Predictive Machine Learning for a Housing Corporation

Using machine learning and subgroup discovery to identify tenants that are more likely to cause payment problems

Author: An Li

Student ID: 5646219

Internal supervisor: Dr. A.J. Feelders

External supervisor: Guus van de Mond

A mater thesis presented for the degree of Master of Science in Computing Science

Algorithmic Data Analysis Group

Department of Information and Computing Science

Utrecht University

The Netherlands

12-11-2017

Universiteit Utrecht

Abstract

In the process of renting a house, payment arrears may happen to some tenants. Normally, the housing corporation can only take actions after the problems occurred. In this thesis, several machine learning and subgroup discovery algorithms are used to detect in advance people who are more likely to cause payment problems. The chosen machine leaning algorithms include logistic regression, random forests, k nearest neighbors, naive bayes and neural networks using model averaging, while the PRIM algorithm is selected for subgroup discovery. Because the skewed distribution of classes in datasets, we utilize the synthetic minority over-sampling technique (SMOTE) to generate more reasonable results. Additionally, feature selection and several ensemble methods are leveraged as well to improve the model performance, such as averaging, majority voting and stacking. By all these approaches, finally, we are able to get a few models that are significantly better than the preliminary one. However, since the available data is limited and incomplete, and important time-based information is missing, we can't obtain a model which is good enough.

Key words: machine learning, subgroup discovery, SMOTE, ensemble, payment problems, housing corporation

Contents

1 Introduction

1.1 Motivation

This thesis presents the results of a data mining project performed at the Machine Learning Company where the author worked as an intern. First, we introduce the customer, namely, the housing corporation we worked for. Then we explain the reason why it is important for the housing corporation to investigate the chosen problem. Because the thesis is the graduation research of a master degree, the scientific motivation will also be given.

1.1.1 The housing corporation

The housing corporation is located in the Netherlands and concentrates especially on two municipalities within the North Brabant province. They have nearly 5000 rented houses and public property such as health centers and care homes. They also rent a lot of garages and parking spaces for organizations and individuals. Generally speaking, the following services are mainly provided by the housing corporation:

1. Renting houses, public buildings, garages and parking places;
2. Managing and maintaining houses, public buildings, garages and parking places;
3. Purchasing real estate;
4. Researching on the livability of neighborhoods.

With approximately 37000 people recorded in the database, the potential of conducting machine leaning or data mining on the housing corporation seems to be promising. If we can use all the profiles and records of these customers, there might be a number of opportunities to generate interesting conclusions. However, it depends on the databases significantly, for instance, what kind of features are there, how many missing records do they have and is there any pollution in the data? In practice, we must take all these circumstances into consideration in order to yield reliable and meaningful results. Besides, the housing corporation can also provide us the complete financial database, in which we can find all the payment transactions of both individual level and company level.

1.1.2 Business motivation

In recent years, with the popularity of machine learning, many traditional companies are seeking such intelligent techniques to enhance their business competitiveness. This is one of the reasons why the housing corporation would like to explore how machine learning can be applied to help them. It is undoubted that a company will dominate the market with the best services and the newest technologies, such as Google in search engine and Amazon in e-commerce. If they don't follow the trend of the development of the technologies, they will finally be eliminated in the future because of the out of date services and ideas. According to [1, 2, 3, 4, 5],

we can find that machine learning has been prevalent in various fields of industries today, and more and more companies have already benefited from it. In a word, integrating machine learning with business will not only save investments, but also make enormous profits.

Additionally, the housing corporation intends to make their employees work more efficiently by getting rid of unnecessary manual work. Meanwhile, most complex operations are normally extremely time consuming when executed by humans, and most of them require, more or less, some predictive abilities. Here we take a small example to explain it more clearly. If the weather condition is awful on a certain day, it might lead to an increase of service requests submitted by tenants, such as leaking roofs and broken windows. Thus, it is wise to assign more employees during these bad weather days to react for larger amount of service requests. By this means, the housing corporation can arrange their manpower more efficiently. However, they can't make a reasonable plan to allocate the manpower yet, because they don't know exactly how many service requests will appear on a day based on the weather condition. But if we apply machine learning on this problem, and build a simple regression model to predict the potential service requests on a future day based on some appropriate features, then, the housing corporation can use the predicted number of service requests to adjust their arrangement in advance. Therefore, machine learning seems to be a good choice to meet the demands of the housing corporation.

1.1.3 Scientific motivation

From scientific perspective, we would like to go through the entire process of managing a machine learning project and gain hands-on experience of data preparation, data analysis, data cleaning, model training, validation and parameter tuning. As far as we know, in the academic field, researchers and students always use well-edited, well-structured and uniform distributed datasets to practice. Nevertheless, in reality, we will probably have to deal with completely raw data or extremely unbalanced data. Sometimes, we even need to manipulate datasets with quite limited features. In some sense, real world problems can help people acquire more experience and improve their engineering abilities. Hence, it is worthwhile to apply what we have learned from the university to real world problems.

Moreover, we will also compare the results generated by different machine learning algorithms to summarize useful experience in practice. Commonly, people can't decide which algorithm is the best, because every algorithm may have a chance to outperform others in some specific situations. So it is recommended to train as many models as time and computing power permits. In this way, we are able to find the algorithm that suits the particular case best.

1.2 Problem statement

A variety of IT technologies have been applied by the housing corporation to improve their service quality and make their employees more efficient, such as CRM systems which record profiles of tenants, self-service portals for submitting service requests and complaints, back-office systems to provide automated invoicing. But only these tools are not enough yet, there are still many scenarios that involve time-consuming labor, low efficiency of resource arrangement, low customer satisfaction and so on.

In this thesis, we choose to present a case that predicts which people are more likely to cause payment problems if they rent a house from the housing corporation. Payment plays a significant role in a rent contract, it is also directly related to the profits of the housing corporation. If every tenant will pay their rent or service cost on time, it will save a lot of extra expenses on collecting the overdue bill. In other words, the housing corporation doesn't have to arrange employees to investigate the unpaid tenants and make additional records to follow these unexpected events if no payment problem occurs. Payment problems will happen now and then, but the housing corporation can't take any early actions before the problems appears.

On the basis of the above evidence, the main goal of the thesis is unambiguous now, that is to say, based on the historical data, we are going to predict which tenants or find which kind of group of people are more likely to have payment arrears so that preventive actions can be assigned and proper attention to the right tenants can be given. Intuitively, the following two categories of data are important for us to start some preliminary analysis.

- Tenants' profiles which has as rich information as possible
- Payment transactions that are made on individual level

Another important point is about the label. In fact, there aren't actual records that label the tenants with having payment problems or not. Thus, the first step for us is figuring out how we can extract valuable information which can help us label the tenants with 1 or 0 (1 means someone had payments problems before while 0 means no payment problems). Once we have the labeled data, we can execute the machine learning phase and the subgroup discovery phase. In figure 1, we create an overview of the proposed methods to solve the entire problem. (In section 4, We will introduce all the data sources we have used in this figure with more details.)

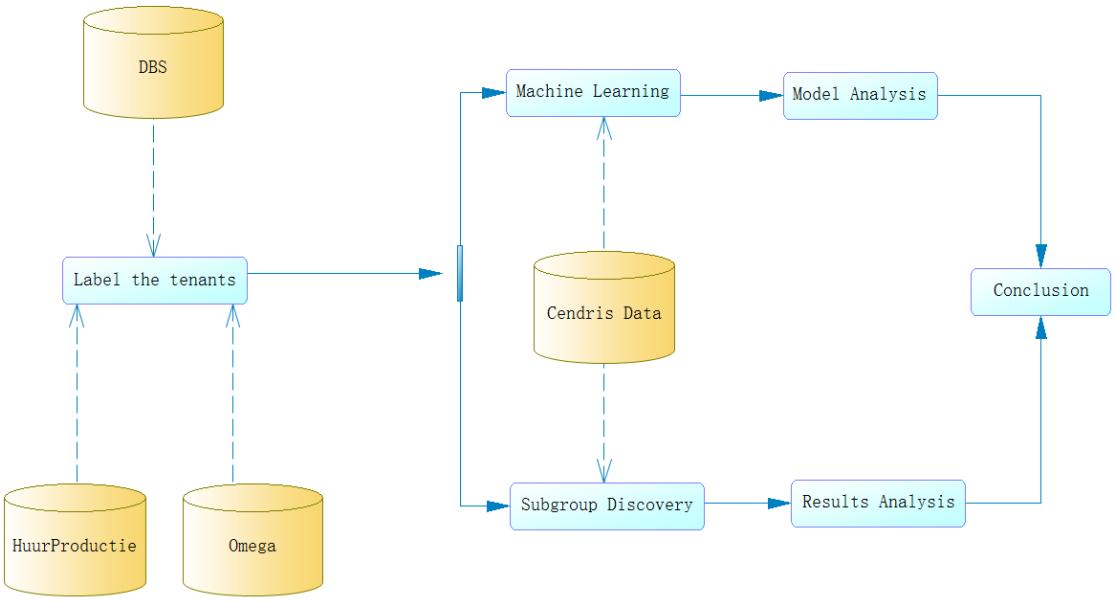

Figure 1: Overview of the proposed methods

1.3 Challenges

Unlike classifying the categories of flowers or predicting if tomorrow will be a rainy day, it is always more difficult for computers to capture humans' behaviors. Therefore, numerous challenges exist from both the business side and the technical view. Before we start the main work, it is considerable for us to think about the limitation of this project and the underlying difficulties we will face. By this way, we are able to set up a reasonable baseline to refer to when the actual results come out during the experiments.

1.3.1 Business perspective

The housing corporation are specialized at sales and marketing but not data and technology, thus, it will always be difficult for them to provide us the expected datasets. Meanwhile, they are not familiar with machine learning and subgroup discovery techniques, which sometimes leads them to overestimate the abilities of these methods. Overall, it is crucial how we introduce machine learning and subgroup discovery algorithms to them as well as how we interpret the results of models or analysis in the end. We can foresee that there might be a huge gap between the housing corporation and us with respect to the understanding of technologies, but from a different perspective, it will also push us to make our work understandable by our customers. Moreover, we can't guarantee that we will get flawless models, because it depends on the scale and quality of the datasets the housing corporation has. In fact, if we can get more data with better quality, we will be more confident to produce better results and more accurate models.

