Combining Images and Tabular Data Using Deep Learning for Classification Tasks

Nicole Van de Weijer[*]

Vrije Universiteit Amsterdam, The Netherlands
MSc. Artificial Intelligence

Abstract. Unstructured real-world data can take numerous forms and typically includes both textual and visual material. This study examines whether combining data from different modalities enhances the performance of deep learning-based classification models. Clappform, a software company, provided data consisting of tabular data and images of real estate properties. For this task, two deep learning models were selected: TabNet for the tabular data and EfficientNet for the images. Contrary to expectations, the model that included both modalities did not outperform the models that concentrated on a single modality. The results suggest that more research is needed.

Keywords: Deep Learning · Convolutional Neural Networks · Binary Classification · Multi-Modality.

[*] Supervisors: Michael Cochez, Jeroen Schoonderbeek and Taewoon Kim,
Studentnumber: 2550019,
Email: n.d.vande.weijer@student.vu.nl

Table of Contents

1 Introduction . . . 4
2 Related Work . . . 5
    2.1 Image Classification . . . 5
        2.1.1 Image Classification in Real Estate . . . 6
    2.2 Tabular Classification . . . 6
        2.2.1 Tabular Classification in Real Estate . . . 6
    2.3 Combining Images and Tabular Data . . . 7
        2.3.1 Combining Images and Tabular Data in Real Estate . . . 7
3 Data Description and Preparation . . . 8
    3.1 Data Description . . . 8
    3.2 Data Exploration . . . 8
        3.2.1 Images . . . 8
        3.2.2 Feature Selection . . . 9
        3.2.3 Missing Values . . . 10
        3.2.4 Outliers . . . 10
    3.3 Preprocessing . . . 11
        3.3.1 Images . . . 11
        3.3.2 Tabular data . . . 11
4 Methodology . . . 12
    4.1 Train, Validation and Test Size . . . 12
    4.2 Part 1: Convolutional Neural Networks (CNNs) . . . 13
        4.2.1 Room Type Classifier . . . 15
        4.2.2 Maintenance Status Classifier . . . 15
        4.2.3 Performance Metrics . . . 15
    4.3 Part 2: TabNet . . . 16
    4.4 Part 3: Combination . . . 16
    4.5 Macro F1 . . . 17
    4.6 Hyperparameter Tuning . . . 17
5 Results . . . 18
    5.1 Results Image Classification . . . 18
        5.1.1 Default . . . 18
        5.1.2 Tuned . . . 20
    5.2 Results Tabular Classification . . . 22
        5.2.1 Default . . . 22
        5.2.2 Tuned . . . 24
    5.3 Results Combination . . . 26
        5.3.1 Default . . . 26
        5.3.2 Tuned . . . 27
6 Discussion . . . 29
    6.1 Discussion of the Results . . . 29
    6.2 Limitations . . . 32

6.2.1 Dataset 32
6.2.2 Computational Power 32
7 Future Work 32
8 Conclusion 33
9 Acknowledgments 33

1 Introduction

Although the world is full of multimodal data, we nevertheless process each modality independently. Unstructured real-world data can take many forms, also known as modalities, and frequently include visual and textual material. Over the last decade, there has been a greater emphasis on merging data from several modalities in order to improve machine learning-based classification models [22].

Multimodality is defined by Lahal *et al.* as a system seen by numerous sensors. The goal of multimodality is to extract and combine significant data from individual sensors, then utilise this combined feature to solve a problem. As a result, the predicted output will be more representative and perform better than the individual modalities [29].

Neural networks, one of the most well-known machine learning models, have played an important role in recent years due to their capacity to train with high accuracy. Deep learning has become a popular study topic in both academia and industry, owing to its superior performance when compared to traditional machine learning models [11]. Deep learning has been shown to be successful with a single modality dataset, however, it still struggles with dealing with multimodal data.

In this study, the performance of deep learning models with a single modal dataset is compared to that of a deep learning model using multimodal data. This research is carried out in collaboration with Clappform. Clappform is a software firm that provides a cloud-based platform that allows advanced analytics through the use of Artificial Intelligence (AI) and Machine Learning (ML) technologies applied in various industries. Clappform's major focus is on integrating such technologies into its clients' business processes and workflows. This study relied on property data from the real estate industry.

Each property listing includes a number of images as well as a table of characteristics. In this digital era, people do not want to waste time browsing through property listings just to find something that does not match their demands. Besides the location, number of bedrooms, and outdoor area, the maintenance status of a property is considered important in most real estate transactions. The maintenance status, however, is a variable that frequently exhibits a subjective bias and is not exactly and objectively defined. Therefore, an attempt is made to predict the maintenance status using the provided images and corresponding tabular data.

To verify the maintenance status reported by real estate agents, computer vision models can extract information from the images of the listing. Computer vision models are created to translate visual input using features and contextual data obtained during training [38]. In this study, a Convolutional Neural Network (CNN) is used to represent a computer vision model [4]. This gets us to the study's first research question:

RQ1: Can a deep learning based computer vision model learn to classify the maintenance status of real estate images?

After the CNN model's performance has been confirmed, Tabular Classification is used to try to validate it. First, the data will be transformed into suitable input for a tabular learning deep learning model. This model, like the previous one, is used to predict a property's maintenance status. This gets us to the study's second research question:

RQ2: What is the performance of tabular deep learning on tabular real estate data?

Finally, the combination of the computer vision model and the tabular deep learning model is examined. This gets us to the third research question:

RQ3: How does the performance change when the models are combined?

The following will be covered in the next sections. First, related work will be discussed. The data description and preparation will be discussed in the third section. The methodology for this research will be detailed in the fourth section, followed by the results of the experiments in the fifth section. Finally, the conclusions and discussion are presented in the sixth and seventh section.

2 Related Work

Since there is such an abundance of data on the real estate markets and transactions, artificial intelligence and machine learning may be quite useful in processing and evaluating the data. The amount of research that has been done on it confirms this, notably predicting property prices, which has received a lot of attention. In the process, evolutionary algorithms, as well as numerous other machine learning algorithms, such as support vector machine (SVM) and random forest (RF), as well as Neural Networks and Deep learning, are utilized, with positive results [5,23,40]. Not only does research look into predicting property prices, but it also delves into other areas. Generating textual descriptions for floor plan images, for example. Goyal *et al.* [15] have been working on it, and has used a transformer and a recurrent neural network to get it accomplished. There are also a number of studies where images are used. Since we acquired a dataset containing both images and tabular data, both will be used in this study. This will be covered in further detail in the following sections.

2.1 Image Classification

The task of assigning a label or class to an entire image is known as image classification. The purpose is to predict a class, given an image. CNNs are commonly employed for image-related tasks owing to the utilization of convolutional layers. It considers spatial features (i.e., the location of the pixels and their neighbors), which are critical for capturing valuable features for visual tasks. The discovery that a CNN could be used to progressively extract higher-level representations of the visual information was a breakthrough in creating models for image classification. Rather than preprocessing the data to extract features such as textures and forms, a CNN uses the image's raw pixel data as input and learns how to extract these features and, in turn, infer what object they represent [12,32].

2.1.1 Image Classification in Real Estate Several previous studies on related topics do not explicitly simulate property maintenance status, but do show that images of a property can be used to estimate relevant characteristics. You *et al.* are the first, to the best of our knowledge, to measure the influence of visual material on real estate price estimation [42]. In their implementation, they used GoogleNet [37], a CNN that is 22 layers deep. In terms of mean absolute error and mean absolute percentage error, the experimental findings show that the model outperforms numerous existing state-of-the-art baseline techniques.

Another study [27], focusing on the attractiveness of a property, used the ResNet50 model [17]. The results clearly reveal that the CNN performed poorly when used with their real-estate data. Pre-trained networks, which were employed for classification issues, yielded more promising outcomes. The ResNet152-hybrid1365 [44] architecture was used for this, which was trained using ImageNet [13] and Places365 [45] datasets. As a result, they proved the usefulness of image features in modeling real-estate attractiveness.

Other research focuses entirely on estimating property prices using images of the urban environment at both street and sky level, rather than images of a property's interior. The model is based on the standard CNN architecture, which employs 3x3 filters that are evaluated on 4, 8, and 13 convolutional layers before proceeding to a series of pooling layers. This research finds promising results in forecasting property prices in London [30].

Furthermore, research is being conducted to provide predictions regarding the building's maintenance status using exterior images. Law *et al.*'s findings [26] suggest that visually estimated building condition may be used as a proxy for appraisers' condition estimations. They created a multiscale patch-based pattern extraction method and combined it with CNNs to predict building maintenance status based on visual inputs.

2.2 Tabular Classification

The task of assigning a class to samples of structured or relational data is known as tabular classification. It requires the use of a categorical column as the target. Tree ensemble models (like Random Forest [10]) are frequently advised when handling classification and regression tasks with tabular data. Several Deep Learning (DL) models for tabular data, on the other hand, have lately been presented, claiming to outperform them in various use cases [6]. Although DL models and tree ensemble models have been compared quite often, no generally superior method has been found [14].

2.2.1 Tabular Classification in Real Estate To the best of our knowledge, no research has been done using tabular data to predict the maintenance status of a property. Other studies on real estate and tabular data, however, have been conducted.

As said before, multiple studies have been done on predicting property prices. The number of articles and blogs written about the subject confirms this. This has also only been studied using tabular data. Some academics concentrate on a single model. A study used a Deep Neural Network to predict boarding property rental prices [7]. Another study looks into using the RF approach to predict property prices [1]. On the other hand, other researchers combine models to get a better outcome. This has been addressed by Afonso *et al.*'s study. They combined DL and Random Forest and discovered that the combined model gave better outcomes than either model separately [2].

