Hedonic Price Models and Indices based on Boosting applied to the Dutch Housing Market

Martijn Kagie*
Michiel van Wezel

Econometric Institute, Erasmus School of Economics and Business Economics
Erasmus University
P.O. Box 1738, 3000 DR, Rotterdam, The Netherlands

Econometric Institute Report EI 2006-17

April 5, 2006

Abstract

We create a hedonic price model for house prices for six geographical submarkets in the Netherlands. Our model is based on a recent data mining technique called boosting. Boosting is an ensemble technique that combines multiple models, in our case decision trees, into a combined prediction. Boosting enables capturing of complex nonlinear relationships and interaction effects between input variables.

We report mean relative errors and mean absolute error for all regions and compare our models with a standard linear regression approach. Our model improves prediction performance with up to 40% compared with Linear Regression. Next, we interpret the boosted models: we determine the most influential characteristics and graphically depict the relationship between the most important input variables and the house price. We find the size of the house to be the most important input for all but one region, and find some interesting nonlinear relationships between inputs and price.

Finally, we construct hedonic price indices and compare these to the mean and median index and find that these indices differ notably in the urban regions of Amsterdam and Rotterdam.

1 Introduction

Hedonic pricing theory hypothesizes that the price p of a product is determined by a function $p = F^*(\mathbf{x})$, where $\mathbf{x}$ is a bundle of characteristics that define the product. Hedonic pricing theory is generally attributed to Court (1939), Lancaster (1966), Griliches (1971b, 1971a) and Rosen (1974).

In practice the hedonic function $F^*(\mathbf{x})$ is estimated by a model $F(\mathbf{x})$ which is fitted on a historical dataset $\{p_i, \mathbf{x}_i\}_1^N$. Traditionally, these models are Linear Regression or Box-Cox type models.

In the context of housing and real estate hedonic models are useful in three ways. In the first place a hedonic model is a very suitable way to predict house prices. House price prediction can be used for bulk appraisal for property tax, but can also help real estate brokers by determining the asking price for a house. However, it is also possible to use a hedonic method for a website

*We thank Jaap Darwinkel and Dree Op 't Veld for their helpful comments and suggestions. Corresponding author: E-mail: kagie@few.eur.nl. Phone: +31 10 4088940. Fax: +31 10 4089031.

feature, where potential costumers can check their house value informally, after which they may decide to sell their house, although they did not intend to do so in advance.

The model structure itself can also be interesting, especially when one wants to find out what the influence is of a characteristic of the house on the price or which characteristics have the highest influence on the price. By interpreting the hedonic model these questions can be answered. Harrison and Rubinfeld (1978) for example use a hedonic model to find a relationship between air pollution and house prices.

The third way in which the hedonic model can be useful, is when it is used to create a hedonic price index. A hedonic price index uses a hedonic model to correct for quality differences over time. Ordinary indices may give a deceptive view, because the average or median product in year t may be a better (or worse) product than in year $t-1$. An average house in the 1930's for instance can in no way be compared with an average house sold in the year 2004 (since these have different characteristics), but nonetheless this is what a regular price index does. Hedonic indices for housing are for instance constructed by Wallace (1996) and Clapp (2004).

Traditional hedonic models have the advantage they are easy to interpret and estimate, but often suffer from misspecification: The assumptions made on functional form do not allow a good representation of reality, i.e., the hedonic model does not fit the data well. Several authors, e.g. Anglin and Gençay (1996), Gençay and Yang (1996), Pace (1998), Clapp (2004), Boa and Wan (2004), Bin (2004), Martins-Filho and Bin (2005), have used semi- and non-parametric methods to estimate a hedonic price model and compared these models with the traditional parametric hedonic models. Usually these new models outperformed the parametric models in terms of prediction performance. Also artificial neural nets, a popular machine learning technique, are frequently used for the estimation of the hedonic function, e.g. by Daniels and Kamp (1999), Kershaw and Rossini (1999), Lomsombunchai, Gan, and Lee (2004). An artificial neural net is a very flexible model, which in theory is a universal function approximator.

A recent successful machine learning method is boosting. Boosting is a relative new method to combine multiple models into a combined prediction. These individual models are called *base learners*. Often regression or classification trees (Breiman, Friedman, Olshen, and Stone 1983) are used as base learners, but a combination of other models, e.g., neural nets is also possible (Drucker 1999) (We will describe regression trees and boosting in more detail in Section 2.). Van Wezel, Kagie, and Potharst (2005) use boosting for hedonic pricing. Boosted hedonic price models where created for 3 small and simple datasets available on the internet: one dataset deals with automobiles, the other two with houses. On two out of the three datasets the boosted models substantially improved out-of-sample performance compared with a stepwise linear regression model.

In this paper, boosted regression trees will be used to create hedonic models for 6 regions in the Netherlands. Real-life data of the year 2004, collected by the largest association of real estate brokers in the Netherlands, the NVM, will be used. These hedonic models will be used for prediction, interpretation and the construction of hedonic price indices.

The remainder of this paper has the following structure. In the next Section we will discuss regression trees and boosting in more detail. Section 3 describes the data used in the experiments. Section 4 describes the performed experiments and their results. In Section 5 two graphical methods, relative importance plots and partial dependence plots are used to interpret the boosted hedonic price models. A hedonic price index based on our boosted models is discussed in Section 6. Finally Section 7 gives a discussion, conclusions and directions for further research.

2 Models & Methods

In this section we will discuss the Machine Learning techniques used in this paper. Generally, in Machine Learning a model is trained (fit, calibrated) on a dataset $D = \{(\mathbf{x}_i; y_i)\}_{i=1}^{N}$. An instance (observation, row) $(\mathbf{x}; y)$ exists of a vector of J attributes $\mathbf{x} = (x_1, \ldots, x_J)$ and a target y. The $\mathbf{x}$ attributes are the explanatory or independent variables and the target is the explained or dependent variable. In Machine Learning a distinction is made between classification and regression models. In the classification case the targets $\{y_i\}_1^N$ have a categorical or binary value,

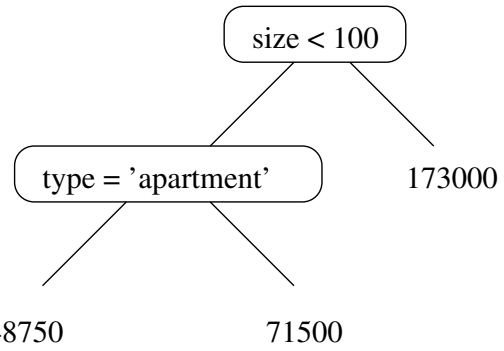


Figure 1: Example of a Regression Tree: The root node divides the houses in the dataset in two parts using the continuous variable *size*. Instances with a size smaller than 100 m^2 go left, all others right. Instances going right get a price of 173000 euros. The instances who were going right, will be split a second time on the categorical *type*. When the type is *apartment*, the instance is going left.

in the regression case a continuous value. In both cases the attributes x_j can be either continuous, categorical or ordinal.

2.1 Classification and Regression Trees

CART (Classification And Regression Trees) is one of the most frequently used methods for constructing decision trees and developed by Breiman, Friedman, Olshen, and Stone (1983). In this paper we will use the CART regression tree.

A regression tree consists of *decision nodes* and *leaf nodes*. Each decision node has two child nodes, which may again be decision nodes or leafs. The *root* of the tree is on the very top – it is the only node in the tree without an ancestor. Every decision node (also called non-terminal node) contains a split criterion, which divides the data at that point in two parts. This split criterion has the shape of $x_j < C$ for continuous variables, where x_j is the j^{th} variable and C is some constant. For a categorical variable the split criterion looks like $x_j \epsilon (V), V \subset W_j$. Here is W_j collection of all possible levels of variable j. The terminal nodes (leafs) contain a $\hat{y}$ value, an estimate for the target value in that leaf. In practice this value is taken to be the average of all observed y values in that leaf.

An example of a regression tree is shown in Figure 1. This tree can be used for prediction as follows. We begin at the root and when the split criterion is met we turn left and when it is not met we turn right. We keep on doing this until we reach a terminal node and use the $\hat{y}$ value in that node as our prediction.

Decision trees are usually built in two phases. The first phase is a *growing* phase, the second phase is a *pruning* phase. In the growing phase, the tree is grown until error reduction on the training set is no longer possible or a predetermined threshold has been reached. The resulting model usually overfits the data, and this is countered in a pruning phase, where the tree is shrunk until the error on a hold-out sample, the *pruning set*, is minimal. Details on the CART procedure for growing and pruning can be found in, e.g., (Breiman, Friedman, Olshen, and Stone 1983; Ripley 1996). Here, we suffice by saying that, given dataset $\{(\mathbf{x}_i; y_i)\}_1^N$, a regression tree B is constructed such that it minimizes the squared error loss

$$B = \arg\min_B E_{\mathbf{x},y}[(B(\mathbf{x}) - y)^2],$$

where $B(\mathbf{x})$ denotes the prediction of tree B for input vector $\mathbf{x}$.

In the context of Boosting, discussed below, the pruning phase of the decision tree algorithm is usually skipped and instead the tree size is limited to a predetermined depth. In the most extreme case the tree depth is 1. The tree then consists of a single decision node and two leaves.

Such a special tree is called a *decision stump*. Although a single decision stump has very limited modelling power, an ensemble of such stumps is able to model complex relationships.

Regression trees have some advantages over linear regression. In the first place regression trees are able to determine themselves which attributes are to be used for modelling the target variable. Another advantage is that regression trees are able to model interactions between attributes and non-linearities, without a required explicit transformation of the inputs. For example, there may be an interaction effect between *lot size* and *location*. In the center of a city houses are built more closely to each other so gardens are smaller or absent. A square meter of building site in the city will therefore be far more expensive than in the country, so there is an interaction between the two variables.

There are many examples of non-linearities in the context of housing in the Netherlands, of which *construction year* is probably the most clear one. Where ancient houses from the 19^{th} century and before are more expensive than same kind of houses from later periods, because of their monumental character, houses from the 1950's on the other hand are relative cheap. Since many houses where destructed in World War II, a lot of new dwellings had to be built as quick as possible in the late 1940's and 1950's, so that buildings built in this period are of a less quality and also not the most beautiful ones. Finally newly built houses are of course again more expensive, so the average relationship between *construction year* and *price* is clearly non-linear. Many other variables show saturation effects, e.g. the utility of a 3^{rd} or 4^{th} *room* is much higher than the utility of the 8^{th}.

Besides the previous advantages there are also two practical ones: In contrast to parametric models regression trees can handle categorical variables and missing values, without transformation of the data.

A drawback of decision trees is their instability – The implemented model depends heavily on the dataset used for model creation, and a small change in the data may have large consequences for the model. Ensemble methods, such as bagging (Breiman 1996) and boosting, have a stabilizing effect by averaging over a number of decision trees. We consider boosting next.

2.2 Boosting

Boosting is a method to combine multiple models to improve performance. Boosting was first applied to and developed for classification problems (with categorical response) by Freund and Schapire (1996, 1997). In a classification context boosting seemed to be able to strongly reduce the error rate on out-of-sample data in many cases (Breiman 1998). The idea behind boosting is to create a sequence of models, called base learners, in which each subsequent base learner focusses on the residual error of the previous base learners. Often, these base learners are decision trees or stumps. The original Freund and Schapire boosting algorithm for classification, AdaBoost.M1, tries to do this by increasing the weight of instances that were wrongly classified by the previous base classifier and decreasing the weight of the correctly classified instances.

The AdaBoost.M1 algorithm was only applicable to binary classification problems. For these problems, the model predicts whether an instance belongs to a class or not. The model thus has a 0/1 output and the quality of the model is measured with the $0-1$ loss function, which basically counts the number of misclassifications. For modelling house prices this loss function is not suitable. Instead, we need a regression loss function that measures the deviance between two numerical values, as usual in regression. This means that we cannot apply the AdaBoost.M1 algorithm to house price prediction, but we have to use a Boosting algorithm for regression. We will pay more attention to loss functions for regression below.

Driven by the success Boosting had in the case of classification various Boosting algorithms were designed for regression (Drucker 1997; Zemel and Pitassi 2001; Duffy and Helmbold 2000; Duffy and Helmbold 2002; Rätsch, Warmuth, Mika, Onoda, Lemm, and Muller 2000; Buhlmann and Yu 2003; Friedman 2001). Friedman (2001) developed LSBoost, LADBoost and MBoost based on the squared, absolute and Huber loss function respectively. (All these loss functions apply to regression problems.) Duffy and Helmbold (2000, 2002), Rätsch, Warmuth, Mika, Onoda, Lemm, and Muller (2000) and Buhlmann and Yu (2003) developed similar algorithms but with other loss

functions. Friedman (2002) also created stochastic variants of his boosting algorithms, where a base learner is repeatedly trained on a sample drawn without replacement from the dataset.

In this paper we will use the non-stochastic versions of Friedman's LSBoost and LADBoost algorithms. These algorithms are chosen because, contrary to many other Boosting algorithms, they have a solid mathematical foundation: they are instantiations of a general Boosting algorithm for general loss functions named GradientBoost. We now give a brief, imprecise description of GradientBoost, LSBoost and LADBoost. More detailed descriptions of these algorithms can be found in (Friedman 2001) and in Appendix A. We will pay more attention to loss functions shortly, in Subsection 2.3.

Contrary to fitting a single model, like the decision tree B above, boosting starts of with an initial guess F_0 and then fits a *sequence* of M models $B_1, \ldots, B_M$ (the base learners) which are subsequently combined in a weighted manner. The final model is thus

$$F_M(\mathbf{x}) = F_0(\mathbf{x}) + \sum_{m=1}^{M} \nu \rho_m B_m(\mathbf{x}).$$

Here, ρ_m denotes the weight for model m and is determined by the algorithm. M, the number of iterations, is to be set by the user. $\nu \in (0; 1]$ denotes a regularization parameter parameter called the *learning rate*. Small values of ν will help prevent the algorithm overfitting the training data.

Note that in the m^{th} iteration, $B_m()$ is added to F_{m-1}:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \rho_m B_m(\mathbf{x}).$$

It makes sense to choose $B_m()$ such that is minimizes the residual error of F_{m-1}. Roughly speaking, given a general loss function $L(y, F)$, B_m attempts to minimize the expected value of this loss function over the dataset:

$$B_m = \arg\min_B \sum_{i=1}^{N} L(y_i, [F_{m-1}(\mathbf{x}_i) + B(\mathbf{x}_i)]).$$

In practice this is done by fitting *pseudo responses* $\tilde{y}_i$ in each iteration

$$B_m = \arg\min_B \sum_{i=1}^{N} \{\tilde{y}_i - B(\mathbf{x}_i)\}^2.$$

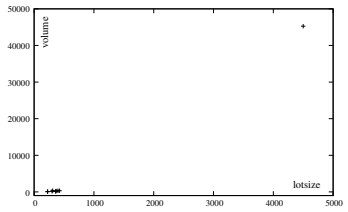
Three remarks must be made about this equation. The first remark is that the $\tilde{y}$ values depend upon the loss function in question. See the appendix for details on how these pseudo-responses are derived. Here, we suffice by stating that the pseudo-responses for the

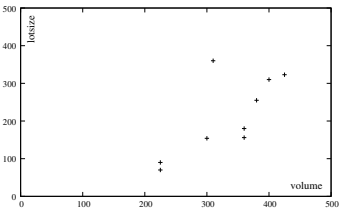
- squared error loss function $L(y, F) = (y - F)^2/2$ are given by $\tilde{y}_{i|m} = y_i - F_{m-1}(\mathbf{x}_i)$, and
- for the absolute deviation loss function $L(y, F) = |y - F|$ are given by $\tilde{y}_{i|m} = sign(y_i - F_{m-1}(\mathbf{x}_i))$.

The squared error loss function and the absolute deviation loss function are used in the LSBoost and LADBoost algorithms respectively.

The second remark is that the minimization over B is done by minimizing over B's parameter space. If B is a tree, these parameters are the split variables and split points in the decision nodes, and B_M is the tree that gives the best fit of the $\tilde{y}$ values in iteration m. If B is a neural network, these parameters are the weights and biases of the neural network.

The third remark is that *no matter what loss function* boosting attempts to minimize, a least squares regression is performed in each step of the Boosting algorithm. So, even if the Boosting algorithm as a whole does not minimize squared error loss but another loss function, the individual base models are fit by least squares regression.





no.	volume	lotsize	price
1	4500	45270	4310912
2	425	323	420000
3	400	310	329000
4	310	360	275000
5	380	255	232500
6	360	180	210000
7	360	156	195000
8	300	154	183500
9	225	90	167000
10	225	70	137500

Table 1: Dataset used in the example. Left: scatter plot with outlier, middle: scatter plot without outlier, right: dataset.

2.3 Loss Functions & Boosting Example

In this subsection we give an illustrative example of the first iterations of the Boosting process for a small dataset. This subsection also serves to illustrate the consequences of using the squared error- or absolute deviation loss function. The first loss function is the most common in regression settings. However, a drawback of this loss function is its sensitivity to outliers, since squaring the errors blows up the large errors caused by the outliers. This drawback can be relieved by using the absolute deviation loss function.

For the example, we use the data in Table 1. These data come from the Apeldoorn dataset (see below) and represent 10 houses in the Apeldoorn region. Both volume, lot size and price are approximately 10 times as high for house 1 as for the 2^{nd} most expensive house, number 2. Clearly, house 1 is an outlier.

In the example below, we use both LSBoost, which attempts to minimize the squared loss function, and LADBoost, based on the absolute deviation. We will see how both these methods cope with the outlier. As base learners, we will use decision stumps – trees with only one decision node and two leaves. The learningrate is set to 1.

2.3.1 LSBoost example

First we start with the LSBoost algorithm. First the initial guess F_0 is computed. This is the mean price of the datapoints:

$$F_0 = \bar{y} = 646041.20$$

The outlier is responsible for the fact that the first guess F_0 is in between the highest and the second highest price. With the use of F_0 the residuals $\tilde{y}$ are computed and on these residuals the first tree is trained. The first stump splits on $lotsize < 22815$, which means that outlier datapoint 1 is excluded from the other points. After this step there are thus two groups: the outlier and the rest. Since our interest is mainly in the normal houses, we actually know nothing at all after this step, which is graphically shown in plot F_1 of Figure 2.

In the second iteration again a stump is trained on the residuals. Outlier datapoint 1 has a residual of 0, because it is correctly predicted by the first base learner. Further datapoints 2 and 3 have a positive residual and the rest a negative one. It is the task of the second stump to find a split that separates the high (positive) from the low (negative) residuals. In fact the second stump uses $lotsize < 282.5$ to separate the 5 less expensive (who had negative residuals)from the 3 expensive houses (who had non-negative residuals). In plot F_2 this division is shown. Datapoint 1 is not shown in the plot, so that the focus is on the "normal" houses. Remarkable is that the prediction of datapoint 1 is not perfect any more after the second iteration, since this second stump adds €76875 to the (correct) price of F_1.

In the third iteration again datapoint 1 has the highest residual in absolute terms. Although the prediction of F_2 has only made a mistake of 1.8% on this point, datapoint 1 will again has a high influence on the third stump. Also points 4 and 8 were highly overestimated up to now, where datapoint 5 is highly underestimated. It is obvious that there is no way to separate these

3 overestimated point from the underestimated one. But when datapoint 1 was excluded from the dataset, we could use the criterion $volume < 335$ to split the overestimated points from the underestimated ones. But the third stump chooses to separate point 1 and 4 from the rest using $lotsize \geq 341.5$. This means that this stump in fact gives a penalty on these high lot sizes, i.e. it modelled a negative effect between *lot size* and *price*. When we look at plot F_3 we see that due to this penalty, houses with a $lotsize \geq 341.5$ (and $lotsize < 22815$) have a lower predicted price than houses with a $lotsize < 341.5$.

