGBoost algorithm consists essentially of two parts. The first part pursues an estimate of y that is as accurate as possible - in the case of regression the mean squared error - and an additional regularization part that regulates the generalization of the model. This regularizer aims to control the number of leaves T in the trees, since deep trees tend to overfit, and regularizes the weights w of the leaves themselves . The weight w_{nj} of a leaf j in the weak learner n corresponds to $\tilde{y_n}$ for all x_i assigned to leaf j. By penalizing high values for the weights e.g. by a L2-norm penalty, the algorithm counteracts the overfitting of the individual weak learners. The regularization parameter γ is used to regularize the number of leaves, and λ to regularize the weights.

$$L(\mathbf{y}, \hat{\mathbf{y}}) = MSE(\mathbf{y}, \hat{\mathbf{y}}) + \sum_n \gamma T + \frac{1}{2}\lambda ||\mathbf{w_n}||^2 \tag{5}$$

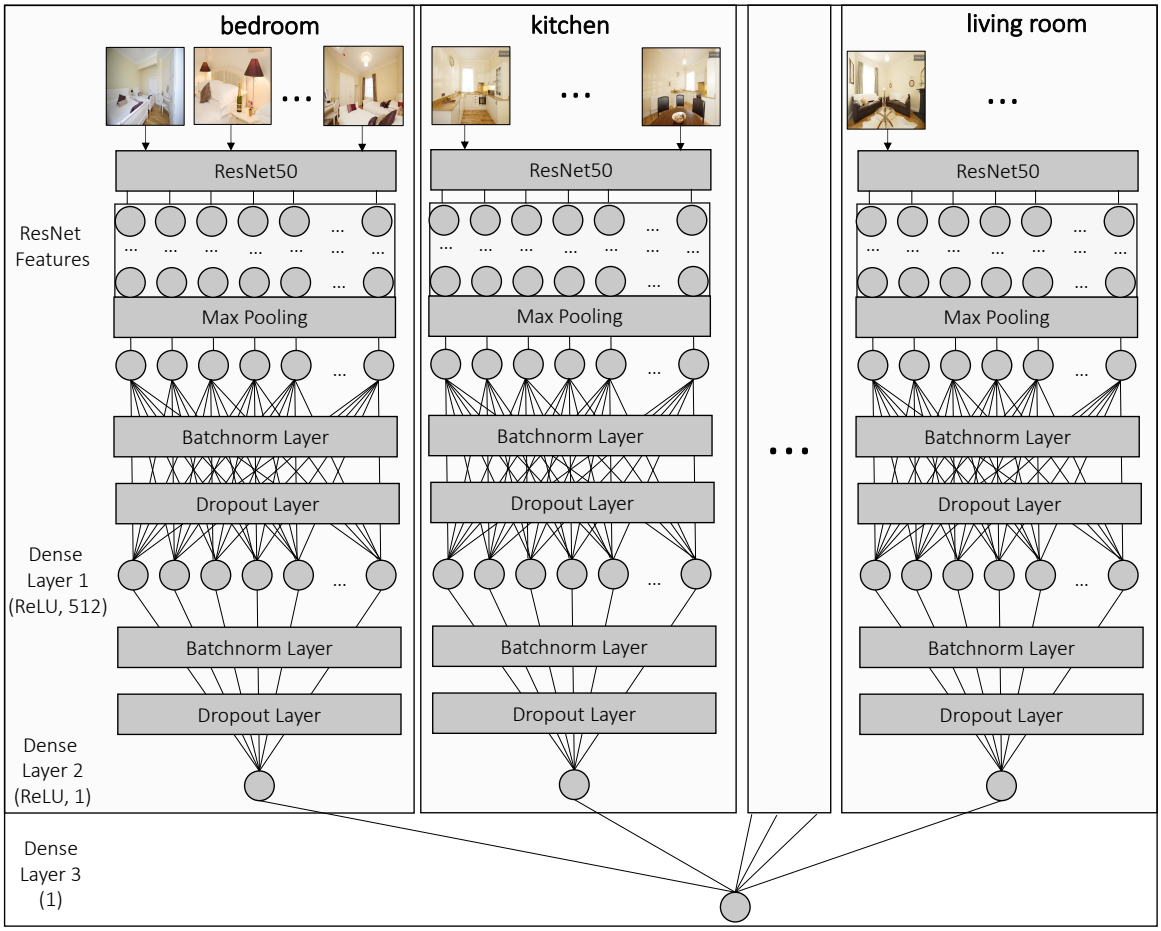


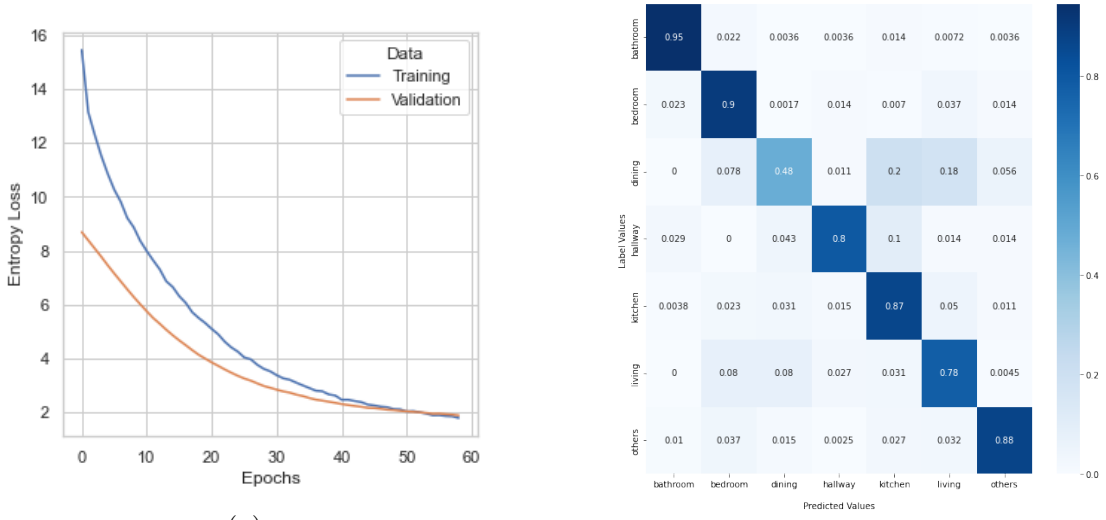
Figure 21: Model Architecture of Image Model