Aside from predicting property prices, researchers have looked at classifying building types using tabular data. The building types were classified in two steps throughout the classification procedure. The first challenge required classification in order to distinguish between residential and non-residential buildings. The second step included classifying properties (single-family homes, multi-family homes, and apartments) among the projected residential buildings. These tasks were completed using a Random Forest classifier. They came to the conclusion that there was a need to improve the accuracy of property type prediction [9].

2.3 Combining Images and Tabular Data

The world presents us with data in a variety of formats. Models that combine data from many modalities, on the surface, appear to outperform their unimodal counterparts because more information can be used. Using tabular data alongside image data in a multimodal approach to solving a task has lately received attention and has the potential to generate more accurate predictions [8].

2.3.1 Combining Images and Tabular Data in Real Estate A recent study in real estate valuation focuses on including image data alongside structured data in the modelling process. Using CNNs, a study assesses the prediction performance of satellite images and structured data. The trained CNN model outperforms the advanced baseline of a neural network trained on structured data by 7% in MAE [28].

Zhao *et al.* investigated the use of deep learning on images of properties and extreme Gradient Boosting (XGBoost) on sales records to estimate property prices. Their research found that replacing the last output layer with XGBoost improves the accuracy of property price predictions [43].

Another research looked into it from the standpoint of disaster aid. Oki *et al.* established a system for estimating a building's structure as well as the year it was built. Again, a CNN was used to analyse exterior images, and a model with sparse modelling (SpM) was utilised to analyse attributes from a property database. According to the study, multimodal learning outperformed the CNN on its own [31].

A study that is more relevant to our research topic developed a method for determining the level of luxury in real estate images. To begin, DenseNet [21] was used to train a classifier to identify images based on room type. Because comparing rooms of the same type is expected to produce better results than comparing rooms of different types. Crowdsourcing was then utilized to estimate luxury levels. Another DenseNet was eventually trained to classify real estate images based on the eight luxury level classifications provided through crowdsourcing. The metadata vector was then combined with the vector representing the average luxury levels of the rooms in the property. The metadata vector comprises all the information on home features such as offered price, size, and so forth. They demonstrated that it gives a more accurate value calculation than a method developed by Zillow, an American internet real-estate marketplace company [33].

3 Data Description and Preparation

This chapter is divided into three sections that include information on the available data. The reader is introduced to the real estate data in the first section 3.1, Data Description. Following that, part 3.2 discusses data exploration, followed by section 3.3, which discusses the applicable pre-processing steps, including making the data appropriate as input for the models.

3.1 Data Description

In the Netherlands, Pararius is a firm that provides an independent housing platform that focuses on private rental properties. Properties may be manually registered on the site by a variety of parties, including real estate agents and individuals. When a registered rental property is rented out to a client, the transaction data is saved. The dataset for this study is made up of all available Pararius transactions that occurred between 2017 and 2021. Each property entry also includes a link to the corresponding images.

3.2 Data Exploration

The data include 20,296 transactions consisting of 55 columns. The party that registers the rental property on the housing platform offers property data, which includes continuous, ordinal, categorical, and binary features. In the Appendix, table 25 shows all the features present in the dataset, along with the feature type and a brief description.

3.2.1 Images A column from the tabular data, `listing_photo_urls`, contains a link to the images of the corresponding properties. The images were taken and published on the Pararius platform by agencies or individuals. Due to the URL's occasional unavailability, not all images could be downloaded.

The images show a wide range of scenes. Images of bathrooms, dining rooms, living rooms, and other rooms, as well as images of the outside and surroundings, are included. Images of the surroundings may include a neighbouring park or café. Floor plans have also been added to a number of properties.

3.2.2 Feature Selection The features that are considered in this study are listed in Table 1. The feature type and unique values have been added to the table as additional information.

Table 1. The used features, including feature types, the number of unique values (cardinality) and the possible values for ordinal and categorical features.

Feature	Type	Unique values	Possible values
construction year	continuous	–	
deposit amount	continuous	–	
lat	continuous	–	
listing price	continuous	–	
listing price sqm	continuous	–	
listing size (m2)	continuous	–	
listing volume (m3)	continuous	–	
lon	continuous	–	
number of photos	continuous	–	
service costs	continuous	–	
balcony	binary	2	
furnished	binary	2	
garage	binary	2	
garden	binary	2	
monumental building	binary	2	
parking availability	binary	2	
protected townscape	binary	2	
shell	binary	2	
storage	binary	2	
upholstered	binary	2	
energy label	ordinal	12	A+++++ – G
floor level	ordinal	36	1 – 88
number of floors building	ordinal	6	1 – 6
total bathrooms	ordinal	5	1 – 5
total bedrooms	ordinal	10	1 – 14
total rooms	ordinal	12	1 – 16
acceptance category	categorical	3	per date, immediately, in consultation
building type	binary	2	existing, newly
interior type	categorical	4	upholstered, furnished, upholstered or furnished, shell
listing residential type	categorical	2	apartment, house
maintenance status	categorical	3	bad, excellent, good
parking type	categorical	6	garage, paid, parking lot, permit, public, unknown

In total, 32 columns were chosen and 23 were removed. Since the columns containing latitude (`lat`) and longitude (`lon`) were chosen as features and already give information about the location, all nine columns that include address information were removed. Furthermore, columns having more than 80% of their values missing have been removed. `Energy index`, `Min rent period` and `Max rent period` were those that satisfied this requirement. Three columns that include information about ID numbers are also deleted, since they are not relevant for the analysis. Additionally, there are columns that offer information on the dates when a property was accepted and published, but they are incorrect. The data scientists at Clappform provided this information. A total of 6 columns of this type have been removed. Moreover, the column `source` was removed because it contained only one possible value (Pararius). Finally, the column containing the download link of the images was removed, due to the fact that it is unrelated to the maintenance status of the property.

3.2.3 Missing Values The dataset is sparse due to a high percentage of missing values in various categorical and binary features. Since the dataset is contributed by people, the process of filling in feature values when a property is registered is not consistent. As a result, the meaning of a missing value may be unclear. A missing value might indicate that there is insufficient information about the feature or that the feature is not existent for a certain property.

Table 2. Percentage of missing values per feature

Feature	% missing values	Feature	% missing values
balcony	38.94%	garage	27.26%
construction year	10.25%	listing volume m3	11.1%
deposit amount	9.98%	number of floors building	4.71%
energy label	35.52%	service costs	78.03%
floor level	25.79%	storage	23.74%
garden	21.42%	total bathrooms	18.5%

3.2.4 Outliers Outliers are caused by two events, according to [19]: measurement errors and variability in the observed feature. Outliers are described as "observations that differ so significantly from other observations that it raises concerns that it was created by a separate mechanism" [16]. Outliers can also have an effect on classification model prediction abilities [25]. Data outliers are typically deleted for both of these reasons.

One method to eliminate the outliers is to seek the advice of professionals with domain knowledge. Machine learning is another option [19]. Because both Pararius and Clappform have data analytics professionals with domain knowledge of the dataset, outliers are eliminated once threshold values are manually adjusted based on their experience and advice. The size of the house and the surroundings are mostly the focus. The eliminated number is reported in section 3.3. Due to confidentiality, the threshold values will not be disclosed.

3.3 Preprocessing

The datasets for all three parts of the study must be cleaned up and made consistent in order to appropriately compare the first and second parts and then go on to the third part of the study. This section will discuss the preprocessing steps.

3.3.1 Images Since the technique we intended to use requires images of the same size, the images, which were originally of different sizes, have been converted to the same size. The images were centrally cropped and then scaled to the required size, to prevent stretching or compressing them. The chosen CNN, which will be discussed later, requires a size of 380 for width and height.

Additionally, the pixel values of images are normalized with relation to the image dataset's mean and standard deviation. This can be helpful for transfer learning and for getting consistent results.

After carefully reviewing several images, the decision was made to merge the classes 'good' and 'excellent'. The images of these classes were comparable, and this would otherwise cause confusion for the model to distinguish between these two classes. As a result, this task became a binary classification.

3.3.2 Tabular data The first step was to check for duplicates, but none were found. Then, as was said in the preceding section, there are some property entries with values that are not valid by the standards of the company. Based on these business rules, 1076 properties are deleted. The property entries in the data set without images are then filtered out. This implies that the download links were no longer active, making it impossible to download the images of 4508 properties.

As a result of the prior preprocessing steps, 1016 duplicate properties were discovered, despite the fact that the whole dataset was free of them initially. The duplicate properties might be the result of properties being listed on the platform a second time and eventually being rented out because the dataset was spread out over such a lengthy period of time. We checked the address to see whether it was the same property, and it was. Due to the removal of the column with the publication date, these duplicates have appeared. Thus, these duplicate properties were removed. The final numbers of the properties and images for each class are shown in Table 3. The large dataset's imbalance is discussed in more detail in section 4.1.

Table 3. The amount of properties and images per class.

	Bad	Good	Excellent	Total
Properties	483	7,690	5,323	13,496
Images	2,868	87,867	69,209	159,944

4 Methodology

In order to answer the research questions presented in the introduction, this research is divided into three parts , as shown in Figure 1. The first part focuses on the images and the CNN. The second part concentrates on the corresponding tabular data. The final part elaborates on the combination of images and tabular data.