1.3.2 Technical perspective

In real world, we will face various kinds of datasets, some datasets might have lots of missing values while others might be polluted when people or programs create them. In this project, there is no doubt that we will encounter such problems as well when we manipulate the datasets before building models. Particularly, when it comes to predicting if a tenant will cause payment problems or not, it is common that we are going to manage an unbalanced dataset. Because people have such problems are always in a minority class in real world so the biggest challenge for us is how we can tackle the unbalance of the dataset in a most appropriate way. But beyond that, we should also put effort into algorithm selection and parameter tuning in the following steps after manipulating the data. Hence, the other challenge is how we select algorithms and the parameters for each model. Additionally, it is possible that the information we extract from the original datasets is not enough so that we need to find extra data sources to enrich the original one. The question is where we can find such data sources if we need them and how we can make connections among different data sources. For example, the tenants' profiles are not complete in the databases and we have to find several important features such as income and family structure from external sources.

1.4 Research questions

The motivations, problem statement and challenges discussed in the previous sections lead us to the following research questions:

1. *Is it possible to build a classification model to predict if a tenant will cause payment problems or not based mainly on the tenant's profile?*

2. *Is it possible to use subgroup discovery algorithm to find which kind of people are more likely to cause payment problems and then build a predictive model based on the most representative rules?*

3. *Which supervised machine learning algorithm performs best with regard to this specific problem?*

4. *How can we interpret the results of both machine learning and subgroup discovery?*

The first question covers the main topic of this project, and this is basically what the housing corporation requires. In consideration of the potential difficulties to handle unbalanced data, we come up with the second research question to help us find more descriptive results. The third question is associated with the first one, which aims to generate models with the best prediction performance. The last research question proposes to investigate the interpretability of machine learning and subgroup discovery as well as compare the results produced by both of them with respect to this specific problem.

1.5 Outline

The structure of the thesis is as follows: in chapter 2, we will firstly introduce some background knowledge of artificial intelligence, machine learning as well as subgroup discovery, at the same time, we will illustrate some related work in these fields. In chapter 3, we discuss all the algorithms we have used in the experiments, and make sure we explain all the techniques clearly so that people can understand the main idea of each algorithm without any deeper reading. Then, we will describe all the available data sources in chapter 4 and give detailed statistic analysis for them, meanwhile, we will also indicate the challenges and issues with respect to the scale and quality of the data. After, we will conduct our experiments in chapter 5, and analyze the results in chapter 6. Finally, the summary and some future work will be presented in chapter 7.

2 Background

Artificial Intelligence(AI) is becoming more and more prominent and has been applied in many areas all over the world, for instance, chat bot like Siri, self-driving car of Baidu and also the famous I-go player AlphaGo from Google. In the meantime, the other term "Machine Learning" is also mentioned frequently along with AI. In 1959, Arthur Samuel firstly defined machine learning as "provides computers with the ability to learn without being explicitly programmed." [6]. Basically, machine learning is an approach to achieve AI and the process of machine learning is as similar as data mining, that is to say, both fields look for patterns from the data. Nevertheless, in data mining, we mainly concentrate on patterns that people can easily understand whilst machine learning can find patterns with deeper insights which might be more difficult for human-beings to detect manually. People often divide machine learning algorithms into supervised algorithms and unsupervised algorithms. Supervised machine learning algorithms require labeled datasets to create models while unsupervised machine learning algorithms are more suitable for dealing with unlabeled datasets [7].

For supervised algorithms, data has labels like True/False, Spam/Not Spam or continuous real number. Based on some well defined features, supervised algorithms can build a predictive model on training dataset and then predict the label of each instance in test dataset. In order to obtain higher accuracy and better performance, several tuning techniques should be applied along with the machine learning algorithms, such as cross-validation. Cross-Validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model. In typical cross-validation, the training and validation sets must cross-over in successive rounds such that each data point has a chance of being validated against [8]. The most basic form of cross validation is K fold cross validation and repeated K fold cross validation is another popular variation.

Furthermore, supervised learning can be divided into regression problems and classification problems. In regression problems, the output takes continuous values, whereas in classification problems, the output must be categorical values. The most representative examples should be linear regression [9] and logistic regression [10]. The former one dedicates to solve regression problems while the latter one is designated to tackle classification problems (the name also contains "regression", but it is because of some historical reasons and it is essentially a classification algorithm). Nevertheless, many widely used algorithms can cope with both regression and classification problems so there is no need to worry about the selection of algorithms too much.

As mentioned before, the training data of unsupervised learning should be unlabeled and the algorithm aims to draw inferences from datasets. The most common unsupervised learning scenario is clustering, which is used for exploratory data analysis to find hidden patterns or classify data into different unknown subgroups. Similarity measures are used to construct clusters, namely, calculating distance between instances. Normally, distance metrics like Euclidean distance and Manhattan dis-

tance are widely used. Another significant type of unsupervised learning is principal components analysis, which is a tool used for data visualization or data preprocessing before supervised techniques are applied. Some well known algorithms are PCA (which is invented in 1901 by Karl Pearson [11] and then independently developed and named by Harold Hotelling [12] in the 1930s) as well as K-means clustering which was firstly introduced by James MacQueen [13].

On the basis of supervised learning, ensemble algorithms [14] are proposed to maximize the accuracy of the model. The underlying principle of ensemble learning is utilizing the strength of multiple models to overcome the weakness of a single model. In real world problems, every model built by a single algorithm has bottlenecks and will probably make mistakes. Given that each model has these "limitations," an ensemble method will result in the best possible overall predictions. Many theoretical and empirical researches have shown that the accuracy of an ensemble model can dramatically outperform a single model. There are a lot of popular algorithms adopt the idea of ensemble learning, for example, Random Forests [15], Adaboost [16] and Gradient Boosting Machines (GBM) [17].

Subgroup discovery is a data mining technique which extracts interesting rules with respect to a target variable. An important characteristic of this task is the combination of predictive and descriptive induction [18]. Similar to supervised learning, subgroup discovery can also be classified based on the type of the target variable. Normally, there are three types of target variables in subgroup discovery: binary, nominal and numeric.

- Binary analysis. The target variable has only two distinct values (positive or negative), and the goal is finding interesting subgroups for each target value.
- Nominal analysis. The methodology for this kind of analysis is similar to the binary case, that is to say, to find subgroups for each possible value, but the target variable can take an arbitrary number of values.
- Numeric analysis. This is the most complex one because the variable can be studied in different ways such as dividing the variable in two ranges with respect to the average.

Another very important component of subgroup discovery is the search strategy. The dimensions of the search space will increase exponentially with respect to the number of features and target values. Thus, a better search strategy can yield a more efficient subgroup discovery algorithm, which is crucial for handling massive datasets. Different strategies have been used so far, for example, beam search based algorithms, exhaustive search based algorithms, genetic algorithm based approaches [19].

3 Theory

In this chapter, we introduce all the theories behind the techniques we have used in experiments. The outline of several machine learning algorithms will be presented at the first place. Then the introduction of the selected subgroup discovery algorithm will be given. Subsequently, the sketch of the ensemble methods will be described. At last, we discuss some popular measures in relation to classification problems.

3.1 Machine learning

Nowadays, numerous machine learning algorithms have been invented and implemented for people to use. However, it is difficult to find the best one for a concrete problem directly. In order to get the best results, the only way is trying as many algorithms as possible. But this doesn't mean we can pick algorithms randomly, on the contrary, we should try the most popular algorithms in the community first. This is the first reason why we choose the following algorithms. The second reason is that we choose algorithms with diverse properties to make sure the subsequent ensemble methods can give us better performance.

3.1.1 Lasso and elastic-net regularized linear models

In order to fit a linear model, we use "glmnet" package in R. The "glmnet" is a package that fits a generalized linear model via penalized maximum likelihood. Normal linear models formulate a linear relationship between a response and one or more predictors. Nevertheless, sometimes, a nonlinear relationship exists. Nonlinear regression or classification describes general nonlinear models. A special class of nonlinear models, called generalized linear models [20], which is a flexible generalization of ordinary linear models that allows for response variables that have error distribution models other than a normal distribution.

Lasso (1) is a regularization technique for estimating generalized linear models. Lasso uses L1 penalty term that constrains the size of the estimated coefficients while another technique ridge (2) regularization adopts L2 norm. Unlike ridge regression, as the penalty term increases, the lasso technique sets more coefficients to zero. This means that the lasso estimator results in a smaller model, with fewer predictors which makes the final model more interpretable. As such, lasso is an alternative to stepwise regression and other model selection and dimensionality reduction techniques. Elastic net is a related technique as well. Elastic net (3) is something similar to a hybrid of ridge regularization and lasso regularization. As with lasso, elastic net can generate reduced models by generating zero-valued coefficients. Empirical studies suggest that the elastic net technique can outperform lasso on data with highly correlated predictors.

$$\hat{\beta} = argmin_{\beta} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} |\beta_j| \tag{1}$$

$$\hat{\beta} = argmin_{\beta} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{2}$$

$$\hat{\beta} = argmin_{\beta_0,\beta} \sum_{i=1}^{n} w_i l(y_i, \beta_0 + \beta^T x_i) + \lambda[(1-\alpha)||\beta||_2^2/2 + \alpha||\beta||_1] \tag{3}$$

In the above three equations, y represents the dependent variable (response or output variable) while x stands for independent variable (predictor or input variable). Equation (2) presents how to calculate the coefficient estimates of ridge regularization, where $\lambda \geq 0$ is a tuning parameter and the term $\lambda \sum_{j=1}^{p} \beta_j^2$, called a shrinkage penalty which can be very small when $\beta_1, ..., \beta_p$ are close to zero. The tuning parameter λ serves to control the relative impact of these two terms on the coefficient estimates. When $\lambda = 0$, the penalty term has no effect, and ridge regularization will produce the least squares estimates. However, as $\lambda \to \infty$, the impact of the shrinkage penalty grows, and the ridge regularization coefficient estimates will approach zero, but will not set any of them exactly zero (unless $\lambda = \infty$). Unlike least squares, which generates only one set of coefficient estimates, ridge regularization will produce a different set of coefficient estimates, $\hat{\beta}$, for each value of λ. Therefore, selecting a good value for λ is critical. Ridge regularization does have one disadvantage. Because L2 norm is used, it will include all p predictors in the end which may affect the interpretation of the final model.