A.3 Results

A.3.1 Room Classification

Table 13: Grid Search Hyperparameters for Image Model

Hyperparameter	Values
learning rate	{0.001, 0.0001, 0.00001}
dropout	{0, 0.2, 0.5}
L2 penalty	{0, 0.1}



(a) (b)

Figure 22: Training Curves for Training and Validation Data (a) and Confusion Matrix of Test Data (b) for Room Classification Model

Figure 23: Examples for Misclassified Images from the Test Data

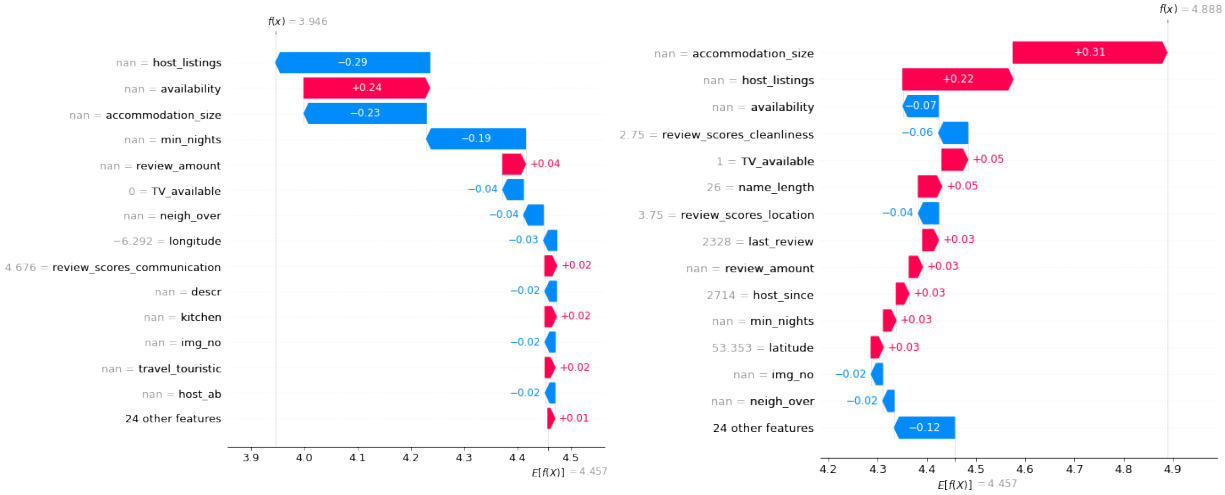
A.3.2 Price Prediction

Table 14: Grid Search Hyperparameters for XGBoost Model

Hyperparameter	Values
n_estimators	{100, 250, 500}
colsample_bytree	{0.1, 0.5, 0.9}
subsample	{0.5, 0.7, 0.9}
max_depth	{3, 10, 50}
gamma	{0, 1, 10}
reg_lambda	{0, 1, 10}

Table 15: Accommodation Size Variables for Example Listings A and B

Variable	Listing A	Listing B
room_type_Private_room	1.0	0.0
room_type_Entire_home/apt	0.0	1.0
bath_number_1	1.0	1.0
bath_number_2	0.0	0.0
bath_kind_Shared	0.0	0.0
bath_kind_Private	1.0	0.0
bath_kind_Normal	0.0	1.0
bedrooms_1	1.0	0.0
bedrooms_2	0.0	1.0
accommodates	1.0	6.0
beds	1.0	2.0
property_type_Private_room_in_residential_home	0.0	0.0
property_type_Entire_rental_unit	0.0	0.0
property_type_Private_room_in_rental_unit	0.0	0.0
property_type_Entire_residential_home	0.0	1.0



(a) (b)

