Predicting the rental turnover rate of a housing corporation

Research Paper Business Analytics

Emma Schipper
Student number: 2584343
Supervisor: Eduard Belitser

9 March 2018

De Vrije Universiteit
Faculty of Science

Abstract

Goal: The aim of this research is trying to predict the rental turnover rate of housing corporation Trivire. This is the percentage of tenants in a housing portfolio (rental data) that moves to another house within a certain time period.

Approach: General approach is to produce tree-based algorithms to make a binary predicting model. Three tree-based algorithms are used: decision tree, bagging tree and random forest.

Practice: The data is unbalanced. For this reason various sampling techniques are used to generate balanced classes in the models. According to the unbalanced data problem only non-misleading performance metrics are evaluated.

Results and conclusion: The overall best performing algorithm is the random forest with SMOTE as sampling technique. In addition a reduced feature model shows good performances on again random forest with undersampling technique. The most important variables are: age of a tenant, information on the length of stay of a current tenant and information on the rented houses. The results of the models show that the prediction of the rental turnover rate is a difficult task. The algorithms have difficulties to achieve high performances on all the metrics. On the other hand some algorithms show high performances on specific metrics. In practice the strengths of different models can be used through combining them, though this is subjected to expert judgement.

Discussion: There are some recommendations within this research to further discover the unbalanced data problem. The first recommendations is to try other and perhaps more suitable types of algorithms to handle unbalanced data. In addition the data could be changed into more balanced classes. This can be achieved by using yearly turnover rate data instead of half yearly. Another recommendations is to elaborate the data set over time. In this way it is possible to involve times dependencies in the models. The last recommendation is to add more informative features to the data set.

Contents

1 Introduction

Housing corporations own real estate and have a lot of rental data at their disposal. The rental data consist of a portfolio of houses given a certain time period with information on the features of houses and the current tenants. Housing corporation Trivire has asked to investigate the prediction of their rental turnover rate. This is the percentage of tenants in the portfolio that moves to another house within a certain time period. Trivire's asset management has two main reasons to cooperate within this research:

The first reason is that the expected rental turnover rate gains insights into rental harmonization, the mechanism that the rental price of a home will be increased by a certain amount after termination of the rental contract. This means that rental harmonization is determined by the rental turnover rate. For housing corporations such as Trivire rental harmonization is an important resource to balance the price to quality ratio of rental house prices. Therefore the rental turnover rate is an important parameter in the valuation of real estate and of great influence on yield calculations and investment analyses of real estate complexes that are made by asset management department. In addition a prediction on specific houses in a portfolio that are going to change from tenant or will stay occupied could be useful for budget analyses on specific housing or building. Summarizing expected rental turnover rate helps to perform future yield and investments calculations for the valuation of real estate and it could be informative on a deeper level to gain insights into houses and buildings and their expected rental turnover rate.

The second reason is to gain insights into the factors that explain and affect the rental turnover rate. This could possibly lead to development of new control measures, for example the flow of tenants.

The reasons above show the importance of predicting the rental turnover rate and gain insights of factors that affect the rental turnover rate for Trivire. In general the housing corporation market is currently focused on the data analysis area. This means translating their portfolios to story telling dashboards with numerical information. There is currently not enough attention for the big opportunities in the prediction area.

The research paper is divided into 7 chapters. In Chapter 2 the data preparation process is discussed divided into an explanation of the raw data and the data transformation. In Chapter 3 the data analysis part is evaluated to get a global view of the data. The predictive part of the research begins in Chapter 4, which starts with the theory of suitable tree-based algorithms. In Chapter 5 the implementation of the model is discussed followed by the results in Chapter 6. Chapter 7 contains the conclusion followed by the final Chapter 8 the discussion.

2 Data preparation

In this section the process of data preparation is discussed. The starting point of data preparation is analyzing the raw characteristics of the received data. Thereafter the raw data need to be transformed. The transformation of a data set starts with cleaning the data. This is followed by adding valuable features to the data with as goal to provide new information into the data set. The data preparation process is applied on the data set of housing corporation Trivire. This process is described in two parts; first a description of the raw data in Section 2.1 followed by the data transformation part in Section 2.2.

2.1 Raw data

The received data from the housing corporation Trivire can be separated into two parts. The first part consists of seven housing portfolios over a period of three years at the following dates: 1-1-2014, 1-7-2014, 1-1-2015, 1-7-2015, 1-1-2016, 1-7-2016 and 1-1-2017. The data sets have on average 15.967 houses (thereafter: units), each unit represents a single row in the data set. The total data set consists of 111.768 units. From the 1-1-2014 until the end date 1-1-2017 the portfolio shrunk by 1.42%, which is quite stable in a period of three years. Each unit in the data set has corresponding features, which are represented in the columns. In total each unit has 18 features. In Table 1 below the features of the units are given.

Feature	Explanation
Date	Date of issue
Garden plot	Unique number for each house in portfolio
Complex	Number of complex (not a unique value)
Contract	Unique number of contract
Date of birth	Current date of birth of tenant
Address	Location details of the house (diffused over different columns)
Date of entry	Current date of entry of tenant
Date of departure	Current date of departure of tenant
House points	Quality of a house on paper
Gross rent	Current gross rent a tenant pays
Net rent	Current net rent exclude service cost
Eligible rent	Part of rent over which the current tenant can receives rent allowance
Unit type	Different types of units: independent living space, dependent living space, parking facility and property (no living space)
Build year	Build year of the house
Valuation of real estate	Valuation of houses through township to provide information for tax authorities
Gender	Gender of tenant (or empty in case of vacancy)
Energy label	Energy label of house divided into six classes (A up to G label)
Energy index	Numerical energy index of house

Table 1: The features of the portfolio data set and their explanations.

The second part of the received data provides a data set from a website. This website offers vacant rented houses from different housing corporations among which Trivire. Property seekers can visit the website and place a reaction if they are interested in an offered house. The website collects a lot of data with information of property seekers. This means that in the first place the data set has information on the offered houses of the Trivire portfolio. Furthermore the data set has information of tenants who are property seekers, among other things tenants of Trivire portfolio. Thereby a property seeker will have a large opportunity to leave their current house and ensures a rental turnover in the portfolio. Overall the information of the data set could be useful in investigating the aim of this research; prediction of the rental turnover of the Trivire portfolio. The following features in Table 2 are part of the data set:

Feature	Explanation
Advertisement number	Number for each offered house
Reaction date	Date that the property seeker placed a reaction on an offered house
Publication date	Date that the house is offered
Address offered house	Location details of the offered house
Address property seeker	Location details of address of the property seeker
Date of birth	Date of birth property seeker
Registration number	Unique number for each reaction on a house
Owner of house	Name of housing corporation that owns the offered house

Table 2: The features of the reactions data set and their explanations.