Comparing equation (2) and equation (1), we see that the only difference between ridge and lasso is that the β_j^2 term in the ridge regularization has been replaced by $|\beta_j|$, which is called L1 norm. As with the ridge regularization, the lasso shrinks the coefficient estimates towards zero. However, in the case of the lasso, the L1 norm has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is large enough. Hence, much like best subset selection, the lasso performs variable selection. As a result, models generated from the lasso are generally much easier to interpret than those produced by ridge. We say that the lasso regularization yields sparse models that involve only a subset of the variables. As in the ridge regularization, selecting a good value of λ for the lasso regularization is also critical.

Equation (3) shows us the elastic net regularization, where $l(y_i, \beta_0 + \beta^T x_i)$ is the negative log-likelihood contribution for observation i. The elastic-net penalty is controlled by α, and bridges the gap between lasso ($\alpha = 1$, the default) and ridge ($\alpha = 0$). The tuning parameter λ controls the overall strength of the penalty. Here, $||\beta||_2^2$ means L2 norm and $||\beta||_1$ is the L1 norm.

Overall, advantages of "glmnet" are:

- The algorithm is extremely fast
- It can exploit sparsity in the input matrix

- It can fit linear, logistic and multinomial, poisson, and cox regression models
- A variety of predictions can be made from the fitted models
- It can also fit multi-response linear regression
- It includes internal feature selection

According to "glmnet" documentation, the regularization path is computed for the lasso or elastic net penalty at a grid of values for the regularization parameter λ. Besides, we also need to tune the α parameter to control elastic net penalty.

3.1.2 Random forests

We choose the package "randomForest" to fit tree based model. Random forests[15] is an ensemble learning algorithm. The main idea of this algorithm is building several weak decision trees with fewer features and then combine these trees to form a single, stronger learner by averaging or majority voting. In practice, random forests has been proven to be one of the most accurate machine learning algorithms. In algorithm 1, we illustrate the pseudocode of random forests.

Algorithm 1 Random Forests

Require: A training set S := (x_1, y_1),...,(x_n, y_n), features X, and number of trees in forest B.

function RandomForests(S, X)
  $H \leftarrow \emptyset$
  for $i \in 1, ..., B$ **do**
    $S^{(i)} \leftarrow$ A bootstrap sample from S
    $h_i \leftarrow RandomizedTreeLearn(S^{(i)}, X)$
    $H \leftarrow H \cup h_i$
  end for
  return H
end function
function RandomizedTreeLearn(S, X)
  for each node **do**
    $f \leftarrow$ Random select a subset of X
    Split on best feature in x
  end for
  return The learned tree
end function

Random forests algorithm really has a lot of superiorities and also a few drawbacks, for example:

- Pros:
  - It is dominant in accuracy among current algorithms
  - It is very efficient on large data sets
  - It can decide the best subset of features

  - It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing
  - It generates an internal unbiased estimate of the generalization error as the forests building progresses

- Cons:
  - Random forests have been observed to overfitting for some datasets with noisy classification/regression tasks
  - The results made by random forests are difficult for humans to interpret
  - For data including categorical variables with different number of levels, random forests are biased in favor of those attributes with more levels

3.1.3 Naive bayes

In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with naive independence assumptions between the features. The Naive Bayes algorithm is called "naive" because it makes the assumption that the occurrence of a certain feature is independent of the occurrence of other features. When it comes to the "Bayes" part, it is because of Thomas Bayes and the theorem named after him, Bayes' theorem, which is the base for Naive Bayes Algorithm. More formally, Bayes' Theorem is formulated as the following equation:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{4}$$

In this equation, P(A|B) (or P(B|A)) means the posterior probability (conditional probability) of occurrence for event A (or B) given B (or A) is true, P(A) and P(B) are prior probabilities of the occurrence for event A and B respectively.

The sketch of the naive bayes learning algorithm is as follows:

1. Build a frequency table for all the features against different classes
2. Create the likelihood table for the features against the classes
3. Calculate the conditional probabilities for all the classes
4. A new instance will belong to the class which has the maximum conditional probability

Naive bayes classifier has the following properties:

- Pros:
  - It is a relatively easy algorithm to build and interpret
  - It is faster to predict classes using this algorithm than many other classification algorithms

  - It can be easily trained using a small dataset

- Cons:

  - One limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we can find such completely independent variables

3.1.4 K nearest neighbors

In machine learning, K nearest neighbors algorithm (KNN) [21] is a non-parametric method used for classification and regression. Given a positive integer K and a test observation x_0, the KNN classifier first identifies the K points in the training data that are closest to x_0, represented by $\mathcal{N}_0$. Algorithm 2 shows the procedure to find the nearest neighbors. It then estimates the conditional probability for class j as the fraction of points in $\mathcal{N}_0$ whose response values equal j, see equation (5):

Algorithm 2 K nearest neighbors

Require: A training set X, class labels Y of X, x: unknown sample, K: an integer

1: **for** $i = 1$ to m **do**
2: Compute distance $d(X_i, x)$
3: **end for**
4: Compute set I containing indices for the K smallest distances $d(X_i, x)$

$$\hat{Pr}(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j) \tag{5}$$

Finally, KNN applies Bayes rule (4) and classifies the test observation x_0 to the class with the largest probability [7], see equation (6).

$$\hat{y} = argmax_y Pr(y|X) \tag{6}$$

Though KNN is a very simple algorithm, it can often produce surprisingly good results. A summary of the benefits and drawbacks of KNN is as follows:

- Pros:

  - It can be regarded as one of the most simplest machine learning algorithm
  - The decision boundary of KNN can take any form
  - Performance is promising with enough representative data

- Cons:

- The main disadvantage of the KNN algorithm is that it is a lazy learner, i.e. the function is only approximated locally and all computation is deferred until classification
- The algorithm must compute the distance and sort all the training data at each prediction, which can be slow if there are a large number of training examples
- It does not learn anything from the training data, which can result in the algorithm not generalizing well and also not being robust to noisy data
- Need to tune K, because it can affect the results significantly

3.1.5 Neural networks using model averaging

Following Ripley [22], the same neural network model is fitted using different random number seeds. All the resulting models are used for prediction. For regression, the output from each network are averaged. For classification, the model scores are first averaged, then translated to predicted classes. Bagging can also be used to create the models.

Hereby, we won't discuss too much about the concept of neural network, because it will go beyond our scope, but we will summarize some pros and cons of neural network and averaging neural networks according to [23]:

- Pros:
  - The performance of averaging neural networks can be better than single neural network
  - Neural network models require less formal statistical training to develop
  - Neural network models can implicitly detect complex nonlinear relationships between independent and dependent variables
  - Neural network models have the ability to detect all possible interactions between predictor variables
  - Neural network can be developed using multiple different training algorithms
- Cons:
  - The training time can be extremely high compared to other algorithms
  - Neural network is a "black box" and sometimes it will be hard to interpret the results
  - Neural network models are prone to overfitting
  - Neural network model development is empirical, and many methodological issues remain to be resolved

3.2 Subgroup discovery

The notion of subgroup discovery has been defined by Klosgen [24] and Wrobel [25] as: *"Given a population of individuals and a property of those individuals that we are interested in, find population subgroups that are statistically 'most interesting', for example, are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest."*

Subgroup discovery focuses on extracting relations among different variables with respect to the property of interest. These relations can be represented by rules as follows: *if: A and B, then: C*, i.e. *if: Conditions, then: Target.*

We choose the PRIM [26] algorithm as an implementation of subgroup discovery. The PRIM algorithm uses bump hunting to find interesting subsets. Generally, the goal of bump hunting is to find regions in the input (attribute/feature) space with relatively high (low) values for the target variable. The regions are described by simple rules like *if: {condition-1* $\vee ... \vee$ *condition-n}, then: estimated target value.* Given the data (or a subset of the data), it is expected to produce a box B within which the target mean is as large as possible.

In practice, finding such regions is valuable for a variety of problems. Especially, these problems that require a decision maker to select the values of the input variables in order to optimize the value of the target variable. Besides, in bump hunting it always follows a so-called covering strategy. This means that the same box construction (rule induction) algorithm is applied sequentially to the subsets of the data.

3.3 Ensemble methods

Generally speaking, ensemble is a technique of combining two or more algorithms in similar or different types as base learners in order to construct a integrated model. By this way, a more powerful and robust model can be made which utilizes all the prediction results from different base learners. The aim of ensemble is improving the performance of a number of weak learners by putting them together. The rest of this section will demonstrate some popular approaches to build ensemble models.

3.3.1 Averaging and weighted averaging

Averaging ensemble is defined as taking the average of predictions from different models in case of regression problem or predicting probabilities for the classification problem. Moreover, in weighted averaging, different weights are assigned to predictions from multiple models and then we can use the weighted average to produce more reasonable results. Table 1 and 2 are examples for averaging ensemble and weighted averaging ensemble respectively.

Model 1	Model 2	Model 3	Model final
50	100	150	100

Table 1: Averaging ensemble

Model	Weight	Prediction
Model 1	0.4	50
Model 2	0.3	100
Model 3	0.3	150
Model final	-	95

Table 2: Weighted averaging ensemble

3.3.2 Majority voting

In a classification problem, majority voting is defined as taking the class with the maximum appearance as the final prediction from multiple models' outcomes, see table 3 for an example.//

Model 1	Model 2	Model 3	Model final
A	B	A	A

Table 3: Majority voting ensemble

3.3.3 Stacking

In stacking ensemble, multiple layers of machine learning models are placed one over another where each model passes their predictions to the model in the layer above it and the top layer model takes decisions based on the outputs of the models in the layer below it.