Figure 24: SHAP Values of two Example Listings for XGBoost Model

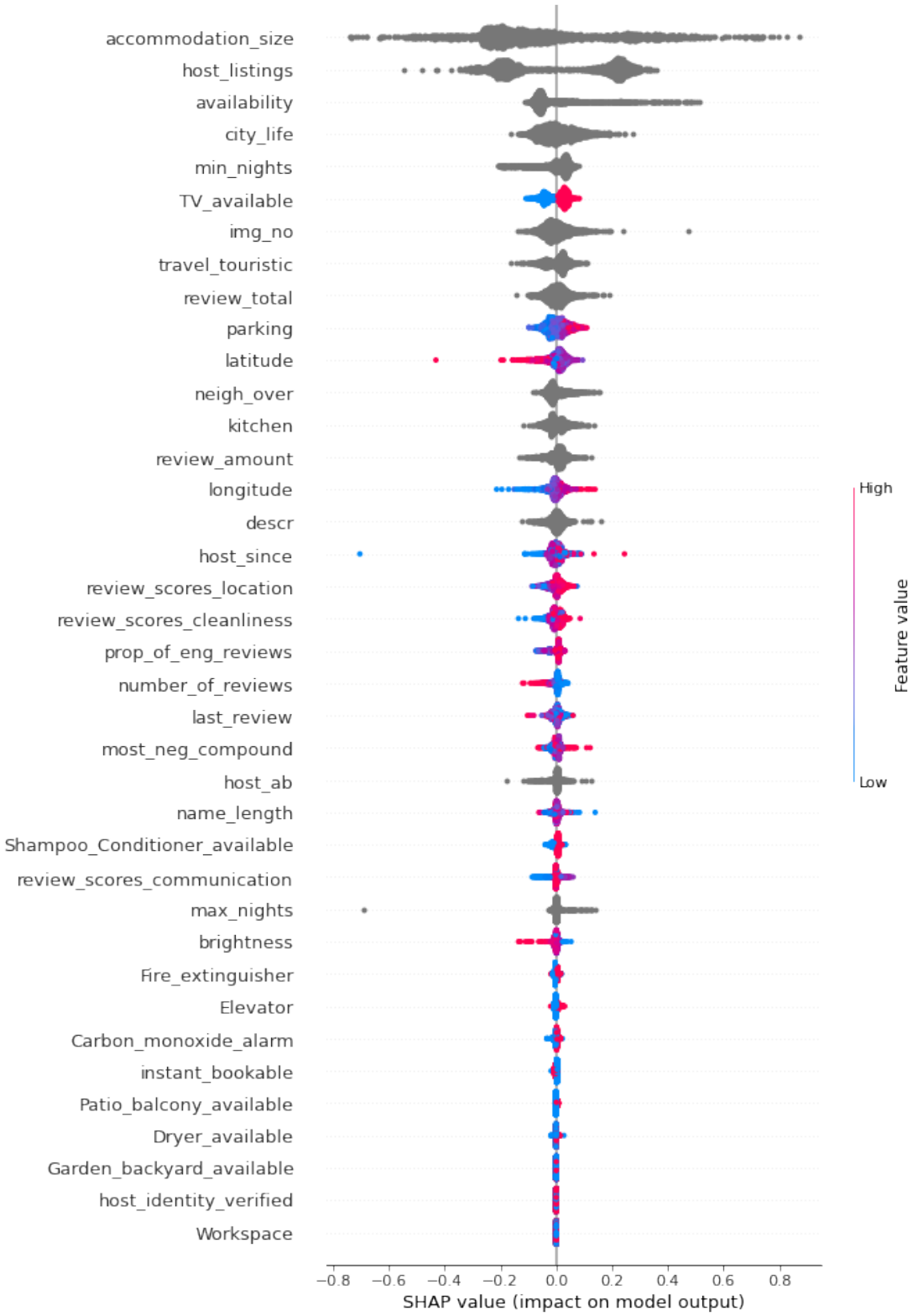
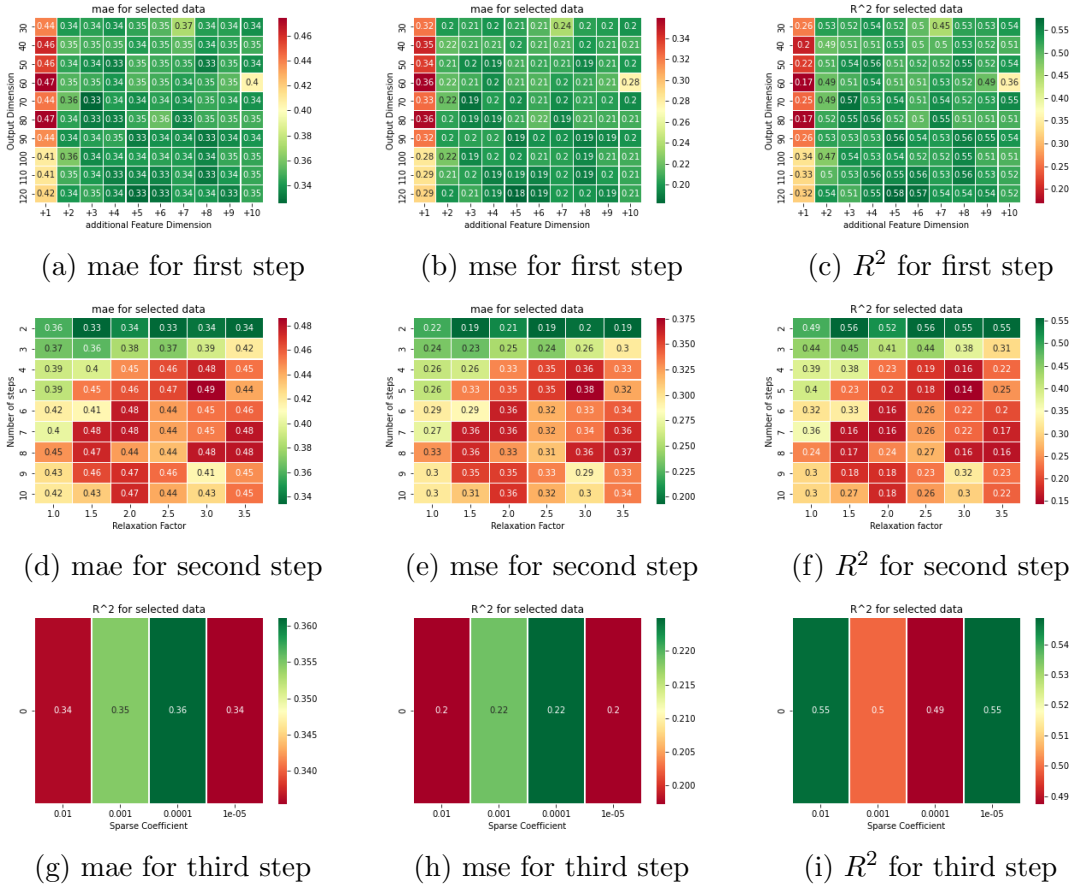