Finally in the fourth iteration the four cheapest houses are separated form the 6 expensive ones again on the variable *lotsize*. The spit criterion is $lotsize < 155$. After 4 iterations LSBoost has produced a model that divides the space of possible houses into 5 parts all based on *lotsize*. A higher *lotsize* leads to a higher *price*, except for the area $341.5 \leq lotsize < 22815$. Only datapoint 4 is in this area. This house has indeed the second largest *lotsize* of the 10 houses, but in terms of *volume* it comes only on a seventh place. This typical split (of the third iteration) was influenced by the outlier, despite the prediction of this house was relative good. And also the first iteration was negative influenced by the outlier, because we had preferred an extra split between the normal houses, which is far more useful for out-of-sample prediction. Of course these effects will be smaller in the real experiments, because of the larger sizes of the datasets and the use of a smaller value for the learningrate parameter.

2.3.2 LADBoost example

LADBoost tries to minimize an absolute loss function. To do this, it uses the sign (-1,0 or 1) of the residual instead of the residual themselves as pseudo responses and medians instead of means. In theory these adaptations should make LADBoost more resistant to outliers. Also with LADBoost we start with an initial guess, in this case the median of the prices of the dwellings in the dataset:

$$F_0 = median(y) = 221250$$

Since it is the median, it is obvious 5 datapoints are overestimated and 5 underestimated. The five more expensive houses get a $\tilde{y} = 1$ and the five cheaper ones get a $\tilde{y} = -1$. The first stump can split these to groups perfect using the split criterion $lotsize < 217.5$. The medians of the residuals of the datapoints in a node are finally used as the prediction. For the left side this is the residual of datapoint 8, -37750, and for the right side this is the residual of datapoint 3, 107750. So after the first iteration datapoints 3 and 8 are correctly classified and the outlier has had no influence in the process. Graphically the first step of LADBoost is shown in plot F_1 of Figure 3.

In the second iteration, shown in plot F_2, it is impossible for a decision stump to separate the positive signed targets from the negative signed ones. The stump splits the data on $volume < 335$. This is the split we wanted LSBoost to make in its third iteration, but that it did not make, caused by the outlier. In the LADBoost algorithm the residual of the outlier has the same influence on the training process as the other residuals due to the fact that the signs of the residuals are taken. This is the reason the split on *volume* is performed here.

In the third iteration the outlier has for the first time influence on the process. There are five datapoint with positive signs, the 2 most expensive houses and houses 6,8 and 9, which are situated relatively in the middle of the data. Since it is impossible to separate datapoints 6,8 and 9 in some way from the other points, it is obvious that the split is made on $volume < 412.5$, so that point 1 and 2 are separated from the rest. House 1 and 2 are thus together one node with residuals of 3962912 and 72000. According to the algorithm these houses are predicted by the base learner by taking the median of the residuals in the node. The median of two points is the mean: 2017456. Obviously this has a terrible influence on the prediction of the second house, which was predicted 72000 euros too low by F_2, but is predicted nearly 2 million too high by F_3. The outlier has in this case a high influence, but in large datasets this situation is unlikely to occur.

Finally in the fourth iteration the same split is made as in the first iteration, only signs are switched: $lot \geq 217.5$. In fact the influence of the first base learner is reduced.

When we compare these steps of LADBoost with LSBoost, we see that the influence of the outlier is less than in the LSBoost algorithm. Only in the third iteration the outlier has an

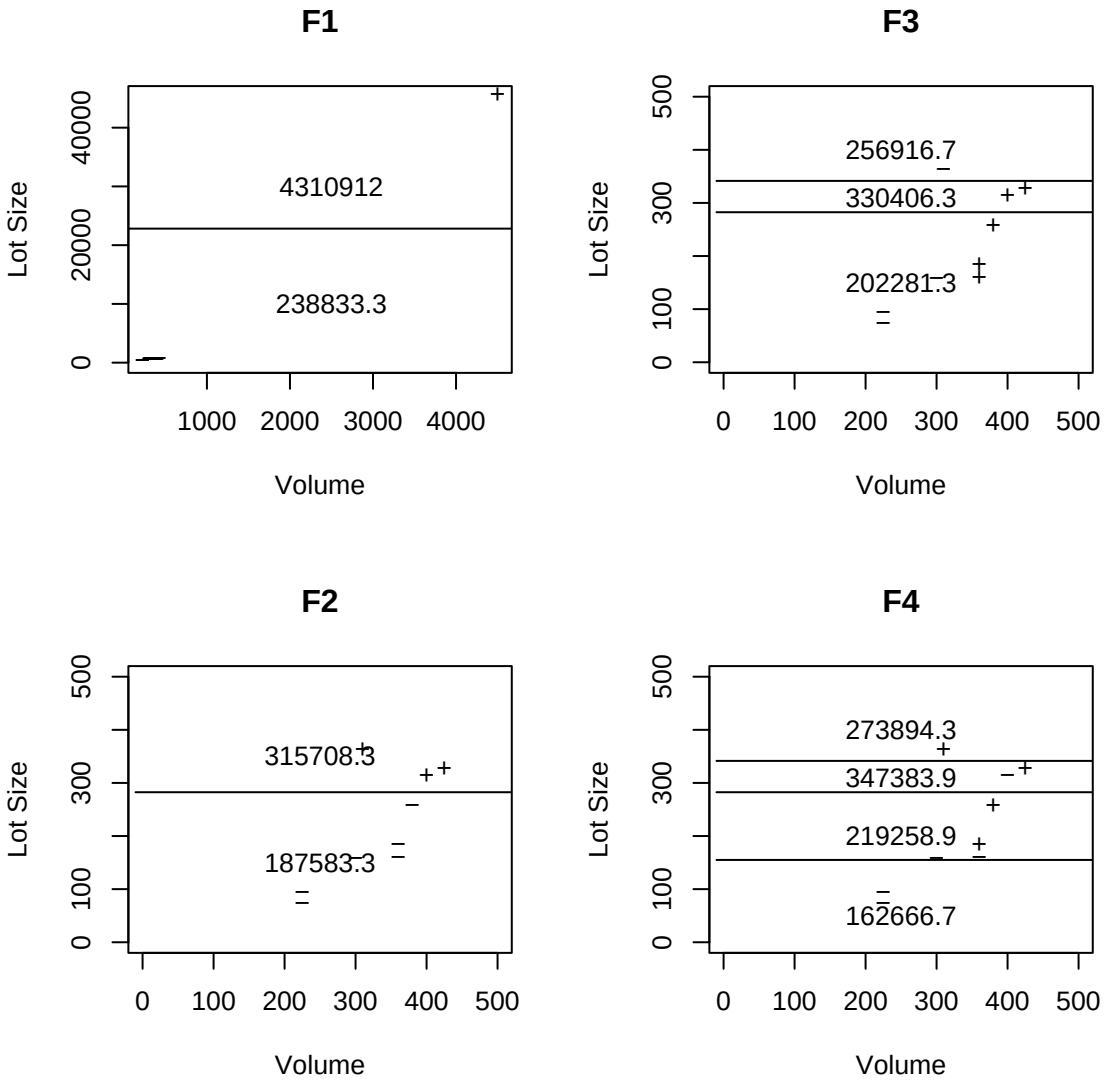


Figure 2: LSBoost process

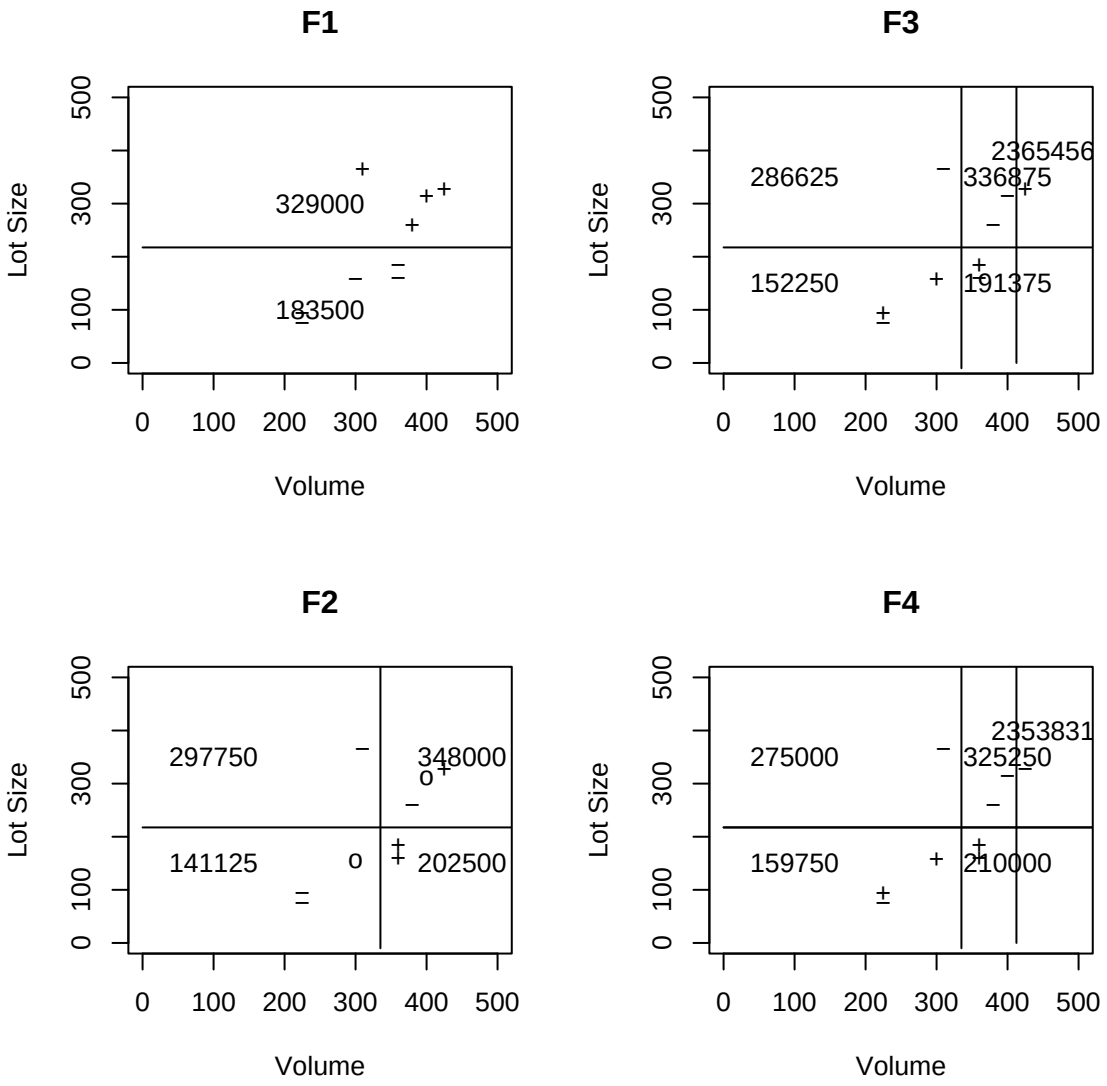


Figure 3: LADBoost process

influence, but this is a rather large one. However, the small size of the dataset is mostly responsible for this influence. LADBoost has the advantage that it is less sensitive to outliers, but this means also that is expected to be worse in the appraisal of expensive houses. Therefore we will use both algorithms in this paper.

2.3.3 Loss Functions for Model Evaluation

We mentioned above that we use either absolute or squared error in the boosting (model fitting, training) procedure. However, for the purpose of interpretation the squared error loss function is not very suitable, as the mean (or sum of) squared error(s) may largely be determined by an outlier.

Figure 4: The situation of the 6 NVM Regions.

Thus, for model evaluation we use either the Mean Absolute Error or the Mean Relative Error:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|F_M(\mathbf{x}_i) - y_i|, \tag{1}$$

$$MRE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{F_M(\mathbf{x}_i) - y_i}{y_i}\right|. \tag{2}$$

where $F_M(\mathbf{x}_i)$ denotes the prediction as before. These loss functions have the advantage of better interpretability, and they are used to report model performance even when a model is fit using the squared loss function.

3 Data

The datasets used in this paper are derived from the database of the Nederlandse Vereniging van Makelaars (NVM, Dutch Association of Real Estate Brokers). The NVM is the largest Dutch association of real estate brokers with 3751 members. 80 % of the sworn brokers in The Netherlands is a member of the NVM. The NVM has divided The Netherlands in 80 living regions or housing markets. Most people move inside such a region and therefore these regions can be seen as distinct markets. For the experiments 6 living regions are used: City of Groningen and environs, Apeldoorn and environs, Eindhoven and environs, Amsterdam, Rotterdam and Zeeland. In the next section the 6 regions will be briefly described and in Section 3.2 an overview of the used variables will be given.

	groningen	apeldoorn	eindhoven	amsterdam	rotterdam	zeeland
Size Dataset	3387	2402	3730	8490	7510	2216
NVM Market Share (2003)	77%	91%	51%	76%	56%	33%
Housing Stock	99034	76330	168342	430733	418467	157377
Percentage Houses in Dataset	3.42%	3.15%	2.22%	1.97%	1.79%	1.41%
Total Area Size (m^2)	288.10	498.38	308.98	303.33	448.83	2689.38
Size Land (m^2)	272.45	496.17	304.20	239.02	337.75	1645.30
Percentage Land	94.57%	99.56%	98.45%	78.80%	75.25%	61.18%
Population Size	217593	189309	392773	855094	899781	349683
Population Density	799	382	1291	3577	2664	213
Apartments	45.73%	22.31%	24.21%	79.41%	61.58%	10.47%
Terraced	34.96%	35.64%	72.65%	18.20%	31.80%	48.29%
Detached	8.92%	25.60%	9.92%	0.94%	2.30%	25.36%
Semi Detached	10.39%	16.44%	17.35%	1.45%	4.31%	15.88%

Table 2: Some characteristics of the 6 regions.

3.1 Regions

Figure 4 shows where the regions are located in the Netherlands and Table 2 summaries some characteristics of the 6 regions. A brief description of the six regions is given below[1]:

City of Groningen and environs: The city of Groningen is located in the province Groningen in the north east of the Netherlands. The region contains the municipalities of Groningen (179185 inhabitants, 76 km^2), Haren (19048, 46 km^2), Eelde (6920, 13 km^2), Norg (7160, 110 km^2) and Peize (5280, 28 km^2). Groningen is the largest city in the north west of the Netherlands and the capital of the province with the same name. It is surrounded by small rural villages. With 77% the NVM has a relatively high market share in this region.

Apeldoorn and environs: Apeldoorn is located in the province Gelderland in the east of the Netherlands. The region contains the municipalities of Apeldoorn (156000 inhabitants, 340 km^2) and Epe (33309, 156 km^2). Like Groningen Apeldoorn is mainly surrounded by small villages. This region is interesting, because all the house types are almost equally present in this region and the NVM has a very high market share of 91% in this region.

Eindhoven and environs: Eindhoven is located in the province Noord-Brabant in the south east of the Netherlands. The region contains the municipalities of Eindhoven (207870 inhabitants, 88 km^2), Nuenen (23367, 34 km^2), Son en Bruegel (15070, 26 km^2), Valkenswaard (31091, 55 km^2), Veldhoven (42545, 32 km^2), Waalre (16502, 22 km^2) and Geldrop (27670, 13 km^2). Eindhoven is the largest city in the south east of the Netherlands and has became this mainly, because of the factories and head quarters of DAF Trucks and Philips, that were located in this city. Eindhoven itself is a typical Dutch city with a large amount of terraced houses. The NVM has a market share of 51% in this region.

Amsterdam: Amsterdam is located in the province Noord-Holland in the west of the Netherlands. The region contains the municipalities of Amsterdam (739104 inhabitants, 161 km^2), Amstelveen (78886, 44 km^2), Diemen (24049, 12 km^2) and Ouder-Amstel (13055, 24 km^2). Amsterdam is the largest city in the Netherlands and the capital of the country. The city is situated on the borders of the river IJ. Amstelveen, Diemen and Duivendrecht (Ouder-Amstel) are suburbs of Amsterdam. The NVM has a market share of 76% in this region.

Rotterdam: Rotterdam is located in the province Zuid-Holland in the west of the Netherlands. The region contains the municipalities of Rotterdam (598923 inhabitants, 206 km^2), Capelle aan den IJssel (65354, 14 km^2), Krimpen aan den IJssel (29046, 8 km^2),Nieuwekerk aan den IJssel (22334,17 km^2), Vlaardingen (74058,24 km^2), Schiedam (75619, 18 km^2), Albrandswaard (19607, 22 km^2) and Nederlek (14831, 28 km^2). Rotterdam, the second largest city in the Netherlands, is one of the largest sea ports in the world and located in delta of

[1]All statistics that are used in the descriptions and Table 2 are derived from the online StatLine database of the CBS (de Jonge, Reedijk, and Sluis 2005), except the NVM market shares.

the rivers Rhine, Maas, Lek and IJssel and near the North Sea coast. The market share of the NVM real estate brokers in this region is 56%.

Zeeland: The region Zeeland contains almost the complete province of Zeeland, located in the south west of the Netherlands. The region contains the municipalities of Borsele (22318 inhabitants, 142 km^2), Goes (36591 inhabitants, 93 km^2), Hulst (27887, 201 km^2), Kapelle (11627, 37 km^2), Middelburg (46350, 49 km^2), Reimerswaal (20996, 102 km^2), Terneuzen (55412, 251 km^2), Veere (22130, 133 km^2), Vlissingen (45236, 34 km^2), Schouwen-Duiveland (34416, 231 km^2), Noord-Beveland (7124, 86 km^2) and Sluis (24596, 280 km^2). Zeeland exist of a number of islands and peninsulas (Schouwen-Duivenland, Noord- and Zuid-Beveland, Walcheren and Zeeuws-Vlaanderen) and because of that 40% of the total area of Zeeland is water. Zeeland is one of the less dense populated regions in the Netherlands and also the NVM has a relative small market share (33%) in this region. Besides the large size of this region these two reasons would make that Zeeland is expected to be the region that is the hardest to predict.

3.2 Variables

For these 6 regions all transactions of existing houses (no newly built houses) that were sold by NVM brokers in the year 2004 are used to create the datasets. The transaction price in euros is the target variable or response. There are 83 explanatory variables, of which 12 variables are continuous, 12 variables are categorical and the other 59 are binary. Both the continuous and the categorical variables have some missing values. Not all categorical variables are compulsory to fill in by the broker who sold the house and continuous variables contain some odd values like 9999 and 0's for number of total rooms and 1's for house sizes. These odd values are replaced by missing values. Binary values measure whether something is present or not, so missing values are replaced by 0 in the dataset. A description of the variables for the 6 regions is given in Table 9 up to 14 in the Appendix. The floor specific variables (except rooms per floor) are summarized in Table 15 for all regions together. When a variable has the same value for all houses in a dataset the variable is omitted in the table. For the districts only the 5 biggest districts are given in the tables, but all districts are used in the computations.

3.3 Demographic Data

According to many real estate brokers location is one of the most important characteristics that determine the house price. In the dataset location is included in the terms of a district code. In the different regions 3 levels of geographic positioning can be used: On municipality, district and neighborhood level. Of course we want to include a positioning at the lowest possible level. However, the disadvantage of the use of neighborhood coding, is its number of possibilities. Because there are many different neighborhoods and some of them are very small, there are neighborhoods where during the time of one year only one or two houses are sold and often even none. The use of neighborhoods coding will thus definitely lead to classification problems (because one tries to estimate the price of a house positioned in a neighborhood unknown to the model) and overfitting of the dataset (because many neighborhoods are represented in the dataset by just a couple of houses). On the other hand the use of district coding has the disadvantage that, especially in big cities as Amsterdam and Rotterdam, the defined districts are quite large, with up to $80,000$ inhabitants and a surface of up to $4,000$ hectares.