For instance, in figure 2, the stacking model has two layers, the bottom layer models (Logistic regression, KNN, naive bayes) receive the original features from the dataset and make predictions respectively. Subsequently, the model at the top layer (random forests) takes the output from the bottom layer models and generates final predictions.

Particularly, we should notice that the out of fold predictions are used when we create the inputs for the top layer. In the above example, the stacking model consists of two layers, but in practice, people can use any number of layers and algorithms to create stacking ensemble model. Besides, we should make sure that the predictions from different models are not correlated with each other, that is to say, it is recommended to select models built by different algorithms.

Below, we conclude some advantages and disadvantages of ensemble methods:

- Pros:
  - Ensemble is a proven method for improving the accuracy of the model and works in most of the cases

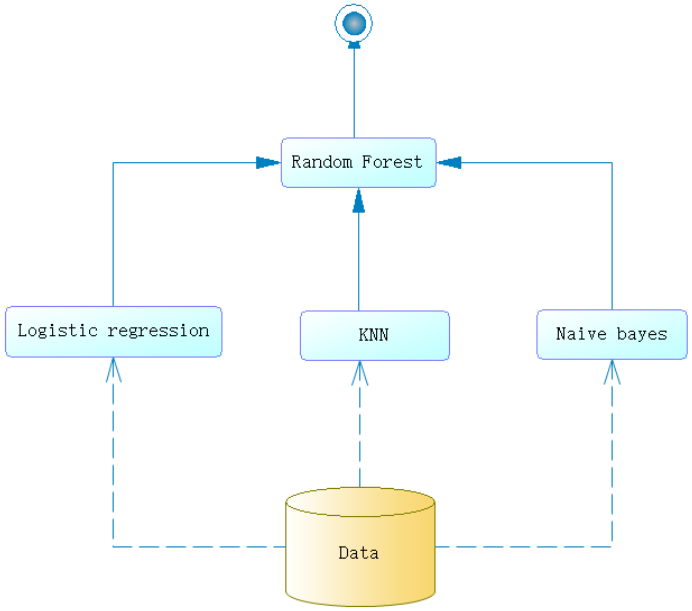

Figure 2: Stacking ensemble

  - Ensemble makes the model more robust and stable

- Cons:
  - Ensemble reduces the interpretation of the model and makes it very difficult to draw any crucial business insights at the end
  - Ensemble learning is time-consuming and thus it might not be a good choice for real-time applications
  - The selection of algorithms for creating an ensemble model is really hard to master

3.4 Measures

Performance measures are ways to evaluate machine leaning models and different machine learning problems require different measures. For instance, regression problems and classification problems will use totally different performance measures. Since we are dealing with a classification problem, we only concentrate on measures for classification problems.

		Actual class	
		Positive	Negative
Predicted class	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Table 4: Classification confusion matrix

Table 4 is a confusion matrix of measures for classification problems. From this table, we can define some representative measures for classification models. To begin

with, we explain the meaning of terms in this table. The first term "True Positive (TP)" means that actual class and predicted class are both positive while "False Negative (FN)" indicates that the actual class should be positive but the predicted class is negative. Then, "True Negative (TN)" and "False Positive (FP)" can be defined in the same way. With these four basic measures, we can formulate a lot of more sophisticated measures below.

- **Accuracy** $= \frac{\mathbf{TP+TN}}{\mathbf{TP+TN+FP+FN}}$: accuracy is the most intuitive performance measure and it is simply the ratio of correctly predicted observation to the total observations. Nevertheless, accuracy requires balanced data, otherwise a high accuracy doesn't mean anything, because in an unbalanced dataset, a high accuracy model will be probably biased to the majority class severely.

- **Precision** $= \frac{\mathbf{TP}}{\mathbf{TP+FP}}$: precision is the ratio of correctly predicted positive observations to the total predicted positive observations. If we are interested in the number of actual positive observations in the predicted positive observations, precision might be the best choice.

- **Recall** $= \frac{\mathbf{TP}}{\mathbf{TP+FN}}$: recall is the ratio of correctly predicted positive observations to all the observations in positive class. For example, if we would like to classify people who has cancer, recall is an extremely crucial measure to evaluate the model. Because if we use a low recall model, it means that most people who has cancer will not be detected successfully by the model.

- $\mathbf{F1} = \mathbf{2} * \frac{\mathbf{Precision*Recall}}{\mathbf{Precision+Recall}}$: F1 Score is the harmonic average of precision and recall. Therefore, F1 score takes both false positive and false negative into consideration.

- **ROC**: receiver operating characteristic curve (ROC curve) is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate (recall) against the false positive rate ($\frac{\mathbf{FP}}{\mathbf{FP+TN}}$) at various threshold settings.

- **AUC**: the overall performance of a classifier, summarized over all possible thresholds, is given by the area under the (ROC) curve (AUC). An ideal ROC curve will hug the top left corner, so the larger the AUC the better the classifier. A reliable and valid AUC estimate can be interpreted as the probability that the classifier will assign a higher score to a randomly chosen positive example than to a randomly chosen negative example. Both ROC and AUC are insensitive to the class distribution (suitable for unbalanced datasets), thus, we choose AUC as one of the most important measures to evaluate models in experiments.

4 Data

In order to predict people who are more likely to cause payment problems, detailed profiles of tenants are necessary, such as age, income and family size. Additionally, records of payment transactions for each tenant are also required to label the tenants if the dataset doesn't contain any labels. In this problem, we indeed have to label the tenants by ourselves based on the payment records. Besides, because of some historical reasons, the profiles of tenants are not recorded completely in the database by the housing corporation, therefore, we also need to seek extra sources in order to enrich the tenants' profiles.

4.1 Internal Data

We derive all the internal data from databases provided by the housing corporation. In total, three Microsoft SQL Server databases are used to create the final dataset.

1 "HuurProductie"

"HuurProductie" is a primary database of the housing corporation, almost all basic information of tenants and houses can be found in it, table 5 will give us a summary of features for both tenants and houses that are registered in the database.

According to table 5, it seems that we already get a decent profile for both tenants and houses. However, though the feature list looks good enough to make something impressive, we still should take the quality and the amount of data into consideration. The completely statistic description for the final datasets will be presented in the subsection "Final datasets" below.

Hereby, we just explain a few possibly ambiguous features in this table. For the features of tenants, the only unclear feature could be the "Mental Status", which is a binary variable indicates whether a people is involved with a mental health clinic or not. As to the features of houses, "Type I" and "Type II" are two different classification ways for houses while "Cluster" represents the group of similar houses, they are all defined by the housing corporation itself. Moreover, "Initial Rent" records the payment of the first month for each house.

2 "Omega"

"Omega" is a reference database which contains addresses of the houses. Because the actual addresses are not stored explicitly in the "HuurProductie" database, we need to use the "Postcode ID" variable to match them from the "Omega" database. Actually, we will not use address as a predictor in the experiments, but it is important for us to join the internal data with the external data by the address.

Type	Features
Tenants	Date of Birth Ethnicity Marital Status Gender Bank Account Mental Status Rent Allowance Indicator Household Size Household Income
Houses	Type I Type II District Cluster Building Year Number of Rooms Initial Rent

Table 5: Raw features derived from the database "HuurProductie"

Table 6 shows the geographical distribution of the houses at postcode level. According to this table, we can see that the business of the housing corporation spreads in 5 places, and mainly in "VALKENSWAARD" and "BERGEIJK".

VALKENSWAARD	BERGEIJK	LUYKSGESTEL	WAALRE	EINDHOVEN
324	29	4	4	1

Table 6: Distribution of the houses at postcode level

3 "DBS"

"DBS" database contains the entire financial data of the housing corporation, including every separate payment for each tenant. The records can be traced back to 2007, it means that we have approximately 10 years data to conduct our analysis.

Nevertheless, there are many difficulties exist in this database when we manage to extract payment features. The first problem is the naming method for tables in the database. All the names can not be recognized by people who has never used this database before, so it will cost us pretty long time to inspect this huge database with over 200 different tables. Hence, we adopt a more efficient method here. First, we calculate the number of rows for each table and then we only check those tables that have over 100000 rows of instances. In this way, only 17 tables are left for inspection. Finally, we find two tables contain the data we expect after scanning these 17 tables. The second problem is about the name of tenants, it is not recorded in the "DBS" database at the beginning which make it impossible for us to link the "DBS" data to a

specific tenant. But fortunately, it is feasible to connect the "DBS" database with the "HuurProductie" database by the bank account number instead of the name, which leads to less instances in the resulting dataset. The third problem appears in above two tables which contain the expected data. The first table called "aobr" which has the bank account number of the tenant, the amount of each payment etc. whilst the other table named "bst" stores more information related to transactions, such as the type of payment, the date of each transaction etc. As mentioned before, we can only find records starting from 2007 in both tables, but many contracts started before 2007, thus, in some sense, the data itself is not complete. Furthermore, we assume the table "aobr" and the table "bst" will record data consistently, but there are a lot of exceptions. For example, the table "aobr" records the payment amount of a tenant since 2010, but the table "bst" may record the same person's transactions since 2007, consequently, if we join both table together, it will generate missing data of payment amount during 2007-2010 for that person. Figure 3 and 4 display a concrete example for this problem and they are also good examples for readers to understand what the table "aobr" and the table "bst" look like. By the way, we have removed some useless columns in both two figures.