Figure 25: SHAP Values for Variables and PCA Blocks Across all Listings Ordered by their Absolute Mean SHAP Value for XGBoost Model

Table 16: Grid Search Hyperparameters for the TabNet over 50 Epochs

Hyperparameter	Values
output dimension N_d	{30, 40, 50, 60, 70, 80, 90, 100, 110, 120}
additional feature dimension N_a	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
number of decision steps N_{steps}	{2, 3, 4, 5, 6, 7, 8, 9, 10}
relaxation factor γ	{1, 1.5, 2, 2.5, 3, 3.5}
sparsity coefficient λ_{sparse}	{1e-2, 1e-3, 1e-4, 1e-5}



(a) mae for first step (b) mse for first step (c) R^2 for first step

(d) mae for second step (e) mse for second step (f) R^2 for second step

(g) mae for third step (h) mse for third step (i) R^2 for third step

Figure 26: Stepwise Grid Search for the TabNet with Uncorrelated Features.

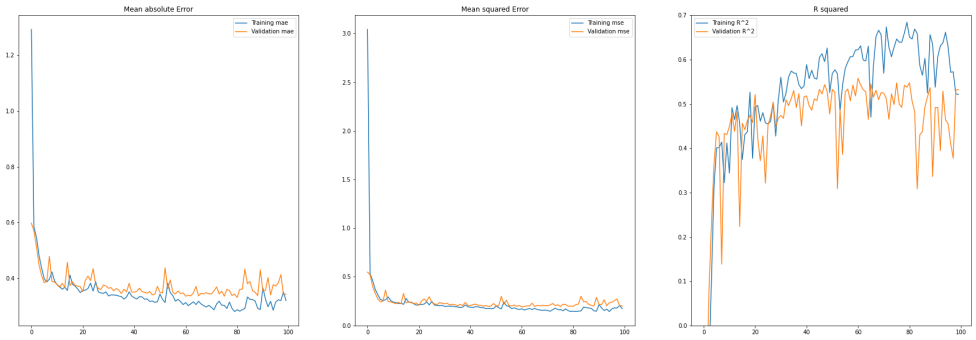
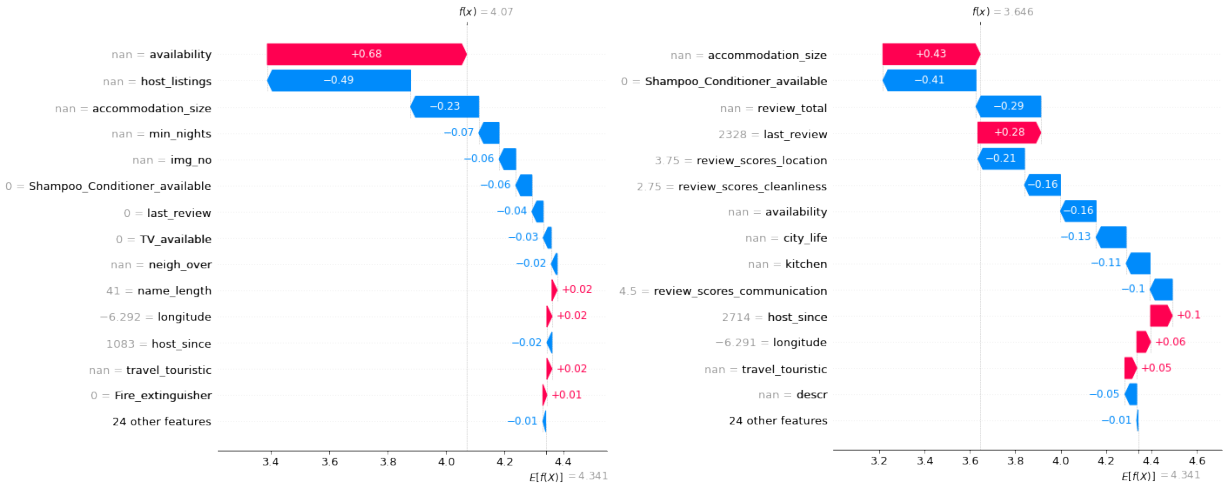


Figure 27: History of the final TabNet for Uncorrelated Features.



(a) (b)