All three parts make use of deep learning: a CNN, a version of EfficientNet [39], in the first and TabNet [6] in the second and third . In the third part, a new feature is added to the initial table. The new feature contains part 1's predictions.

When optimizing these classification models, cross-entropy is used as the loss function. To reduce the cross entropy loss, gradient-based optimization is being done. The number of epochs used to train the models is 20. The number of epochs used in studies for an image and tabular classifier led to the selection of this value. It was decided on a number between the two. Instead of the most recent epoch, the best weights from the best epoch (during validation) are automatically kept.

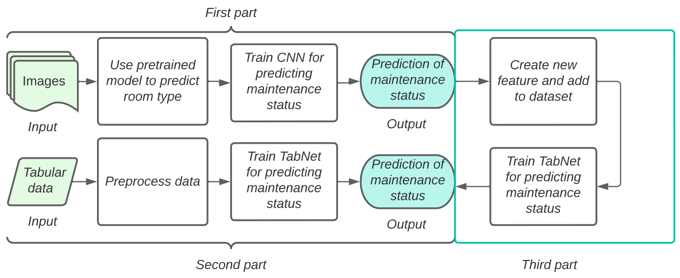
First part
Images
Input
Use pretrained model to predict room type
Train CNN for predicting maintenance status
Prediction of maintenance status
Output
Create new feature and add to dataset
Tabular data
Input
Preprocess data
Train TabNet for predicting maintenance status
Prediction of maintenance status
Output
Train TabNet for predicting maintenance status
Second part
Third part

Fig. 1. Pipeline of this research.

4.1 Train, Validation and Test Size

A json file containing all properties is created, and for each property, the features with associated values and paths to the images are listed. The sets are drawn at random from the total data set, with a proportion of 80%, 10%, and 10% set for the training, validation, and test sets, respectively. The model is trained using the training set. The validation set is used to validate the model's performance during training, and a test set is needed to measure the models' performance.

A train set of 10796 properties, a validation set of 1350 properties, and a test set of 1350 properties result from the splits. This can be seen in Table 4. This table also clearly shows that there is a large imbalance in the classes.

Table 4. An overview of the unbalanced sets, including the number of properties and the number of images. The amount of properties and images per class are also indicated.

	Train	Validation	Test
Properties	10,796	1,350	1,350
Bad	407	31	45
Good	10,389	1,319	1,305
Images	127,993	16,349	15,602
Bad	2,451	185	232
Good	125,542	16,164	15,370

The train and validation sets are balanced during training to guarantee that both classes are treated equally. It is balanced in terms of the number of real estate properties. Since there are different numbers of images for each real estate property, there are not an equal amount of images for each class. The test set remains unbalanced. Table 5 exhibits the new sets.

Table 5. An overview of the balanced sets, including the number of properties and the number of images. The amount of properties and images per class are also indicated.

	Train	Validation	Test
Properties	814	62	1,350
Bad	407	31	45
Good	407	31	1,305
Images	7,297	502	15,602
Bad	2,451	185	232
Good	4,846	317	15,370

4.2 Part 1: Convolutional Neural Networks (CNNs)

This section is about the computer vision part. This part was completed using Convolutional Neural Networks. First, a classifier, a CNN, is used to predict the room type. Taewoon Kim, one of my supervisors, created the room type classifier, which is accessible on GitHub [1]. Section 4.2.1 offers further information. This

[1] Taewoon Kim. (2022). tae898/room-classification: v0.2 (v0.2). Zenodo. https://doi.org/10.5281/zenodo.6338716

classifier is used to exclude some images depending on the type of room, and then a maintenance status classifier, also a CNN, is made using the remaining images.

Both classifiers use a EfficientNet model. EfficientNets, a family of models, was introduced by Google AI and compared to existing CNNs using an image dataset called ImageNet [13]. EfficientNet models generally outperform the existing CNNs in terms of accuracy and efficiency, while requiring fewer parameters. They are also evaluated on eight commonly used transfer learning datasets to see if it performed well on additional datasets. In 5 of the 8 datasets, EfficientNets achieved state-of-the-art accuracy [39].

Tan and Le [39]investigated the effects of scaling the models' various dimensions. While increasing individual dimensions enhances model performance, they discovered that balancing all network dimensions—width, depth, and resolution—against available resources boosts overall performance the most. This uniform scaling of these dimensions is also called the compound scaling approach.

Table 6, from [39], shows the structure of EfficientNet-B0 as described by Tan and Le. The basic building component of EfficientNet-B0 is mobile inverted bottleneck MBConv [35,39], with squeeze-and-excitation optimization [20]. Beginning with the baseline EfficientNetB0 and employing the compound scaling approach, the network may be scaled up to many bigger and superior forms. The number of parameters increases as we move from EfficientNet-B0 to EfficientNet-B7. The EfficientNet models that were used in this research are mentioned in sections 4.2.1 and 4.2.2.

Table 6. EfficientNet-B0 baseline network – Each row describes a stage i with $\hat{L}_i$ layers, with input resolution $\langle \hat{H}_i, \hat{W}_i \rangle$ and output channels $\hat{C}_i$.

Stage	Operator	Resolution	#Channels	#Layers
i	$\hat{F}_i$	$\hat{H}_i$ x $\hat{W}_i$	$\hat{C}_i$	$\hat{L}_i$
1	Conv3x3	224 x 224	32	1
2	MBConv1, k3x3	112 x 112	16	1
3	MBConv6, k3x3	112 x 112	24	2
4	MBConv6, k5x5	56 x 56	40	2
5	MBConv6, k3x3	28 x 28	80	3
6	MBConv6, k5x5	14 x 14	112	3
7	MBConv6, k5x5	14 x 14	192	4
8	MBConv6, k3x3	7 x 7	320	1
9	Conv1x1 & Pooling & FC	7 x 7	1280	1

Note. From *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.*, by Tan, M. & Le, Q., 2019, *Proceedings of the 36th International Conference on Machine Learning*, PMLR 97:6105-6114 [39].

Also, both EfficientNet models use a pretrained model, trained on ImageNet [13]. ImageNet contains 1000 classes, so the pre-trained model has been trained to work on a lot of different things. The usage of a pretrained model was chosen, because Hendrycks *et al.* [18] claim pre-training can improve model robustness and uncertainty. In addition to these advantages, it may also provide faster model training.

4.2.1 Room Type Classifier Each property entry in the dataset includes a link to images of the property and its surroundings. In order to properly evaluate the property's maintenance status, it was decided to filter out particular room types. The following scenes are included if the model's confidence score was equal to or more than 80%: bathrooms, kitchens, and living rooms. These room types were selected because they have distinctive features, and it was believed that doing so would make it easier to predict the maintenance status.

The model was trained using images of bathrooms, bedrooms, living rooms, kitchens, dining rooms, exteriors, and interiors. It is a fine-tuned *EfficientNet-B3* using pytorch-lightning.

4.2.2 Maintenance Status Classifier After filtering the images, developing the maintenance status classifier was the next step. The images used to train the model include images of properties in a particular maintenance status, among the images of bathrooms, kitchens and living rooms. It is not feasible to claim with certainty that just these types of images were included in the set, because the room type classifier can also misclassify them.

This classifier also uses pytorch lightning with a pretrained EfficientNet model. A number of Efficientnets were tested on given data, but the more parameters the model contained, the more frequently the program crashed since the service we used *(Google Colab Pro)* did not have enough RAM. Because of an additional restricted runtime of up to 24 hours and a GPU limit of 12 GB, *EfficientNetB4* was eventually chosen. This problem prevented us from experimenting with the batch size, thus we decided on a batch size of 32.

4.2.3 Performance Metrics The goal is to predict the maintenance status of the property, but in the training phase, several images per property are used. The used CNN predicts on the basis of an image instead of a property. As a result, to predict the maintenance status of a property, two alternative metrics have been developed, called Property Maintenance Status (PMS) Average and Mode. After evaluating the results generated by the model with default hyperparameters using both metrics, the metric with the highest score on the validation set is chosen to continue working with. This model's architecture is saved after the epoch with the best performance.

1. *PMS Average*
   The PMS average metric operates on a per-class basis. Every real estate property has several images, and each image has associated probabilities for each class. The probabilities are averaged per class, after which the class with the highest probability is chosen.

2. *PMS Mode*
   The PMS Mode metric operates image by image. Here again, every real estate property has several images, and each image has associated probabilities for each class. It now approaches each image separately rather than tackling it per class. It determines which class has the highest probability for each image and selects that one. The most often mentioned class within the entire collection of chosen classes is then determined, and that class is selected.

In both cases, the macro F_1 is then calculated by comparing this to the target value. In section 4.5, this score is explained in more detail. In addition to the F_1, a confusion matrix and the training loss of the best performing metric is included in the results section.

4.3 Part 2: TabNet

TabNet, a deep learning model for tabular learning [6], is used to predict the maintenance status of a property using tabular data. This tabular classifier was chosen because it outperforms other neural network and decision tree variations on numerous tabular datasets and produces interpretable feature attributions as well as insights into overall model behavior [6].

At each decision step, the model employs sequential attention to choose a subset of significant features to process, resulting in improved learning since learning capacity is allocated to the most useful features. The feature selection is instance-based, which means that it can differ for each row of the training dataset.