The separated two data sets need to be merged to get interesting information as input for the prediction of the rental turnover rate of Trivire. In the next section the data transformation part will be described.

2.2 Data transformation

The first step of the data transformation is to merge the two data sets, the Trivire portfolio data set and the Reaction data set from the website. The merging process of those two data sets needs firstly a base data set, the starting point. The Trivire portfolio data set is used as the base data set. Within the portfolio data set the real rental turnover rate can be distracted by noticing the departed tenants over the three year period. In addition the Trivire portfolio data set has information per house in each row of the data set which is the easiest way to make predictions on house level. The data set Reaction will be used to add valuable features to the base data set.

The base data set needs to be cleaned. Below an overview is given numeration of the most important cleaning actions of existing features which will be part of the final data set.

Feature	Cleaning action
Date	Changed to numbers of periods instead of dates
Gender	Changed to factors instead of words
Day of entry	Changed to length of stay where vacancy will have a length of zero
Date of birth	Changed to the age of the current tenant
Type unit	Changed the names into factors

Table 3: Cleaning actions on various features.

There are two features Property seeker and Number of reactions that are derived in the Reaction data set and added to the base data set. For these features the added information to the base data set is filtered and only applicable to tenants in the Trivire portfolio. Thereby the features Turnover type, Rental classes and Rental turnover rate are derived from the base data set. On the next page the new features are given with a description. These features will be part of the final data set:

New feature	Description
Property seeker	Binary feature with information if the current tenant in the portfolio had placed a reaction of interest in another house (1=yes, 0=no)
Number of reactions	Number of reactions that a current tenant in the portfolio placed at different houses where he/she is interested in
Turnover type	Derived from the last number of the feature Complex: single-family house (1), multi-family house (2), parking facility (3) and other (for example business space) (4)
Rental classes	Each house belongs to a rental class based on their eligible rent
Rental turnover rate	Binary variable which gives information based on realization whether the current tenant at time t has moved in the next period t+1 (1=yes, 0=no)

Table 4: Added features.

In Table 4 the last new added feature is Rental turnover rate. In order to discover the aim of this paper the feature Rental turnover rate is needed to create a model which could possibly predict the rental turnover rate. For this reason the Rental turnover rate is the response variable and will be coded as the following binary variable: 1 if a turnover takes place between t and t+1. Otherwise the code will 0 if the tenant remains at the same house in the Trivire portfolio between t and t+1.

In the end the following features were removed and not part of the final data set:

Removed feature	Reason
Contract	Feature Garden plot is used as unique feature for each house
Address & Location	There is already information about the address and location through Garden plot and hard to classify into suitable variable type
Eligible rent	Many similarities with the feature net rent

Table 5: Removed features.

As earlier mentioned the feature Rental turnover rate is part of the final data set. This feature will be used in the model as binary response variable. The final data set consists of the following 18 features (explanatory variables): Period, Garden plot, Complex, Mutation type, Type unit, Age, Gender, Length of stay, Property seeker, Number of reactions, Gross rent, Net rent, Rental classes, House points, Build year, Energy label, Energy index and Valuation of real estate.

In Chapter 3 the statistics of the Rental turnover type in the final model are given and the distributions of the final model features will be analyzed.

3 Data analysis

Rental turnover rate is the response variable of the final model. For this reason it is required to investigate the statistics of the response variable to understand its behaviour. Trivire has different types of turnover as discussed in Table 4. The most important turnover types are the housing units: single- and multi-family houses. This is the core business of Trivire. This type covers on average 88.7% of the portfolio over the last three years. The other 11.3% of units represents a collection of parking facilities and other for example business spaces. In the rest of the paper solely the housing units are relevant.

Historical information of the rental turnover rate in the given time period (01-01-2014 up to 01-01-2017) are given in Figure 1 below.

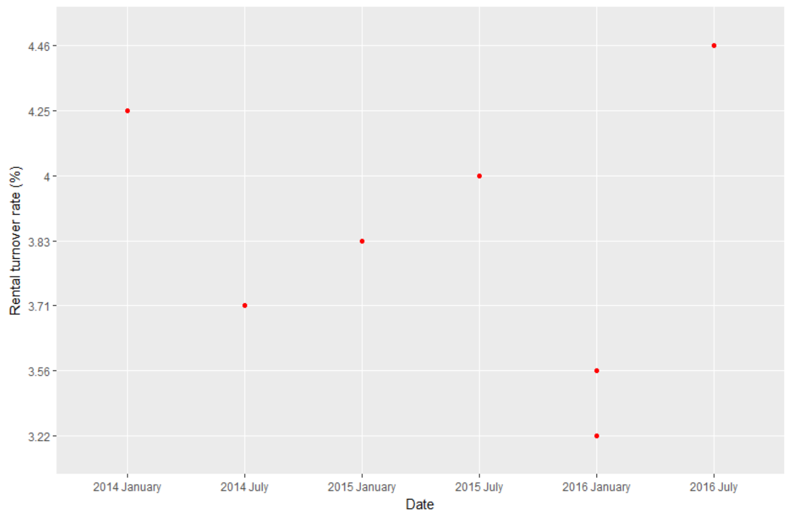

Figure 1: Rental turnover rate (%) over period 01-01-2014 up to 01-01-2017.

Statistics arithmetic mean (AM) and standard deviation (SD) are used to investigate the (historical) behaviour of the rental turnover rate. First the formulas for arithmetic mean (1) and sample standard deviation (2) over at a given period [1], [3]:

$$AM = \mu = \sum_{i=1}^{n} \frac{x_i}{n} \tag{1}$$

$$SD = \sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n-1}} \tag{2}$$

In Table 6 below the arithmetic mean and standard deviation of the response variable Rental turnover rate are given at specific periods: January, July and Ultimate.

Date	AM	SD
January (01-01)	3.72%	0.44%
July (01-07)	4.06%	0.38%
Ultimate cumulative (31-12)	7.82%	0.14%

Table 6: Statistics of the response variable Rental turnover rate over the periods: January, July and Ultimate.

Table 6 above gives a clear indication on the distribution of the Rental turnover rate. The response variable obviously has unbalanced classes, which means that the class of tenants that has moved is underrepresented and so the minority class. The class with tenants who want to stay in their house is over represented and so the majority class. The probability that a random tenant in the portfolio will move in the next period will be hard to predict with unbalanced classes. A learning algorithm is based on the assumption to maximize the accuracy. For realistic problems this gives the following situation: the response variable is binary and if 98% of the binary classes is labeled as one class then a learning algorithm is hardly pressed to do better than the 98% accuracy achievable in the true majority class. This results into 100% prediction of the majority class [11]. In Chapter 4 this problem will be evaluated in detail.