Figure 28: SHAP Values of two Example Listings for TabNet Model

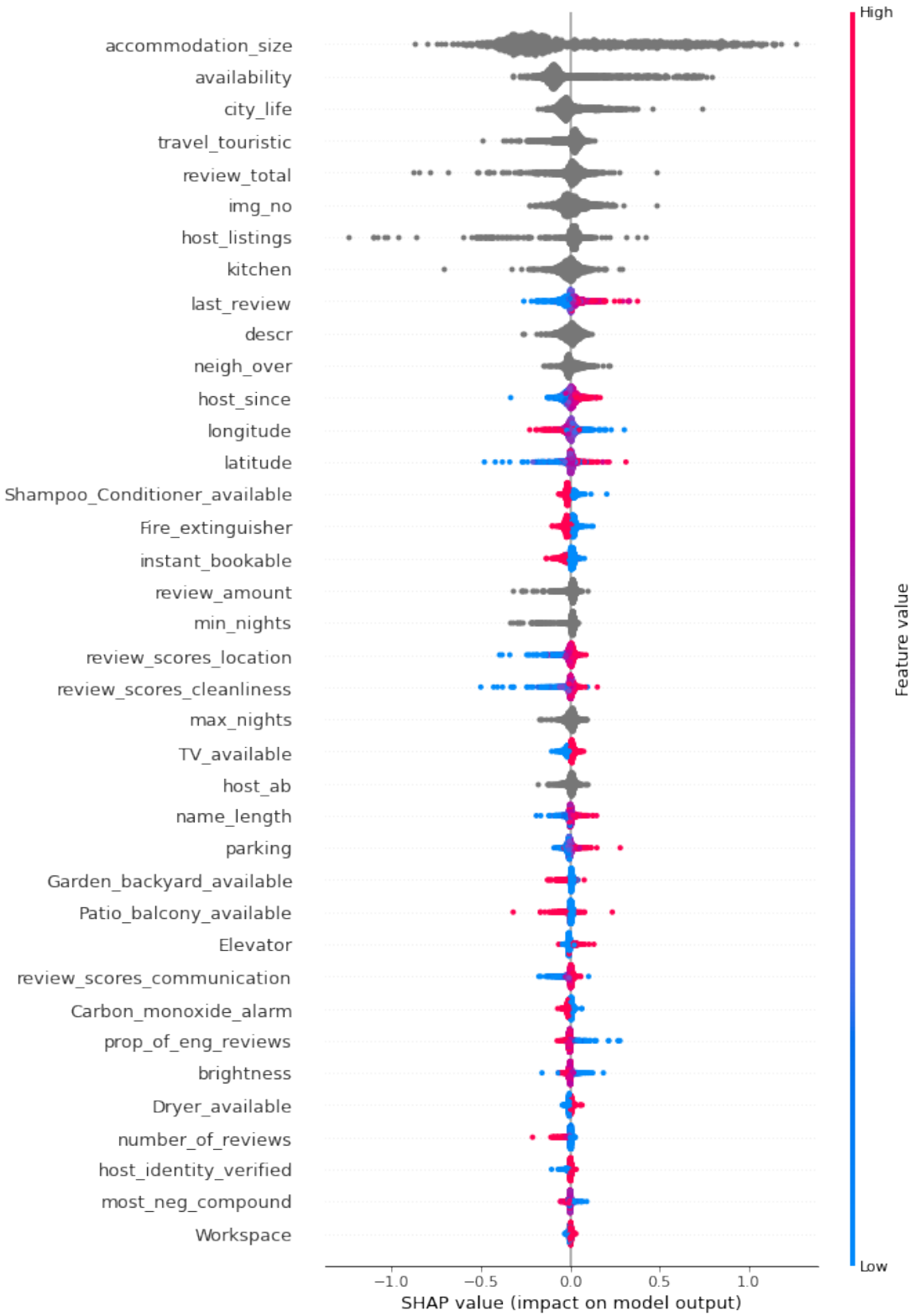


Figure 29: SHAP Values for Variables and PCA Blocks Across all Listings Ordered by their Absolute Mean SHAP Value for TabNet Model

Figure 30: TabNet Grid Search Results with correlated Variables and History of the best Model

Table 17: Grid Search Hyperparameters for Image Model

Hyperparameter	Values
dropout rate	{0, 0.2, 0.5}
L2 regularization	{0.0, 0.01, 0.001}
nodes	{64, 512}

Table 18: Grid Search Results for Image Model

Dropout	L2	Nodes	Val. MSE	Val. R^2	Train MSE	Train R^2
0.0	0.000	64	0.340157	0.158453	0.009895	0.975213
0.0	0.000	512	0.320462	0.212454	0.008252	0.979390
0.0	0.010	64	0.472604	0.156609	0.195085	0.780643
0.0	0.010	512	0.553412	0.196390	0.293380	0.717599
0.0	0.001	64	0.400306	0.159742	0.085048	0.905126
0.0	0.001	512	0.423162	0.165022	0.115019	0.887467
0.2	**0.000**	**64**	**0.308538**	**0.251441**	**0.299438**	**0.272605**
0.2	0.000	512	0.305444	0.256064	0.286204	0.304451
0.2	0.010	64	0.625010	0.227250	0.756768	-0.063806
0.2	0.010	512	0.896431	0.215529	1.093091	-0.176111
0.2	0.001	64	0.487127	0.234388	0.518781	0.145123
0.2	0.001	512	0.609311	0.235500	0.664445	0.105202
0.5	0.000	64	0.343687	0.167180	0.748588	-0.830711
0.5	0.000	512	0.343053	0.163491	0.713657	-0.742474
0.5	0.010	64	0.809308	0.091218	1.274529	-1.046673
0.5	0.010	512	1.259100	0.089910	1.752104	-0.990438
0.5	0.001	64	0.606932	0.117446	1.017460	-0.881257
0.5	0.001	512	0.817512	0.120507	1.257922	-0.914159