	deb_kre_id	bst_id	bedrag	op_bj	ba	ban
26374	5000051	136608	361.49	2010	NL^^1529.37.110	152937110
72574	5000051	140428	361.49	2010	NL^^1529.37.110	152937110
180472	5000051	144261	361.49	2010	NL^^1529.37.110	152937110
132271	5000051	149047	361.49	2010	NL^^1529.37.110	152937110
184307	5000051	152845	361.49	2010	NL^^1529.37.110	152937110
26513	5000051	156690	365.84	2010	NL^^1529.37.110	152937110
80840	5000051	160370	365.84	2010	NL^^1529.37.110	152937110
118400	5000051	164288	365.84	2010	NL^^1529.37.110	152937110
175671	5000051	168136	365.84	2010	NL^^1529.37.110	152937110
177743	5000051	171860	-472.00	2010	NL^^1529.37.110	152937110
33073	5000051	172190	365.84	2010	NL^^1529.37.110	152937110
199540	5000051	176038	365.84	2010	NL^^1529.37.110	152937110
183473	5000051	179954	365.84	2011	NL^^1529.37.110	152937110
241220	5000051	183824	365.84	2011	NL^^1529.37.110	152937110
33710	5000051	187706	365.84	2011	NL^^1529.37.110	152937110
100070	5000051	192412	365.84	2011	NL^^1529.37.110	152937110
104064	5000051	196427	365.84	2011	NL^^1529.37.110	152937110
112836	5000051	200313	365.84	2011	NL^^1529.37.110	152937110
241630	5000051	204196	370.59	2011	NL^^1529.37.110	152937110

Figure 3: Example of the table "aobr"

4.2 External Data

Table 5 already gives us many useful features, but the missing data in the table "aobr" and "bst" is too much to handle, so we have to look for external sources to enrich the tenants' profiles. Then, a company named Cendris enters our line of

	bj	deb_id	bst_id	dat	oms	dat_aanmaak	dat_lst_wijz	ext_ref
64	2007	5000051	265	2007-03-01	Huur 01-03-2007 t/m 31-03-2007	2007-02-16 11:19:06	2007-03-01	Van Cuykstraat 27
1916	2007	5000051	4188	2007-04-01	Huur 01-04-2007 t/m 30-04-2007	2007-03-20 12:28:03	2007-04-01	Van Cuykstraat 27
2686	2007	5000051	8039	2007-05-01	Huur 01-05-2007 t/m 31-05-2007	2007-04-16 10:16:06	2007-05-01	Van Cuykstraat 27
1190	2007	5000051	11926	2007-06-01	Huur 01-06-2007 t/m 30-06-2007	2007-05-21 14:27:02	2007-06-01	Van Cuykstraat 27
5861	2007	5000051	15873	2007-07-01	Huur 01-07-2007 t/m 31-07-2007	2007-06-18 13:35:06	2007-07-01	Van Cuykstraat 27
824	2007	5000051	19690	2007-08-01	Huur 01-08-2007 t/m 31-08-2007	2007-07-16 12:48:58	2007-08-01	Van Cuykstraat 27
499	2007	5000051	23562	2007-09-01	Huur 01-09-2007 t/m 30-09-2007	2007-08-20 12:50:00	2007-09-01	Van Cuykstraat 27
15949	2007	5000051	27438	2007-10-01	Huur 01-10-2007 t/m 31-10-2007	2007-09-18 13:48:14	2007-10-01	Van Cuykstraat 27
2448	2007	5000051	31296	2007-11-01	Huur 01-11-2007 t/m 30-11-2007	2007-10-17 08:48:07	2007-11-01	Van Cuykstraat 27
3066	2007	5000051	35154	2007-12-01	Huur 01-12-2007 t/m 31-12-2007	2007-11-19 11:30:27	2007-12-01	Van Cuykstraat 27
21424	2008	5000051	39010	2008-01-01	Huur 01-01-2008 t/m 31-01-2008	2007-12-17 11:31:55	2008-01-01	Van Cuykstraat 27
22805	2008	5000051	42766	2008-02-01	Huur 01-02-2008 t/m 29-02-2008	2008-01-17 08:26:01	2008-02-01	Van Cuykstraat 27
23005	2008	5000051	46546	2008-03-01	Huur 01-03-2008 t/m 31-03-2008	2008-02-18 09:42:35	NA	Van Cuykstraat 27
3605	2008	5000051	50326	2008-04-01	Huur 01-04-2008 t/m 30-04-2008	2008-03-17 10:00:08	NA	Van Cuykstraat 27
185	2008	5000051	53971	2008-05-01	Nieuwe betalingsregeling	2008-04-10 07:42:44	NA	NA
9637	2008	5000051	55134	2008-05-01	Huur 01-05-2008 t/m 31-05-2008	2008-04-17 08:22:44	NA	Van Cuykstraat 27
14214	2008	5000051	58635	2008-05-01	Aflossing betalingsreg 5-2008	2008-04-17 08:41:08	NA	NA
5900	2008	5000051	58888	2008-06-01	Huur 01-06-2008 t/m 30-06-2008	2008-05-19 10:55:39	NA	Van Cuykstraat 27
12327	2008	5000051	62389	2008-06-01	Aflossing betalingsreg 6-2008	2008-05-19 11:15:02	NA	NA

Figure 4: Example of the table "bst"

sight, which is a subsidiary of PostNL and can provide us more detailed data of some tenants at household level. We should notice that they can only provide the data of people who are currently living at the addresses we apply for, so that we will get more features at the cost of the decreasing of the instances.

We send Cendris all the postcodes that the housing corporation's houses locate in, and the dataset we receive contains all the people who are living in those streets at present, owning or renting a house respectively. Therefore, we obtain extra features for tenants who still live in the same houses or the same clusters now. But one thing we should take care is that we are using current profiles of tenants to predict the past behavior of them. It sounds like non-logical, but it is quite hard for the housing corporation to collect the data of the past years for each tenant. As a consequence, we have to run our experiments based on the available datasets. For more information about variables of Cendris dataset, please see Appendix I.

4.3 Label

In this part, we are going to describe how we label the people with having payment problem or not.

From figure 4, we can see there are extra two different descriptions besides the normal description for monthly rent, they are "Nieuwe betalingsregling" and "Aflossing betalingsregling". The first term means there is a new payment strategy applied to the tenant, it can reveal that something happens to the tenant so the housing corporation has to change the strategy for that person. The second term says there is a redemption or a repayment appear which directly indicates a payment problem

in that month. As a result, we decide to use these two descriptions as references to label the tenants. Firstly, we look at the distribution of these two descriptions in the database, see table 7. There are 756 people that the housing corporation once changed the payment strategy because of some reasons while 739 people encountered payment problems in the past. Actually, each tenant who has a "Aflossing betalingsregling" record can also be found in the "Nieuwe betalingsregling" group. Thus, we only need to focus on people who has a "Aflossing betalingsregling" description and figure out how to label the data using it.

Nieuwe betalingsregling	Aflossing betalingsregling
756	739

Table 7: People with "Nieuwe betalingsregling" and "Aflossing betalingsregling"

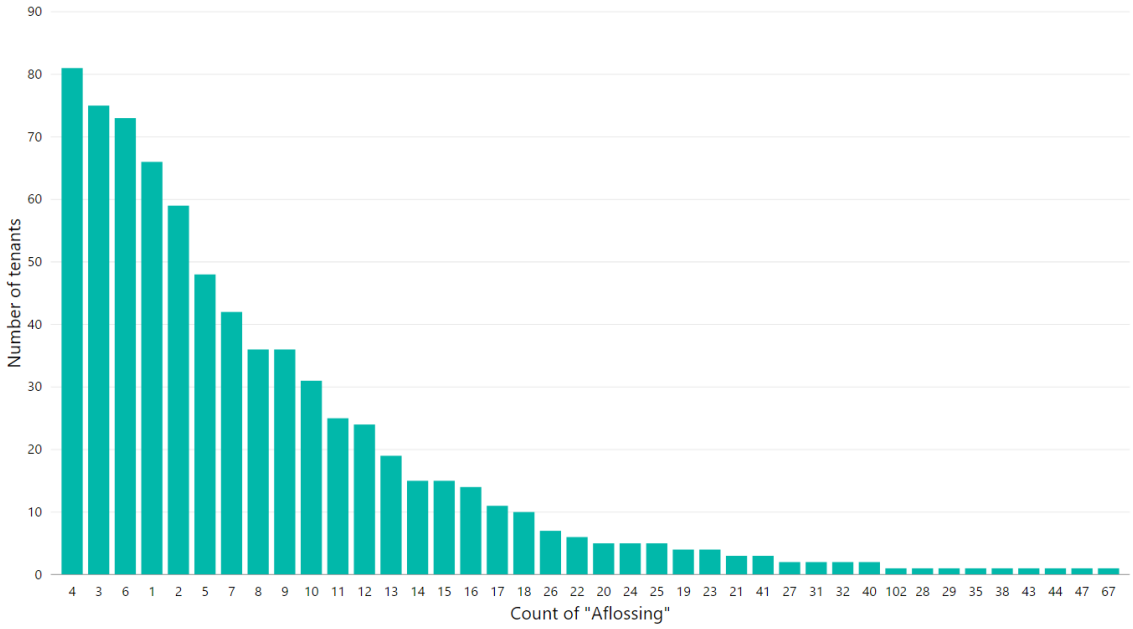

Figure 5: Distribution of "Aflossing betalingsregling" by count for tenants

In order to label the tenant with having payment problems, we have narrowed our searching space to those people has "Aflossing betalingsregling" description in the database. Next, we attempt to visualize this group of people in figure 5 to help us obtain more insights. The histogram shows that most people has less than 10 times "Aflossing betalingsregling" records. But how can we label the tenants based on this summary? If we take the threshold too small then we will weaken the representativeness of people who are more likely to cause payment problems, because some people may cause payment problems once or twice but they may not in the similar situation with those people who has payment arrears more than 20 or 30 times. On the contrary, if we set a large threshold then we will lose the generalization of different kinds of people who are likely to cause payment problems. Figure 6 presents another distribution of "Aflossing betalingsregling". By this histogram, we know that there are more than 400 "Aflossing betalingsregling" records appear during each year since 2007, and up to 794 in the maximum year. Accordingly, we can conclude that the count of "Aflossing betalingsregling" is approximately uniform

for each year which means we can use all the records from 2007 to 2016. However, we still would like to know the number of years where problems appears for most people, see figure 7. This figure demonstrates the distribution of problematic years for all tenants, it is evident that almost half of the tenants (331 of 739) who once had payment problems only in a single year. Therefore, we can't merely filter out those people who only had payment problems in one year, because it was possible for people to cause quite a lot payment problems in only one year, and tenants like this should also be labeled as problematic.