The features (explanatory variables) are important in the final model, because they will form the explanation of the prediction part. Therefore understanding the behaviour of these features is required. A couple of interesting features will be discussed. First Figure 2 and 3 below give the distributions of the following features: Age tenant, Length of stay, Gross rent and Valuation real estate. The figures show the most recent developments in the portfolio at 01-01-2017.

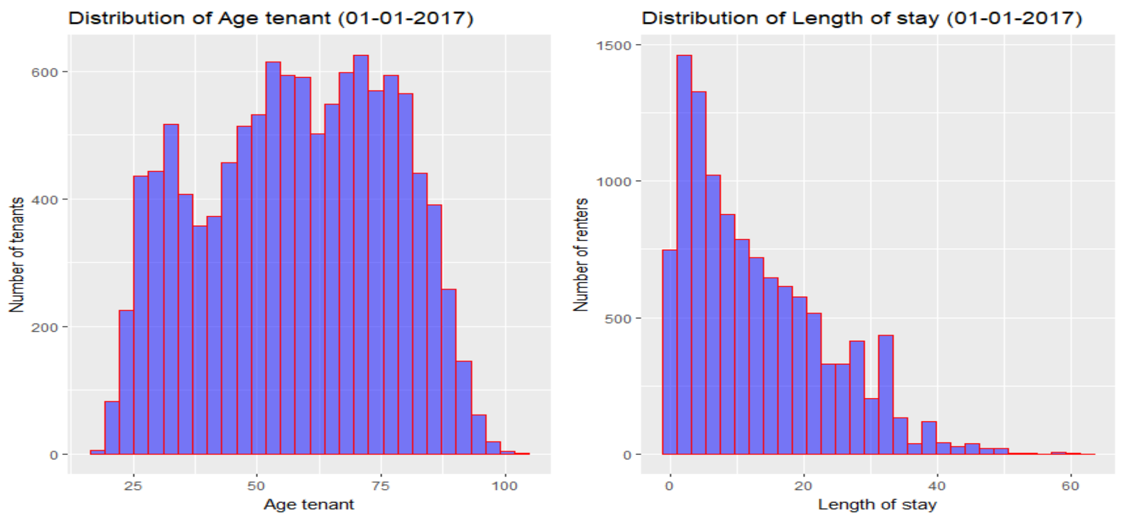

Figure 2: Distribution of Age tenant (left) and distribution of Length of stay (right).

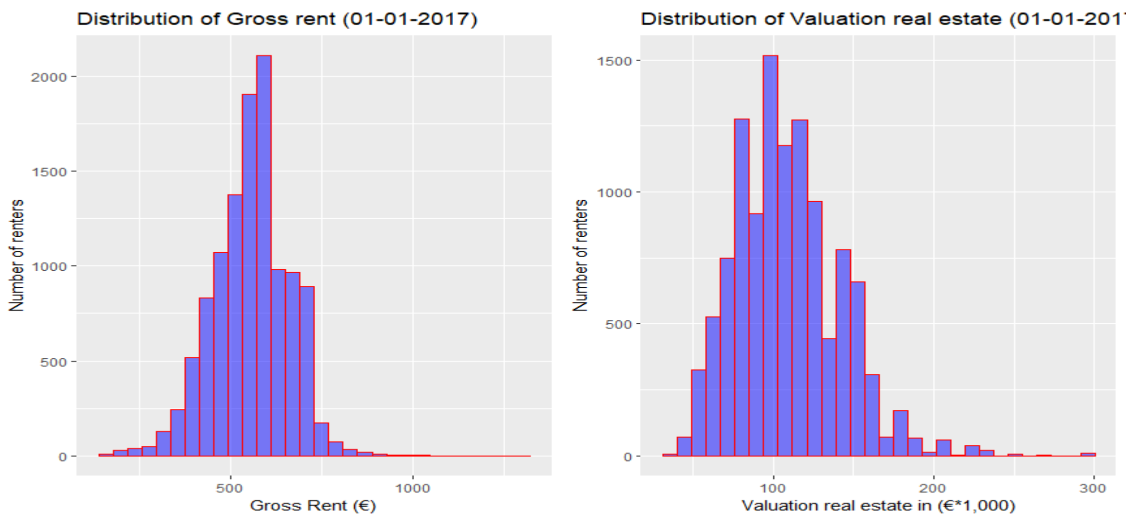

Figure 3: Distribution of Gross rent (left) and distribution of Valuation real estate (right).

In order to support Figure 2 and Figure 3 above, Table 7 below is given with various statistics about the features:

Statistics	Age tenant	Length of Stay	Gross rent (€)	Valuation of real estate (€)
Minimum	18	0	143	35,000
First quarter	43	4	194	85,000
Median	59	11	563	105,000
Third quarter	73	20	622	130,000
Maximum	104	62	1,286	295,000
Average	58	13	557	109,386
SD	19	11	105	31,915

Table 7: Statistics of the features Age tenant, Length of stay, Gross rent and Valuation of real estate.

Figure 2, Figure 3 and Table 7 together give the following observations for each feature:

- Age tenant: the portfolio has a wide age spread, but exists predominantly of older ages with an average of 58 years old;
- Length of stay: on average tenants at 01-01-2017 live 13 years in the same rent at house, the plot is obviously skewed to the right with as first quarter (share of 25%) a stay of 4 years;
- Gross rent: distribution has the biggest share in the cheaper rents with as first quarter (share of 25%) €194 a median of €563 and third quarter of €622. The gross rent is predominantly low;
- Valuation of real estate: conclusion of the cheaper gross rents corresponds to the lower valued real estate, where the first quarter (share of 25%) €85,000 a median of €105,000 and third quarter of €130,000. The amounts are centered together and relatively cheap.

4 Model theory: Tree-based algorithms

In this section the model theory will be discussed. The response variable of the data set is the Rental turnover rate and is a binary variable described in Section 2.2. There are two important issues by choosing suitable algorithms: the binary response variable leads to a classification problem and the data set has unbalanced classes as explained in Chapter 3. In response to these two issues the tree-based algorithms will be chosen as learning algorithms. The reason for this is that tree-based algorithms often perform well on unbalanced data. The hierarchical structure of trees permit them to learn from signals which will be received from both classes movers and non-movers. In practice tree-based algorithms are widely used in medical and credit risk cases. For example predicting risk of getting cancer based on medical data of patients or the credit risk of loan applicants based on their individual data features. For predicting the rental turnover the algorithms decision tree, bagging tree and random forest will be applied to the data set. In Section 4.1 the decision tree algorithm is evaluated and thereafter in Section 4.2 the bagging tree and the random forest algorithm.