Global interpretability is one of TabNet's advantages. The importance of each feature to the trained model across the whole dataset is quantified by global interpretability. The importance of the individual features adds up to 1. In this manner, it is simple to determine which feature had the most impact. This is displayed for each model in the results section. Since TabNet does not allow missing values, it was decided to impute the missing values in this study. This was done because there was not a property entry without missing values. The imputation method used is Multiple Imputation by Chained Equations (MICE) [36]. It uses random forests in an iterative way to try to estimate the best prediction for each missing value in a dataset by looking at data from other columns. Furthermore, all the features noted in Table 1 are included, and the categorical features are one hot encoded.

The macro F_1 and confusion matrix are once again used to measure performance. The results also include the training loss of the model. The batch size selected for the training is 1024, which is the default value.

4.4 Part 3: Combination

By including a new feature to the part 2's dataset, the combination of images and tabular data is created. The new feature contains the probability that the property belongs to class 'good'. It was decided not to add a 0, a bad maintenance status, or a 1, a good maintenance status, since this would provide less certainty.

The dataset with the new feature is then fed into TabNet as input.

The macro F_1 and confusion matrix are once again used to measure performance. The results also include the training loss of the model. Also, the batch size selected for this is 1024, which is the default value.

4.5 Macro F1

By calculating the harmonic mean of a classifier's precision and recall, the F_1 integrates both into a single statistic. Comparing the effectiveness of two classifiers is its main purpose. Assume classifiers A and B have greater recall and precision, respectively. The F_1 for both classifiers in this situation may be used to assess which one yields superior results. The F_1 of a classification model is calculated as follows:

$$F_1 = \frac{2(P * R)}{P + R}$$

where P is the precision and R the recall. A generalization of the F_1 called the Fbeta-score includes a beta configuration parameter. The default beta value is 1.0, which is the same as the F_1. In the calculation of the score, a smaller beta value, such as 0.5, gives more weight to precision and less weight to recall, whereas a larger beta value, such as 2.0, gives less weight to precision and more weight to recall [34]. In this study, beta is set to 1, because recall and precision are of equal importance. False negatives are just as essential here as false positives.

By first calculating the F_1 for each class and then average them, macro F_1-averaging is carried out. This is done because there is a large imbalance in the dataset's classes. This metric ignores class imbalances and treats all classes as equal. Each class receives the same weighting under macro F_1. It is low for models that only excel in popular classes while failing miserably at uncommon ones. The best result is a macro F_1 of 1, while the poorest value is 0. The macro F_1 is defined as the mean of class-wise F_1:

$$\mathit{Macro}\ F_1 = \frac{1}{N}\sum_{i=0}^{N} F_{1_i}$$

where i is the class index and N the number of classes.

4.6 Hyperparameter Tuning

The tuning or optimization of hyperparameters is critical to the prediction accuracy of machine learning algorithms. Hyperparameter tuning is referred to choosing a set of optimal hyperparameters for a learning algorithm. The relationship between machine learning algorithm performance and hyperparameters is unclear, but in practice, it is necessary to continuously adjust hyperparameters and train a number of models with different combinations of values, then compare model performance to select the best model. As a result, tuning hyperparameters becomes a critical issue in a machine learning problems [41].

An automatic search approach, the Bayesian optimization algorithm, was chosen for this research because it outperforms other global optimization algorithms, according to experiments published in another work [24]. By iteratively creating a probabilistic model of the function mapping from hyperparameter values to the objective function, Bayesian optimization methods search for global optimization. To create a posterior distribution over the objective function, the probabilistic model captures ideas about the function's behavior. The posterior distribution is then utilized to create an acquisition function that selects the next point with the highest chance of improvement [41].

Optuna [3], which uses a Bayesian optimization algorithm by default, is a hyperparameter optimization software framework and is implemented to perform hyperparameter optimization in this study. Each model has the learning rate tuned within a range of 1e-5 and 1e-1. This is completed after 50 trials. This number was arbitrarily chosen. The learning rate determines the step size at each iteration while moving toward a minimum of a loss function. The optimizer that is used in both models is AdamW with a default learning rate of 0.001.

Because of limitations in computational capacity, the model trained on images (part 1) uses a subsample of the data. First, the dataset is balanced, and then 20% of the properties are included. The other model uses the entire dataset.

5 Results

In this section, the results of the deep learning models are described. This is done separately for image classification, tabular classification and combination models in subsections 5.1, 5.2, and 5.3 respectively. Each model has also been fine-tuned, and therefore there is a subsection for the default model and a subsection for the tuned model, each providing the findings.

5.1 Results Image Classification

5.1.1 Default The validation sets were evaluated using the two metrics. The macro F_1s of the two metrics were not very different, as shown in Figure 2. But since the PMS average produced some higher macro F_1s than the PMS mode, the PMS average was chosen.

The train and validation macro F_1s for 20 epochs are displayed in Figures 3 and 4. Both graphs demonstrate how well the training is progressing and that the model is learning. However, the model appears to perform poorly when evaluated on the validation set.

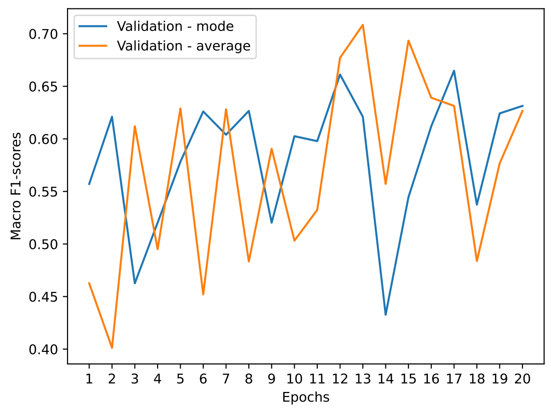

Fig. 2. The validation set was evaluated every epoch using the PMS mode and PMS average.

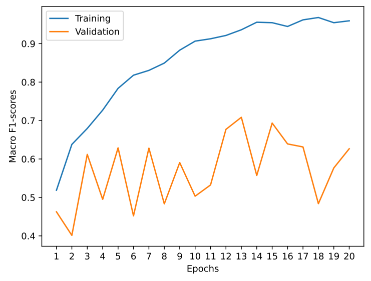

Fig. 3. Train and validation macro F_1s evaluated by PMS average (default learning rate)

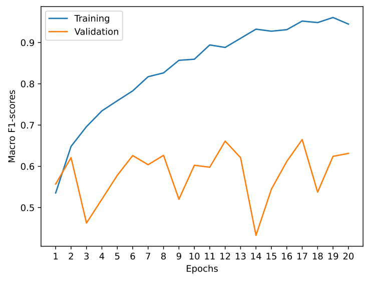

Fig. 4. Train and validation macro F_1s evaluated by PMS mode (default learning rate)

The train loss in Figure 5 is decreasing, but it is also going up and down nearing the end. Also, it appears like the line is still decreasing. Note that steps, not epochs, are shown in this picture. The number of batches has an impact on the steps.

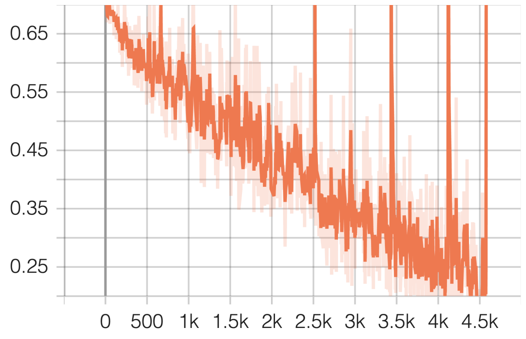

Fig. 5. Train loss (default learning rate)

Table 7 demonstrates that the model predicts that the attributes of class 'bad' belong almost as much to class 'bad' as they do to class 'good'. Table 8 shows that many more properties are predicted to be in the 'bad' class than are actually present.

Table 7. Confusion matrix for the validation set (default learning rate)

		Predicted	
		Bad	Good
Actual	Bad	17	14
	Good	6	25

Table 8. Confusion matrix for the test set (tuned learning rate)

		Predicted	
		Bad	Good
Actual	Bad	26	19
	Good	234	1071

5.1.2 Tuned In order to perform better than the model with the default learning rate, the learning rate is optimized. Optuna found that trial 27, which resulted in the model's best macro F_1 of 0.693 and learning rate of 0.0005, was the best trial after 50 trials, as shown in table 9.

The optimized learning rate is less than the 0.001 default learning rate. A lower learning rate could enable the model to learn a better set of weights. The aim is for the model to perform well on both the training and validation sets.

Table 9. Tuning EfficientNet with Optuna: the best trial information.

Best trial information	
Number of finished trials	50
Best trial	27
Best macro F_1	0.693
Best learning rate	0.0005

The training progressed nicely, just like with the default model, as evidenced by the increasing line in Figure 6. However, the model is assessed once again using a validation set, and again it appears to perform poorly even though the learning rate is optimized. The epoch with the best macro F_1 on the validation set after 20 epochs, with a score of 0.6418, is epoch 5.

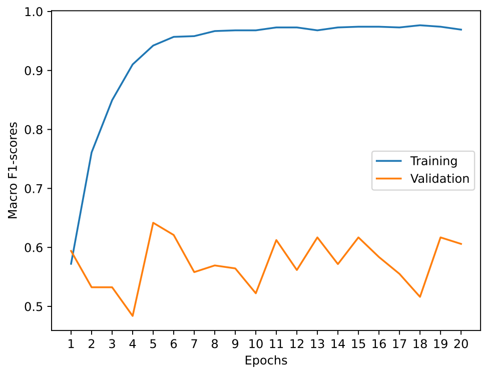

Fig. 6. Train and validation Macro F_1s evaluated by the average metric (tuned learning rate)

The train loss of the model with the tuned learning rate (see Figure 7) is comparable with the loss of the model with the default learning rate. The line is, however, moving up and down less sharply; this may be related to a lower learning rate. A lower learning rate could enable the model to learn a better set of weights.