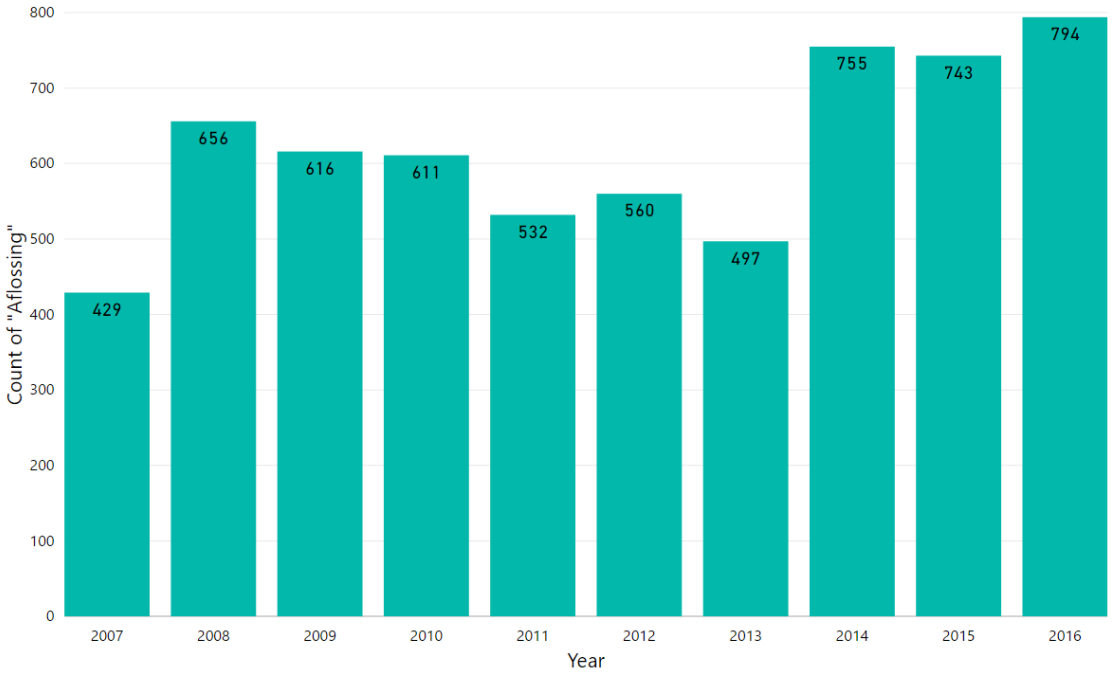

Figure 6: Count of the "Aflossing betalingsregling" per year

Given the above, we set the threshold as 3, namely, if a tenant has more than 3 times "Aflossing betalingsregling" records we will label this person with a "1", otherwise we will label the person with a "0". At last, 539 tenants are labeled with "1" in the "DBS" database. In this way, we filter out people who causes payment problems just a few times, but we remain those people who has more obvious patterns to cause problems, for example, people causes a lot of problems in one year and people has payment problems for many years. Specially, according to section 1.1.1, we know that the housing corporation has approximately 37000 tenants in the database, thus, except these 539 problematic tenants, other tenants are all labeled with a "0". Nonetheless, this is not the final label distribution yet, since we still need to join all the data from different data sources together and some tenants might not be matched successfully. The final datasets we use in the experiments will be discussed in the following section.

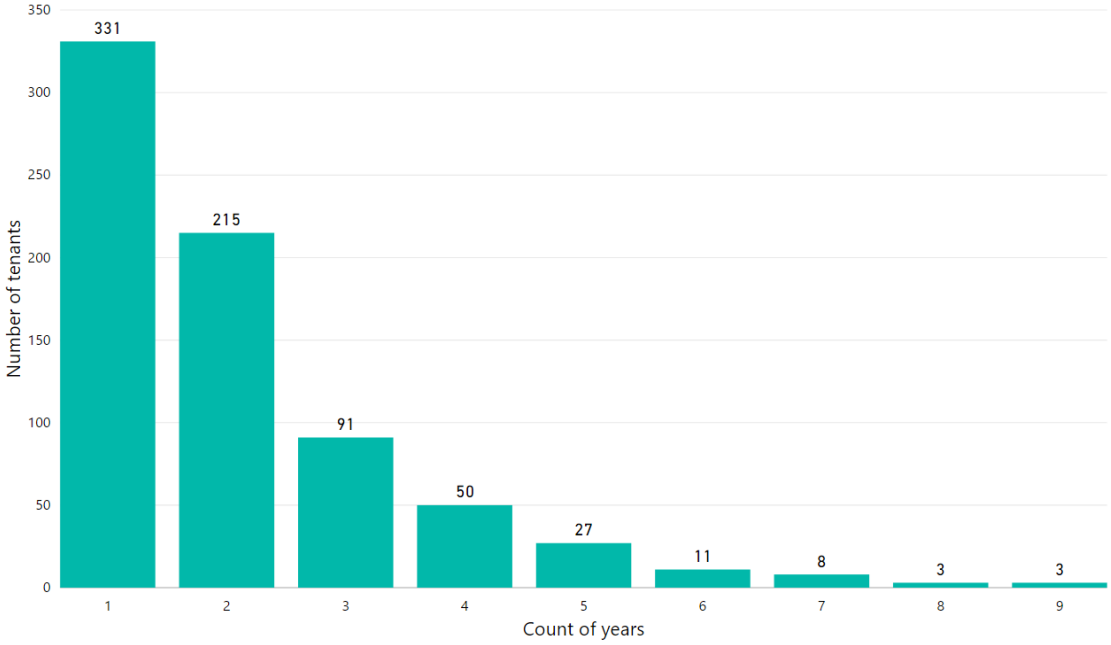

Figure 7: Distribution of problematic years among tenants

4.4 Final datasets

In this section, we describe the datasets we eventually use in the experiments. Ultimately, we create two datasets for use, one is the original dataset without Cendris data, the other one is the dataset joined with Cendris data.

Table 8 and 9 display the total number of instances and the distribution of labels for both datasets after joining all the data we have mentioned in previous sections. It is notable that there are only 258 tenants having payment problems left in the dataset without Cendris because we can't match the other 281 tenants from both the "HuurProductie" database and the "DBS" database. Simultaneously, the number of tenants with negative label decline from around 36000 to 2981. Moreover, if we join Cendris data as well, the number of instances will reduce even worse, with only 97 positive instances and 1074 negative instances left. Another observation is that both datasets suffer from significantly unbalanced distribution (the positive label only accounts for 8%). Unbalanced data can cause the model to be biased towards one class, hence, we will manage to solve it in the chapter of experiments.

Label	Number	Percentage
0	2931	91.9%
1	258	8.1%
Total	3189	100%

Table 8: Label distribution of the dataset without Cendris

Label	Number	Percentage
0	1074	91.7%
1	97	8.3%
Total	1171	100%

Table 9: Label distribution of the dataset with Cendris

4.4.1 Pre-processing

Pre-processing is an important step in machine learning and data mining. It aims to create datasets with high quality. Kotsiantis et al. present a decent summary of methods in different steps of pre-processing [27]. Because we have two datasets with some non-overlapping features, so we will deal with them respectively.

1 Dataset without Cendris

(a) Remove messy rows:
When we join data from different sources, sometimes, it is inevitable to generate some messy rows. Messy data will affect the accuracy of the prediction and lead to a lot of unknown errors when we try to fit models on them. So, we delete rows involved with data dislocation as well as rows with all blanks.

(b) Feature construction:
Because age is not recored directly in the database, we use the date of birth as well as the year the transaction happened to construct the actual age of a tenant at that year. In this manner, we can get the real age of tenants when they caused the first payment problem, which makes age become a reliable feature.

(c) Replace illegal values:
We extract age of tenants by the date of birth, however, it is strange that some people are over 200 years old when we examine the results, which is completely impossible in real life. After tracking back to the database, we find that the original database sets default value "1800-1-1" for tenants whose date of birth is unregistered. For around 100 rows of data like this, we replace wrong ages in each label with the mean of all the other reasonable ages belonging to the same label.

(d) Fill missing values:
Incomplete data is an unavoidable problem in dealing with most of the real world data sources [27]. It is also a very common problem exists in this project. In table 5, certain potentially significant features, such as "Household Size", "Household Income", "Number of Rooms" and "Initial Rent" have a lot of missing value, see table 10 for an overview of the number of missing values. If we just ignore all the missing values in these four fields we will lose much valuable information and we have to eliminate a lot of instances. Therefore, we must figure out a more reasonable method to handle it. Because each feature has different nature so that we have to tackle them using the most appropriate approach respectively.

Features	Number of missing values	Number of valid data	Total
Household size	2360	829	3189
Household income	2379	810	3189
Number of rooms	37	3152	3189
Initial rent	2275	914	3189

Table 10: Overview of missing values

i. "Household Size"
"Household Size" is correlated with the marital status, thus, we can calculate the mean household size for each type of marital status and use it to fill in the missing values of "Household size". Table 11 is a reference table for the feature "Marital status" and also presents the mean household size for each type of marital status. It is interesting to see that people who is married, living together or in a registered partnership owns a bigger household size, which means the data is reliable enough to fill in the missing values.

Marital Status	Description	Mean Household Size
X	Unknown	1.43
O	Unmarried	1.43
S	Unmarried living together	2.29
P	Registered partnership	1.75
G	Married	2.21
GO	Married (partner deceased)	1.15

Table 11: Mean household size for different "Marital Status"

ii. "Household Income"
When it comes to "Household Income", we adopt the same approach as we described before. We list the mean household income for different "Marital Status" in table 12. According to this table, we can summarize that people who is married, living together or in a registered partnership has more household income, which accords with common sense as well.

Marital Status	Description	Mean Household Income
X	Unknown	17225.76
O	Unmarried	20544.90
S	Unmarried living together	27218.90
P	Registered partnership	29823.97
G	Married	28506.78
GO	Married (partner deceased)	22453.69

Table 12: Mean household income for different "Marital Status"

iii. "Number of Rooms"
In fact, the "Number of Rooms" is not associated with tenants, but

it is dominated by the address and the type of the house. Hence, we should take those houses with the same postcode and category into account, more specifically, we can fill in the missing values by the "Number of Rooms" from similar houses. Fortunately, there are only 37 rows' data are missing with respect to the "Number of Rooms". Furthermore, 12 of them are garages, gardens, park places and storages, all of these kind of estates are not in our interest, so, we directly delete these 12 instances from the dataset. Eventually, 25 houses are left to be filled with the number of rooms.