Due to these considerations we use another way represent location, namely with the use of demographic data. Statistics Netherlands (Centraal Bureau voor Statistiek, CBS) publishes demographic information for all the municipalities, districts and neighborhoods in the Netherlands. This information is free accessible on the internet via StatLine (de Jonge, Reedijk, and Sluis 2005). The neighborhood codes of the houses available in the dataset are matched with the CBS neighborhood information. When a characteristic is missing for some neighborhood, this characteristic is replaced by the value for the district the neighborhood belongs to. When this information is

also unavailable for the district, the municipality data is used. When information of neither of the 3 levels is available a missing value is inserted. Table 16 shows which information is included from the StatLine database.

4 Experiments and Results

In this section the experiments in which the boosting algorithms are tested on the datasets are described and their results are reported. Subsection 4.1 describes the setup for the experiments, the software and the parameter values used. Next we describe two experiments, one without demographic data and one with demographic data included, respectively in Subsections 4.2 and 4.3.

4.1 Setup for the Experiments

For the experiments the `R` programming environment (R Development Core Team 2005) is used. In each experiment the dataset is partitioned in a test set and a training set for 100 times. Each time, the training set contains 90% of the instances from the dataset, the other 10% of the instances form the test set. Both training and test set are random samples (without replacement) of the total dataset. Nevertheless, in each experiment with the same dataset the n^{th} training and test set in experiment A are equal to the n^{th} training and test set in experiment B. All error measures reported below are *test set errors* or *out of sample errors.* The datasets of the 6 regions described in the previous section are used. (Remember that the total dataset contains all transactions concerning existing houses in 2004.) We will now describe the configurations for the various model types.

Linear Regression. We use linear regression as a reference model in our experiments. As implementation of the linear regression model the `lm` function in `R` is used. We use the most simple version of Linear Regression, without any transformation of variables. This is done because the boosted regression trees also use untransformed input variables. Also a stepwise procedure is not used, because these procedures are too slow, when having large numbers of variables. Since linear regression is unable to use categorical variables, dummies are created for this kind of variable. Due to the very large number of possible values of the *district* variable this variable is not used in the linear regression analysis. The second disadvantage of linear regression is that it is unable to handle missing values. Therefore missing values have been replaced by the mean of the non missing values of the variable.

CART. For the CART experiments the `rpart` package (Therneau, Atkinson, and Ripley. 2005) in `R` is used. Pruning is done by 10 fold cross validation. Furthermore the minimal split criterion is set to 10, which means that a node split is only attempted if the node contains at least 10 instances. The minimum bucket size is 1 which means that a newly created node after a split must contain at least 1 instance. Finally, the complexity parameter (a regularization parameter) is set to 0 so that the tree will get all opportunity grow before it get pruned.

LSBoost and LADBoost. The base of both the boosting algorithms is also `rpart`, because CART is used as base learner for these algorithms. We implemented the LSBoost and LADBoost algorithms ourselves. The base learners have the same parameters as stand alone CART, except no cross validation and thus no pruning is performed and the depth of the tree is constrained to 1, 2, 3, 4 or 5. When the tree has a depth of 1, a so called stump is created with only the root and 2 terminal nodes. A depth of 2 and 3 are associated with maximums of 4 and 8 terminal nodes respectively et cetera. Depths higher than 5 are not evaluated because those kind of interactions are not likely to occur in reality.

The LSBoost and LADBoost algorithms themselves also have 2 parameters: the number of base learners M and the learning rate ν, for which we used values 300 and 0.1 respectively. These setting lead to good results, but there is no guaranty these are the optimal ones. There is a quite high possibility that with higher M and a lower ν better results will be produced, but the finding

Region	Best Model	MAE	Improvement	MRE	Improvement
groningen	LSBoost (5)	€20110.40	26.47%	11.32%	32.01%
apeldoorn	LSBoost (5)	€26643.74	24.38%	9.31%	32.39%
eindhoven	LSBoost (5)	€21856.95	26.71%	8.06%	31.98%
amsterdam	LADBoost (5)	€29320.47	32.84%	9.38%	44.00%
rotterdam	LADBoost (5)	€22637.88	34.60%	10.91%	40.15%
zeeland	LSBoost (5)	€28914.29	31.16%	14.83%	38.03%

Table 3: Out-of-sample performance of the best models per region. Improvement is the improvement with respect to the Linear Regression model. The number between the parentheses represents the tree depth of the base learner. See text for more comments.

	MAE			MRE		
Region	LSBoost	LADBoost	Impr. LADB	LSBoost	LADBoost	Impr. LADB
groningen	€20110.40	€20141.72	-0.16%	11.32%	11.62%	-2.65%
apeldoorn	€26643.74	€27824.49	-4.43%	9.31%	9.86%	-5.91%
eindhoven	€21856.95	€21822.29	0.16%	8.06%	8.09%	-0.37%
amsterdam	€29283.76	€29320.47	-0.13%	9.55%	9.38%	1.78%
rotterdam	€22067.10	€22637.88	-2.59%	10.96%	10.91%	0.46%

Table 4: Comparison of the results of LSBoost and LADBoost

of the optimal parameters is a time-consuming task that we considered to be beyond the scope of our research.

4.2 Experiment 1: Data with Districtcoding

This subsection gives the results of the experiments using districtcoding but *without* demographic variables. In Table 3 a summary of the best results is given. A complete overview of all results can be found in Appendix D, Table 17.

On each dataset 10 boosting experiments are performed (5 LSBoost and 5 LADBoost, with depths 1 up to 5), except on the Amsterdam and Rotterdam regions, where some less interesting experiments were skipped, because of their long computation times. All these boosting experiments outperformed the LR model on their dataset. Adding an extra layer always reduced both absolute and relative error, but the effect of every extra layer is less than the previous one. The improvement of a Boosting model with depth 5 with respect to a depth 4 model in terms of absolute error is around €200. It is therefore expected that the effect of adding a 6^{th} layer is negligible. There are two possible reasons why extra layers increase the prediction performance. In the first place because extra layers allow for extra interaction and in the second place, since the number of models is constant, extra layers just can make more splits on the data and therefore predict more accurate.

The boosting models with depth 5 reduce the absolute error with around 25% – 35% and relative error with around 30% – 45% with respect to the LR model. But which of the two boosting algorithms, LSBoost and LADBoost, performed best differed by dataset. When we look at the relative error, LADBoost is the best algorithm in the large cities of Amsterdam and Rotterdam, where LSBoost performed best in the other 4 regions. However, the difference in performance between the two algorithms is very small. Only in Apeldoorn and environs LSBoost is really better than LADBoost. Since LADBoost tries to ignore outliers, it is possible that the worse performance in Apeldoorn is explained by the bad prediction of expensive houses. In Table 4 a comparison is made between the results of both the boosting algorithms.

4.2.1 Submarkets

It is possible that prediction accuracy is very different among different types of houses or market segments. In real estate literature market segments are often called submarkets. We analyze the predictive performance of our models after a very simple market segmentation, in which each of the six possible house types (detached, apartments etc.) represents a segment.

The prediction error (MAE or MRE) of a submarket is the weighted mean test error of that submarket, i.e. in every test set the houses belonging to the submarket are collected and the

Region	Best	Size	LSBoost	LADBoost	Worst	Size	LSBoost	LADBoost
groningen	mid-terraced	901	9.67%	9.24%	detached	302	24.30%	24.98%
apeldoorn	mid-terraced	548	5.08%	5.15%	detached	615	14.39%	14.82%
eindhoven	apartment	903	6.92%	6.93%	detached	370	13.11%	12.26%
amsterdam	apartment	6742	9.06%	8.92%	detached	80	23.68%	21.90%
rotterdam	stepped	61	8.00%	8.04%	detached	173	19.66%	18.57%
zeeland	stepped	34	9.58%	9.83%	detached	562	20.29%	20.82%

Table 5: Best and worst predicted house type per region

mean is taken and multiplied by the number of houses belonging to the submarket in that test set. Afterward the sum of all these means is taken and divided by the sum of the house counts.

Table 5 gives the best and worst predicted house types per region. More detailed results can be found in Appendix D, Table 18.

The terraced houses, especially the mid-terraced house, are predicted relatively well in all regions. There are many terraced houses and when looking at the variables included, we may conclude that these datasets are especially created for terraced houses (and other single household residences). Many of the variables are uninteresting for apartments (gardens, floors etc.). Apartments are also predicted well, except in Zeeland. Detached houses seem to be the biggest problem, also when there are relative many houses of this type in the region, e.g. in Apeldoorn and Zeeland. Even then detached houses are quite complex to predict. Detached houses are often quite unique and besides the included variables also the architectonic style of the house may have a high influence. Also by the other house types we see when a house becomes more exotic prediction performance decreases. There are two approaches that may improve the prediction of these exotic houses, but these approaches create also new problems. Both approaches are based on extending the datasets. First we can include houses sold in other years in the dataset, but then we have to correct the prices to compensate for market dynamics. The second approach is to combine multiple regions. Disadvantage is we get an enormous number of districts, but maybe the use of demographic data (and preferable other environmental variables) can replace the districts.

It was expected that LADBoost would predict ordinary houses better and LSBoost reached better results on the larger dwellings. The submarket results don't confirm this expectation. Sometimes LADBoost predicts detached dwellings even better than LSBoost. Overall there does not seem to be a pattern in which algorithm predicts best on the different house types.

4.3 Experiment 2: Data with Demographic Variables

In Section 3.3 datasets were created where the district codes were replaced by a number of demographic variables on neighborhood level. The LSBoost and LADBoost, with depth 5, were also trained on these datasets. In Table 6 the best models per region are listed. A complete overview of the results is given in Table 19 in the Appendix.

The results of all the boosting models in all 6 regions are better on these new datasets compared with the results on the old ones, discussed in Section 4.2. The size of the performance increase differs among regions and models. Overall, the adaptation of the datasets has more influence on the LSBoost models than on the LADBoost models. In the region of Amsterdam prediction performance of the models has increased by about 10%, but in the region of Rotterdam this is only 3% to 4%. It does not come as a surprise that Amsterdam is the region where performance gain is highest, since the districts in Amsterdam are extremely large. With the use of demographic information the model can discover a better separation between different locations.

We can conclude that LSBoost or LADBoost models with depth 5 and built with the use of demographic data are the best choices among the models evaluated so far. These models improve the Mean Relative Error with 35% up to 49% compared with the original Linear Regression model.

Region	Best Model	MAE	Improvement	MRE	Improvement
groningen	LSBoost (5)	€18892.04	6.06%	10.41%	8.04%
apeldoorn	LSBoost (5)	€25497.43	4.30%	8.64%	7.20%
eindhoven	LSBoost (5)	€20748.22	5.07%	7.71%	4.34%
amsterdam	LADBoost (5)	€26373.36	10.05%	8.51%	9.28%
rotterdam	LADBoost (5)	€21936.93	3.10%	10.42%	4.49%
zeeland	LSBoost (5)	€27189.76	5.96%	13.90%	6.27%

Table 6: Results of the best models per region with the use of demographic data. Improvement is the improvement with respect to the model trained on the normal dataset.

5 Interpretation

Parametric techniques often have the advantage that a useful interpretation can be given to the model parameters, e.g., in linear regression the model parameters can be interpreted as the partial prices of the product characteristics. Although not parametric, regression trees are also highly interpretable and can be written as an equivalent set of if-then rules. Boosted trees lack both these appealing properties. Not all is lost, however, because two useful tools exist that can be used for the interpretation of these ensemble models:

Relative importance plots, that visualize how important the various independent variables are relative to one another in predicting the dependent variable. In regression trees, the relative importance of a variable is measured examining the effect of each split on that variable on the model outcome. Roughly speaking, this effect is high if the split results in a large difference in model outcome for the right and the left subtree (with preferably an equal probability to turn right or left) and/or the split is likely to occur. (The probability that a split occurs is indicated by the number of patterns that travel through the corresponding node relative to the number of patterns in the dataset. Generally this probability is higher for nodes close to the root.) In an ensemble of regression trees such as built by boosting, the relative importances are simply averaged over all trees in the ensemble.

Partial dependence plots, that visualize the partial dependence of the implemented function on a subset of the independent variables. For a grid of values for this variable subset, the 'expected' output of the model at that point is computed. Ideally, this expectation at a grid point should be computed with respect to the conditional distribution (given the grid point) of the variables *not* in the subset. In practice it is obtained by averaging the model outputs over all instances in the dataset, keeping the values in the selected variable subset fixed to the grid point, thus using the dataset to approximate the distribution.

A more detailed description of these plot types is given in Appendix B.

Both relative importance plots and partial dependence plots are made for the six datasets using an LSBoost model. An LSBoost model, with 300 CART base learners with depth 5 and learningrate 0.1, is trained on the complete 2004 dataset of a region. First we will use the datasets with *district* coding, but in Subsection 5.2 we will also create relative importance plots with the inclusion of demographic variables and partial dependence plots of some demographic variables.

5.1 Interpretation: Data with Districtcoding

Figure 5 shows the relative importance plots of the 6 regions. Only the 10 most important variables in a dataset are plotted. Except in Zeeland the most important variable in all regions is a variable that measures some size: *volume*, *size* or *lot size*. In Zeeland *kind* is the most important variable, but *kind* measures house size implicit. In general there are 6 variables that are in the top tens for all 6 datasets: *volume*, *size*, *lot size*, *kind*, *construction year* and *district*. It is remarkable that, although the size of the house is very important in the hedonic model, the number of rooms in a house hardly has an influence. The total size of these rooms is far more important.

We created partial dependence plots for 5 of the 6 most important variables mentioned in the previous section. Only for *district* no plots are created, because of the large number of different values this variable can have.

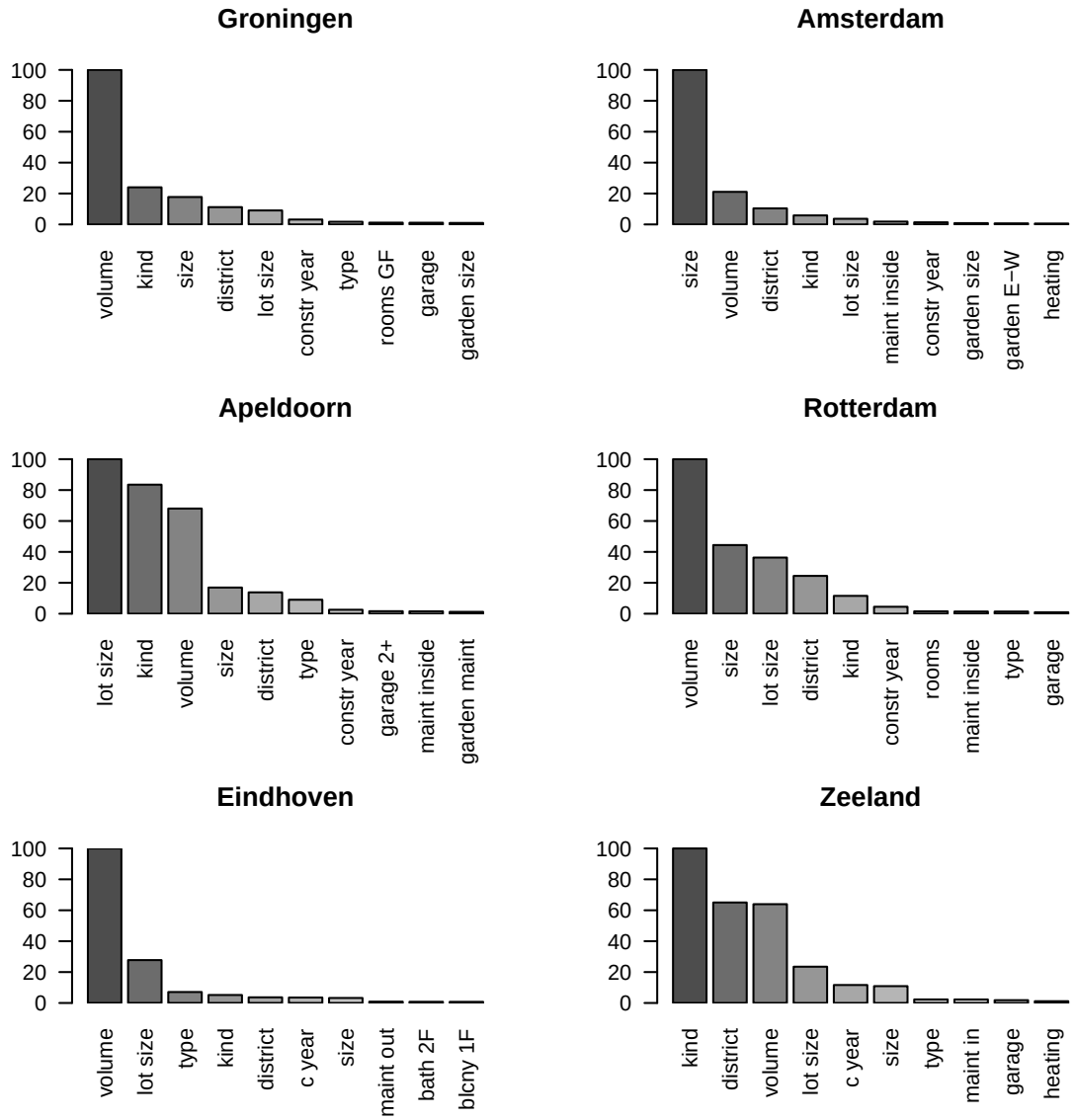


Figure 5: Relative Importance of predictors with Districtcoding.