4.1 Decision tree

In general a decision tree can be visualized as a reversed tree which is based on partitioning idea. The tree starts from the root which represents a certain feature, also known as input value. The root will be split up recursively according to the rest of its features, every split up represents a decision node. At the bottom of the reserved tree in the terminal nodes (in the leaves of the tree) the response variable is classified. The constructed decision tree ends up with different paths from the root to the decision nodes and will end in the last stage the terminal nodes.

In contrary to traditional statistical models, the decision tree algorithm makes no assumptions on distributions of the dependent and independent variables. Variables do not have to follow a specific distribution. This means that the whole learning algorithm is not based on a probabilistic model. Therefore the predicting part does not rely on a confidence level, but on the accuracy of historical records.

The base theory of the decision tree is described and found in Classification and Regression Trees (CART) and discovered through Breiman, Friedman, Olshen and Stone [8]. In this paper the CART method is applied as learning algorithm. The CART method involves classification as well as regression trees. The target value is a binary response variable and is modelled as a classification tree. The CART method allows features (explanatory variables) to be categorical or continuous. Another characteristic of the CART method is that the splits in the nodes are always binary splits. In this paper the regression tree is out of scope, because the data has a binary response variable. The model theory of the classification tree by the use of CART [13] is described in the rest of this section and applied on the rental data to predict the Rental turnover rate.

In the decision tree algorithm prior probabilities plays a crucial role. As earlier mentioned a decision tree does not have any assumptions about distributions of variables. In constructing a classification tree the CART method uses prior probabilities. A prior probability is defined as the probability of an event before the collection of new data and evidence. In other words there are established beliefs about the probability distribution of the prior probability before new evidence is taken into account. In the decision tree the number of rental turnover records (quantity to predict) in the data set D are denoted as N. The distribution of the rental turnover records divided over two classes are denoted in equation (3):

$$N_j = \# \, of \, records \, in \, class \, j \, in \, the \, data \, set \, (j = 1, 2) \tag{3}$$

The prior probability of a class is given in equation (4):

$$\pi_j = \frac{N_j}{N} \tag{4}$$

The prior probability (4) is estimated as the assumption of priors data following that the proportion of the classes of the response variable have the same distribution as the population. This is the only base assumption that is made to construct the tree.

The decision tree starts in the root of the tree with a chosen feature (explanatory variable) to further split on. The root node feature is chosen after a selection procedure. This selection procedure choose the best feature to split on and is determined through a certain measure of impurity. Let j = 1,2 the two classes of the response variable as defined in (3). The estimated probability of class j within node t is given in equation (5). A node t is representing one of the 18 features (explanatory variables) in the rental data set.

$$p(j|t) = p(1|t) + p(2|t) = 1 \qquad\qquad (t = 1, .., 18) \tag{5}$$

Given an impurity function $\emptyset$ the impurity measure *i(t)* at node t will be defined in equation (6):

$$i(t) = \emptyset[p(1|t), p(2|t)] \tag{6}$$

In equation (6) the impurity measure is given as a function of the class probabilities. Furthermore, the impurity measure uses a splitting rule to determine the feature to split on. This splitting rule is called the Gini diversity index. The splitting rule starts with the Gini impurity measure *i(t)* (7) and is further distracted through the Gini impurity function S (8):

$$i(t) = 1 - S \tag{7}$$

$$S = \sum_{j=1}^{2} p^2(j|t) \tag{8}$$

Gini impurity function S (8) can reach the following minimum and maximum values (9):

$$S = \begin{cases} 0.5 & if\, p(1|t) \ = \ p(2|t) \\ 1 & if\, p(1|t) = 1 \vee p(2|t) \ = \ 1 \end{cases} \tag{9}$$

This means that the Gini impurity measure *i(t)* (7) varies between the following values at node t (10):

$$i(t) = \begin{cases} 0 & if\, S = 1 \\ 0.5 & if\, S = 0.5 \end{cases} \tag{10}$$

If impurity measure *i(t)* (10) is equal to 0 then this means that the classification rate is 0. In other words all the records in a node t belong to one class which will result in 100% confidence in the split from node t to the 100% confidence class.

The information above has given the distraction of the Gini impurity measure *i(t)* at a node *t*. In the last step the best feature is chosen based on the best split criterion. The best split criterion is also known as 'Goodness-of-Split Criteria'. Let s be the split at node t. The best split is chosen by the largest reduction in impurity (11):

$$\Delta i(s,t) = i(t) - p_L[i(t_L)] - p_R[i(t_R)] \tag{11}$$

Where,

$$\begin{aligned}
&s = split\,at\,node\,t\\
&p_L = proportion\,of\,records\,at\,node\,t\,that\,go\,into\,left\,child\,node\\
&p_R = proportion\,of\,records\,at\,node\,t\,that\,go\,into\,right\,child\,node\\
&i(t_L) = impurity\,of\,left\,child\,node\\
&i(t_L) = impurity\,of\,right\,child\,node
\end{aligned}$$

In general the selecting feature process in every decision node follows a greedy approach, a mathematical process that search for easy to implement solutions to complex multi-step problems. In practice the decision tree algorithm has a greedy approach in choosing the best split. The choice is made on the available information at that moment and will not be reconsidered which could possibly lead to sub optimal choices instead of optimal choices [12].

The theory above has explained the selection procedure from the root to various decision nodes within the use of the CART method. The last crucial role in the decision tree is to assign classes to the terminal nodes. This process is part of the class assignment rules. The rule applied to the model is called the Plurarity Rule. Assign terminal node t to a class for which *p(j|t)* (5) is the highest. In other words the majority of the records in the terminal node determine the class of that terminal node. Under the assumption that the misclassification costs are equal for each of the classes. Misclassification cost can be applied to the distribution of misclassified instances. Only if there is a reason to assume that a certain kind of error is more costly than another which is not directly applicable on the rental data, because moving is not more costly in comparison to not moving for the rental turnover rate. Let $c(i \mid j)$ be the cost of classifying a class j record as a class i record (12):

$$c(i \mid j) >= 0 \quad if\, i \neq j, \quad c(i \mid j) = 0 \quad if\ i = j \tag{12}$$

In equations 13 and 14 the application of the Prularity Rule is further distracted:

$$r_1 = \pi_1 * c(2 \mid 1) \tag{13}$$

$$r_2 = \pi_2 * c(1 \mid 2) \tag{14}$$

Where,

$$\begin{aligned}
&\pi_1 = prior\,probability\,of\,class\,1\,at\,node\,t\\
&\pi_2 = prior\,probability\,of\,class\,2\,at\,node\,t\\
&r_1 = cost\,of\,assigning\,node\,t\,to\,class\,1\\
&r_2 = cost\,of\,assigning\,node\,t\,to\,class\,2
\end{aligned}$$

According to the Prularity rule where $c(2 \mid 1) = c(1 \mid 2)$ the class with the highest prior probability is assigned to the terminal node.