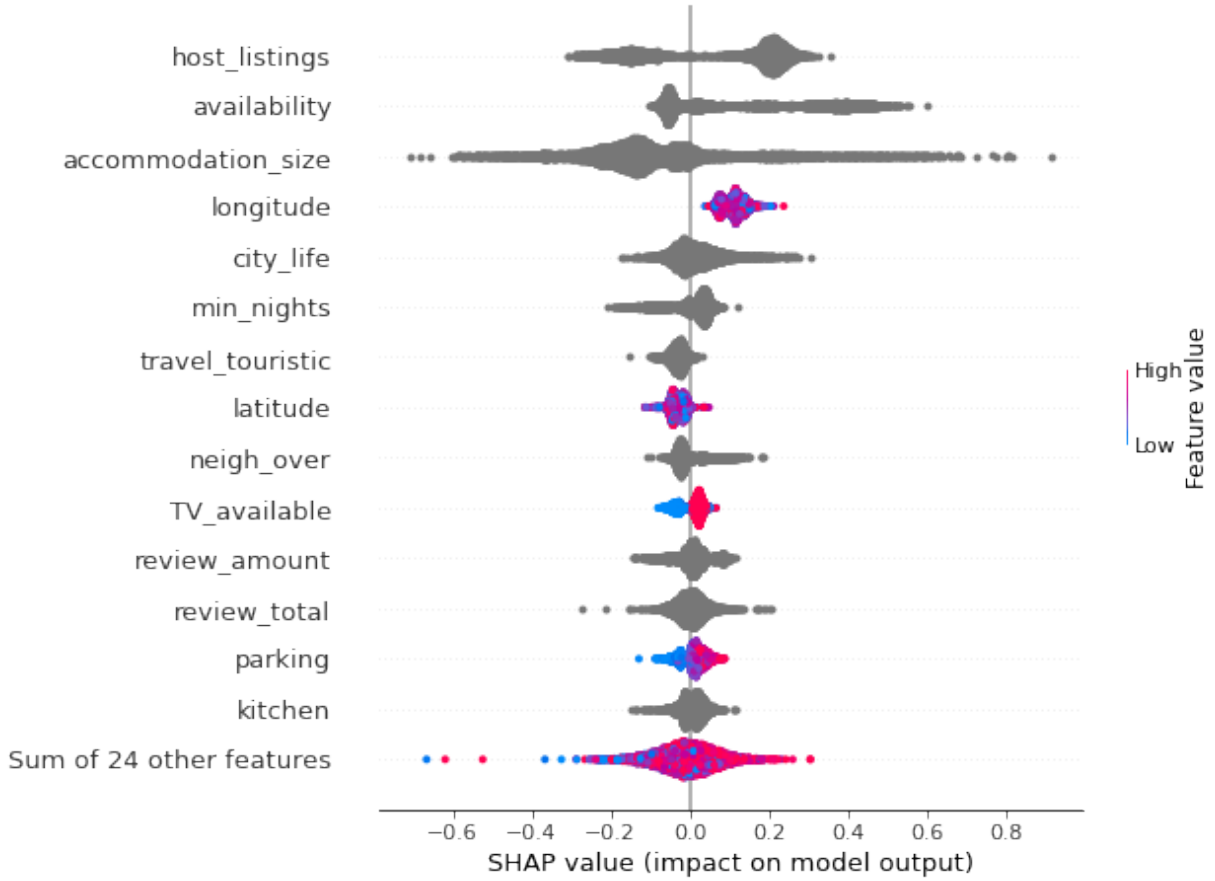


Figure 31: Aggregated SHAP Values of the 15 most important Variables and PCA Blocks across all Listings for Munich from the XGBoost Model