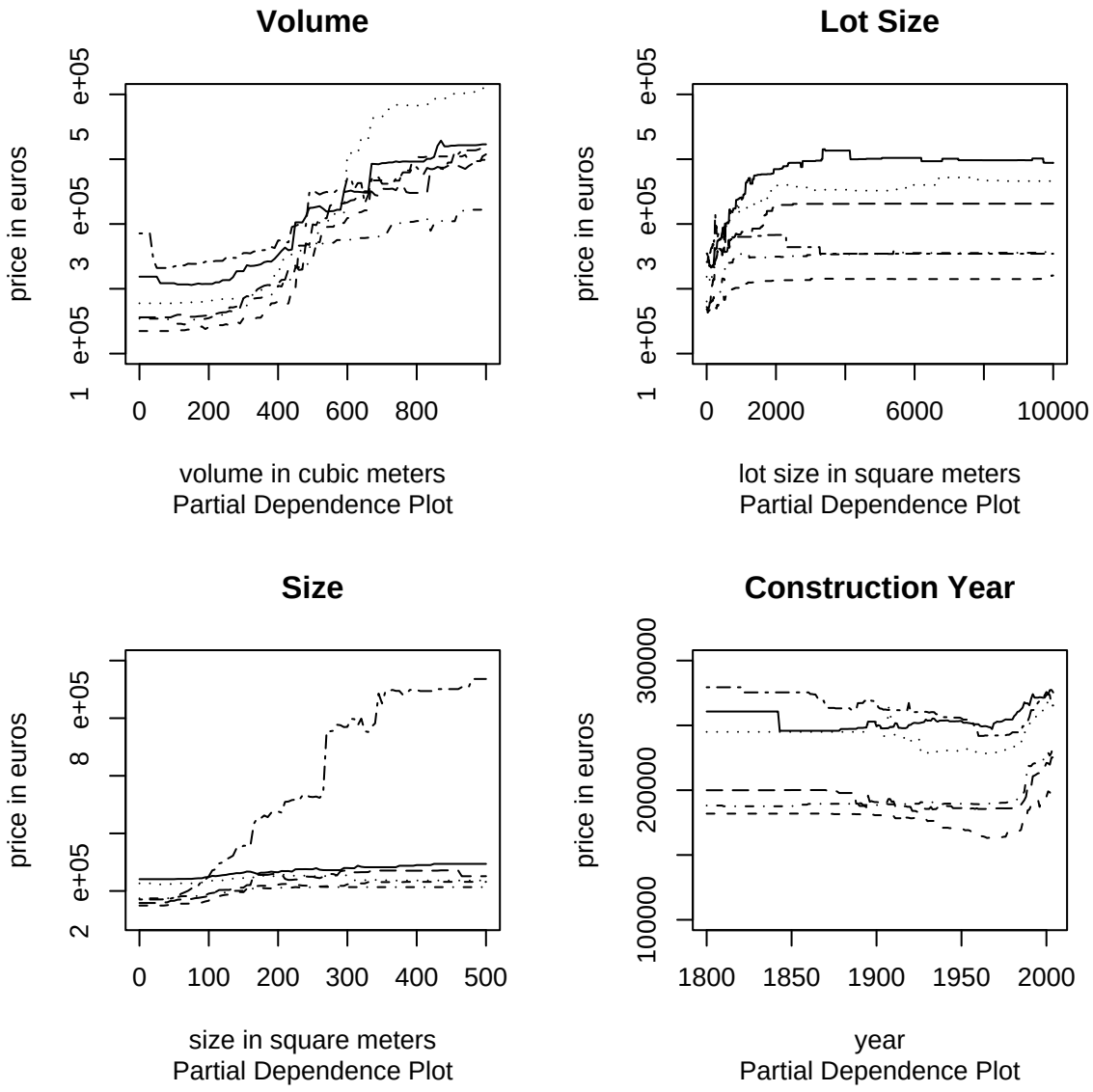


Figure 6: Partial Dependence Plots Continuous Variables: Groningen (dashed), Apeldoorn (solid), Eindhoven (dotted), Amsterdam (longdash), Rotterdam (twodash) and Zeeland (dotdash)

In Figure 6 the partial dependence plots of the four important continuous variables (*volume*, *size*, *lot size* and *construction year*) are shown. When interpreting these plots we must remember that only average relations are shown and that, for certain values of correlated predictors, the relationship may be entirely different.

Figure 7 shows the partial dependence plots of the categorical variable *kind house*. The plots are centered to have zero mean over the coresponding dataset. We can see in these plots that for example a country-estate in Apeldoorn has a partial dependence of $+60,000$ euros and a bungalow a partial dependency of $-20,000$. This means that the average effect of being a country-estate and not a bungalow is 60,000-(-20,000)=80,000. This seems a little odd, but we have to realize that we in fact ignore the other variables, although in fact a country-estate has probably a larger *lot size* and *volume*.

5.2 Interpretation: Data with Demographic Variables

Figure 8 shows these new relative importance plots. In all the 6 regions 2 up to 5 demographic variables are in the top 10 of most important variables. *Average income per income receiver* is 4 times in the top 10 and 3 times together with *average income*, which is also the total of top 10's for *average income*. Also *western immigrants* is 3 times in the top 10, which is remarkable. Further *address density* and *percentage 25-45 years* are both mentioned twice. Figure 9 shows partial dependence plots of the 4 continuous demographic variables. In general we see that these partial effects are much smaller than the ones shown in the previous subsection, since these variables are less important.

6 Hedonic Price Indices & Boosting

Hedonic models can be used to construct a so called hedonic price index. A hedonic price index uses hedonic models to correct for quality changes. In the next section we will describe which hedonic price indices exist and how they work. In Section 6.2 it is explained how boosting is combined with these indices. Finally in the last section results are given of the hedonic indices.

6.1 Hedonic Price Indices

In a recent paper by Pakes (2003) an overview of different hedonic indices is given and compared with a standard matched model index (or repeated sales index). Besides some hybrid indices Pakes describes three types of hedonic indices: Dummy variable indices, Laspeyres-like indices and Paasche-like indices. Wallace (1996) mentions the Fischer's Ideal index which is a combination of the Laspeyres- and Paasche-like hedonic index. In the next subsections we will briefly describe these 4 methods.

We will use the following notation. We construct an index over a number of periods $\{t\}_1^T$. For each of these periods we have a dataset of houses $\{(\mathbf{x}_i^t, p_i^t)\}_1^{n^t}$ with characteristics $\mathbf{x}^t$ and price p^t. Each of these datasets can be used to train a hedonic model $h^t(\mathbf{x})$. We will use period t_1 as the comparison period and period t_0 as the base period.

6.1.1 Dummy Variable Hedonic Index

The dummy variable hedonic index is the most frequently used hedonic index in economic research, mainly because it is very easy to construct. This index is obtained by pooling all the data together and then regressing the log of price on the characteristics and period specific dummy variables. The difference between two dummies coefficients is then the growth rate of that period. This method assumes that model parameters are constant over time, so the valuation of the characteristics of the houses does not change.

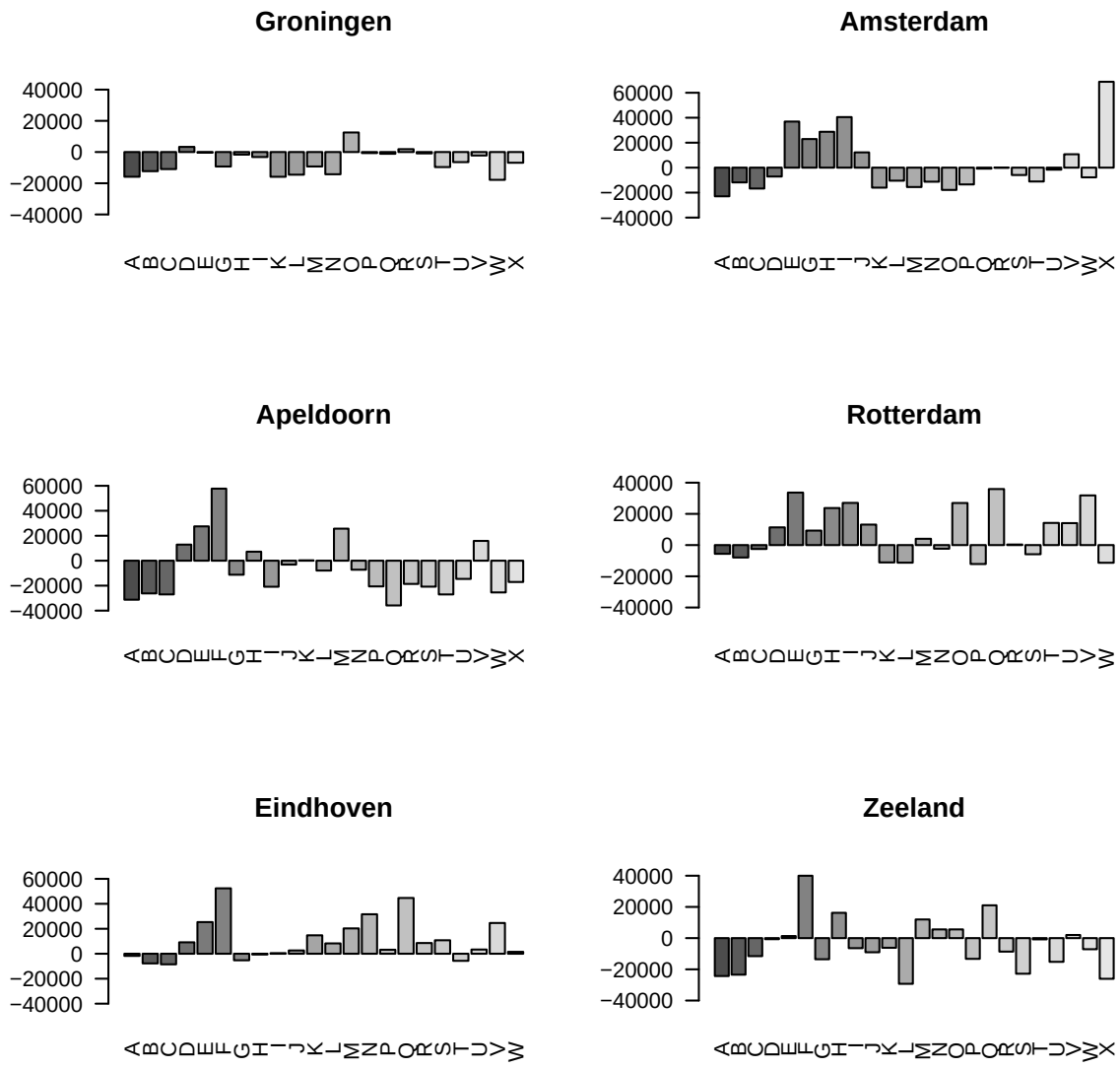


Figure 7: Partial Dependence Plot House Kind. From left to right the following kinds are plotted: simple house (A), middle class house (B), mansion (C), villa (D), country-house (E), country-estate (F), bungalow (G), patio-bungalow (H), semi bungalow (I), split-level (J), downstairs house (K), upstairs house (L), up- + downstairs house (M), staircase-access flat (N), canal house (O), maisonette (P), service flat (Q), flat with elevator (R), flat without elevator (S), house with office (T), drive-in house (U), farm house (V), apartment (W), holiday home (X)

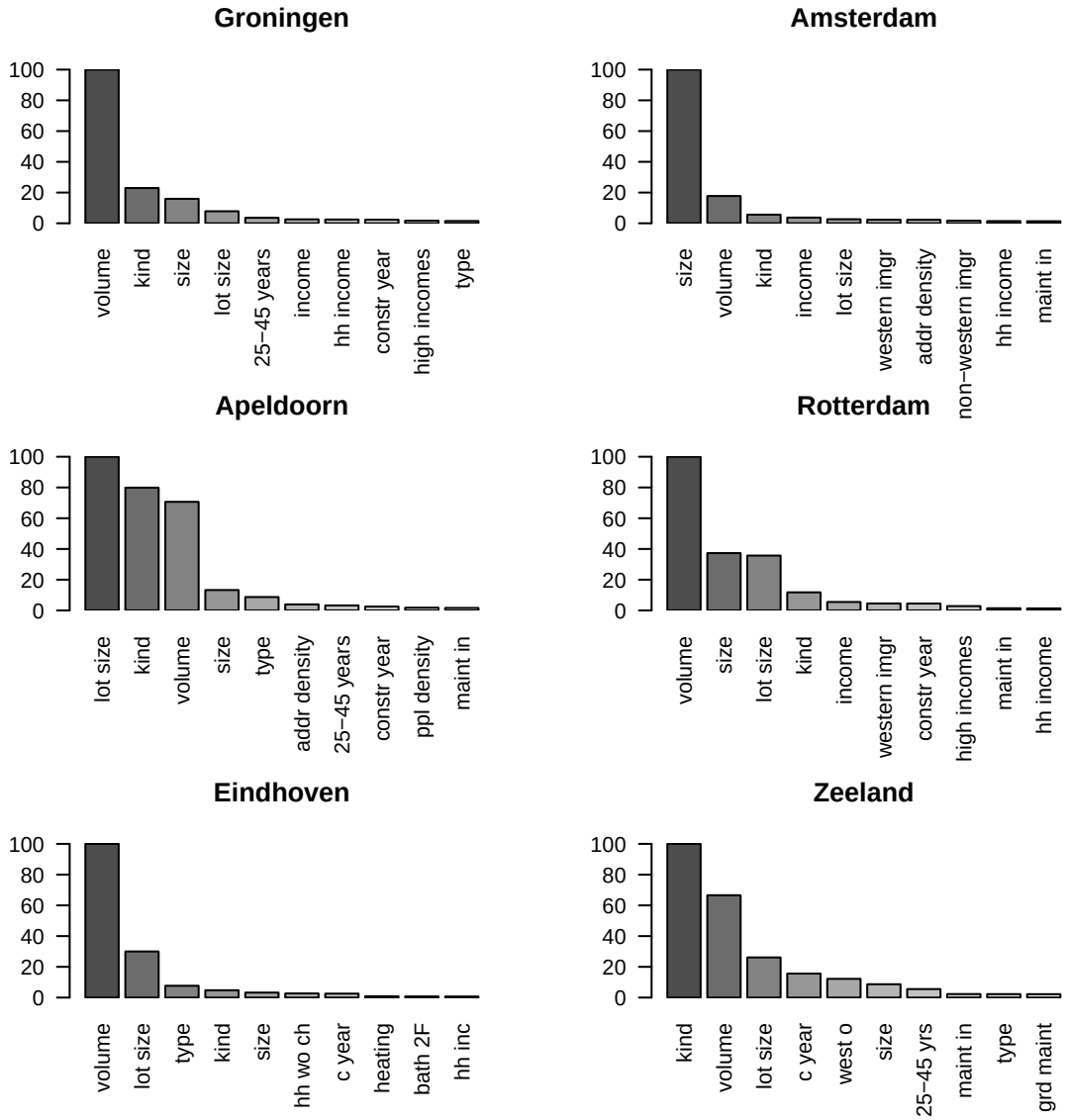


Figure 8: Relative Importance with demographic data

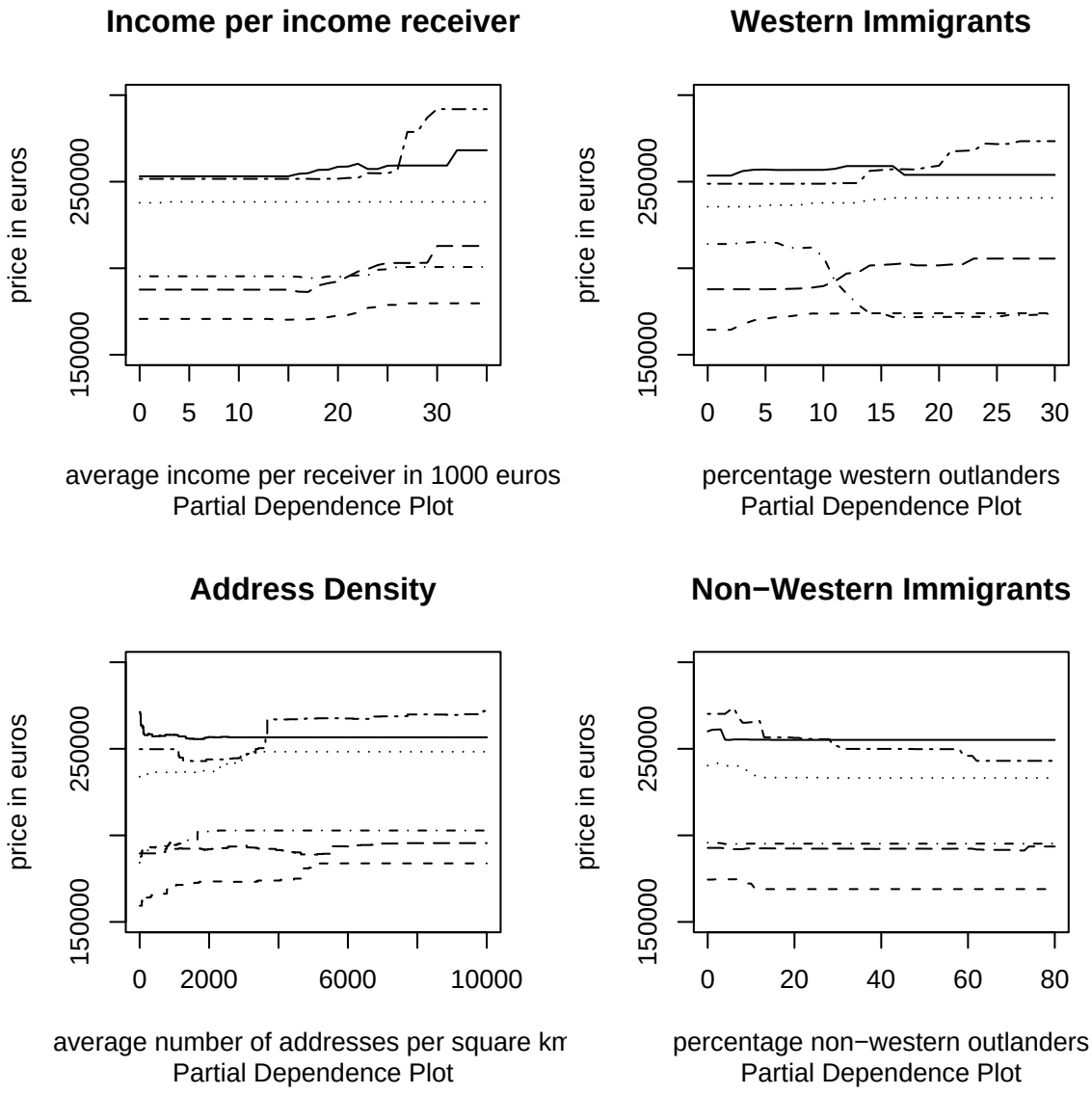


Figure 9: Partial Dependence Plots of Demographic Variables: Groningen (dashed), Apeldoorn (solid), Eindhoven (dotted), Amsterdam (longdash), Rotterdam (twodash) and Zeeland (dotdash)

6.1.2 Laspeyres-like Hedonic Index

According to Pakes(2003) the Laspeyres-like hedonic index tries to model the amount of income a consumer needs in the comparison period to buy the same good (in this case a house) as in the base period. We use the average over all consumers, thus over all houses sold in a period.

The average amount of income used for buying a house in the base period is known: It is the mean price of the houses in the base period. Now we want to know what amount of income a consumer needs to have in the comparison period to buy the same house, i.e. we need to know how expensive the houses sold in the base period are in the comparison period. Since it is very likely these houses are not sold again in the comparison period, we need to estimate these comparison period prices, using the characteristics of the houses. For the estimation of the comparison period prices we use a hedonic model h^{t_1} trained on the houses sold in the comparison period:

$$\tilde{p}_i^{t_1} = h^{t_1}(\mathbf{x}_i^{t_0})$$

The last thing we need to do to create the index is dividing the estimated mean price of the comparison period by the mean price of the base period:

$$I_{Laspeyres}^{t_1} = \frac{\sum_{i=1}^{n_{t_0}} h^{t_1}(\mathbf{x}_i^{t_0})}{\sum_{i=1}^{n_{t_0}} p_i^{t_1}}$$

Since the Laspeyres-like index holds the product bundle constant, it does not account for substitution effects. It is well possible that a consumer, given the "prices" of the characteristics in the comparison period, would buy a different house, by, for instance, substituting a wooden shed by an extra room (used as store room), because of increased prices of wood and thereby increased prices of wooden sheds. The Laspeyres-like index just computes the extra costs for the wooden shed, but the consumer derives the same utility to a house with an extra room, which is cheaper. The index thus overestimates the amount of income compensation that is necessary to stay at the same utility level.

6.1.3 Paasche-like Hedonic Index

For estimating a series of T price indices using a Laspeyres-like method, one needs to train $T-1$ models. The advantage of the Paasche-like hedonic index, is that only one model needs to be trained, the model of the base period. Disadvantage is that data quality of datasets is often worse in earlier years and a relatively bad model will be used for the construction of the index.