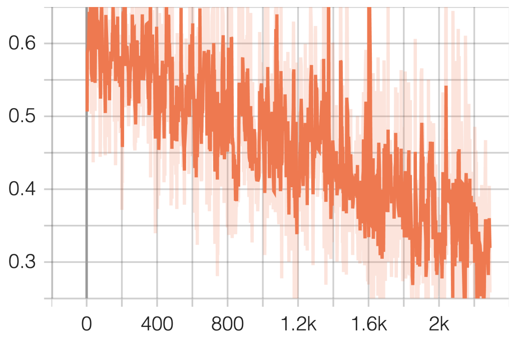

Fig. 7. Train loss (tuned learning rate)

The confusion matrices presented in Tables 10 and 11 are similar to the confusion matrices of the model with the default learning rate. Table 10 demonstrates that the model predicts that the properties of class 'bad' belong almost as much to class 'bad' as they do to class 'good' and Table 11 shows that many more properties are predicted to be in the 'bad' class than are actually present.

Table 10. Confusion matrix for the validation set (tuned learning rate)

		Predicted	
		Bad	Good
Actual	Bad	16	15
	Good	6	25

Table 11. Confusion matrix for the test set (tuned learning rate)

		Predicted	
		Bad	Good
Actual	Bad	31	14
	Good	251	1054

Table 12 demonstrates that no model performed better than the other. There is not much of a difference between the macro F_1s of the models.

Table 12. The mean and standard deviation (SD) of five runs of the models with the default and tuned learning rate on the validation and test set.

	Default	Tuned
Validation macro F_1	$.6763 \pm .0252$	$.6392 \pm .0167$
Per class macro F_1: bad	$.6288 \pm .0260$	$.5741 \pm .0239$
Per class macro F_1: good	$.7239 \pm .0239$	$.7042 \pm .0239$
Test macro F_1	$.5398 \pm .0239$	$.5410 \pm .0239$
Per class macro F_1: bad	$.1755 \pm .0239$	$.1770 \pm .0239$
Per class macro F_1: good	$.9040 \pm .0239$	$.9051 \pm .0239$

5.2 Results Tabular Classification

5.2.1 Default As illustrated in Figure 8, the model appears to have chosen a random choice during training and validation, since the macro F_1 is around 0.5. The epoch with the best macro F_1 on the validation set after 20 epochs, with a score of 0.62816, is epoch 11.

The deposit amount, latitude, and construction year are the features that are shown to be the most relevant in Figure 9's evaluation of feature importance. As described in section 4.3, most relevant means that the model uses sequential attention to select which model features to reason from at each step in the model.

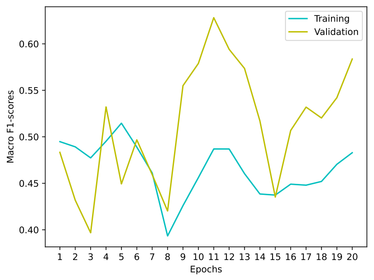

Fig. 8. Train and validation Macro F_1s (default learning rate)

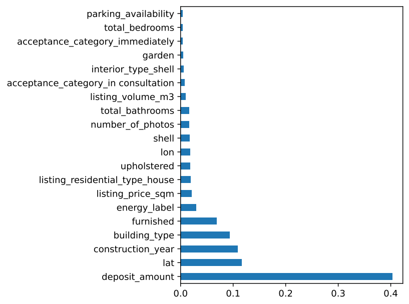

Fig. 9. Relative feature importance (default learning rate)

The loss is calculated on training, and as seen in Figure 10, the line is decreasing and becoming more static as more epochs pass.

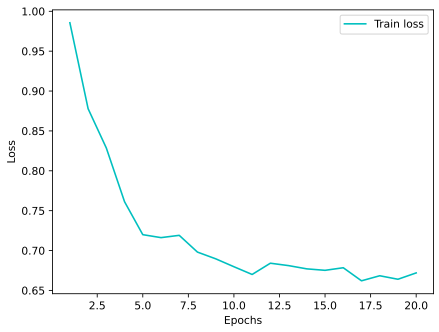

Fig. 10. Train loss (default learning rate)

Tables 13 and 14 demonstrate that while the properties in class ’good’ are more frequently predicted correctly, the properties in class ’bad’ are misclassified more often.

		Predicted Bad	Predicted Good
Actual	Bad	12	19
	Good	6	25

Table 13. Confusion matrix for the validation set (default learning rate)

		Predicted Bad	Predicted Good
Actual	Bad	7	38
	Good	349	956

Table 14. Confusion matrix for the test set (default learning rate)

5.2.2 Tuned After 50 trials, Optuna determined that trial 45 was the best trial when the model obtained the best macro F_1 of 0.694 and a learning rate of 0.0037 as shown in table 15.

Table 15. Tuning TabNet with Optuna: the best trial information.

Best trial information	
Number of finished trials	50
Best trial	45
Best macro F_1	0.694
Best learning rate	0.0037

The tuned learning rate model appears to make random predictions, just like the default model. However, the increasing lines from epoch 11 demonstrate that the model is getting better at learning. The epoch with the best macro F_1 on the validation set after 20 epochs, with a score of 0.52622, is epoch 19.

The construction year, listing residential type apartment, and energy label are the features that are shown to be the most relevant in Figure 12's evaluation of feature importance.

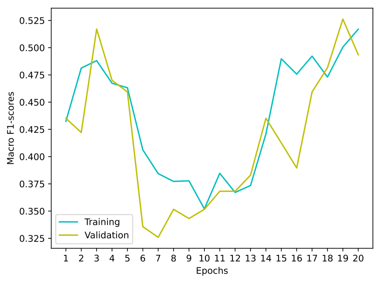

Fig. 11. Train and validation Macro F_1s (tuned learning rate)

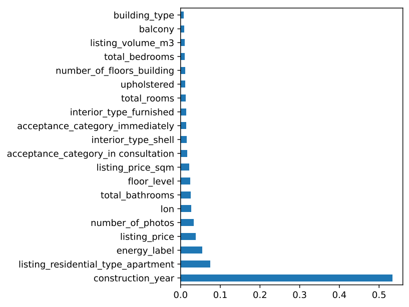

Fig. 12. Relative feature importance (tuned learning rate)

The train loss in Figure 13 is going more up and down than the training loss in Figure 10. The increased learning rate could be the cause of this. Also, it appears like the line is still decreasing. This indicates that the model is still learning.

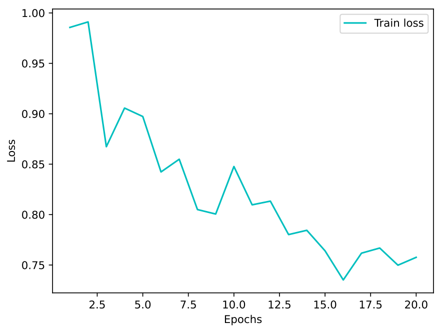

Fig. 13. Train loss (tuned learning rate)

Table 16 demonstrates that while the model performs better when predicting properties of class 'bad', it more often misclassifies properties of class 'good'. Table 17 shows that the model misclassifies more than half of the properties of class 'good' to the class 'bad'. Compared to the model with the default learning rate, shown in Table 14, this model predicted the properties of the label 'good' more frequently wrong. Also, the model predicts the properties, labeled as 'bad', as often as 'good', although Table 14 shows that 'bad' properties are more often forecasted as 'good'.

		Predicted	
		Bad	Good
Actual	Bad	21	10
	Good	14	17

Table 16. Confusion matrix for the validation set (tuned learning rate)

		Predicted	
		Bad	Good
Actual	Bad	24	21
	Good	826	479

Table 17. Confusion matrix for the test set (tuned learning rate)

As shown in table 18, the model with the default learning rate outperformed the model with the tuned learning rate in terms of the validation and test set's mean macro F_1.

Table 18. The mean and standard deviation (SD) of five runs of the models with the default and tuned learning rate on the validation and test set.

	Default	Tuned
Validation macro F_1	$.5960 \pm .0472$	$.5300 \pm .0587$
Per class macro F_1: bad	$.5813 \pm .1007$	$.5411 \pm .1591$
Per class macro F_1: good	$.6120 \pm .0690$	$.5189 \pm .1224$
Test macro F_1	$.3940 \pm .0976$	$.3140 \pm .1341$
Per class macro F_1: bad	$.0733 \pm .0187$	$.0589 \pm .0122$
Per class macro F_1: good	$.7166 \pm .1783$	$.5701 \pm .2734$

5.3 Results Combination

5.3.1 Default As illustrated in Figure 14, the model appears to have chosen a random choice during training and validation, since the macro F_1 is around 0.5. The epoch with the best macro F_1 on the validation set after 20 epochs, with a score of 0.55889, is epoch 2.

The listing price, construction year, and listing volume (m^3) are the features that are shown to be the most relevant in Figure 15's evaluation of feature importance. Notably absent from the list of the top 20 features is the new feature that contains the probability score that it would be class 1.