Besides the address and the type of houses, there is another feature called "Cluster" available. It represents a group of similar houses. Generally, houses with the same cluster will have almost the same size, number of rooms and the amount of rent. Thus, for each house that the number of rooms is unknown, we calculate the average rooms of the other houses with valid values in the cluster it belongs to, then, we fill in the missing value with the average.

iv. "Initial Rent"
As far as "Initial Rent" is concerned, the same process as the feature "Number of Rooms" is done to deal with the missing values, because the amount of rent also depends on the type of houses. A slightly different step is that we calculate the overall mean of "Initial Rent" using all valid data, and we make it as a default to fill in all the instances that belong to a cluster without any valid "Initial Rent" records at all.

(e) Eliminate useless features:
Both "Ethnicity" and "Mental Status" have only one unique value and will not provide any meaningful information, so we should remove these useless features. It makes sense that these two features are not effective in this case, because the housing corporation locates in a small city of the Netherlands and the citizens are mostly dutch and it is also reasonable that most people has ability to rent a house will not be involved with a mental health clinic.

(f) Transform feature types:
Because we use the programming environment R to run all experiments of this project, it is vital that we elaborate the correct data types for each feature. In general, "data.frame" is the desired type for the entire dataset, and "factor" and "numeric" are two basic data types for the features. Especially, we should note that "factor" represents categorical variables in R. A summary of data types and possible values for all the features in the final dataset without Cendris is in table 13.

Data types	Features	Possible values
Numeric	Age	17 - 95
	Household Size	1 - 6
	Household Income	954 - 148965
	Building Year	1920 - 2016
	Number of Rooms	1 - 5
	Initial Rent	107 - 1332
Factor	Marital Status	6 categories
	Gender	3 categories
	Type I	20 categories
	Type II	27 categories
	Rent Allowance Indicator	Binary
	Label	Binary

Table 13: Final dataset without Cendris

2 Dataset with Cendris

Since tenants in the dataset with Cendris are a subset of the dataset without Cendris as well as Cendris data is already well-structured when we received it, we only need to join the original dataset with Cendris data by person ID in order to get extra features for subset of the tenants. Furthermore, as the same as previous section, we should transform the data type of all the features correctly, however, Cendris already made all the features categorical initially, so the only thing we should take care is making sure all features from Cendris are in "factor" type. In Appendix I, 24 features with "MX" prefix are newly added from Cendris data to the second dataset. Though this dataset has less instances compared with the dataset without Cendris, we still use it in the experiments because we assume that more features might improve the model performance as well.

3 The label distribution after pre-processing

After pre-processing, both datasets are cleaned thoroughly and well-formatted. In table 14 and 15, we illustrate new label distributions of the latest datasets and we compare them with the old distributions. It seems that the distributions remain the same, which means the pre-processing phase didn't lead to much loss of the number of instances. Nevertheless, we can't guarantee all the features are effective predictors, some of them are informative but a few features might be noisy. As a result, we will apply feature selection on the datasets in next section.

4.4.2 Feature selection with variable importance

The objective of variable (feature) selection is three-fold: improving the prediction performance of the predictors, providing faster and more cost-effective predictors,

Label	Previous number	Previous percentage	Latest number	Latest percentage
0	2931	91.9%	2916	91.9%
1	258	8.1%	258	8.1%
Total	3189	100%	3174	100%

Table 14: Label distribution of the final dataset without Cendris

Label	Previous number	Previous percentage	Latest number	Latest percentage
0	1074	91.7%	1058	91.7%
1	97	8.3%	96	8.3%
Total	1171	100%	1154	100%

Table 15: Label distribution of the final dataset with Cendris

and providing a better understanding of the underlying process that generated the data [28]. In addition, the reasons for using feature selection also include: avoiding the curse of dimensionality, simplifying the model and making it easier to interpret. Thus, it is wise to apply feature selection and check whether it will improve the results or not, especially for the dataset with Cendris (which has 35 features). In fact, many machine learning algorithms perform feature selection implicitly as part of their overall operation, such as algorithms using L1 norm regularization and random forests. Therefore, we use the selected features to enhance only those algorithms without feature selection internally.

1 Create train set and test set:

In order to make models in the experiments comparable, for both datasets, we create a fixed train set and test set by setting the splitting proportion of the original dataset as 0.7, namely, 70% of instances will be distributed into the train set and 30% will remain in the test set. The reason why we set the proportion as 0.7 is that it can keep both train set and test set with enough instances.

Notably, we can't randomly split the original datasets, on the contrary, we should split the datasets by stratified sampling. In stratified sampling, we select a single column for which we want values to be apportioned equally among train and test sets. In other words, it is still a kind of random splitting but it is done within the levels of selected column when this column is a factor in an attempt to balance the class distributions within the splits. In this case, the chosen column is the class label. Table 16 shows the resulting datasets and label distributions after splitting.

With fixed train and test sets, we are able to fit different algorithms on the train set and compare their results on the test set. Importantly, we should only perform feature selection on the train set instead of the entire dataset, because the latter option may lead to biased performance estimate.

datasets	Types	Train			Test		
	Label	0	1	Total	0	1	Total
Without Cendris	Number	2042	181	2223	874	77	951
	Percentage	91.8%	8.2%	100%	91.9%	8.1%	100%
With Cendris	Number	741	68	809	317	28	345
	Percentage	91.6%	8.4%	100%	91.9%	8.1%	100%

Table 16: Label distribution after splitting

2 Filter based variable importance

Variable importance are typically presented in two classes, the first class is filter based while the other one is model based. Because model based variable importance methods are customized for different algorithms so that we can't find a way to generalize them, moreover, the results of model based variable importance are difficult to interpret. Because of this, we only use filter based variable importance in this section. Filter based variable importance method selects variables regardless of the model and use the chosen metric to calculate the correlation between the potential features and the target variable. Generally, we use the selected metric to identify irrelevant features, and filter out redundant variables from the feature list. By choosing the most relevant features, it can potentially improve the accuracy and efficiency for classification problems. Paper [29] presents a good example of filter based feature selection by correlation between dependent variables and independent variables.

In the community of R, there are many available implementations of filter based variable importance and we choose "filterVarImp" function from "caret" package for use. In a word, we pick features with the highest score of importance as final predictors. For classification, ROC curve analysis is conducted on each predictor within "filterVarImp" function. For two class problems, a series of cutoffs is applied to the predictors to predict the class. The sensitivity and specificity are computed for each cutoff and the ROC curve is computed. The trapezoidal rule is used to compute the area under the ROC curve. This area (i.e. AUC) is used as the measure of variable importance. Finally, we only need to decide the number of features we would like to pick from the ordered list.

The scores of importance for each feature are listed in table 17 and table 18. Because there are 35 features in the dataset with Cendris and features with a low score are non-significant, we only display the top 20 of them here.

Features	Score
Age	0.6159436
Marital Status	0.5432641
Household Income	0.5326662
Gender	0.5203435
Household Size	0.5167423
Rent Allowance Indicator	0.5153787
Type I	0.5140394
Type II	0.5077151
Number of Rooms	0.5068641
Initial Rent	0.5068425
Building Year	0.5003179

Table 17: Filter based feature importance for the dataset without Cendris

Features	Score
Age	0.6285425
MXinkomen	0.6017504
MXwelstand	0.5775482
MXnvolw	0.5675359
Marital Status	0.5656009
MXbrink	0.5606494
MXleeftijd	0.5569084
MXTwinkkw	0.5459633
MXctht	0.5445046
MXlevfas	0.5440680
MXauto	0.5388386
Initial Rent	0.5383524
Household Size	0.5382333
Gender	0.5328848
Number of Rooms	0.5244205
Type I	0.5205307
Household Income	0.5200742
Rent Allowance Indicator	0.5196674
MXopleid	0.5166706
MXauleas	0.5138922

Table 18: Filter based feature importance for the dataset with Cendris (Top 20)

5 Experiments

In this chapter, we set up all the procedures for the experiments. First of all, the tools we use to run the experiments will be introduced and then we will explain how oversampling and undersampling are employed on the train sets. To end with, the detailed steps of how we build models are presented for each algorithm.

5.1 Experiments setup

We conduct all the experiments within Rstudio, the R version is 3.3.3 and the version of Rstudio is 1.0.153. At the beginning, we also tried using Python and the Microsoft Azure Machine Learning Studio as experimental options, but we finally found that Rstudio makes it easier for us to manipulate data because it has very handy visualization tools, furthermore, R outperforms Azure Machine Learning Studio due to more flexibility as well as more packages available.

5.2 Train/Test and SMOTE

In chapter 4.4.2, we already split the original datasets by 70% for training and 30% be testing, see table 16. Because of unbalance of the datasets, the predictions will always biased to the majority class. In order to overcome this problem, we attempt to apply oversampling or undersampling techniques on the train sets. One of the most well-known sampling methods is Synthetic Minority Oversampling Technique (SMOTE) [30]. The paper claims that a combination of their method of oversampling the minority (abnormal) class and undersampling the majority (normal) class can achieve better performance (in ROC space) than only undersampling the majority class.

We apply SMOTE on both datasets with three different pairs of oversampling and undersampling rates, table 19 and 20 list all the rates we set for both datasets and the distributions of the label in the resulting train sets. The terms "Over" and "Under" control the amount of oversampling of the minority class and undersampling of the majority classes respectively. Here we take the third column in table 19 for an example, "Over=100" means 100% (181) of minority class instances are newly created, while "under=200" indicates that 200% of the number of new created minority instances, namely, 362 instances will be randomly selected from the majority class to constitute the final balanced dataset.