According to Pakes (2003) the Paasche hedonic models the amount of income that would make a consumer who bought the comparison periods bundle in the base period at least as well off in the comparison period as in the base period. This makes the Paasche-like hedonic the opposite to the Laspeyres-like hedonic, since it uses the comparison product bundle instead of the base period bundle. Now we know the prices of the bundle of houses sold in the comparison period and need to estimate these bundle's price in the base period:

$$\tilde{p}_i^{t_0} = h^{t_0}(\mathbf{x}_i^{t_1})$$

And this leads to the Paasche-like hedonic index:

$$I_{Paasche}^{t_1} = \frac{1}{n_{t_1}} \sum_{i=1}^{n_{t_1}} \frac{p_i^{t_1}}{h^{t_0}(\mathbf{x}_i^{t_1})}$$

Also the Paasche-like hedonic doesn't account for substitution effect, but because the Paasche hedonic works opposite to the Laspeyres hedonic, the index underestimates the amount of income compensation that is necessary to stay at the same utility level. Both Pakes(2003) and Wallace(1996) show that the Laspeyres-like and Paasche-like hedonic indices are the upper bound and lower bound of the real hedonic index.

Region	Mean	Median	Fisher's Ideal Hedonic
groningen	354.18	376.01	388.60
apeldoorn	396.79	377.36	408.49
eindhoven	367.48	356.19	404.94
amsterdam	408.60	402.42	511.95
rotterdam	334.73	335.80	430.24
zeeland	317.64	311.71	341.40

Table 7: Indices for the year of 2004. (1985=100)

6.1.4 Fisher's Ideal Hedonic Index

Since we can see the Laspeyres-like and Paasche-like index as the bounds of the real index, we know that the real index is in between those two. According to Wallace (1996) the Fisher's Ideal index will be the best estimation of the real index, when the Laspeyre-like and Paasche-like index lie "close" to each other. The Fisher's Ideal index is defined as the geometric mean of the Laspeyres-like and Paasche-like index:

$$I^{t_1}_{Fisher} = \sqrt{I^{t_1}_{Paasche} \cdot I^{t_1}_{Laspeyres}}$$

6.2 Using Boosting to Create a Hedonic Price Index

Boosting will be used to construct the indices mentioned in the previous chapter, except for the dummy variable index. This index is omitted in the experiments. In the first place because it is expected that the assumption of constant "prices" of the characteristics would not hold in reality. Second boosting is not an appropriate method to use in combination with the dummy variable index. Since we want in the case of the dummy variable index model a *ceteris paribus* effect of the dummy variables on the price no interaction with other variables is allowed. However, the power of boosting is in these interaction effects.

The other indices will be constructed on data from the period 1985 until 2004 for the same 6 regions as the other experiments. These datasets are also derived from the NVM database and have the same variables. Hence, the index gives only information about the dwellings that were sold by NVM brokers during these years and not about all dwellings. Since demographic data is not available for all years, we will use *district* coding instead. For the hedonic models $h^t(\mathbf{x})$ LSBoost models will be used with regression trees with depth 5 as their base learners. Further all parameters are the same as in the previous experiments. 1985 will be used as the base period and is set to 100. The Paasche and Laspeyres Index are only used for the creation of the Fisher's Ideal Index.

6.3 The Hedonic Price Indices for the Six Dutch Regions

Table 7 shows the hedonic indices for 2004 with 1985 as base period. The hedonic indices are compared with the mean and median index. This comparison is interesting, because we can see in this way, whether quality of the houses has improved during the period. Because both mean and median index don't correct for quality, they are just the mean or median of the comparison period divided by the mean or median of the base period, the difference between the hedonic index and the mean or median index can be used as a measure of quality improvement. When the hedonic index is below the mean or median index, this means that quality has improved, since the quality corrected prices are lower than the real prices. We see that house prices has raised impressive in all 6 regions. An avarage house was in 2004 3 up to 4 times more expensive than in 1985. Of course it is possible that houses were become that expensive, because houses have reached a higher level of quality during these years. The hedonic indices show that this is not the case. It is even the other way around, all hedonic indices in all 6 regions are higher than the mean and median index, what means that quality has decreased.

Figure 10 show the course of the indices over time for the 6 regions during the period 1985 until 2004. The plots show the Fisher's Ideal hedonic index together with the mean and median

	Lot Size			Volume			Apartments		
Region	1985	2004	%	1985	2004	%	1985	2004	%
Groningen	587.39	240.50	-59.06%	356.21	327.93	-7.94%	49.12%	46.06%	-6.22%
Apeldoorn	864.85	654.82	-24.29%	384.59	379.00	-1.45%	13.33%	22.04%	65.32%
Eindhoven	335.85	254.53	-24.21%	425.88	389.39	-8.57%	9.25%	23.48%	153.80%
Amsterdam	170.72	108.14	-36.65%	337.70	269.60	-20.17%	71.24%	79.26%	11.26%
Rotterdam	292.01	138.13	-52.70%	420.17	305.51	-27.29%	49.54%	61.46%	24.06%
Zeeland	675.99	467.29	-30.87%	428.07	377.89	-11.72%	16.20%	10.93%	-32.57%

Table 8: Differences of variable values between 1985 and 2004

index.

In all 6 regions the same pattern is visible only the absolute effects differ. Prices raises in the first 10 years with 40% in Zeeland up to 90% in Amsterdam, according to the mean and median indices. The Fisher hedonic shows raises of 58% in Zeeland up to 106% in Amsterdam, which means that the price raise was accompanied by a quality decline.

In the next 5 years, 1995 up to 2000 these effects became even stronger. In 2000 houses were almost 4 times so expensive in Amsterdam than in 1985. Corrected for quality this is even 5 times. Although both effects are the strongest in Amsterdam this pattern is also visible in the other 5 regions. The quality decline is the largest in big cities, such as Amsterdam and Rotterdam, but also in Eindhoven. In the 3 more rural regions the quality decline is less impressive. Due to these differences in quality decline, mean and median indices give a wrong notion about in which regions prices have increased the most. There is no discussion about the fact that Amsterdam is the region with the highest price increase and Zeeland the region with the lowest. But for the other 4 regions the mean and median indices give a deceptive view. The most striking example is Rotterdam. According to the mean and median indices this region is positioned fith out of the six regions in terms of price increase, but according to the hedonic indices it is positioned second, since the quality decline of the dwellings in Rotterdam is relatively high.

Table 8 shows some means of important variables in the years of 1985 and 2004. When these important characteristics have become of less quality (i.e. the average house volume has become smaller, the average lot size has decreased or more dwellings are apartments) this would be a decent explanation of the quality decline the hedonic models found between 1985 and 2004. Both *lot size* and *volume* have decreased during these 20 years in all of the 6 regions. *Lot size* with 24% in Eindhoven up to 59% in Groningen and *volume* with 1% in Apeldoorn up to 27% in Rotterdam. Also the relative amount of apartments has increased in 4 of the 6 regions. The regions with a large decline in the average house volume are also the 2 regions with a large quality decline according to the hedonic price index: Amsterdam and Rotterdam. The quality decline during the 90's can therefore mainly be explained by the fact that houses have become smaller during this period.

These hedonic indices give only a general overview of all possible dwellings. Since we can see in Table 8 that in some regions one reason of the decline in quality is the increase of the portion of apartments, it is possible that the average qualities of the different house types did not decrease. When one wants to check whether this is the case one could create hedonic indices per house type.

Although the hedonic indices give a good impression of the quality differences during a period, they are not 100% reliable. There are three reasons for this.

The first reason is a theoretic one and discussed in the previous section. Both the Paasche and Laspeyres index do not account for substitution effects. The Fisher's Ideal index only estimates these effect by using the geometric mean of the other two indices. However, the Laspeyres and Paasche index are situated very close to each other and thus the Fisher's index seems to be a good estimation.

In the second place the boosting models make prediction errors. Since the data quality and quantity become less going further back in history, this will be a problem. Especially for the Paasche-like index, because this approach uses only the base period model.

The third reason has to do with the meaning of the variables. There are two variables which meanings change over time. The first is *construction year*. This variable measures in fact two things: the age of the house and the period it is built in, e.g. the architectonic style and average quality of houses built in that period. This means a *construction year* of 1980 has a different

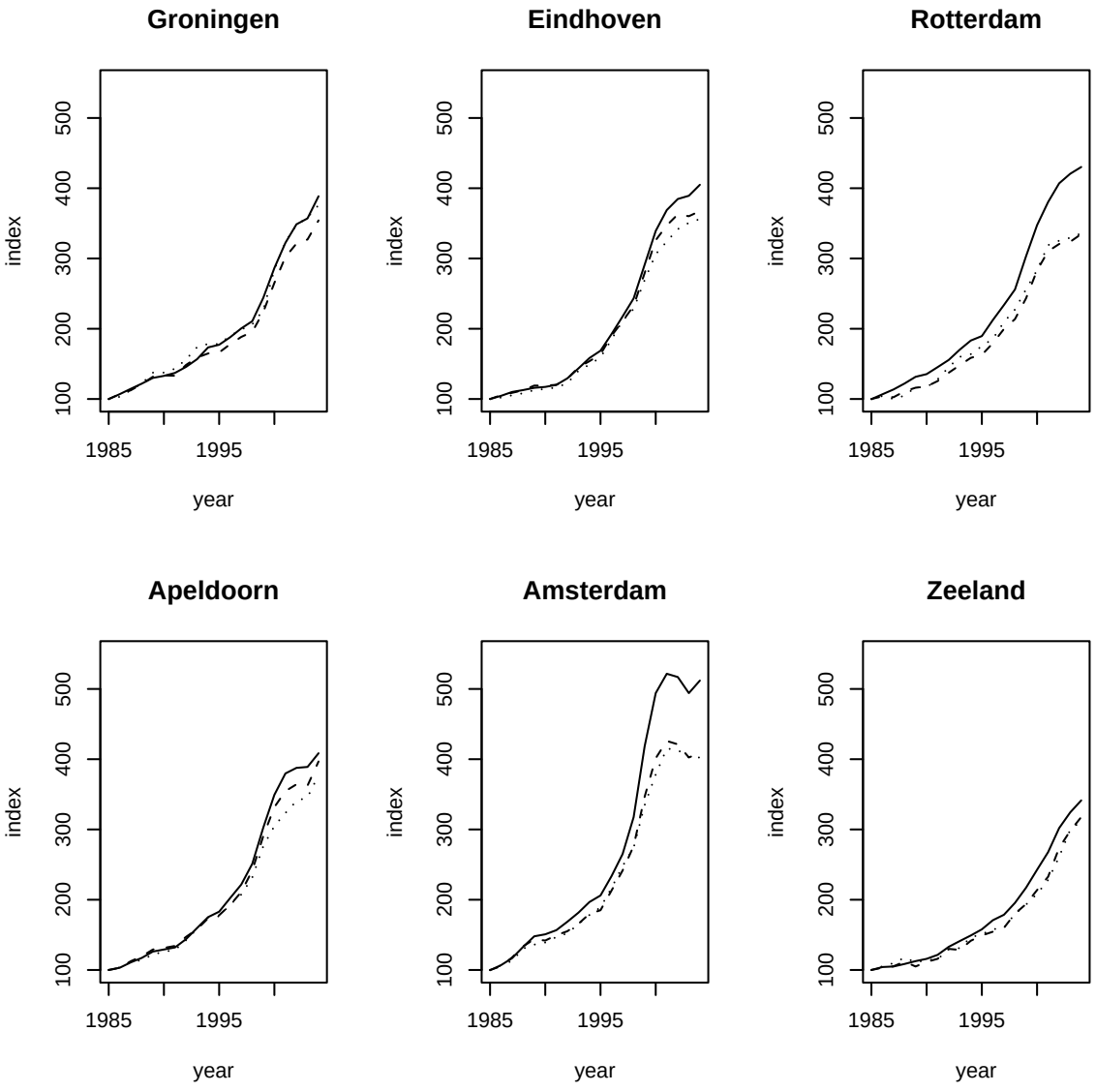


Figure 10: Hedonic and normal indices of the six regions: Fisher's Hedonic Index (solid), Mean Index (dashed) and Median Index (dotted).

meaning in 1985 than in 2004. Another problem is that houses in 1985 cannot have construction years higher than 1985 and the boosting model also cannot extrapolate. In the Paasche Hedonic this means that all construction years higher than 1985 were treated as built in 1985. The second variable which can cause problems is *district*. This is due by the fact that houses can be torn down and new houses can be built in a district during the period.

Overall the hedonic index gives a better notion of the real price index than mean and median index, despite its theoretical and practical flaws. The hedonic indices show that not only prices have increased, but also the quality of the dwellings has decreased, mainly because houses have become smaller.

7 Conclusions

Boosted hedonic models were used in this thesis in 3 applications in a housing context: Prediction, interpretation and construction of hedonic indices.

The predictions of a boosted hedonic model can be used for bulk appraisal. Of course models with a high prediction performance and low errors are preferred. The experiments in Section 4.2 show that the boosting models, LSBoost and LADBoost, improve prediction performance (MRE) with up to 40% in relation to an ordinary Linear Regression model. Further we tried to improve prediction performance by replacing the *district* codes by a number of demographic variables on neighborhood level. This improves the prediction performance of the boosting models by 5% to 10%. Also simple submarkets were created to see whether prediction performance differed a lot among different types of houses. These experiments show that terraced houses and apartments in some regions are the best predicted houses. Detached houses on the other hand are a problem for the models.

Prediction performance of the boosting models may possible be increased in the future. First the model itself can be approved. Model parameters M and ν were not optimized in the experiments in this thesis. Further other base learners can be used to improve model parameters. These base learners can be for instance monotone decision trees (Potharst and Bioch 2000) or multivariate adaptive regression splines (MARS) (Friedman 1991). Monotone decision trees have the advantage that these trees are monotone on a chosen set of variables, which forces the tree to construct a more plausible function. However, the boosting model also has to be made monotone itself for optimal performance. MARS models have the advantage that these functions are more smooth than regression trees and are able to extrapolate.

A second way in which the prediction performance of the boosting models may be approved is an extension of the dataset. First we can increase the number of instances in the training set by adding more years from the past to the model, but then a good way of correcting for price fluctuations is necessary. A good method for price fluctuation also corrects for in-year price fluctuations, which are also a problem for the prediction. Second we can use larger regions or even the complete country as region, but the use of *district* codes is not possible any more in that case, because of the high number of different possible values. An inclusion of demographic variables and preferable also some environmental variables will take away this problem. Besides increasing the number of instances also an increase of the number of variables may increase prediction performance, like in the case of the demographic data. However, this data has to be available or collected. One way to collect extra information may be the use of textmining to collect information out of descriptions brokers give about the house and its location.

For the interpretation of the boosting models two graphical models were used: Relative importance and partial dependence plots. Although interpretation of these plots seems easy at first, interpretation of these plots, especially the partial dependence plots, is sometimes rather difficult. The reason is the large number of variables used in the experiments that are also sometimes highly correlated with each other. For interpretation it would be better to use a smaller set of variables or perform factor analysis before training the model.

In the last chapter boosted hedonic models were used for the construction of hedonic price indices for the different regions. These indices were compared with median and mean price indices

to see whether there was a quality increase or decline during the evaluated period 1985 to 2004. The hedonic indices show in all six regions a quality decline of the dwellings. This quality decline is mainly explained by the fact that houses have become smaller and lot sizes have decreased dramatically during the last 20 years in the evaluated regions. An interesting extension of the use of these hedonic models in the future can be to use this hedonic index to compare different regions with each other. The mixture of houses differs between regions, so we cannot compare mean or median prices of regions with each other. A hedonic index corrects for these differences in housing quality between regions. The index has only be adapted so it uses a base and comparison region instead of a base and comparison period.

References

Anglin, P. and R. Gençay (1996). Semiparametric estimation of a hedonic price function. *Journal of Applied Econometrics 11*, 633–648.

Bin, O. (2004). A prediction comparison of housing sales prices by parametric versus semiparametric regression. *Journal of Housing Economics 13*, 68–84.

Boa, H. and A. Wan (2004). On the use of spline smoothing in estimating hedonic housing price models: Empirical evidence using hong kong data. *Real Estate Economics 32*(3), 487–507.

Breiman, L. (1996). Bagging predictors. *Machine Learning 24*, 123–140.

Breiman, L. (1998). Arcing classifiers. *Annals of Statistics 26*(2), 801–824.

Breiman, L., J. Friedman, R. Olshen, and C. Stone (1983). *Classification and Regression Trees*. New York: Chapman & Hall.

Buhlmann, P. and B. Yu (2003). Theory and methods - boosting with the l2 loss: Regression and classification. *Journal of the American Statistical Association 98*(462), 324–339.

Clapp, J. (2004). A semiparametric method for estimating local house price indices. *Real Estate Economics 15*(1), 127–160.

Court, A. (1939). Hedonic price indexes with automotive examples. In *The Dynamics of Automobile Demand*, pp. 99–117. New York: General Motors Corporation.

Daniels, H. and B. Kamp (1999). Application of mlp networks to bond rating and house pricing. *Neural Computing & Applications 8*, 226–234.

de Jonge, E., A. Reedijk, and W. Sluis (2005). *StatLine On The Internet*. Voorburg, The Netherlands: Centraal Bureau voor Statistiek. version 3.1, `http://statline.cbs.nl`.

Drucker, H. (1997). Improving regressors using boosting techniques. In D. Fisher (Ed.), *Proceedings of the 14^{th} International Conference on Machine Learning*, San Francisco, pp. 107–115. Morgan Kaufmann.

Drucker, H. (1999). Boosting using neural nets. In A. Sharkey (Ed.), *Combining Artificial Neural Nets: Esemble and Modular Learning*, pp. 51–77. Berlijn: Springer.

Duffy, N. and D. Helmbold (2000). Leveraging for regression. In *Proceedings of the 13^{th} Annual Conference on Computational Learning Theory*.

Duffy, N. and D. Helmbold (2002). Boosting methods for regression. *Machine Learning 47*, 153–200.

Freund, Y. and R. Schapire (1996). Experiments with a new boosting algorithm. In L. Saitta (Ed.), *Proceedings of the 13^{th} International Conference on Machine Learning*, San Francisco, pp. 148–156. Morgan Kaufmann.

Freund, Y. and R. Schapire (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences 55*(1), 119–139.

Friedman, J. (1991). Multivariate adaptive regression splines (with discussion). *Annals of Statistics 19*(1), 1–141.

Friedman, J. (2001). Greedy function aproximation: A gradient boosting machine. *Annals of Statistics 29*(5), 1189–1232.

Friedman, J. (2002). Stochastic gradient boosting. *Computational Statistics and Data Analysis 38*(4), 367–378.

Gençay, R. and X. Yang (1996). A forecast comparison of residential housing prices by parametric versus semiparametric conditional mean estimators. *Economics Letters 52*, 129–135.

Griliches, Z. (1971a). Hedonic price indexes for automobiles: An econometric analysis of quality change. In Z. Griliches (Ed.), *Price Indexes and Quality Change: Studies in New Methods of Measurement*, pp. 55–87. Cambridge: Harvard University Press.

Griliches, Z. (1971b). Hedonic price indexes revisited. In Z. Griliches (Ed.), *Price Indexes and Quality Change: Studies in New Methods of Measurement*, pp. 3–15. Cambridge: Harvard University Press.

Harrison, D. and D. Rubinfeld (1978). Hedonic prices and the demand for clean air. *Journal of Environmental Economics & Management 5*, 81–102.

Kershaw, P. and P. Rossini (1999). Using neural networks to estimate constant quality house price indices. In *Proceedings of the Fifth Annual Pacific-Rim Real Estate Society Conference*, Kuala Lumpur, Malaysia.

Lancaster, K. (1966). A new approach to consumer theory. *Journal of Political Economy 74*, 132–157.