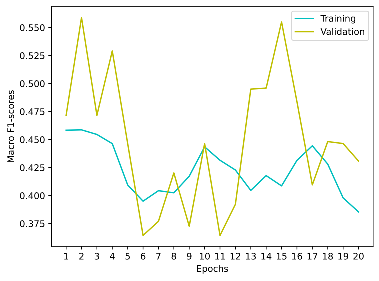

Fig. 14. Train and validation macro F_1s (default learning rate)

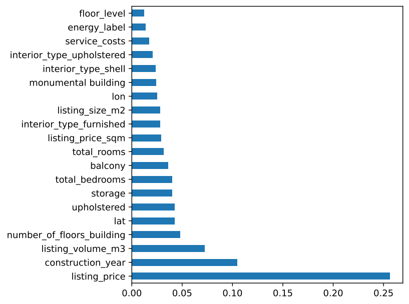

Fig. 15. Relative feature importance (default learning rate)

The train loss, as seen in Figure 16, is decreasing over time. Although it is doing so more slowly, it appears that the loss may even fall more after 20 epochs.

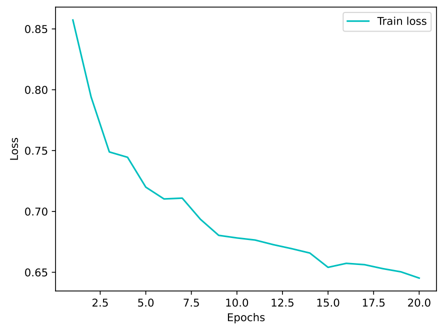

Fig. 16. Train loss (default learning rate)

Tables 19 and 20 show that the properties, labeled as 'bad' are more often predicted belonging to class 'good'. The properties, labeled as 'good', are predicted correctly more frequently than not.

		Predicted	
		Bad	Good
Actual	Bad	7	24
	Good	3	28

Table 19. Confusion matrix for the validation set (default learning rate)

		Predicted	
		Bad	Good
Actual	Bad	14	31
	Good	353	952

Table 20. Confusion matrix for the test set (default learning rate)

5.3.2 Tuned Optuna selected trial 12 as the best trial after 50 trials when the model achieved the best macro F_1 of 0.645 with a learning rate of 0.0058.

Table 21. Tuning TabNet with Optuna: the best trial information.

Best trial information	
Number of finished trials	50
Best trial	12
Best macro F_1	0.645
Best learning rate	0.0058

Since the macro F_1 is close to 0.5, as shown in Figure 17, it appears that the model made a random selection throughout training and validation. With a score of 0.55889, epoch 6 has the highest macro F_1 on the validation set after 20 epochs. Given that both the training and validation curves are increasing as of the 14th epoch, it seems that this model is starting to learn. The default learning rate model does not exhibit this behavior.

The listing price per square meter has received the majority of the model's attention (see Figure 18). This feature in particular has played an important role to the majority of the predictions due to the tuned learning rate. It is worth noting once more that the additional feature did nothing to assist the model.

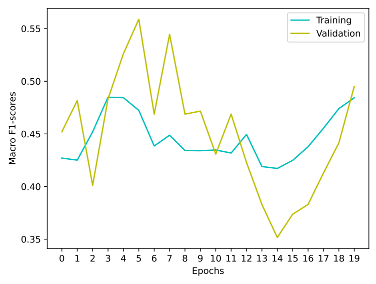

Fig. 17. Train and validation macro F_1s (tuned learning rate)

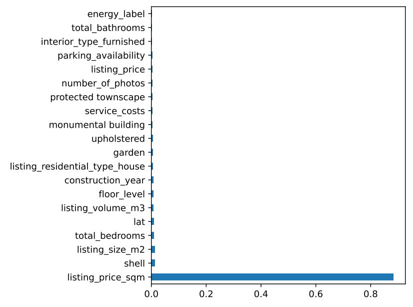

Fig. 18. Relative feature importance (tuned learning rate)

The loss does not seem to have entirely stagnated, as shown in Figure 19. It is possible that it decreases more after 20 epochs. The loss is going up and down more in this case than it did in the model with the default learning rate (Figure 16), which may be connected to the higher learning rate.

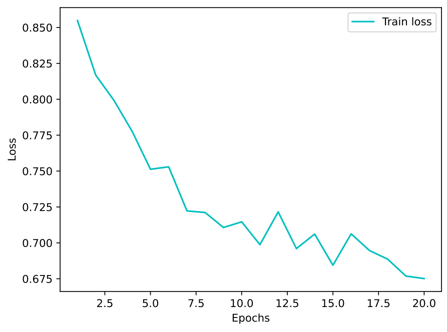

Fig. 19. Train loss (tuned learning rate)

Tables 22 and 23 show similar outcomes as Tables 19 and 20. The properties, labeled as 'bad' are more often predicted belonging to class 'good'. The properties, labeled as 'good', are predicted correctly more frequently than not.

		Predicted	
		Bad	Good
Actual	Bad	15	16
	Good	9	22

Table 22. Confusion matrix for the validation set (tuned learning rate)

		Predicted	
		Bad	Good
Actual	Bad	13	32
	Good	321	984

Table 23. Confusion matrix for the test set (tuned learning rate)

Table 24 demonstrates that no model performed better than the other. There is not much of a difference, and the best model depends on the situation.

Table 24. The mean and standard deviation (SD) of five runs of the models with the default and tuned learning rate on the validation and test set.

	Default	Tuned
Validation macro F_1	$.5480 \pm .0383$	$.5740 \pm .0546$
Per class macro F_1: bad	$.4783 \pm .0473$	$.5819 \pm .1098$
Per class macro F_1: good	$.6157 \pm .0892$	$.5675 \pm .1300$
Test macro F_1	$.4100 \pm .0574$	$.3900 \pm .0809$
Per class macro F_1: bad	$.0624 \pm .0709$	$.0709 \pm .0104$
Per class macro F_1: good	$.7613 \pm .1198$	$.7112 \pm .1674$

6 Discussion

This section is divided into the following sections. The discussion of the results of the three parts are described at the beginning. An explanation of the study's limitations and issues that were encountered is then given. Finally, several ideas for future research are given.

6.1 Discussion of the Results

When looking at section 5.1, Figures 3,4 and 6, it can be concluded that the model overfits to the training set by memorizing the noise and fitting too closely to it, which makes it less able to generalize to new data. The validation is not following the training curve, which is moving in the direction of 100. Also, as shown in Table 12, the model predicts properties labelled as 'good' fairly well, but properties labelled as 'bad' very poorly. The confusion matrices, Tables 7, 8, 10 and 11, show this as well. It is noticeable that the model frequently interprets images with a 'bad' label as images with a 'good' label. Since the train and validation sets are balanced to ensure that the classes are treated equally, you would not expect this behavior.

This might be caused by a number of factors, including the non-equal distributions of images by class. The properties are balanced, but the images are not. There are far more images with the label 'good' than there are with the label 'bad'. The large variety of images can also make it difficult for the model to correctly predict images that were not observed during training. Additionally, the lighting in each image may differ, or possibly a better camera is used at a different property. Another reason could be that real estate agents have mislabelled the images. After all, real estate agents themselves have labelled the images, so it is subjective: a property can be considered 'good' or 'bad' by different people. This could also be reflected in the dataset used. Moreover, the model with the tuned learning rate did not produce noticeably better results.

Figures 20, 21 and 22 display some examples of properties. Figure 20 shows three images of a property labelled as 'good'. The model correctly predicted in this case the label 'good' with a probability of 93.19 percent. Figure 21 also includes three images, and they are labelled as 'bad'. The model is 81.22 percent confident that it is a property with a bad maintenance status. It is also accurately predicted this time. However, Figure 22 displays a collection of images that have been labelled as 'bad', despite the model classifying them as 'good' with an 86.46 percent likelihood. This is a good illustration of the mentioned factors. The model could be confused because the labels are subjective, the lighting is varied in the images, and the images do not really differ that much from one another.

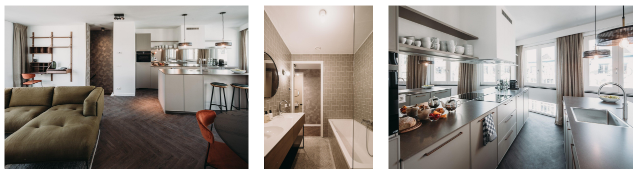

Fig. 20. A few images of a property from the class good (1)

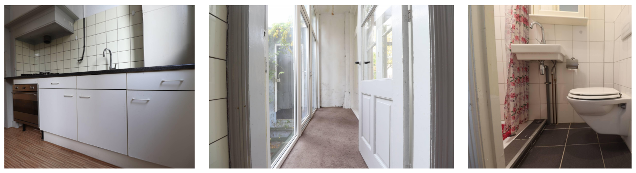

Fig. 21. A few images of a property from the class bad (0)

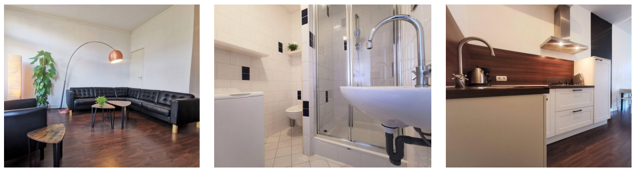

Fig. 22. A few images of a property from the class bad (0)

The predictions of the model of the second part in section 5.2 with the default learning rate and the tuned learning rate seems to be random, as may be inferred from Figures 8 and 11 and Table 18. There is a potential that there is no relationship between the features and classes. These figures also demonstrate how poorly the training is progressing. The macro F_1 vary between 0.5 and even less than that. But it is likely that if the model with the tuned learning rate ran for a longer duration (more epochs), it would perform better. Since it is on the rise from epoch 18 and the best macro F_1 occurs during epoch 19. It is possible that because the tuned learning rate is higher than the default learning rate (0.001), the maximum macro F_1 cannot be achieved. This is also seen in the train loss in Figure 13: the training loss appears to be decreasing.