In the next section extended versions of decision trees will be discussed.

4.2 Bagging tree & random forest

Bagging tree and random forest are well known extended tree-based algorithms. In this section first the bagging tree algorithm will be described and thereafter the random forest algorithm.

The bagging tree is also known as a bootstrap aggregation tree. In general a bootstrap method is any test that relies on random sampling with replacement. The algorithm can be defined in the following steps [5]:

1. Define *Bagging(D, K, L)* with D the data as earlier defined in Section 4.1, K as the size and L as the learning algorithm in this case decision tree as described in Section 4.1;
2. For *k=1* to K build a bootstrap sample D_k from D by sampling D data points with replacement then run L to D_k to produce a model M_k. So a new model k is based on corresponding bootstrapped sample k;
3. Return models M_k given *k=1* up to K where M_k for *k=1* up to K are fixed decision trees models;
4. The aggregation process through averaging M_k models given *k=1* up to K.

In the last and fourth step of *Bagging(D, K, L)* the aggregation process is applied on the bootstrapped models. The statistical accuracy can be evaluated by looking at the variability of predictions between the different bootstrap models. Within decision trees the method is applied to average the random sampled decision trees which will reduce the variance.

The random forest algorithm is a further extended method of the bagging tree algorithm. The difference in relation to the bagging tree is that the random forest draws a random sample of m features at each tree split and only those m features are considered for splitting. Typically $m =$ *sqrt(t)* where t is 18 number of features. Within bagging trees only random samples of the decision trees are drawn, there is no option for random sampling in features. The model can be defined through the following steps [6]:

1. Define *Bagging(D, K, L, m)* with D the data as earlier defined in Section 4.1, K as the size, L as the learning algorithm in this case decision tree as described in Section 4.1 and m as subspace dimension;
2. For *k=1* to K build a bootstrap sample D_k from D by sampling D data points with replacement, select m features at random and reduce dimensionality of D_k accordingly run L to D_k without pruning to produce a model M_k;
3. Return models M_k given *k=1* up to K where M_k for *k=1* up to K are fixed decision trees models with different feature selections;
4. The aggregation process through averaging M_k models given *k=1* up to K.

In this chapter the theory of tree-based models was discussed. The performance of the tree-based model will be evaluated in Chapter 6, but first in the next chapter the implementation process of the tree-based models.

5 Implementation

In this chapter the implementation of the tree-based algorithms discussed in Chapter 4 will be evaluated. An important part of the implementation is the unbalanced data problem as explained in Chapter 3. The first step is to choose algorithms which are suitable for handling unbalanced data. The second step is to try different techniques which are able to generate balanced classes. In Section 5.1 the different implemented techniques will be discussed, followed by an explanation of cross validation in Section 5.2. This is a method for training and testing the data and will be implemented in the tree-based algorithms. Furthermore in Section 5.3 the final model implementation in R will be showed and explained.

5.1 Generating balanced classes

Unbalanced data models can be improved by trying different techniques to generate balanced data classes. In this case the classes are unbalanced, because the class 'Yes' (coded as 1; tenants that are moved) of the response variable is strong underrepresented and the minority class. Whereas the class 'No' (coded as 0; tenants that are not moved) of the response variable is over represented and stands for the majority class. In the model the following four techniques are implemented:

- Undersampling: select the majority 'No' samples of the Rental turnover rate randomly without replacement and drop them out the model, this will result in equal minority and majority classes;
- Oversampling: select the minority 'Yes' samples of Rental turnover rate and duplicate them by randomly sampling with replacement from this class, this will result in equal minority and majority classes;
- Synthetic Minority Over-sampling Technique (SMOTE): oversamples the minority class 'Yes' by using bootstrapping and k-nearest neighbor to synthetically create additional observations of that event [10];
- Random Over-Sampling Examples (ROSE): especially suited for binary unbalanced data and the balanced samples are generated according to a smoothed bootstrap approach [9].

The techniques are implemented in Section 5.3 and applied on the trainingsset. The testset consists only of original and (not adjuted) data. In the next Section 5.2 the partitioning into trainings- and testset is explained.

5.2 Cross validation

The data is split into a trainingsset and a testset. The trainingsset is 70% of the data and the testset the remaining 30%. The tree-based algorithms will be built in the trainingsset and the testset will test the performance of the model.

The cross validation is part of the trainingsset. This means cross validation is a technique which will 'test' the model in training phase. In other words cross validation is used to improve the performance of the trainingsset. In this paper the k-fold cross-validation is used. This technique takes the training data and partitions it into k groups of equal sizes. Then k - 1 groups are used to train a set of models that are then evaluated on the remaining group. This procedure is repeated for all k possible choices for the held-out group and performance scores from the k runs are then averaged [4].

5.3 R parameters

In this section the implementation and parameter part in R are discussed. The software R is used for implementation of predicting the rental turnover. The package caret [7] is used. Below the program code of the model is shown with an explanation of each command below the program code.

```
caret::train(Response variable(1) ~ .,
                    data(2) = train_data,
                    method(3) = "algorithm",
                    preProcess(4) = c("scale", "center"),
                    trControl(5) = trainControl(6)(method(7) = "repeatedcv",
                                        number(8) = 5,
                                        repeats(9) = 5,
                                        verboseIter(10) = FALSE,
                                        sampling(11) = "samplingmethod"))
```

1. Response variable: variable which is used as the response variable in the model in this case Rental turnover rate;
2. data: the training set of the model which is described in the previous section;
3. method: chosen tree-based algorithm described in Section 4;
4. preProcess: transformation (centering, scaling etc.) can be estimated from the training data and applied to any data set with the same variables;
5. trControl: a list of values that define how this function acts;
6. trainControl: command of trControl;
7. repeatedcv: repeated cross validation described in Section 5.2;
8. Number: number of k-fold cross validation;
9. Repeat: number of cross validations to repeat;
10. verboseIter: a logical for printing a training log;
11. sampling: sampling methods to generate balanced classes explained in the Section 5.1.

According to the program code above; the implementation in R is done for three tree-based algorithms (numeration number 3) described in Chapter 4. Each model is applied on the different sampling techniques (numeration number 11), but the results of the original sample will first be considered.

6 Results

In this section the results of the tree-based algorithms will be discussed. First in Section 6.1 the chosen performance metrics will be evaluated. In Section 6.2 the results of the models are discussed. The last Section 6.3 introduces models with reduced features. These models are trying to improve the performances of the first model results in Section 6.2.