Lomsombunchai, V., C. Gan, and M. Lee (2004). House price prediction: Hedonic price models vs. artificial neural nets. *American Journal of Applied Statistics 1*(3), 193–201.

Martins-Filho, C. and O. Bin (2005). Estimation of hedonic price functions via additive nonparametric regression. *Empirical Economics 30*, 93–114.

Pace, R. (1998). Appraisal using generalized additive models. *Journal of Real Estate Research 15*, 77–99.

Pakes, A. (2003). A reconsideration of hedonic price indexes with an application to pc's. *American Economic Review 93*(5), 1578–1596.

Potharst, R. and J. Bioch (2000). Decision trees for ordinal classification. *Intelligent Data Analysis 4*(2), 97–112.

R Development Core Team (2005). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. ISBN 3-900051-07-0, `http://www.R-project.org`.

Rätsch, G., M. Warmuth, S. Mika, T. Onoda, S. Lemm, and K.-R. Muller (2000). Barrier boosting. In *Proceedings COLT*, San Francisco, pp. 170–179. Morgan Kaufmann.

Ripley, D. B. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge.

Rosen, S. (1974). Hedonic prices and implicit markets: Product differentiation in perfect competition. *Journal of Political Economy 82*(1), 34–55.

Therneau, T. M., B. Atkinson, and B. Ripley. (2005). *rpart: Recursive Partitioning*. R package version 3.1-22, S-PLUS 6.x original at `http://www.mayo.edu/hsr/Sfunc.html`.

van Wezel, M., M. Kagie, and R. Potharst (2005). Boosting the accuracy of hedonic pricing models. Technical Report Econometric Institute Report EI 2005-50, Econometric Institute, Erasmus School of Economics and Business Economics, Erasmus University Rotterdam.

Wallace, N. (1996). Hedonic based price indexes for housing: Theory, estimation and index construction. *Economic Review - Federal Reserve Bank of San Francisco 3*, 34–48.

Zemel, R. and T. Pitassi (2001). A gradient-based boosting algorithm for regression problems. In T. Leen, T. Dietterich, and V. Tresp (Eds.), *Advances in Neural Information Processing Systems 13*, Cambridge, pp. 696–702. MIT Press.

A Boosting Algorithms

A.1 GradientBoost

Parametric statistical models $F(\mathbf{x}|\theta)$ for finding some relationship between a target y and inputs $\mathbf{x}$ are fit by minimizing an objective function. This objective function is a function of the parameters θ:

$$F_{\text{best}} = F(\mathbf{x}|\theta_{\text{best}})$$
$$\theta_{\text{best}}(\mathbf{x}) = \arg\min_{\theta} E_{\mathbf{x},y}[L(y, F(\mathbf{x}|\theta))]$$

In theory the loss function L can be every differentiable function of the parameters, but it is often a negative log likelihood function. In many cases the optimal parameter vector is found in a number of steps. Also the GradientBoost (Friedman 2001) procedure uses a number of steps to find an approximation of the target function, but instead of optimizing model parameters, boosting adds a new base model to the model each iteration. The algorithm starts with an initial guess F_0 and then in M steps M base models B_m are added to the function:

$$F_{\text{best}} = F_0 + \sum_{m=1}^{M} \rho_m B_m(\mathbf{x})$$

Of course, we want to find a function F that minimizes the loss function of our choice:

$$F_{\text{best}} = \arg\min_{F} \sum_{i=1}^{N} L(y_i, F(\mathbf{x}_i))$$

GradientBoost finds this minimum by taking M steps along the negative gradient direction of the objective function $\sum_{i=1}^{N} L(y_i, F(\mathbf{x}_i))$. When a large enough number of steps would be taken by the algorithm, it will certainly reach a local minimum. The negative gradient at instance i at the m^{th} step is given by:

$$-g_m(\mathbf{x}_i) = -\frac{\partial L(y_i, F_{m-1}(\mathbf{x}_i))}{\partial F_{m-1}(\mathbf{x}_i)}|\mathbf{x}_i$$

To reach the minimum we would like to take steps $-\rho_m g_m$ along the negative gradient direction, where ρ_m is a step size parameter. However, when a finite dataset is used the negative gradient can only be estimated at the datapoints. GradientBoost uses a certain model type, the base learner, to approximate this step as good as possible. These base learners B_m are all of the same type (e.g. neural nets or like in this case regression trees) and have parameters $\mathbf{a}$ (e.g. weights or split points). To find the best step size these parameters have to be optimized (i.e. the model has to be trained). A least squares method is used to find the optimal parameters of the model:

$$\mathbf{a}_m = \arg\min_{\mathbf{a}} \sum_{i=1}^{N} \{-g_m(\mathbf{x}_i) - B(\mathbf{x}_i|\mathbf{a})\}^2$$

After the training of the model the step size parameter ρ_m has to be estimated:

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho B_m(\mathbf{x}_i))$$

Finally this base learner is added to the aggregated function:

$$F_m(\mathbf{x}) = F_{m-1} + \rho_m B_m(\mathbf{x})$$

The different instantiations of the GradientBoost procedure, summarized in Figure 11, differ from each other by the use of different loss functions. First the LSBoost algorithm, based on squared loss, will be discussed in the next subsection, then the LADBoost algorithm (absolute loss) is described in subsection A.3.

Input: dataset with instances $\{\mathbf{x}_i; y_i\}_1^N$
number of iterations M
Output: Model $F(\mathbf{x})$
$F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^N L(y_i, \rho)$ **for** $m = 1$ *to* M **do**
  $\left\{\tilde{y}_i = -\frac{\partial L(y_i, F_{m-1}(\mathbf{x}_i))}{\partial F_{m-1}(\mathbf{x}_i)}\right\}_1^N$
  train $B_m(\mathbf{x})$ using $\{\mathbf{x}_i; \tilde{y}_i\}_1^N$
  $\rho_m = \arg\min_\rho \sum_{i=1}^N L(y_i, F_{m-1}(\mathbf{x}_i) + \rho B_m(\mathbf{x}_i))$
  $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m B_m(\mathbf{x})$
end

Figure 11: GradientBoost algorithm

A.2 LSBoost

The LSBoost (Friedman 2001) algorithm is an instantiation of the GradientBoost algorithm where the loss function is the squared loss function:

$$L(y, F) = (y - F)^2/2$$

When we follow GradientBoost first an initial guess F_0 has to be computed. The initial guess is a constant that minimizes the total squared loss over the instances:

$$F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^N L(y_i, \rho) = \arg\min_\rho \sum_{i=1}^N (y_i - \rho)^2/2 = \bar{y}$$

Each iteration GradientBoost replaces the targets with the negative gradient. In the case of LSBoost one gets:

$$\tilde{y}_i = -g_m(\mathbf{x}) = -\frac{\partial L(y_i, F_{m-1}(\mathbf{x}_i))}{\partial F_{m-1}(\mathbf{x}_i)} = -\frac{\partial((y_i - F_{m-1}(\mathbf{x}_i))^2/2)}{\partial F_{m-1}(\mathbf{x}_i)} = y_i - F_{m-1}(\mathbf{x}_i)$$

This means that the target is replaced by the difference of the real value of y and the y value predicted by the additive model up to now. Now a new base learner $B_m(\mathbf{x})$ will be trained on the attributes and the replaced targets $\{\mathbf{x}_i; \tilde{y}_i\}_1^N$. After the training the ρ parameter needs to be computed:

$$\rho_m = \arg\min_\rho \sum_{i=1}^N (\tilde{y}_i - \rho B_m(\mathbf{x}_i))^2$$

In practice ρ_m will almost always equal 1 when using CART as base learner. At this point a new parameter will be introduced: the learningrate ν. In practice it is not always preferable to reach the optimum as quickly as possible, because data will then be predicted too closely and overfitting will occur. Therefore the learningrate is introduced, which slows down the learning process by decreasing the influence of a single base model on the total function. The update of $F(\mathbf{x})$ now becomes:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu\rho_m B_m(\mathbf{x})$$

When a small learningrate is used more iterations are necessary to train the algorithm, so there exists a trade off between speed and precision. However, the marginal effect of lowering the learningrate is decreasing, which means that after some point lowering the learningrate is hardly effective. The LSBoost algorithm is summarized in Figure 12.

A.3 LADBoost

LADBoost (Friedman 2001) like LSBoost is an instantiation of the GradientBoost algorithm. LADBoost uses absolute loss as its loss function:

$$L(y, F) = |y - F|$$

Input: dataset with instances $\{\mathbf{x}_i; y_i\}_1^N$
number of iterations M
learning rate ν
Output: Model $F(\mathbf{x})$
$F_0(\mathbf{x}) = \bar{y}$
for $m = 1$ *to* M **do**
  $\{\tilde{y}_i = y_i - F_{m-1}(\mathbf{x}_i)\}_1^N$
  train $B_m(\mathbf{x})$ using $\{\mathbf{x}_i; \tilde{y}_i\}_1^N$
  $\rho_m = \arg\min_\rho \sum_{i=1}^N [\tilde{y}_i - \rho B_m(\mathbf{x}_i)]^2$
  $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu\rho_m B_m(\mathbf{x})$
end

Figure 12: LSBoost algorithm

The new initialization becomes:

$$F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^N |y_i - \rho| = median\{y_i\}_1^N$$

Again the new targets are the negative gradient of the loss function with respect to the current model F in the datapoints:

$$\tilde{y}_i = -\frac{\partial |y_i - F_{m-1}(\mathbf{x}_i))|}{\partial F_{m-1}(\mathbf{x}_i)} = sign(y_i - F_{m-1}(\mathbf{x}_i))$$

In the case that the base learner $B(\mathbf{x})$ is a regression tree $T(\mathbf{x})$ a line search is not necessary to determine ρ_m. Friedman (2001) shows, when having a regression tree $T(\mathbf{x})$ with terminal nodes $\{R_l)\}_1^L$, ρ_m can be replaced by a γ_{lm} for each terminal node. The γ_{lm} 's are the medians of the residuals in the specific node:

$$\gamma_{lm} = median_{\mathbf{x_i}\epsilon R_{lm}}\{y_i - F_{m-1}(\mathbf{x}_i)\},$$

Finally $F(\mathbf{x})$ is updated:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu\gamma_{lm} 1(\mathbf{x}\epsilon R_{lm})$$

Here function $1(\mathbf{x}\epsilon R_{lm})$ has a value of 1 when the datapoint belongs to terminal node R_{lm} and 0 otherwise. An overview of LADBoost for regression trees, LADTreeBoost, is given in Figure 13.

Input: dataset with instances $\{\mathbf{x}_i; y_i\}_1^N$
number of iterations M
learning rate ν
Output: Model $F(\mathbf{x})$
$F_0(\mathbf{x}) = median\{y_i\}_1^N$
for $m = 1$ *to* M **do**
  $\{\tilde{y}_i = sign(y_i - F_{m-1}(\mathbf{x}_i))\}_1^N$
  train tree $T_m(\mathbf{x})$ having terminal nodes $\{R_{lm})\}_1^L$ using $\{\mathbf{x}_i; \tilde{y}_i\}_1^N$
  $\gamma_{lm} = median_{\mathbf{x_i}\epsilon R_{lm}}\{y_i - F_{m-1}(\mathbf{x}_i)\}$
  $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu\gamma_{lm} 1(\mathbf{x}\epsilon R_{lm})$
end

Figure 13: LADTreeBoost algorithm

B Methods for Interpretation

In this appendix we describe two graphical methods to interpret LSBoost models: Relative importance plots and partial dependence plots. Both methods are introduced by Friedman (2001).

B.1 Relative Importance Plots

A relative importance plot shows, graphically, how much influence the different variables have on the output of the model. Breiman et al. (1983) developed the following formula to measure importance of a variable in a single CART model:

$$\hat{I}_j^2(T) = \sum_{t=1}^{J-1} \hat{i}_t^2 1(v_t = x_j)$$

Here the summation is over the non-terminal nodes in the J-terminal-node tree T and $1()$ denotes the indicator function. v_t is the split variable of node t. $\hat{i}_t^2$ measures the improvement in squared error as a result of the split in t, and can be computed with the equation:

$$\hat{i}^2 = \frac{w_l w_r}{w_l + w_r}(\bar{y}_l - \bar{y}_r)^2$$

Here w_l and w_r are the probabilities an instance turns to the left or right child node, $\bar{y}_l$ and $\bar{y}_r$ are the mean target values for both children. Both the probabilities and the means are computed on the training set and saved in the CART model.

$\hat{I}_j^2$ is used to measure importance in the case of regression trees and not its square root $\hat{I}_j$. To compute the $\hat{I}_j^2$'s of a boosting model it is sufficient to average the $\hat{I}_j^2$'s of the base learners:

$$\hat{I}_j^2 = \frac{1}{M} \sum_{m=1}^{M} \hat{I}_j^2(T_m)$$

A variable gets a high importance I^2 when it is used in many splits, but more importantly when it is used in splits that divide the data in two almost equally large parts with a high difference in the mean value of the target value.

Finally the variable with the highest importance gets an index of $RI = 100$ and the other indices are adjusted to this:

$$RI_j = \frac{\hat{I}_j^2}{\hat{I}_{max}^2} \cdot 100$$

B.2 Partial Dependence Plots

Relative importance plots show which variables have the highest influence on $F(\mathbf{x})$, but don't show anything about the sign or the shape of the influence of variable x_j on $F(\mathbf{x})$. Partial dependence plots do show this.

For a single regression tree T, input variable x_j and a specified set of values $\{x_{ij}\}_{i=1}^N$ for variable x_j, a partial dependence plot shows the average output of T for all values of x_j in the set. For one point x_{ij} this is denoted $\bar{T}(x_{ij})$. The average output is computed with respect to the training data. The average output of T is computed over all points in the training set, keeping x_j fixed to one value.

In the case of a tree, this average output can be computed by a a weighted pass-through of T. At the root of the tree a weight $w_{ij} = 1$ is initialized. For each non-terminal node with x_j as split variable that will be visited, the right or left child node will be visited depending on the value of x_{ij}. When a non-terminal node will be visited with another split variable than x_j, both its children will be visited and the weight multiplied with the probability to turn right or left:

$$w_l = wp_l; w_r = w(1 - p_l)$$

Each visited terminal node will be assigned the current value of w_{ij}. The value of $\bar{T}(x_{ij})$ is the weighted sum of the terminal nodes:

$$\bar{T}(x_{ij}) = \sum_{k=1}^{K} (w_k y_k)$$

Here the summation is over the number of visited terminal nodes and w_k and y_k are weight and the mean target value of terminal node k.

In the case of an LSBoost model the results of the base learners $T_m(\mathbf{x})$ are summed in the same way as in the normal LSBoost algorithm (see Figure 12):

$$\bar{F}(x_{ij}) = F_0 + \sum_{m=1}^{M} (\nu \rho_m \bar{T}_m(x_{ij}))$$

Because partial dependence plots display an average relationship, one must be cautious when interpreting these plots. Partial dependence plots ignore correlations that exist between the 'fixed' variable and the other variables. This may lead to a deceptive result where the true influence of the 'fixed' variable is not correctly depicted. Nevertheless, they sometimes reveal interesting patterns.

C Tables with Data Statistics

Categorical	
Name	Levels
Kind House	simple house (122) middle class house (1201) mansion (271) villa (45) country-house (35) bungalow (32) patio-bungalow (7) semi bungalow (10) downstairs house (318) upstairs house (387) up- + downstairs house (10) staircase-access flat (144) canal house (7) maisonette (73) service flat (25) flat with elevator (106) flat without elevator (390) house with office (18) drive-in house (24) farm house (33) apartment (96) holiday home (33)
Type House	apartment (1549) end-terraced house (283) mid-terraced house (901) detached house (302) semi detached house(312) stepped house (40)
Garden Size	no garden (1221) 0 to 5 meters (161) 5 to 10 m (657) 10 to 15 m (746) 15 to 20 m (205) 20 to 50 m (124) 50+ m (13) missing (260)
Situation Garden N-S	no garden (1221) east or west (513) north (481) south (760) missing (412)
Situation Garden E-W	no garden (1221) north or south (584) east (478) west (692) missing (412)
Garage	no garage (2672) attached brick (250) detached brick (185) attached wood (12) detached wood (53) built-in (215)
Living Room	L-room (602) T-room (15) Z-room (57) through living room (344) room and suite (236) missing (2133)
Garden Maintenance	no garden (1221) to be laid out (53) normal (1815) pretty (291) neglected (7)
Shed	no shed (775) attached brick (284) detached brick (757) attached wood (121) detached wood (618) built-in (832)
Heating	central(gas) (2528) central(oil) (21) city heating (234) room heater with back boiler (131) warm-air (48) gas heaters (376) missing (49)
Situation	open (50) normal (3016) covered (321)
District	groningen herewegwijk en helpman (557) groningen noorddijk (476) groningen oranjewijk (305) groningen schilders- en zeeheldenwijk (299) haren centrum (293) other(12 districts) (1450) missing (7)

Numerical	
Name	Min. 1st Qu. Median Mean 3rd Qu. Max. (Missing)
Transaction Price (€)	13000 115000 137500 172800 190300 1450000
Lot Size (m^2)	12 80 110 242.8 177 14950 (70)
Construction Year	1650 1935 1965 1958 1980 2004
Rooms Basement	0 0 0 0.01063 0 6
Rooms Ground-Floor	0 1 1 1.832 3 10
Rooms 1^{st} Floor	0 0 2 1.793 3 6
Rooms 2^{nd} Floor	0 0 0 0.3862 1 5
Rooms 3^{rd} Floor	0 0 0 0.01151 0 4
Total Rooms	1 3 4 4.044 5 22 (8)
Volume (m^3)	76 225 300 327 375 3500 (4)
House Size (m^2)	22 80 100 109.1 125 640(4)
Subsidized House	0 0 0 0.0008857 0 1
Listed Building	0 0 0 0.0062 0 1
Upholstered	0 0 0 0.00561 0 1
Partially Upholstered	0 0 0 0.03956 0 1
Furnished	0 0 0 0.003543 0 1
Maintenance Outside	2 8 8 8.026 8 10
Maintenance Inside	2 8 8 7.997 8 10
Garden: Patio	0 0 0 0.002952 0 1
Garden: Place	0 0 0 0.04163 0 1
Garden: Terrace	0 0 0 0.08001 0 1
Garden: Lawn	0 0 0 0.07411 0 1
Carport	0 0 0 0.02834 0 1
Possibility for garage	0 0 0 0.01535 0 1
Garage: 2+ cars	0 0 0 0.02126 0 1
Garage: Central Heated	0 0 0 0.01329 0 1
Parking Space	0 0 0 0.05403 0 1
Rear Access	0 0 0 0.3 1 1
Floor Heating	0 0 0 0.01358 0 1
Fireplace	0 0 0 0.04547 0 1
Isolation: Roof	0 0 0 0.2899 1 1
Isolation: Wall	0 0 0 0.2203 0 1
Isolation: Floor	0 0 0 0.1751 0 1
Isolation: Double Glazing	0 0 0 0.2359 0 1
Isolation: Double Glazing Part.	0 0 0 0.3646 1 1
Isolation: Double Windows	0 0 0 0.0248 0 1