The relative importances of the features are not markedly different (see Figures 9 and 12), but it is worth noting that the top two features of the model with the default learning rate do not even appear in the top 20 of the model with the tuned learning rate. It is possible that if all the features are used, some of them cause confusion and thus the model cannot predict well.

The model with the default learning rate appears to predict the label 'good' more frequently in the confusion matrices (Tables 13 and 14), but the model with the tuned learning rate predicts the label 'bad' more frequently (Tables 16 and 17). This could occur as a result of the model's random decision.

The model's performance in the third part, see section 5.3, was expected to be better than the models' individual performances in parts 1 and 2. The addition of model 1's output as a new feature to model 3's input raised the probability that model 3 would be influenced and consider this feature relevant. Figures 15 and 18 show that, contrary to predictions, neither of the models—with either a default or tuned learning rate—focused on it. It does not rank among the top 20 features for either model. This could be because the hyperparameters were set incorrectly. If these are changed, the model may adapt to the data and begin focusing more on the new feature.

The macro F_1s are roughly 0.5, as seen in Figures 14, 17 and Table 24. This indicates that a random prediction is made. Nevertheless, the line of training increases starting at epoch 14 in figure 17. It is possible that performance could improve if the model had to train for more epochs. Similar to part 2, it is probable that the maximum macro F_1 cannot be achieved in this case because the tuned learning rate is higher than the default learning rate. Also, similar to part 2, the training loss looks to be decreasing further (see Figure 19). But this does not only hold for the model with the tuned learning rate. The same is true for the model with the default learning rate (see Figure 19). However, there is a trend toward class 'good' predictions being made more frequently: the Tables 19, 20, 22 and 23 show that both models predict the properties as 'good' more frequently than 'bad'. More understanding could be gained by running more than 20 epochs.

6.2 Limitations

6.2.1 Dataset There are some limitations to the dataset. The data is created by people. As a result, the data's quality may be questionable. Data flaws discovered included, for example, values for the feature 'floor level' that exceeded the value for the feature 'number of floors building'. The same problem was discovered with the features 'total bedrooms' and 'total rooms'. Furthermore, the data contained a substantial portion of missing values in several features. This could also be due to incorrect human entry. An imputation procedure was employed, however it is possible that it was frequently inaccurate. The data collection process could be improved to obtain a higher quality dataset. For example, the data providing parties could be given explicit descriptions of each feature, or there could be predetermined answers from which the real estate agent could choose.

6.2.2 Computational Power Since the laptop used is a 2020 model Macbook Pro, GPU is not supported. Because of this, local CNN training with images was not an option. To resolve this issue, I was able to use Google Colab Pro, however I was restricted to a runtime of up to 24 hours and a GPU limit of 12GB. This prevented me from conducting a number of experiments, such as adjusting more hyperparameters than simply the learning rate or testing out different EfficientNet versions with a variety of parameters.

7 Future Work

The knowledge introduced in this paper can be expanded upon in a number of different ways. The first part, which focuses on the images, can be approached in a variety of ways. Performance metrics may be more thoroughly tested. Other approaches could be developed, such as saving probabilities from the image of the worst maintenance state rather than the mode and average.

Furhermore, the images could be analyzed using the equipment in the room. The equipment in the images could be labeled and determined whether it is in a good or bad maintenance status. By doing this, rather than relying entirely on the features that a model extracted from an image, you have more control over how a room is labeled as 'good' or 'bad'. This approach, however, requires more work than the approach used in this study.

Another approach would be to develop a classifier for each type of room, such as one for kitchens and bathrooms separately. This way, the model concentrates on a specific kind of room and its features. As mentioned in the Computational Power section, there could also be experiments with different versions of EfficientNet.

In the second part, which focuses on the tabular data, feature engineering can be performed to see if it yields a better result.

In the final part, which is a combination of the first two parts, future work can focus on a multi input model. In this case, both modalities are fed simultaneously into the model.

Finally, future research could concentrate on fine-tuning more hyperparameters for all models. All parts can benefit from hyperparameter tuning.

8 Conclusion

In this research, deep learning models for classification tasks were the main focus. Images and tabular data were both used as inputs to the models. The outcomes of our work using EfficientNet and TabNet were shown. The results of these models alone and in combination were compared, and tuned models were also added to see if they improved performance. Although the model trained using images appears to learn well, it performs poorly when tested on the validation and test sets. However, it performs better than the model that was trained on tabular data. Contrary to expectations, the combined model did not perform any better than the individual models, and the models with the tuned learning rate are not a valuable addition in terms of enhanced performance. The results indicate that this needs to be investigated further.

9 Acknowledgments

I would like to thank Taewoon Kim for his supervision, help, and creation of the room type classifier throughout the writing of this thesis. I would also like to thank Michael Cochez for his additional supervision. This research, however, would not have been possible without Clappform. Therefore, I would like to thank Clappform for providing me with the opportunity and resources to complete this research. Thereby, I would like to thank Clappform's data analytics, product development, and business development teams, as well as my external supervisor, Jeroen Schoonderbeek, for his constant supervision throughout the internship.

References

1. Adetunji, A.B., Akande, O.N., Ajala, F.A., Oyewo, O., Akande, Y.F., Oluwadara, G.: House Price Prediction Using Random Forest Machine Learning Technique. Procedia Computer Science **199**, 806–813 (2022). https://doi.org/https://doi.org/10.1016/j.procs.2022.01.100, https://www.sciencedirect.com/science/article/pii/S1877050922001016, the 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 & 2021): Developing Global Digital Economy after COVID-19
2. Afonso, B., Melo, L., Dihanster, W., Sousa, S., Berton, L.: Housing Prices Prediction with a Deep Learning and Random Forest Ensemble. In: XVI Encontro Nacional de Inteligência Artificial e Computacional. pp. 389–400 (01 2020). https://doi.org/10.5753/eniac.2019.9300
3. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A Next-Generation Hyperparameter Optimization Framework. In: Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019)
4. Albawi, S., Mohammed, T.A., Al-Zawi, S.: Understanding of a Convolutional Neural Network. In: 2017 International Conference on Engineering and Technology (ICET). pp. 1–6 (2017). https://doi.org/10.1109/ICEngTechnol.2017.8308186
5. Angrick, S., Bals, B., Hastrich, N., Kleissl, M., Schmidt, J., Doskoc, V., Katzmann, M., Molitor, L., Friedrich, T.: Towards Explainable Real Estate Valuation via Evolutionary Algorithms. CoRR **abs/2110.05116** (2021), https://arxiv.org/abs/2110.05116
6. Arik, S.O., Pfister, T.: Tabnet: Attentive Interpretable Tabular Learning. Proceedings of the AAAI Conference on Artificial Intelligence **35**(8), 6679–6687 (May 2021), https://ojs.aaai.org/index.php/AAAI/article/view/16826
7. Aziz, M.A., Nurrahim, F., Susanto, P.E., Windiatmoko, Y.: Boarding House Renting Price Prediction Using Deep Neural Network Regression on Mobile Apps. CoRR **abs/2101.02033** (2021), https://arxiv.org/abs/2101.02033
8. Baltrusaitis, T., Ahuja, C., Morency, L.P.: Multimodal Machine Learning: A Survey and Taxonomy. IEEE Trans. Pattern Anal. Mach. Intell. **41**(2), 423–443 (feb 2019). https://doi.org/10.1109/TPAMI.2018.2798607, https://doi.org/10.1109/TPAMI.2018.2798607
9. Bandam, A., Busari, E., Syranidou, C., Linssen, J., Stolten, D.: Classification of Building Types in Germany: A Data-Driven Modeling Approach. Data **7**(4) (2022). https://doi.org/10.3390/data7040045, https://www.mdpi.com/2306-5729/7/4/45
10. Breiman, L.: Random Forests. Machine Learning **45**(1), 5–32 (2001). https://doi.org/10.1023/A:1010933404324, http://dx.doi.org/10.1023/A%3A1010933404324
11. Chen, M.Y., Chiang, H.S., Lughofer, E., Egrioglu, E.: Deep Learning: Emerging Trends, Applications and Research Challenges. Soft Computing **24**, 1–4 (04 2020). https://doi.org/10.1007/s00500-020-04939-z
12. Cireşan, D.C., Meier, U., Masci, J., Gambardella, L.M., Schmidhuber, J.: Flexible, High Performance Convolutional Neural Networks for Image Classification. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two. p. 1237–1242. IJCAI'11, AAAI Press (2011)
13. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A Large-Scale Hierarchical Image Database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009). https://doi.org/10.1109/CVPR.2009.5206848