6.1 Performance metrics

The algorithms decision tree, bagged and random forest described in Chapter 4 are used as prediction models. These algorithms are applied on different sampling techniques for unbalanced classes described in Section 5.1 and implemented as prediction model described in Section 5.3. The results of the models are measured by a couple of well known machine learning performance metrics which are suitable for unbalanced data. The performance metrics are explained through their position in the confusion matrix (2x2) of binary classification problems. The positions in the matrix are filled with the following content:

$$\begin{bmatrix} \#TP & \#FP \\ \#FN & \#TN \end{bmatrix}$$

The abbreviations in the confusion matrix above have the following meaning:

- #TP: number of true positive correctly classified instances (in the model the instances which are not going to turnover);
- #FP: number of false positive wrong classified instances (in the model the instances which are going to turnover, but predicted as not going to turnover);
- #TN: number of true negative correctly classified instances (in the model the instances which are going to turnover);
- #FN: number of false negative wrong classified instances (in the model the instances which are not going to turnover, but predicted as going to turnover).

The first metric which is derived from the confusion matrix is the balanced accuracy which is a derivative of the total accuracy. The total accuracy is a measure that counts the total correct predictions as a percentage of the total instances. The balanced accuracy metric is more suitable for unbalanced data, because the total accuracy can give a misleading view for unbalanced data. For example if the majority class represents 98% of the cases and the model results show that all the predictions are assigned to the majority class then the total accuracy will be 98% correct classified and 2% false classified. Whereas the balanced accuracy measure will look separately into the true positive and true negative instances. The balanced accuracy (15) is defined as:

$$\frac{1}{2}\left(\frac{\#TP}{\#TP+\#FN}+\frac{\#TN}{\#TN+\#FP}\right) \tag{15}$$

The second performance metric is the Precision (16) which presents the proportion of predicted positives which are actual by positive:

$$\frac{\#TP}{\#TP+\#FP} \tag{16}$$

The third performance metric is the Recall (17) which presents the proportion of actual positives which are predicted positive:

$$\frac{\#TP}{\#TP + \#FN} \tag{17}$$

The fourth performance metric is the F1 measure. F1 is the harmonic mean of Precision and Recall. It is an aggregate measure where score 1 indicates perfect precision and recall and score 0 indicates poor Precision and Recall. F1 (18) is defined as:

$$\frac{2 * (Precision/Recall)}{Precision + Recall} \tag{18}$$

The last and fifth performance measure is the Area Under Curve (AUC) which represents the area under the Receiver Operating Characteristic curve (ROC) [2]. The ROC curve is a plot of true positive (TP) rate on the vertical axis against the true negative (TN) rate on the horizontal axis. A random model represents a diagonal line as ROC curve and has an AUC value of 0.5. This means the chance of classifying a record as TP is the same as classifying a record as TN. Therefore the higher the AUC value the better the classifier. An excellent classifier has an AUC value of 1.

6.2 Results models

The results of the models are given in Table 8 below. In this table three tree-based models are evaluated; decision tree, bagging tree and random forest. The models are evaluated with the performance metrics described in the previous section.

	Original sample
Balanced accuracy decision tree	0.5182
Balanced accuracy bagging tree	0.53
Balanced accuracy random forest	0.5934
Precision decision tree	0.9606
Precision accuracy bagging tree	0.9618
Precision random forest	0.9683
Recall decision tree	1
Recall bagging tree	0.9971
Recall random forest	0.9197
F1 decision tree	0.98
F1 bagging tree	0.9791
F1 random forest	0.9434
AUC decision tree	0.6123
AUC bagging tree	0.6598
AUC random forest	0.7119

Table 8: Performance metrics Balanced accuracy, Precision, Recall, F1 and AUC for the algorithms decision tree, bagging tree and random forest.

In Table 8 above the scores lead to the conclusion that the random forest algorithm has the best overall performance. In the original sample of the random forest the model has a high accuracy in predicting the people who are not going to turnover in the next period and a low accuracy in predicting the people who are going to turnover the next period. This is not surprising, the unbalanced data problem shows that algorithms without sampling techniques assign all the predictions to the majority class. The sampling techniques should probably solve this problem. Table 9 below shows the scores on the sampling techniques as explained in 5.1.

	Under	Over	SMOTE	ROSE
Balanced accuracy decision tree	0.6594	0.7727	0.6343	0.5639
Balanced accuracy bagging tree	0.6384	0.5253	0.6086	0.5593
Balanced accuracy random forest	0.632	0.5206	0.5936	0.5629
Precision decision tree	0.9774	0.9758	0.9726	0.9648
Precision bagging tree	0.9762	0.9614	0.9693	0.9642
Precision random forest	0.9754	0.9610	0.9684	0.9851
Recall decision tree	0.7018	0.7072	0.8043	0.9338
Recall accuracy bagging tree	0.6533	0.9876	0.8607	0.9642
Recall random forest	0.6825	0.9931	0.9197	0.1983
F1 decision tree	0.8170	0.82	0.8805	0.9491
F1 bagging tree	0.7827	0.9743	0.9118	0.9642
F1 random forest	0.8031	0.9768	0.9434	0.3301
AUC decision tree	0.6915	0.6722	0.6803	0.5641
AUC bagging tree	0.6988	0.6551	0.6944	0.656
AUC random forest	0.6916	0.7135	0.7093	0.6709

Table 9: Sampling method and their performance metrics Balanced accuracy, Precision, Recall, F1 and AUC for the algorithms decision tree, bagging tree and random forest.

The random forest has still the best overall performance with SMOTE as sampling technique. Precision (0.9684), Recall (0.9197) and F1 (0.9434) are quite high. The balanced accuracy (0.6343) is lower; too many cases are predicted as the minority class. This is the consequence of applying sampling techniques; it ensures more correct predictions on the minority class, but at the same time more predictions which will be classified as minority class and actually belong to the majority class. The AUC performance metric has the best scores on the original (0.7119) and SMOTE (0.7093) sampling techniques with random forest as algorithm.

The second best algorithm is the bagging tree and the worst algorithm is the decision tree. This is not surprising, because the bagging tree and random forest are advanced decision tree versions with bootstrapping techniques. The sampling techniques on the tree-based algorithms have quite varying results. Overall the performance metrics of SMOTE and under sampling techniques show the best results.

The analysis above is focused on an overall good performance of the metrics on the different algorithms. Another interpretation of the results is to focus on models which have quite high scores on specific metrics. For this research the rate of good predictions is an interesting measure. This means models which have a specific high accuracy in predicting the majority class or the other way around in predicting the minority class. For the majority class the original sample in Table 8 on random forest has the best performance with 92% correctly classified followed by the SMOTE sampling technique with 90% correctly classified. The minority class is best predicted through ROSE with the decision tree algorithm with 99% correctly classified, either in this model the false negative predictions are really high. The under sampling technique has the second best performance on predicting the minority class by using the decision tree algorithm, where 63% is correctly classified.