Table 9: City of Groningen and environs: Statistics of variables

Categorical	
Name	Levels
Kind House	simple house (63) middle class house (1219) mansion (195) villa (153) country-house (15) country-estate (1) bungalow (50) patio-bungalow (2) semi bungalow (37) split-level (6) downstairs house (18) upstairs house (19) up- + downstairs house (4) staircase-access flat (35) maisonette (41) service flat (42) flat with elevator (137) flat without elevator (170) house with office (27) drive-in house (22) farm house (58) apartment (76) holiday home (18)
Type House	apartment (536) end-terraced house (272) mid-terraced house (584) detached house (615) semi detached house(345) stepped house (50)
Garden Size	no garden (484) 0 to 5 meters (44) 5 to 10 m (297) 10 to 15 m (767) 15 to 20 m (222) 20 to 50 m (172) 50+ m (18) missing (398)
Situation Garden N-S	no garden (484) east or west (381) north (314) south (664) missing (559)
Situation Garden E-W	no garden (484) north or south (412) east (428) west (519) missing (559)
Garage	no garage (1512) attached brick (234) detached brick (405) attached wood (9) detached wood (130) built-in (129)
Living Room	L-room (393) T-room (20) Z-room (132) through living room (115) room and suite (78) missing (1721)
Garden Maintenance	no garden (484) to be laid out (28) normal (1408) pretty (463) neglected (19)
Shed	no shed (637) attached brick (219) detached brick (666) attached wood (49) detached wood (402) built-in (429)
Heating	central(gas) (1908) central(oil) (26) city heating (133) room heater with back boiler (48) warm-air (76) gas heaters (173) missing (38)
Situation	open (70) normal (2106) covered (226)
District	apeldoorn west (355) apeldoorn zuidoost (346) apeldoorn zuid (285) apeldoorn oost (258) apeldoorn noord (221) other(15 districts) (935) missing (2)
Numerical	
Name	Min. 1st Qu. Median Mean 3rd Qu. Max. (Missing)
Transaction Price (€)	27500 175000 210000 256700 300000 4311000
Lot Size (m^2)	11 124 184 643.1 335.3 45270 (46)
Construction Year	1830 1955 1972 1966 1984 2004
Rooms Basement	0 0 0 0.02165 0 4
Rooms Ground-Floor	0 1 1 1.758 2 8 (1)
Rooms 1^{st} Floor	0 1 3 2.216 3 10
Rooms 2^{nd} Floor	0 0 0 0.46 1 5
Rooms 3^{rd} Floor	0 0 0 0.004163 0 3
Total Rooms	1 4 4 4.472 5 20 (6)
Volume (m^3)	10 300 350 376.8 400 4500 (11)
House Size (m^2)	10 100 120 127.2 140 1500 (11)
Subsidized House	0 0 0 0.0008326 0 1
Listed Building	0 0 0 0.002914 0 1
Upholstered	0 0 0 0.006661 0 1
Partially Upholstered	0 0 0 0.006661 0 1
Furnished	0 0 0 0.001249 0 1
Maintenance Outside	2 8 8 8.204 8 10
Maintenance Inside	2 8 8 8.153 8 10
Garden: Patio	0 0 0 0.004163 0 1
Garden: Place	0 0 0 0.00458 0 1
Garden: Terrace	0 0 0 0.1478 0 1
Garden: Lawn	0 0 0 0.1561 0 1
Carport	0 0 0 0.1224 0 1
Possibility for garage	0 0 0 0.03081 0 1
Garage: 2+ cars	0 0 0 0.05579 0 1
Garage: Central Heated	0 0 0 0.04455 0 1
Parking Space	0 0 0 0.132 0 1
Rear Access	0 0 0 0.39 1 1
Floor Heating	0 0 0 0.03164 0 1
Fireplace	0 0 0 0.1116 0 1
Isolation: Roof	0 0 0 0.4933 1 1
Isolation: Wall	0 0 0 0.4184 1 1
Isolation: Floor	0 0 0 0.2889 1 1
Isolation: Double Glazing	0 0 0 0.2627 1 1
Isolation: Double Glazing Part.	0 0 0 0.44 1 1
Isolation: Double Windows	0 0 0 0.01582 0 1

Table 10: Apeldoorn and environs: Statistics of variables

Categorical	
Name	Levels
Kind House	simple house (61) middle class house (2037) mansion (420) villa (100) country-house (34) country-estate (1) bungalow (62) patio-bungalow (18) semi bungalow (37) split-level (9) downstairs house (41) upstairs house (64) up- + downstairs house (2) staircase-access flat (33) maisonette (66) service flat (1) flat with elevator (254) flat without elevator (327) house with office (34) drive-in house (16) farm house (8) apartment (115)
Type House	apartment (903) end-terraced house (514) mid-terraced house (1296) detached house (370) semi detached house(532) stepped house (115)
Garden Size	no garden (831) 0 to 5 meters (51) 5 to 10 m (533) 10 to 15 m (1358) 15 to 20 m (519) 20 to 50 m (278) 50+ m (22) missing (138)
Situation Garden N-S	no garden (831) east or west (655) north (628) south (940) missing (138)
Situation Garden E-W	no garden (831) north or south (619) east (770) west (834) missing (138)
Garage	no garage (2465) attached brick (411) detached brick (447) attached wood (4) detached wood (7) built-in (396)
Living Room	L-room (789) T-room (44) Z-room (180) through living room (416) room and suite (83) missing (2218)
Garden Maintenance	no garden (831) to be laid out (17) normal (2079) pretty (784) neglected (19)
Shed	no shed (881) attached brick (283) detached brick (1393) attached wood (41) detached wood (274) built-in (858)
Heating	central(gas) (3032) central(oil) (121) city heating (331) room heater with back boiler (60) warm-air (26) gas heaters (89) missing (71)
Situation	open (24) normal (3563) covered (143)
District	eindhoven woensel-noord (660) eindhoven stratum (445) eindhoven woensel-zuid (374) best (290) eindhoven strijp (246) other(16 districts) (1711) missing (4)
Numerical	
Name	Min. 1st Qu. Median Mean 3rd Qu. Max. (Missing)
Transaction Price (€)	79500 165000 198000 237700 269000 1690000
Lot Size (m^2)	12 115 166 265.1 249 46370 (14)
Construction Year	1694 1960 1974 1971 1987 2005
Rooms Basement	0 0 0 0.01287 0 3
Rooms Ground-Floor	0 1 1 1.495 2 8
Rooms 1^{st} Floor	0 2 3 2.335 3 6
Rooms 2^{nd} Floor	0 0 0 0.6062 1 6
Rooms 3^{rd} Floor	0 0 0 0.01206 0 3
Total Rooms	1 4 5 4.469 5 13 (5)
Volume (m^3)	84 300 365 389 450 3115 (24)
House Size (m^2)	11 100 125 130 150 854 (5)
Subsidized House	0 0 0 0.0005362 0 1
Listed Building	0 0 0 0.004558 0 1
Upholstered	0 0 0 0.001609 0 1
Partially Upholstered	0 0 0 0.01072 0 1
Furnished	0 0 0 0.0002681 0 1
Maintenance Outside	2 8 8 8.099 8 10
Maintenance Inside	2 8 8 8.116 8 10
Garden: Patio	0 0 0 0.01394 0 1
Garden: Place	0 0 0 0.005898 0 1
Garden: Terrace	0 0 0 0.1319 0 1
Garden: Lawn	0 0 0 0.1332 0 1
Carport	0 0 0 0.04692 0 1
Possibility for garage	0 0 0 0.03619 0 1
Garage: 2+ cars	0 0 0 0.03405 0 1
Garage: Central Heated	0 0 0 0.05684 0 1
Parking Space	0 0 0 0.06005 0 1
Rear Access	0 0 1 0.52 1 1
Floor Heating	0 0 0 0.06595 0 1
Fireplace	0 0 0 0.119 0 1
Isolation: Roof	0 0 0 0.4724 1 1
Isolation: Wall	0 0 0 0.3547 1 1
Isolation: Floor	0 0 0 0.296 1 1
Isolation: Double Glazing	0 0 0 0.2662 1 1
Isolation: Double Glazing Part.	0 0 0 0.3129 1 1
Isolation: Double Windows	0 0 0 0.005094 0 1

Table 11: Eindhoven and environs: Statistics of variables

Categorical	
Name	Levels
Kind House	simple house (145) middle class house (1040) mansion (325) villa (61) country-house (3) bungalow (6) patio-bungalow (3) semi bungalow (1) split-level (20) downstairs house (763) upstairs house (3557) up- + downstairs house (61) staircase-access flat (86) canal house (62) maisonette (377) service flat (20) flat with elevator (609) flat without elevator (346) house with office (32) drive-in house (42) farm house (7) apartment (923) holiday home (1)
Type House	apartment (6742) end-terraced house (335) mid-terraced house (1210) detached house (80) semi detached house(104) stepped house (19)
Garden Size	no garden (5408) 0 to 5 meters (602) 5 to 10 m (1196) 10 to 15 m (924) 15 to 20 m (134) 20 to 50 m (90) 50+ m (16) missing (120)
Situation Garden N-S	no garden (5408) east or west (761) north (461) south (1413) missing (447)
Situation Garden E-W	no garden (5408) north or south (973) east (691) west (971) missing (447)
Garage	no garage (8018) attached brick (78) detached brick (58) attached wood (6) detached wood (13) built-in (317)
Living Room	L-room (560) T-room (12) Z-room (24) through living room (568) room and suite (537) missing (6789)
Garden Maintenance	no garden (5408) to be laid out (146) normal (2524) pretty (394) neglected (16) missing (2)
Shed	no shed (4262) attached brick (210) detached brick (578) attached wood (80) detached wood (581) built-in (2779)
Heating	central(gas) (5464) central(oil) (316) city heating (1473) room heater with back boiler (208) warm-air (23) gas heaters (569) missing (437)
Situation	open (126) normal (8057) covered (307)
District	amsterdam oud-zuid (1187) amsterdam binnenstad (1109) amstelveen (1029) amsterdam zuideramstel (623) amsterdam oost/watergraafsmeer (507) other(14 districts) (4015) missing (20)

Numerical	
Name	Min. 1st Qu. Median Mean 3rd Qu. Max. (Missing)
Transaction Price (€)	22500 160000 210000 257600 292000 4250000
Lot Size (m^2)	10 65 85 108.7 115 30050 (155)
Construction Year	1584 1920 1955 1946 1985 2004
Rooms Basement	0 0 0 0.02886 0 4
Rooms Ground-Floor	0 1 2 2.19 3 10
Rooms 1^{st} Floor	0 0 0 0.867 2 7 (1)
Rooms 2^{nd} Floor	0 0 0 0.257 0 5
Rooms 3^{rd} Floor	0 0 0 0.04523 0 4
Total Rooms	1 3 3 3.469 4 22 (164)
Volume (m^3)	12 175 230 269.4 315 4240 (181)
House Size (m^2)	18 65 85 97.59 115 1400 (6)
Subsidized House	0 0 0 0.0007067 0 1
Listed Building	0 0 0 0.02521 0 1
Upholstered	0 0 0 0.01625 0 1
Partially Upholstered	0 0 0 0.1353 0 1
Furnished	0 0 0 0.001413 0 1
Maintenance Outside	2 8 8 8.355 8 10
Maintenance Inside	2 8 8 8.274 9 10
Garden: Patio	0 0 0 0.01201 0 1
Garden: Place	0 0 0 0.008716 0 1
Garden: Terrace	0 0 0 0.07986 0 1
Garden: Lawn	0 0 0 0.05536 0 1
Carport	0 0 0 0.008245 0 1
Possibility for garage	0 0 0 0.02108 0 1
Garage: 2+ cars	0 0 0 0.006596 0 1
Garage: Central Heated	0 0 0 0.005536 0 1
Parking Space	0 0 0 0.04523 0 1
Rear Access	0 0 0 0.08 0 1
Floor Heating	0 0 0 0.01048 0 1
Fireplace	0 0 0 0.01331 0 1
Isolation: Roof	0 0 0 0.2465 0 1
Isolation: Wall	0 0 0 0.2212 0 1
Isolation: Floor	0 0 0 0.236 0 1
Isolation: Double Glazing	0 0 0 0.2845 1 1
Isolation: Double Glazing Part.	0 0 0 0.1254 0 1
Isolation: Double Windows	0 0 0 0.006125 0 1

Table 12: Amsterdam: Statistics of variables

Categorical	
Name	Levels
Kind House	simple house (86) middle class house (2006) mansion (518) villa (113) country-house (8) bungalow (30) patio-bungalow (11) semi bungalow (7) split-level (19) downstairs house (527) upstairs house (1206) up- + downstairs house (9) staircase-access flat (148) canal house (9) maisonette (461) service flat (2) flat with elevator (734) flat without elevator (1227) house with office (23) drive-in house (45) farm house (10) apartment (308)
Type House	apartment (4625) end-terraced house (635) mid-terraced house (1753) detached house (173) semi detached house(263) stepped house (61)
Garden Size	no garden (3930) 0 to 5 meters (247) 5 to 10 m (1089) 10 to 15 m (1542) 15 to 20 m (315) 20 to 50 m (165) 50+ m (11) missing (211)
Situation Garden N-S	no garden (3930) east or west (912) north (540) south (1407) missing (721)
Situation Garden E-W	no garden (3930) north or south (865) east (936) west (1058) missing (721)
Garage	no garage (6788) attached brick (224) detached brick (127) attached wood (9) detached wood (22) built-in (340)
Living Room	L-room (587) T-room (14) Z-room (66) through living room (480) room and suite (408) missing (5955)
Garden Maintenance	no garden (3930) to be laid out (81) normal (3012) pretty (447) neglected (36) missing (4)
Shed	no shed (2656) attached brick (370) detached brick (794) attached wood (140) detached wood (917) built-in (2633)
Heating	central(gas) (4342) central(oil) (89) city heating (1748) room heater with back boiler (261) warm-air (48) gas heaters (535) missing (487)
Situation	open (127) normal (7086) covered (297)
District	rotterdam noord (735) rotterdam prins alexander (731) rotterdam hilligersberg-schiebroek (586) rotterdam kralingen-krooswijk (518) rotterdam delfshaven (436) other(38 districts) (4354) missing (11)

Numerical	
Name	Min. 1st Qu. Median Mean 3rd Qu. Max. (Missing)
Transaction Price (€)	11500 120000 160000 192400 220000 2045000
Lot Size (m^2)	15 75 100 137.9 140 9719 (98)
Construction Year	1580 1936 1964 1960 1984 2005 (1)
Rooms Basement	0 0 0 0.02863 0 5
Rooms Ground-Floor	0 1 2 1.903 3 7 (1)
Rooms 1^{st} Floor	0 0 2 1.489 3 30
Rooms 2^{nd} Floor	0 0 0 0.4638 1 6
Rooms 3^{rd} Floor	0 0 0 0.0249 0 5
Total Rooms	1 3 4 3.916 5 16 (16)
Volume (m^3)	75 210 270 305.5 360 2500 (16)
House Size (m^2)	25 76 100 109.6 130 650 (10)
Subsidized House	0 0 0 0.0001332 0 1
Listed Building	0 0 0 0.004927 0 1
Upholstered	0 0 0 0.002796 0 1
Partially Upholstered	0 0 0 0.01771 0 1
Furnished	0 0 0 0.0009321 0 1
Maintenance Outside	2 8 8 7.96 8 10
Maintenance Inside	2 8 8 7.876 8 10
Garden: Patio	0 0 0 0.006658 0 1
Garden: Place	0 0 0 0.01025 0 1
Garden: Terrace	0 0 0 0.03222 0 1
Garden: Lawn	0 0 0 0.06232 0 1
Carport	0 0 0 0.01238 0 1
Possibility for garage	0 0 0 0.01691 0 1
Garage: 2+ cars	0 0 0 0.009055 0 1
Garage: Central Heated	0 0 0 0.01265 0 1
Parking Space	0 0 0 0.06099 0 1
Rear Access	0 0 0 0.18 0 1
Floor Heating	0 0 0 0.02503 0 1
Fireplace	0 0 0 0.04008 0 1
Isolation: Roof	0 0 0 0.2097 0 1
Isolation: Wall	0 0 0 0.1903 0 1
Isolation: Floor	0 0 0 0.1758 0 1
Isolation: Double Glazing	0 0 0 0.2574 1 1
Isolation: Double Glazing Part.	0 0 0 0.1905 0 1
Isolation: Double Windows	0 0 0 0.003728 0 1

Table 13: Rotterdam: Statistics of variables

Categorical	
Name	Levels
Kind House	simple house (130) middle class house (1379) mansion (101) villa (46) country-house (25) country-estate (1) bungalow (76) patio-bungalow (2) semi bungalow (62) split-level (8) downstairs house (17) upstairs house (60) up- + downstairs house (7) staircase-access flat (5) canal house (5) maisonette (20) service flat (1) flat with elevator (27) flat without elevator (58) house with office (17) drive-in house (12) farm house (41) apartment (37) holiday home (79)
Type House	apartment (232) end-terraced house (356) mid-terraced house (714) detached house (562) semi detached house(318) stepped house (34)
Garden Size	no garden (249) 0 to 5 meters (64) 5 to 10 m (387) 10 to 15 m (635) 15 to 20 m (214) 20 to 50 m (234) 50+ m (29) missing (404)
Situation Garden N-S	no garden (249) east or west (376) north (319) south (577) missing (695)
Situation Garden E-W	no garden (249) north or south (362) east (443) west (467) missing (695)
Garage	no garage (1362) attached brick (386) detached brick (275) attached wood (11) detached wood (35) built-in (147)
Living Room	L-room (341) T-room (6) Z-room (43) through living room (434) room and suite (94) missing (1298)
Garden Maintenance	no garden (249) to be laid out (47) normal (1699) pretty (201) neglected (20)
Shed	no shed (706) attached brick (315) detached brick (569) attached wood (70) detached wood (288) built-in (268)
Heating	central(gas) (1835) central(oil) (21) city heating (52) room heater with back boiler (15) warm-air (18) gas heaters (205) missing (70)
Situation	open (78) normal (2025) covered (113)
District	goes (177) vlissingen oost-souburg (118) middelburg centrum (109) middelburg zuidoost (96) middelburg noord (95) other(126 districts) (1596) missing (25)
Numerical	
Name	Min. 1st Qu. Median Mean 3rd Qu. Max. (Missing)
Transaction Price (€)	31000 125000 162000 195200 230400 1400000
Lot Size (m^2)	11 130 191 454.4 344.5 22420 (33)
Construction Year	1545 1936 1970 1956 1983 2004 (1)
Rooms Basement	0 0 0 0.01489 0 3
Rooms Ground-Floor	0 1 1 1.507 2 7
Rooms 1^{st} Floor	0 2 3 2.288 3 9
Rooms 2^{nd} Floor	0 0 0 0.4152 1 6
Rooms 3^{rd} Floor	0 0 0 0.0194 0 4
Total Rooms	1 4 4 4.536 5 15 (140)
Volume (m^3)	66 280 340 377.1 410 3600 (2)
House Size (m^2)	25 90 110 122.5 140 800 (3)
Listed Building	0 0 0 0.01309 0 1
Upholstered	0 0 0 0.01805 0 1
Partially Upholstered	0 0 0 0.01083 0 1
Furnished	0 0 0 0.03565 0 1
Maintenance Outside	2 8 8 7.787 8 10
Maintenance Inside	2 8 8 7.807 8 10
Garden: Patio	0 0 0 0.003159 0 1
Garden: Place	0 0 0 0.02347 0 1
Garden: Terrace	0 0 0 0.03294 0 1
Garden: Lawn	0 0 0 0.09657 0 1
Carport	0 0 0 0.02211 0 1
Possibility for garage	0 0 0 0.02527 0 1
Garage: 2+ cars	0 0 0 0.037 0 1
Garage: Central Heated	0 0 0 0.02527 0 1
Parking Space	0 0 0 0.03655 0 1
Rear Access	0 0 0 0.41 1 1
Floor Heating	0 0 0 0.02392 0 1
Fireplace	0 0 0 0.07356 0 1
Isolation: Roof	0 0 0 0.4215 1 1
Isolation: Wall	0 0 0 0.3308 1 1
Isolation: Floor	0 0 0 0.2446 0 1
Isolation: Double Glazing	0 0 0 0.2541 1 1
Isolation: Double Glazing Part.	0 0 0 0.3384 1 1
Isolation: Double Windows	0 0 0 0.009025 0 1