14. Gorishniy, Y., Rubachev, I., Khrulkov, V., Babenko, A.: Revisiting Deep Learning Models for Tabular Data. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 18932–18943. Curran Associates, Inc. (2021), https://proceedings.neurips.cc/paper/2021/file/9d86d83f925f2149e9edb0ac3b49229c-Paper.pdf
15. Goyal, S., Chattopadhyay, C., Bhatnagar, G.: Knowledge Driven Description Synthesis for Floor Plan Interpretation. CoRR **abs/2103.08298** (2021), https://arxiv.org/abs/2103.08298
16. Hawkins, D.: Identification of Outliers. Monographs on applied probability and statistics, Chapman and Hall, London [u.a.] (1980), http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+02435757X&sourceid=fbw_bibsonomy
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 770–778 (June 2016). https://doi.org/10.1109/CVPR.2016.90
18. Hendrycks, D., Lee, K., Mazeika, M.: Using Pre-Training Can Improve Model Robustness and Uncertainty. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 2712–2721. PMLR (09–15 Jun 2019), https://proceedings.mlr.press/v97/hendrycks19a.html
19. Hoogendoorn, M., Funk, B.: Machine Learning for the Quantified Self: On the Art of Learning from Sensory Data. Springer (2017)
20. Hu, J., Shen, L., Sun, G.: Squeeze-and-Excitation Networks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7132–7141 (2018). https://doi.org/10.1109/CVPR.2018.00745
21. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely Connected Convolutional Networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2261–2269 (2017). https://doi.org/10.1109/CVPR.2017.243
22. Iv, W.C.S., Kapoor, R., Ghosh, P.: Multimodal Classification: Current Landscape, Taxonomy and Future Directions. ACM Comput. Surv. (jun 2022). https://doi.org/10.1145/3543848, https://doi.org/10.1145/3543848, just Accepted
23. Jha, S.B., Babiceanu, R.F., Pandey, V., Jha, R.K.: Housing Market Prediction Problem Using Different Machine Learning Algorithms: A Case Study. CoRR **abs/2006.10092** (2020), https://arxiv.org/abs/2006.10092
24. Jones, D.: A Taxonomy of Global Optimization Methods Based on Response Surfaces. J. of Global Optimization **21**, 345–383 (12 2001). https://doi.org/10.1023/A:1012771025575
25. Kalisch, M., Michalak, M., Sikora, M., Wróbel, L., Przystałka, P.: Influence of Outliers Introduction on Predictive Models Quality. In: BDAS. vol. 613, pp. 79–93 (04 2016). https://doi.org/10.1007/978-3-319-34099-9_5
26. Koch, D., Despotovic, M., Sakeena, M., Döller, M., Zeppelzauer, M.: Visual Estimation of Building Condition with Patch-Level ConvNets. In: Proceedings of the 2018 ACM Workshop on Multimedia for Real Estate Tech. p. 12–17. RETech'18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3210499.3210526, https://doi.org/10.1145/3210499.3210526
27. Kostic, Z., Jevremovic, A.: What Image Features Boost Housing Market Predictions? IEEE Transactions on Multimedia **PP**, 1–1 (01 2020). https://doi.org/10.1109/TMM.2020.2966890

28. Kucklick, J.P., Müller, O.: Location, Location, Location: Satellite Image-Based Real-Estate Appraisal. ArXiv **abs/2006.11406** (2020)
29. Lahat, D., Adali, T., Jutten, C.: Multimodal Data Fusion: An Overview of Methods, Challenges and Prospects. Proceedings of the IEEE **103**(9), 1449–1477 (Aug 2015). https://doi.org/10.1109/JPROC.2015.2460697, https://hal.archives-ouvertes.fr/hal-01179853
30. Law, S., Paige, B., Russell, C.: Take a Look Around: Using Street View and Satellite Images to Estimate House Prices. ACM Trans. Intell. Syst. Technol. **10**(5), 54:1–54:19 (2019). https://doi.org/10.1145/3342240, https://doi.org/10.1145/3342240
31. Oki, T., Ogawa, Y.: Model for Estimation of Building Structure and Built Year Using Building Facade Images and Attributes Obtained from a Real Estate Database, pp. 549–573. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-76059-5_27, https://doi.org/10.1007/978-3-030-76059-5_27
32. O'Shea, K., Nash, R.: An Introduction to Convolutional Neural Networks (2015). https://doi.org/10.48550/ARXIV.1511.08458, https://arxiv.org/abs/1511.08458
33. Poursaeed, O., Matera, T., Belongie, S.: Vision-Based Real Estate Price Estimation. Machine Vision and Applications **29**(4), 667–676 (Apr 2018). https://doi.org/10.1007/s00138-018-0922-2, http://dx.doi.org/10.1007/s00138-018-0922-2
34. Rijsbergen, C.J.V.: Information Retrieval. Butterworth-Heinemann, 2nd edn. (1979)
35. Sandler, M., Howard, A.G., Zhu, M., Zhmoginov, A., Chen, L.: Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation. CoRR **abs/1801.04381** (2018), http://arxiv.org/abs/1801.04381
36. Shah, A.D., Bartlett, J.W., Carpenter, J., Nicholas, O., Hemingway, H.: Comparison of Random Forest and Parametric Imputation Models for Imputing Missing Data Using MICE: A CALIBER Study. American Journal of Epidemiology **179**(6), 764–774 (01 2014). https://doi.org/10.1093/aje/kwt312, https://doi.org/10.1093/aje/kwt312
37. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going Deeper with Convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1–9 (2015). https://doi.org/10.1109/CVPR.2015.7298594
38. Szeliski, R.: Computer Vision - Algorithms and Applications. Texts in Computer Science, Springer (2011)
39. Tan, M., Le, Q.: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 6105–6114. PMLR (09–15 Jun 2019), https://proceedings.mlr.press/v97/tan19a.html
40. Wang, Z., Wang, Y., Wu, S.: House Price Valuation Model Based on Geographically Neural Network Weighted Regression: The Case Study of Shenzhen, China (2022). https://doi.org/10.48550/ARXIV.2202.04358, https://arxiv.org/abs/2202.04358
41. Wu, J., Chen, X.Y., Zhang, H., Xiong, L.D., Lei, H., Deng, S.H.: Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization. Journal of Electronic Science and Technology **17**(1), 26–40 (2019). https://doi.org/https://doi.org/10.11989/JEST.1674-862X.80904120, https://www.sciencedirect.com/science/article/pii/S1674862X19300047

42. You, Q., Pang, R., Cao, L., Luo, J.: Image-Based Appraisal of Real Estate Properties. IEEE Transactions on Multimedia **19**(12), 2751–2759 (Dec 2017). https://doi.org/10.1109/tmm.2017.2710804, http://dx.doi.org/10.1109/TMM.2017.2710804
43. Zhao, Y., Chetty, G., Tran, D.: Deep Learning with XGBoost for Real Estate Appraisal. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1396–1401 (2019). https://doi.org/10.1109/SSCI44817.2019.9002790
44. Zhou, B., Khosla, A., Lapedriza, A., Torralba, A., Oliva, A.: Places: An Image Database for Deep Scene Understanding. Journal of Vision **17** (10 2016). https://doi.org/10.1167/17.10.296
45. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A.: Places: A 10 Million Image Database for Scene Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(6), 1452–1464 (2018). https://doi.org/10.1109/TPAMI.2017.2723009

Appendices

Appendix A: All features in the dataset

Table 25. The features present in the dataset, including feature types and a brief description.

Feature	Type	Description
construction year	continuous	The year the property was built
deposit amount	continuous	The deposit amount associated with the listed property
energy index	continuous	The energy index
lat	continuous	The latitude of the property
listing price	continuous	The rental price per month for the listed property
listing price sqm	continuous	The rental price per square meter for the listed property
listing size (m2)	continuous	The size of the property in square meters
listing volume (m3)	continuous	The volume of the property in cubic meters
lon	continuous	The longitude of the property
number of photos	continuous	The number of photos of the property
publication year	continuous	The year the listing was published
publication date	continuous	When the apartment is rented
published on	continuous	When the apartment is listed on the platform
service costs	continuous	The service costs associated with the listed property
balcony	binary	Indicator whether a balcony is present
furnished	binary	Indicator whether the property is furnished
garage	binary	Indicator whether a garage is present
garden	binary	Indicator whether a garden is present
monumental building	binary	Indicator whether the property is a monumental building
parking availability	binary	Indicator whether there is parking availability
protected townscape	binary	Indicator whether the property is part of a protected townscape
publication published	binary	Indicator whether the publication is published
shell	binary	Indicater whether the property is delivered shell or not
storage	binary	Indicator whether a storage is present
upholstered	binary	Indicator whether the property is upholstered
energy label	ordinal	The energy label
floor level	ordinal	The floor level at which the property is located
max rent period	ordinal	The maximum rent period of the property
min rent periode	ordinal	The minimum rent period of the property
number of floors building	ordinal	The total number of floors of the building in which the property is located
total bathrooms	ordinal	Total number of bathrooms in the listed property
total bedrooms	ordinal	Total number of bedrooms in the listed property
total rooms	ordinal	Total number of rooms in the listed property, excluding bathroom and toilet
acceptance category	categorical	The type of acceptance
acceptance date	categorical	The date of acceptance of the property
available since	categorical	The date that the property is available since
building type	categorical	Indicator whether the property is new or existing
city	categorical	The city in which the property is located
district name	categorical	The district in which the property is located
interior type	categorical	The interior type of the property
listing city name	categorical	The city in which the property is located
listing photo urls	categorical	The photo urls of the listing
listing postal code	categorical	The postal code in which the property is located
listing residential type	categorical	Indicator whether it is a house or apartment
listing street name	categorical	The street in which the property is located on
maintenance_status	categorical	The maintenance status
municipal name 2020	categorical	The municipal in which the property is located
neighbourhood name	categorical	The neighbourhood in which the property is located
parking type	categorical	The type of parking availability
postal code	categorical	The postal code in which the property is located
source	categorical	The source of the listing