Figure 4 below gives an overview of the variable importance of the best performed random forest algorithm with the SMOTE sampling technique.

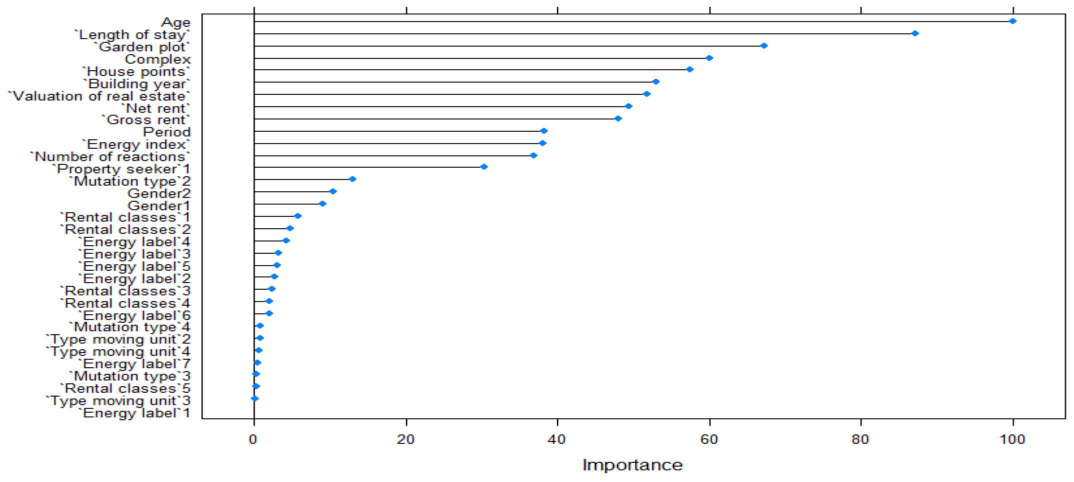

Figure 4: Variable importance.

Figure 4 above shows the following three variables as most important for the results of the model:

1. Age (100): information of the age of the tenant;
2. Length of stay (87): information about the length of stay of a current tenant;
3. Garden plot (67): information to recognize a rented house in the portfolio as unique over the time.

The above top three of features importance are as expected. The three variables contains a lot of useful information which will lead to predictable decisions on the rental turnover of tenants. For example age can influence the frequency mutation, because a reasonable thought could be that old tenants are not that flexible in moving to another location in contrary to student tenants. The length of stay can influence the rental turnover for example in case that most of the tenants move after a given number of years. Or the information that tenants do not move after a given number of many years. In the Chapter 3 the variables Age and Length of stay are further evaluated. The last important feature is the Garden plot. This feature could possibly give information on the frequency of relocation of a specific garden plot. Some houses for example student complexes have a higher rental turnover over the time, compared to complexes with senior citizens.

6.3 Results of reduced features models

The performance of the model given in Section 6.2 can be improved by feature selection. This means selecting the most important features out of the variable importance overview given in Figure 4. Reducing of redundant and misleading features in the model could possibly lead to:

- Reduction of over fitting: less misleading data could lead to less opportunity to make decisions based on noise;
- Improvement of accuracy: less misleading data could lead to improvement of modeling accuracy.

The features of the model given in Figure 4 with significantly low variable importance are filtered out of the model. The features with an importance lower than score 30 are reduced out of the model. This means that the following features are selected and applied to the reduced feature model:

- Age
- Length of stay
- Complex
- House points
- Building year
- Valuation of real estate
- Net rent
- Gross rent
- Period
- Energy index
- Number of reactions

All other features in Figure 4 have been deleted. The model with the reduced features is trained and tested on the same algorithms and first on the original sample with the results in Table 10.

	Original
Balanced accuracy decision tree	0.5030
Balanced accuracy bagging tree	0.5107
Balanced accuracy random forest	0.5095
Precision decision tree	0.9607
Precision accuracy bagging tree	0.9613
Precision random forest	0.9612
Recall decision tree	1
Recall bagging tree	0.9964
Recall random forest	0.9971
F1 decision tree	0.9799
F1 bagging tree	0.9785
F1 random forest	0.9788
AUC decision tree	0.6439
AUC bagging tree	0.6694
AUC random forest	0.7365

Table 10: Performance metrics on Balanced accuracy, Precision, Recall, F1 and AUC for the algorithms decision tree, bagging tree and random forest.

The scores in Table 10 shows the results of the tree-based algorithms models with the reduced features applied on the original sample. The reduced feature model does not show significantly better results compared to the full feature model. Table 12 gives an overview of the delta between the reduced and full feature models. First the results of the reduced feature model on the different sampling techniques. This leads to the following performances in Table 11:

	Under	Over	SMOTE	ROSE
Balanced accuracy decision tree	0.6463	0.6362	0.6456	0.5278
Balanced accuracy bagging tree	0.6293	0.5123	0.6012	0.539
Balanced accuracy random forest	0.6489	0.6489	0.512	0.6045
Precision decision tree	0.9783	0.9745	0.9783	0.9964
Precision accuracy bagging tree	0.9758	0.9615	0.9696	0.9981
Precision random forest	0.9612	0.9614	0.9613	0.9908
Recall decision tree	0.6347	0.7466	0.7823	0.0608
Recall bagging tree	0.6535	0.9867	0.8517	0.0816
Recall random forest	0.9971	0.9915	0.9971	0.1311
F1 decision tree	0.7699	0.8456	0.8684	0.1146
F1 bagging tree	0.7828	0.9739	0.9069	0.151
F1 random forest	0.9788	0.9762	0.9788	0.2316
AUC decision tree	0.6822	0.6337	0.6914	0.5713
AUC bagging tree	0.6876	0.6335	0.6838	0.5642
AUC random forest	0.7073	0.6952	0.7197	0.651

Table 11: Sampling techniques and their performance metrics on Balanced accuracy, Precision, Recall, F1 and AUC for the algorithms decision tree, bagging tree and random forest.

The random forest shows overall the best performance on the sampling techniques and slightly better than the full feature models. Especially techniques undersampling and SMOTE are showing slightly better results. Table 12 below gives the delta between: scores reduced feature model - scores full feature model.