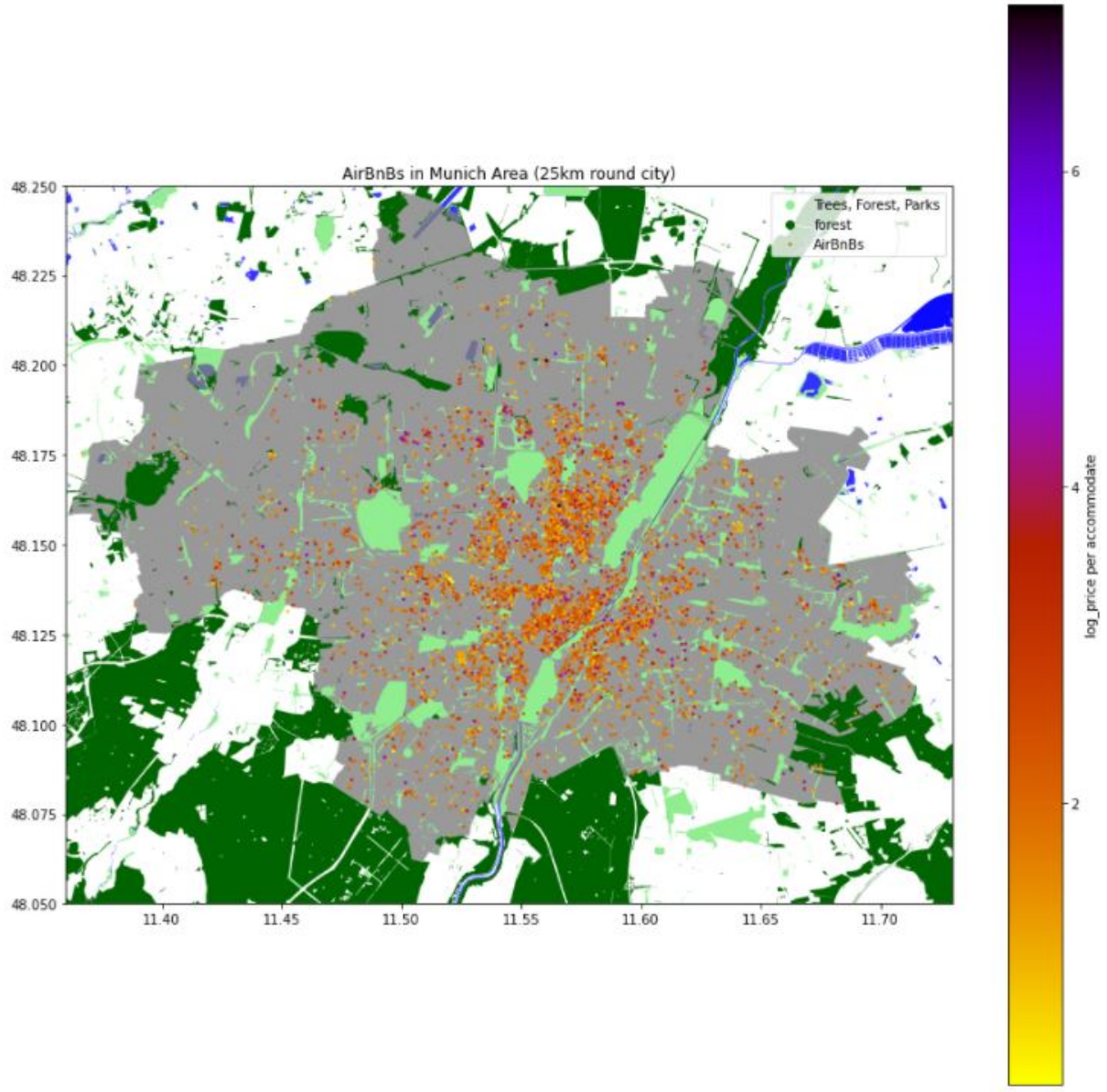


Figure 32: City of Munich with log Price per Accommodate.

A.4 Another Image Based Approach

In addition to the image based model presented in Section 2.2.2 we also tried another approach utilizing the scraped images. For this model we used the first 5 images of each listing side by side as a gallery view as input. An example of a gallery view for a listing is given in Figure 33 at the top. Listings with missing image are imputed using black images as replacement. A gallery view can be understood as the first visual impression of the apartment when visiting the listing page on textitAirBnB. The log price as target variable is replaced by a discretized version of it to transform the problem into a classification for price categories instead of estimating the exact log price. This discretization is done using

the 5, 25, 75, 95 and 100 percent quantiles in the respective accommodation sizes of the listings. As there are only a few observations for listings that can accommodate more than 6 people these observations were aggregated and their log price was divided by their capacity to obtain the quantiles. The goal of this discretization is to relativize the influence of the accommodation size on the price. The model itself first extracts image features from the gallery views using a ResNet50. These are then processed using an FC layer with 128 nodes and ReLU activation function which is followed by another FC layer with 5 nodes and a softmax activation function, where each node represents one of the 5 price classes. The model uses dropout of 0.5 and L2 regularization of 0.01 for the weights of the FC layers. Moreover batch normalization layers are placed between the dense layers and after the feature extraction (see Figure 33). The model was trained using an exponential learning rate decay of 0.95 with an adjustment every 500th iteration step and inital learning rate of 0.001. Dropout, L2 regularization and the learning rate were optimized using grid search with three different values each.

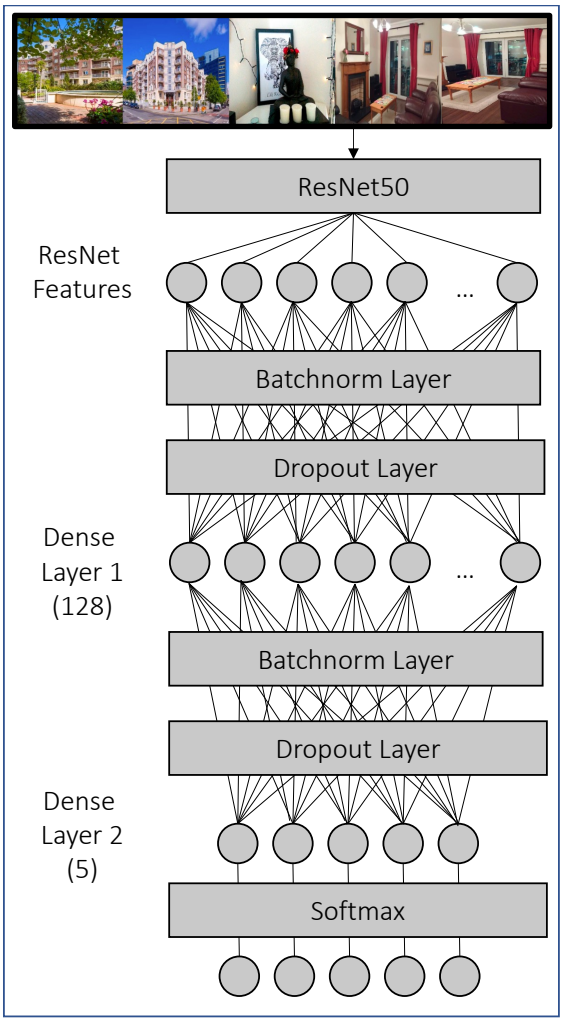


Figure 33: Architecture of the "Gallery" Model

The results in the form of a confusion matrix are shown in Figure 34. It becomes quickly visible that the model does not achieve the desired performance. Therefore the approach was not pursued further.