Table 14: Zeeland: Statistics of variables

Name	Groningen	Apeldoorn	Eindhoven	Amsterdam	Rotterdam	Zeeland
			Basement			
Kitchen	0.0002952	0.0004163	0	0.0004711	0.001731	0.0004513
Bathing Facilities	0.001476	0.002914	0.0005362	0.0053	0.004527	0.002256
Understairs Pantry	0.2247	0.2856	0.3732	0.07479	0.1092	0.1417
Cellar Pantry	0.007086	0.01166	0.009115	0.01567	0.01838	0.004513
Basement Boiler	0.007676	0.007078	0.00563	0.007774	0.01691	0.00361
Garage	0	0.002082	0.001609	0.0007067	0.0002663	0.0009025
			Ground-Floor			
Study	0.01476	0.06661	0.03056	0.009894	0.006525	0.009928
Garden Room	0.02126	0.03705	0.02574	0.02521	0.01931	0.02347
Kitchen	0.4618	0.4092	0.2729	0.3514	0.3742	0.3159
Scullery	0.1963	0.2423	0.2147	0.03757	0.05393	0.282
Living-Room-Cum-Kitchen	0.07942	0.1694	0.1086	0.08987	0.06312	0.1187
Open-Plan Kitchen	0.3888	0.3855	0.4397	0.3847	0.381	0.3308
Sun-Room	0.0109	0.04704	0.01716	0.004594	0.005859	0.0167
Bathroom with Bath	0.1119	0.1415	0.1453	0.2744	0.1764	0.1187
Bathroom with Shower	0.279	0.2356	0.1252	0.3585	0.2663	0.1697
Shower	0.07263	0.03997	0.0185	0.05041	0.04341	0.03971
			First Floor			
Balcony	0.1069	0.122	0.1643	0.0978	0.178	0.04919
Kitchen	0.052	0.02498	0.04879	0.08657	0.08921	0.02978
Store-room	0.1414	0.1936	0.1493	0.07786	0.10932	0.08303
Bathroom with Bath	0.2578	0.3177	0.5166	0.166	0.3103	0.3299
Bathroom with Shower	0.2628	0.2785	0.2284	0.1064	0.1718	0.2748
Shower	0.03484	0.01832	0.01877	0.01932	0.02636	0.02843
			Second Floor			
Balcony	0.007086	0.002914	0.005898	0.01625	0.02357	0.00361
Bathing Facilities	0.04872	0.0295587	0.02949	0.04888	0.06764	0.03159
Attic Storage Space	0.2524	0.4488	0.5311	0.09046	0.1763	0.3005
Loft	0.06466	0.1082	0.1381	0.02049	0.03356	0.1588
Dormer	0.01949	0.06869	0.1657	0.0338	0.05899	0.01083
Fixed Staircase	0.09123	0.1936	0.3335	0.02214	0.06764	0.1079
Room Possibility	0.0508	0.07535	0.1209	0.006832	0.02916	0.04016
			Third Floor			
Balcony	0	0	0	0.003534	0.001065	0.0004513
Bathing Facilities	0.001771	0	0.001072	0.01107	0.004927	0.002256
Ceiling	0.01329	0.006245	0.02332	0.01013	0.01438	0.008574
Attic	0.01476	0.005828	0.02601	0.004829	0.01505	0.006769
Dormer	0.0002952	0.0004163	0.001072	0.001178	0.0009321	0.0004513
Fixed Stairs	0.002362	0.0008326	0.00563	0.0009423	0.002397	0.002708
Room Possibility	0.001181	0.0004163	0.002413	0.0008245	0.001731	0.0009025

Table 15: Means of floor specific variables

Variable	Description
address density	the average number of adresses per km^2 within a circle with a radius of 1 km with the address as the center of the circle.
women	proportion of women in the total population
0-15 years	percentage of 0-15 years old people in the total population
15-25 years	percentage of 15-25 years old people in the total population
25-45 years	percentage of 25-45 years old people in the total population
45-65 years	percentage of 45-65 years old people in the total population
65^+ years	percentage of 65^+ years old people in the total population
population density	average number of inhabitants per km^2
western immigrants	percentage of immigrants coming from Europe, North-America, Oceania, Japan and Indonesia with respect to the total population
non-western immigrants	percentage of immigrants coming from Turkey, Africa, Latin-America and Asia (except Japan and Indonesia) with respect to the total population
moroccan immigrants	percentage of immigrants coming from Morocco with respect to the total population
antillean immigrants	percentage of immigrants coming from the Dutch Antilles and Aruba with respect to the total population
surinamese immigrants	percentage of immigrants coming from Surinam with respect to the total population
turkish immigrants	percentage of immigrants coming from Turkey with respect to the total population
other non-western immigrants	percentage of immigrants coming from other non-western countries with respect to the total population
single person households	percentage of households which consist of only one person
households without children	percentage of multi person households without children
households with children	percentage of multi person households with children
average household size	average number of persons in a household
average income per income receiver	average annual net income of individuals who received 52 weeks of the previous year an income (2003)
average income	total annual net income of the population divided by the number of inhabitants (2003)
low incomes	percentage of income receivers with an annual gross income below €14200 per year (2003)
high incomes	percentage of income receivers with an annual gross income above €25200 per year (2003)
non-actives	percentage of income receivers which had social welfare as major source of their income (2003)
water	proportion of surface water with respect to the total surface

Table 16: Description of the variables taken from the StatLine database. All information is from the year 2004, except the variables where is stated otherwise.

D Tables with Results

	Mean Absolute Error			Mean Relative Error		
			Groningen region			
Model	Mean	Sd	Improvement	Mean	Sd	Improvement
Linear Regression	€27348.79	€1625.07	–	16.65%	1.584%	–
CART	€30369.98	€3313.63	-11.05%	18.21%	2.626%	-9.37%
LSBoost (1)	€24666.29	€2052.02	9.81%	14.97%	2.131%	10.09%
LSBoost (2)	€21259.41	€1523.75	22.27%	12.57%	1.546%	24.50%
LSBoost (3)	€20283.89	€1389.76	25.83%	11.82%	1.415%	29.01%
LSBoost (4)	€20170.40	€1459.15	26.25%	11.54%	1.475%	30.69%
LSBoost (5)	€20110.40	€1520.25	26.47%	11.32%	1.480%	32.01%
LADBoost (1)	€23744.68	€2251.29	13.18%	13.63%	1.613%	18.14%
LADBoost (2)	€21287.20	€1771.59	22.16%	12.33%	1.677%	25.94%
LADBoost (3)	€20499.94	€1618.75	25.04%	11.86%	1.639%	28.77%
LADBoost (4)	€20229.31	€1593.25	26.03%	11.68%	1.659%	29.85%
LADBoost (5)	€20141.72	€1591.84	26.35%	11.62%	1.639%	30.21%
			Apeldoorn region			
Linear Regression	€35231.94	€3216.74	–	13.77%	0.9960%	–
CART	€43445.45	€10161.24	-23.31%	16.52%	5.223%	-20.00%
LSBoost (1)	€31767.77	€4080.93	9.83%	11.81%	0.9014%	14.23%
LSBoost (2)	€28563.06	€4190.22	18.93%	10.47%	0.8362%	23.97%
LSBoost (3)	€27648.55	€4115.88	21.52%	9.95%	0.7727%	27.74%
LSBoost (4)	€26877.13	€4159.03	23.71%	9.56%	0.6932%	30.57%
LSBoost (5)	€26643.74	€4176.01	24.38%	9.31%	0.7084%	32.39%
LADBoost (1)	€31153.46	€5182.99	11.58%	11.05%	0.9471%	19.75%
LADBoost (2)	€29128.91	€4838.08	17.32%	10.28%	0.8913%	25.34%
LADBoost (3)	€28394.42	€4542.81	19.41%	10.08%	0.9324%	26.80%
LADBoost (4)	€28255.91	€4484.44	19.80%	10.03%	0.9566%	27.16%
LADBoost (5)	€27824.49	€4348.22	21.02%	9.86%	0.8823%	28.40%
			Eindhoven region			
Linear Regression	€29821.35	€2729.40	–	11.85%	0.7173%	–
CART	€31020.14	€2692.29	-4.02%	12.05%	1.035%	-1.67%
LSBoost (1)	€25108.28	€1881.82	15.80%	9.68%	0.5560%	18.30%
LSBoost (2)	€23532.32	€1594.52	21.09%	8.94%	0.4881%	24.56%
LSBoost (3)	€22549.92	€1512.67	24.38%	8.50%	0.4205%	28.30%
LSBoost (4)	€22025.48	€1581.09	26.14%	8.22%	0.3945%	30.63%
LSBoost (5)	€21856.95	€1516.26	26.71%	8.06%	0.3844%	31.98%
LADBoost (1)	€24326.20	€2080.84	18.43%	9.03%	0.4787%	23.83%
LADBoost (2)	€22870.33	€1838.51	23.31%	8.48%	0.4062%	28.43%
LADBoost (3)	€22398.23	€1790.51	24.89%	8.26%	0.4054%	30.29%
LADBoost (4)	€21986.68	€1655.48	26.27%	8.13%	0.4070%	31.39%
LADBoost (5)	€21822.29	€1594.00	26.82%	8.09%	0.4191%	31.73%
			Amsterdam region			
Linear Regression	€43659.71	€2014.69	–	16.75%	0.5359%	–
CART	€44375.37	€3596.66	-1.64%	16.32%	1.609%	2.57%
LSBoost (1)	€37825.20	€2045.186	13.36%	13.65%	0.4418%	18.54%
LSBoost (2)	€33582.29	€1594.52	23.08%	11.66%	0.3986%	30.41%
LSBoost (3)	€30746.69	€1843.06	29.58%	10.28%	0.3543%	38.61%
LSBoost (4)						
LSBoost (5)	€29283.76	€1991.59	32.93%	9.55%	0.3373%	42.99%
LADBoost (1)						
LADBoost (2)						
LADBoost (3)	€30043.13	€2063.33	31.19%	9.68%	0.3368%	42.21%
LADBoost (4)						
LADBoost (5)	€29320.47	€2014.00	32.84%	9.38%	0.3418%	44.00%
			Rotterdam region			
Linear Regression	€34613.74	€1730.56	–	18.23%	0.6519%	–
CART	€31390.28	€2091.12	9.31%	15.86%	1.061%	13.03%
LSBoost (1)	€29720.83	€1738.70	14.14%	15.08%	0.7008%	17.28%
LSBoost (2)	€25388.71	€1540.41	26.65%	12.87%	0.6044%	29.41%
LSBoost (3)	€23677.78	€1376.89	31.59%	11.91%	0.5614%	34.68%
LSBoost (4)						
LSBoost (5)	€22067.10	€1283.30	36.25%	10.96%	0.4982%	39.88%
LADBoost (1)						
LADBoost (2)						
LADBoost (3)	€23707.98	€1633.10	31.51%	11.36%	0.5112%	37.69%
LADBoost (4)						
LADBoost (5)	€22637.88	€1508.22	34.60%	10.91%	0.5178%	40.15%
			Zeeland region			
Linear Regression	€41999.26	€3459.74	–	23.93%	1.849%	–
CART	€43666.48	€4198.79	-3.97%	23.23%	2.412%	2.93%
LSBoost (1)	€32815.32	€3081.94	21.87%	17.46%	1.572%	27.03%
LSBoost (2)	€29608.46	€2782.55	29.50%	15.68%	1.524%	34.48%
LSBoost (3)	€29466.44	€2803.23	29.84%	15.42%	1.399%	35.58%
LSBoost (4)	€28915.97	€3027.87	31.15%	14.96%	1.333%	37.48%
LSBoost (5)	€28914.29	€3127.07	31.16%	14.83%	1.299%	38.03%
LADBoost (1)	€33194.28	€3953.89	20.96%	16.77%	1.552%	29.92%
LADBoost (2)	€31245.20	€3736.43	25.61%	15.79%	1.471%	34.01%
LADBoost (3)	€30446.38	€3416.95	27.51%	15.36%	1.385%	35.83%
LADBoost (4)	€29875.19	€3430.52	28.87%	15.17%	1.348%	36.61%
LADBoost (5)	€29567.88	€3332.21	29.60%	15.10%	1.288%	36.90%

Table 17: Results for the experiments without demographic data. The numbers in the parentheses behind the boosting models represent the depths of the base learners.

Groningen Region									
		Lin Regr		CART		LSBoost		LADBoost	
House Type	Size	MAE	MRE	MAE	MRE	MAE	MRE	MAE	MRE
Apartment	1549	€17827.19	15.38%	€18587.10	16.63%	€12479.71	10.32%	€12598.37	10.37%
End-terraced	283	€23691.92	14.68%	€26598.52	15.36%	€17731.11	11.21%	€18502.50	11.54%
Mid-terraced	901	€22052.74	14.37%	€24138.70	15.25%	€15295.61	9.67%	€14958.97	9.24%
Detached	302	€64154.15	30.19%	€77635.62	34.73%	€55537.34	24.30%	€54995.75	24.98%
Semi Detached	312	€36969.56	15.99%	€38286.19	16.70%	€26185.88	11.30%	€26951.49	11.42%
Stepped	40	€44411.16	20.63%	€35043.76	15.67%	€27394.72	12.74%	€25903.62	11.64%
Apeldoorn Region									
		Lin Regr		CART		LSBoost		LADBoost	
House Type	Size	MAE	MRE	MAE	MRE	MAE	MRE	MAE	MRE
Apartment	536	€23555.67	18.24%	€27050.15	21.14%	€14499.10	10.13%	€15460.91	11.92%
End-terraced	272	€20101.79	9.56%	€26216.55	12.69%	€14368.29	6.72%	€14860.05	6.93%
Mid-terraced	584	€17340.62	8.90%	€17018.77	8.88%	€10049.46	5.08%	€10250.35	5.15%
Detached	615	€67918.12	17.47%	€87852.79	21.95%	€59389.71	14.39%	€62723.96	14.82%
Semi Detached	345	€31012.65	12.20%	€36953.09	14.75%	€23058.85	9.10%	€23544.35	9.33%
Stepped	50	€23233.14	10.85%	€23409.40	10.95%	€14563.45	6.68%	€14488.86	6.77%
Eindhoven Region									
		Lin Regr		CART		LSBoost		LADBoost	
House Type	Size	MAE	MRE	MAE	MRE	MAE	MRE	MAE	MRE
Apartment	903	€20699.26	12.05%	€21040.66	12.27%	€12640.84	6.92%	€12943.35	6.93%
End-terraced	514	€24740.35	11.81%	€28418.24	13.07%	€18093.28	8.38%	€18277.05	8.39%
Mid-terraced	1296	€22067.13	11.16%	€20493.94	10.35%	€14236.11	7.05%	€14749.54	7.19%
Detached	370	€79229.99	15.22%	€86893.90	16.57%	€68535.87	13.11%	€63773.48	12.26%
Semi Detached	532	€33417.15	11.32%	€39114.38	13.16%	€27277.07	9.18%	€28441.11	9.52%
Stepped	115	€29136.63	11.05%	€39109.76	15.04%	€24721.68	8.89%	€26390.02	9.46%
Amsterdam Region									
		Lin Regr		CART		LSBoost		LADBoost	
House Type	Size	MAE	MRE	MAE	MRE	MAE	MRE	MAE	MRE
apartment	6742	36678.28	16.43%	35854.45	15.94%	23032.78	9.06%	23275.45	8.92%
end-terraced	335	55853.86	16.04%	59701.44	15.06%	41154.55	10.14%	41761.38	9.59%
mid-terraced	1210	62159.73	17.95%	63989.83	16.69%	45668.35	10.97%	44288.03	10.63%
detached	80	213587.50	27.06%	252667.24	32.53%	182074.62	23.68%	173400.37	21.90%
semi-detached	104	94040.41	19.31%	122671.10	22.26%	73634.77	14.15%	75611.02	14.01%
stepped	19	82789.66	24.98%	63989.42	18.83%	35195.60	10.82%	29103.85	8.87%
Rotterdam Region									
		Lin Regr		CART		LSBoost		LADBoost	
House Type	Size	MAE	MRE	MAE	MRE	MAE	MRE	MAE	MRE
apartment	4625	26331.68	18.66%	23959.67	16.87%	16710.27	11.30%	16952.31	11.18%
end-terraced	635	43616.01	16.94%	40319.59	15.42%	26620.20	9.75%	28617.72	10.06%
mid-terraced	1753	37521.85	16.45%	33116.94	14.42%	22258.11	9.58%	23026.87	9.53%
detached	173	136534.87	26.24%	152763.75	27.88%	108303.86	19.66%	105308.32	18.57%
semi-detached	263	62357.89	17.19%	74818.91	20.35%	46406.91	12.88%	48668.82	13.32%
Stepped	61	34317.44	12.71%	38894.28	14.30%	21140.52	8.00%	22721.22	8.04%
Zeeland Region									
		Lin Regr		CART		LSBoost		LADBoost	
House Type	Size	MAE	MRE	MAE	MRE	MAE	MRE	MAE	MRE
Apartment	232	€35878.69	27.50%	€41320.77	30.59%	€27387.84	19.72%	€28948.27	19.89%
End-terraced	356	€34625.24	23.10%	€29782.00	19.93%	€19796.46	12.58%	€19869.47	12.71%
Mid-terraced	714	€31529.63	22.89%	€29386.76	20.88%	€18588.37	12.90%	€18875.97	12.99%
Detached	562	€67684.37	27.08%	€80484.99	30.21%	€56843.39	20.29%	€57309.8	20.82%
Semi Detached	318	€33454.68	20.76%	€32656.20	19.29%	€19508.23	11.39%	€20197.78	11.96%
Stepped	34	€38145.65	18.47%	€44478.35	18.54%	€21023.92	9.58%	€20446.54	9.83%

Table 18: Results for Submarkets

	MAE			MSE		
Model	Mean	Std	Impr.	Mean	Std	Impr.
Groningen region						
LSBoost(5)	€18892.04	€1565.54	6.06%	10.41%	1.357%	8.04%
LADBoost(5)	€19147.28	€1514.23	4.94%	10.93%	1.600%	5.94%
Apeldoorn region						
LSBoost(5)	€25497.43	€4095.57	4.30%	8.64%	0.5955%	7.20%
LADBoost(5)	€26635.71	€4704.50	4.27%	9.26%	0.7960%	6.09%
Eindhoven region						
LSBoost(5)	€20748.22	€1471.96	5.07%	7.71%	0.3384%	4.34%
LADBoost(5)	€20895.34	€1583.24	4.25%	7.78%	0.3718%	3.83%
Amsterdam region						
LSBoost(5)	€26301.93	€1885.80	10.18%	8.61%	0.2967%	10.92%
LADBoost(5)	€26373.36	€1803.08	10.05%	8.51%	0.2933%	9.28%
Rotterdam region						
LSBoost(5)	€21410.43	€1322.24	2.98%	10.56%	0.4302%	3.65%
LADBoost(5)	€21936.93	€1498.14	3.10%	10.42%	0.4302%	4.49%
Zeeland region						
LSBoost(5)	€27189.76	€2936.54	5.96%	13.90%	1.114%	6.27%
LADBoost(5)	€28165.62	€3026.54	4.74%	14.64%	1.196%	3.05%

Table 19: Results with the use of demographic data