Δ *Reduced model − full model*	Original	Under	Over	SMOTE	ROSE
Balanced accuracy decision tree	-0.02	-0.02	-0.014	0.01	-0.04
Balanced accuracy bagging tree	-0.02	-0.01	-0.01	-0.01	-0.02
Balanced accuracy random forest	-0.08	0.02	-0.01	0.01	-0.01
Precision decision tree	0	0	0	0.01	0.03
Precision bagging tree	0	0	0	0	0.03
Precision random forest	-0.01	-0.01	0	-0.01	0.01
Recall decision tree	0	-0.07	0.04	-0.02	-0.87
Recall bagging tree	0	0	0	-0.01	-0.88
Recall random forest	0.08	0.31	0	0.08	-0.07
F1 decision tree	0	-0.05	0.03	-0.01	-0.83
F1 bagging tree	0.06	0	0	0	-0.81
F1 random forest	0.04	0.18	0	0.04	-0.1
AUC decision tree	0.03	-0.01	-0.04	0.01	0.01
AUC bagging tree	0.01	-0.01	-0.02	-0.01	-0.09
AUC random forest	0.02	0.02	-0.02	0.01	-0.02

Table 12: Delta in performance metrics on Balanced accuracy, Precision, Recall, F1 and AUC for random forest algorithm rounded at two decimals of reduced feature model versus full feature model.

Table 12 above shows that the random forest reduced feature models performs better on the under sampling technique and slightly better on the SMOTE sampling techniques. Whereas the original, over and ROSE sampling techniques are not showing better results overall. In addition the random forest with under sampling technique performs way better on Recall (0.31). This means the under sampling method in this model is better in predicting the majority cases. The bagging tree and decision tree do overall perform worse compared to the full feature model. This is not surprising, because the selected features in the reduced features model are based on the measure variable importance of random forest algorithm.

In the random forest reduced feature models the rate of actual predictions has approximately remained the same for the majority class. For the minority class the under sampling technique with the random forest algorithm shows a significant improvement from 63% correctly classified to 67% correctly classified. ROSE sampling technique with the decision tree algorithm still shows the best way of predicting the minority class with 99%, whereas the predictions of the false classified instances in the majority class is very high.

7 Conclusion

The aim of this research was to investigate the prediction of the rental turnover rate of housing corporation Trivire. The reasons behind this research were in the first place to support future yield and investments calculations with an expected turnover rate and on a deeper level gain insights into houses and housing complexes and their turnover rates in the next period. Secondly, the research aim to gain insights into the factors that explain and affect the rental turnover rate.

The prediction of the rental turnover rate in general is a difficult task. The biggest challenge in this prediction was to handle high unbalanced data. For this unbalanced data problem variety of methods have been taken into account to challenge unbalanced data; using suitable tree-based algorithms, trying different sampling techniques and evaluating on non-misleading metrics. The performances on the original sample had high accuracy in predicting the people who are not going to turnover, but lower accuracy in predicting the people that are going to turnover. The highest overall performance metrics is achieved with the SMOTE sampling technique on the random forest algorithm. However, it turned out that this model had very high scores on predicting the majority class and lower scores (false negative instances) on predicting the minority class. Though the real world data is used in the models is very unbalanced and it is a difficult task to achieve a high balanced accuracy.

The research also aimed to improve the performance metrics with a reduced feature model. The model with reduced features showed an improvement in the performance metrics of the random forest with under sampling techniques and a slightly better performance on sampling technique SMOTE. Even the sampling technique does not however lead to the desirable results in overall performances. On the other hand based on the best overall performance model it is possible to make predictions of the rental turnover rate, otherwise this is subject to expert judgement. This use in practice will be discussed in the next Chapter 8.

In addition there are a couple of specific performances which could be very helpful to gain insights on a deeper level into houses and housing complexes. The housing corporation is interested in specific houses and housing complexes that will become vacant or not. In this case it would be better to choose a model with a specific high performance on correctly classified minority or majority classes. For example the original sample with random forest algorithm has zero false negative observations. Another option is to choose ROSE sampling technique which has far less false positive observations.

Finally the data analysis and variable importance showed that age of the current tenants is the most important factor in predicting rental turnover rate. In addition the length that a current tenant stays in a rental house is the most important factor followed by the garden plot.

8 Discussion

In this section the discussion will be evaluated and divided in two parts. The first one considers the practical use of the predicting model. The second part gives some recommendations which should be involved within further research and support the unbalanced data problem to improve the performance.

In practice the model could be used with a correction on the standard false predictions. This means the model is trained and tested on more than 100.000 data points. For example if the model predicts 6% rental turnover rate and the realization is 4% then a standard error correction on the model could be made. In this example the model has higher estimates and so for the intuitive prediction the correction will be made on the results of the model. In this way the model could be used in practice with the use of expert judgement. A recommendation within further research is to scientifically prove the correction on the model. Therefore the expert judgement is not involved in the research part of this paper. Another way of practical use based on expert judgement is to combine the results of a prediction model which has specific good performances on correctly classified minority classes with a prediction model which has specific good performances on correctly classified majority classes.

Finally the following recommendations specially focuses on the unbalanced data problem:

1. The model can be further improved by trying other types of algorithms. At this moment only tree-based algorithms are used. There are various other types of algorithms which may perform better on unbalanced data.
2. Currently the model is modelled with half yearly rental turnover rate data, with as underlying reason that there was a lot of portfolio data available over the last three years every. If there is for example six years of historical data available then the model will perform better on yearly rental turnover rate, because the yearly mutation degree is slightly more balanced.
3. The model can be further elaborated when there is a larger data set over time available. For example with 10 years of data it could be possible to also involve times dependency models.
4. Adding more informative features to the data set. The performance will probably improve if some features were added, because more valuable features could lead to a better performance.

References

[1] Average.

[2] Receiver operating characteristic.

[3] Standard deviation.

[4] Christopher M. Bishop. *Pattern recognition and machine learning.* Springer Science+Business Media, 33, 2006.

[5] Peter Flach. *Machine Learning: The art and science of algorithms that make sense of data.* Cambridge University Press, 332, 2012.

[6] Peter Flach. *Machine Learning: The art and science of algorithms that make sense of data.* Cambridge University Press, 333, 2012.

[7] Max Kuhn. Package 'caret'. (1), 2017.

[8] Charles J. Stone R.A. Olshen Leo Breiman, Jerome Friedman. *Classification and Regression Trees.* Taylor Francis, 1984.

[9] Nicola Torelli Nicola Lunardon, Giovanna Menardi. Rose: A package for binary imbalanced learning'. 2014.

[10] Lawrence O. Hall W. Philip Kegelmeyer Nitesh V. Chawla, Kevin W. Bowyer. Smote: Synthetic minority over-sampling technique'. 2002.

[11] Foster Provost. Machine learning from imbalanced data sets 101.

[12] Akash Sharma. Basic of greedy algorithms.

[13] Yisehac Yohannes and Patrick Webb. Classification and regression trees, cart. pages 14–17, 1999.