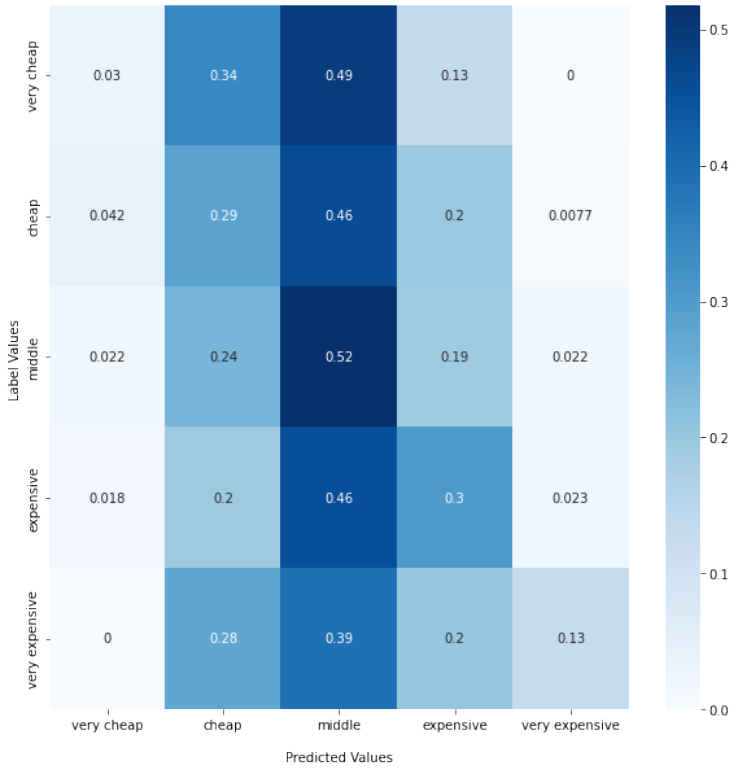


Figure 34: Confusion Matrix for Gallery Model

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer.

Boeing, G. (2917). Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, 65:126–139.

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, and Kasneci, G. (2021). Deep neural netwoks and tabular data: A survey. *arXiv:2110.01889v1*.

Burch, N. (2021). Traditional irish names. http://www.namenerds.com/irish/trad.html. [Online; accessed last 10-February-2022].

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Dauphlin, Y. N., Fan, A., Auli, M., and Grangier, D. (2016). Language modeling with gated convolutional networks. *arXiv:1612.08083*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Fischmann, S., Carvalho, E., Abiteboul, J., Raquet, Q., Grankin, M., Thewsey, A., and Abeywardana, S. (2020). Pytorch implementation of tabnet paper. [https://github.com/dreamquark-ai/tabnet].

FOSSGIS e.V. (2004). Osm. [https://www.openstreetmap.de].

Freedman, D., Pisani, R., and Purves, R. (2007). Statistics (international student edition). *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*.

Goswami, A., Chittar, N., and Sung, C. H. (2011). A study on the impact of product images on user clicks for online shopping. In *Proceedings of the 20th international conference companion on World wide web*, pages 45–46.

Hart, S. (1989). *Shapley Value*, pages 210–216. Palgrave Macmillan UK, London.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Heo, C. Y. and Hyun, S. S. (2015). Do luxury room amenities affect guests' willingness to pay? *International Journal of Hospitality Management*, 46:161–168.

Hinz, T. and Auspurg, K. (2017). Diskrimierung auf dem Wohnungsmarkt. pages 387–406. Springer.

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441.

Huggins, J. (2008). Selenium module.

Hutto, C. and Gilbert, E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text. *Proceedings of the International AAAI Conference on Web and Social Media, 8(1)*, pages 216–225.

Ilse, M., Tomczak, J., and Welling, M. (2018). Attention-based deep multiple instance learning. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2127–2136. PMLR.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

Jaccard, P. (1912). The distribution of the flora in the alpine zone.1. *New Phytologist*, 11(2):37–50.

Jayalakshmi, T. and Santhakumaran, A. (2011). Statistical normalization and back propagation for classification. *International Journal of Computer Theory and Engineering*, 3(1):1793–8201.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., and Lee, S.-I. (2020). From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2(1):2522–5839.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Majumdar, S., Jablons, Z., and Tamazian, A. (2019). A tensorflow 2.0 implementation of tabnet. [https://github.com/titu1994/tf-TabNet].

Martins, A. F. T. and Astudillo, R. F. (2016). From softmax to sparsemax: A sparse model of attention and multi-label classification. *arXiv:1602.02068v2*.

Michelson, A. (1927). Studies in optics," p. 135. univ. of chicago press, chicago, illi-nois.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Icml*.

Name-Census (2021). Name census top 100 name list. https://www.kaggle.com/namecensus/name-census-top-100-name-list. [accessed last 23-January-2022].

Nowak, A. S. and Radzik, T. (1994). The shapley value for n-person games in generalized characteristic function form. *Games and Economic Behavior*, 6(1):150–161.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572.

Prechelt, L. (1998). Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer.

Richardson, L. (2007). Beautiful soup module.

Rosenblatt, F. (1961). Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY.

Schulz, E., Speekenbrink, M., and Krause, A. (2018). A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematial Psychology*, 85:1–16.

Sercan, A. and Pfister, T. (2019). Tabnet: Attentive interpretable tabular learning. *arXiv:1908.07442*.

Shwartz-Ziv, R. and Armon, A. (2021). Tabular data: Deep learning is not all you need. *arXiv:2106.03253v2*.

Singh, D. and Singh, B. (2020). Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, 97:105524.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Tate, R. F. (1954). Correlation between a discrete and a continuous variable. point-biserial correlation. *The Annals of mathematical statistics*, 25(3):603–607.

Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.

Welch, B. L. (1947). The Generalization of 'student's' Problem when several different population variances are involved. *Biometrika*, 34(1-2):28–35.

Wilson, T., Wiebe, J., and Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, pages 347–354.