Label	Original	Over=100 Under=200	Over=200 Under=150	Over=1000 Under=110
0	2042	362	543	1991
1	181	362	543	1991
Total	2223	724	1086	3982

Table 19: Label distributions after SMOTE (without Cendris)

Label	Original	Over=100 Under=200	Over=400 Under=125	Over=900 Under=111
0	741	136	340	679
1	68	136	340	680
Total	809	272	680	1359

Table 20: Label distributions after SMOTE (with Cendris)

5.3 Procedures

We create two datasets for the experiments, one is generated without Cendris data, the other one is enriched with Cendris data. Therefore, the following procedures are applicable for both datasets. Besides, we use the "caret" package to streamline the model training process. The package utilizes a number of R packages but tries not to load them all at package start-up, which is very handy and time-saving.

5.3.1 Single algorithm

Because "glmnet" and "random forests" implement feature selection implicitly, we only fit extra models with explicit feature selection (filter based variable importance in chapter 4) for other algorithms. More specifically, both "glmnet" and "random forests" will generate 3 models, while other algorithms will yield 6 models.

1. Lasso and elastic-net regularized linear models
    - Train set: 3 SMOTE datasets
    - Family: binomial, which fits a traditional logistic regression model
    - Tuning parameters: lambda, which is the regularization parameter (the other parameter alpha is set as a constant 1, namely, the lasso regularization is used)
    - Tuning grid: automatically
    - Tuning method: 3 times repeated 10 fold cross validation
    - Selected measure for optimization: ROC, in fact, AUC is calculated (the same in the following settings)
    - Extra models with feature selection: no
2. Random forests
    - Train set: 3 SMOTE datasets
    - Tuning parameters: mtry, which is the number of variables randomly sampled as candidates at each split
    - Tuning grid: automatically
    - Tuning method: 3 times repeated 10 fold cross validation
    - Selected measure for optimization: ROC
    - Extra models with feature selection: no

3. K nearest neighbors
    - Train set: 3 SMOTE datasets
    - Tuning parameters: K, which is the number of neighbors considered
    - Tuning grid: automatically
    - Tuning method: 3 times repeated 10 fold cross validation
    - Selected measure for optimization: ROC
    - Extra models with feature selection: yes
4. Naive bayes
    - Train set: 3 SMOTE datasets
    - Tuning parameters: usekernel, which is a binary parameter used to estimate the densities of numeric predictors
    - Tuning grid: automatically
    - Tuning method: 3 times repeated 10 fold cross validation
    - Selected measure for optimization: ROC
    - Extra models with feature selection: yes
5. avNNet
    - Train set: 3 SMOTE datasets
    - Tuning parameters: size, which is the number of units in the hidden layer; decay, which is the parameter for weight decay
    - Tuning grid: automatically
    - Tuning method: 3 times repeated 10 fold cross validation
    - Selected measure for optimization: ROC
    - Extra models with feature selection: yes

5.3.2 Ensemble methods

Sometimes, a single algorithm is not enough to yield satisfactory prediction results, so we also try to build ensemble models. Three types of ensemble methods are employed in the experiments: averaging, majority voting and stacking.

1. For each algorithm, we select the best model based on accuracy, recall, F1 score, AUC respectively
2. Firstly, we create averaging ensemble models using the best three models of different algorithms with respect to the same measure
3. Then, we tune the weights for each components of the best averaging ensemble model in step 2, it is so-called the weighted averaging ensemble method
4. Next, we build majority voting ensemble model using the same components of the best averaging ensemble model in step 2
5. At last, we construct stacking ensemble models by setting 5 different top layers (with 5 algorithms we have used in the previous section). As far as the bottom layer is concerned, we will give more specific explanations in chapter 6.

5.3.3 Subgroup discovery

There are many packages in R that have implemented subgroup discovery or rule mining algorithms. In our experiments, we use an R package called "subgroup.discovery". This package aims to assist in discovering interesting subgroups in multi-dimensional data, it is an implementation of the PRIM algorithm [26]. PRIM involves finding a set of "rules" which combined imply unusually large (or small) values of some other target variable. Specifically one tries to find a set of subregions in which the target variable is substantially larger than overall mean.

Because SMOTE data may mislead the generated rule due to its unreality, we run the subgroup discovery algorithm on the original datasets only. We will list the mined rule (includes a number of conditions) for both datasets in next chapter. Additionally, we will convert the generated rule to a predictive model and test the model with the test set as well.

The concrete approach to transform the rule to a predictive model is as below: for each condition of the rule, we check if the test instance meet the condition; supposing one test instance meets all the conditions, we will predict it as a positive sample. In this way, we are able to compare subgroup discovery rules with machine learning models by the same measures like accuracy recall, F1 score and AUC.

6 Results

In this chapter, we display the results produced from the experiments. First of all, we show the results of all models fitted by single machine leaning algorithm, subsequently, we illustrate the results generated by ensemble methods. In the end, we list the mined rule of subgroup discovery and the results of the converted model. The selected measures to evaluate models are accuracy, recall, F1 score and AUC. Particularly, the most significant measure should be AUC, because it is more suitable for unbalanced datasets.

6.1 Single algorithms

6.1.1 Lasso and elastic-net regularized linear models

Lasso and elastic-net regularized linear models correspond to the "glmnet" package in R. Table 21 and 22 present results of the linear models for the dataset with Cendris and the dataset without Cendris respectively. In the following tables, there are a lot of terms like "glmnet1wo". Actually, the letters indicate which algorithm or package is used, and the number "1" means we fit this model on the first SMOTE train set, namely the SMOTE train set with the least number of instances, the postfix "wo" means this model is built on the dataset without Cendris. Another abbreviation will be used frequently is "fs", it means the model is fitted using features produced by explicitly feature selection method. The rest paper will use similar naming method for other models. For example, "glmnet1wo" shows that it is a "glmnet" model fitted on the first SMOTE train set of the dataset without Cendris.

Models	Accuracy	Recall	F1-score	AUC
glmnet1wo	60.2	57.1	18.9	58.8
glmnet2wo	61.1	54.5	18.5	58.1
glmnet3wo	58.6	45.4	15.1	52.6

Table 21: "GN" models on dataset without Cendris

Models	Accuracy	Recall	F1-score	AUC
glmnet1w	63.0	39.3	14.7	52.1
glmnet2w	68.7	39.2	16.9	55.3
glmnet3w	69.6	46.4	19.8	59.0

Table 22: "GN" models on dataset with Cendris

According to table 21 and 22, the best models should be "glmnet1wo" and "glmnet3w" if we take all measures into account.[i] But both models have a low F1-score, it is because of the influence of low precision. Moreover, it is interesting to see that the model becomes worse if we oversample more data for the dataset without Cendris, while it is completely opposite with respect to the dataset with Cendris.

[i]The confusion matrix of the model in bold can be found in Appendix II

Features	Coefficient estimates
Marital Status	0.11778597034
Gender	-0.42014234657
Rent Allowance Indicator	-0.64440136134
Household Size	-0.41945110102
Household Income	-0.00002305128
Type I	0.00420757808
Type II	0.08840304898
Building year	-
Number of Rooms	-
Initial Rent	0.00027108745
Age	-0.02938079041

Table 23: Coefficients estimate of "glmnet1wo"

Features	Coefficient estimates	Features	Coefficient estimates
MXleeftijd	-0.014189272	MXsentype	0.052645401
MXctht	-0.039399759	MXlevfas	-
MXmanvrw	-	MXnvolw	0.640711466
MXopleid	-	MXwelstand	-0.176857183
MXinkomen	0.209406016	MXbrink	0.031442583
MXtwinkkw	0.047361187	MXprijs	0.077322829
MXsockl	.	MXchartype	.
MXmedia	-0.004529825	MXkwdb	.
MXpopdb	0.357463585	MXpostord	0.040877751
MXduurzaam	.	MXactie	-0.015300159
MXcc	-0.195286314	MXadopter	-0.087778967
MXauto	-0.499737714	MXauleas	0.167397031
Maritial Status	0.271749751	Gender	.
Rent Allowance Indicator	-0.656517530	Household Size	-0.103873000
Household Income	-	Type I	-0.043028400
Type II	0.084240094	Building Year	-0.003131642
Number of Rooms	0.136078169	Initial Rent	0.001919512
Age	-0.033785996		

Table 24: Coefficients estimate of "glmnet3w"

We extract the coefficient estimates of model "glmnet1wo" and model "glmnet3w" using the function "coef()". Features that have positive coefficient are most predictive for the positive class (there is a positive correlation between these features and the positive class), while features that have a negative sign are most predictive for the negative class. As shown in table 23, "Rent Allowance Indicator" has a coefficient of 0.644, because it is a binary variable, it indicates that tenants who don't have rent allowance are more likely to cause payment problems which makes sense. Another example is "Household Income", which has a coefficient of -0.000023, this means that an increase in household income is associated with an decrease in the probability of having payment problems which is also sensible. Besides, we can see that if an initial rent is high, the tenants may have more probability to cause pay-

ment problems in the future. More interesting, the building year and the number of rooms are not correlated with the payment problems, because the coefficients of these two features are not provided. However, because "glmnet" doesn't handle factor variables as same as "glm" (basic logistic regression) package, normally we have to transform factor variables into dummy variables. But in this problem, when we check the values of "Type I" and "Type II", we can see that the possible values are already in an ascending order of the potential rent price, for example, in "Type I", "1" represents a single room while "4" stands for a semi-detached villa, thus, we can regard these variables as numeric features. As to the "Marital status", we transform the possible values to indicate the status from unmarried to married, such as "0" represents "Unknown", "1" means "Unmarried" and "4" expresses "married". In this way, we can interpret the results for these three factor features as same as numeric features. Both "Type I" and "Type II" have a positive parameter, which means that tenants who rent expensive houses are more likely to cause payment problems. "Marital status" also has a positive coefficient, it might reveal that people who have a larger family tend to cause more payment problems.

Nevertheless, in table 24, the positive coefficient estimates are unreasonable for features "MXinkomen", "MXbrink", "MXtwinkkw", it means that more income will lead to more probability of having payment problems. The reason of this may involve with the generated date of Cendris data, as mentioned before, we are using current data (Cendris data is made in 2016) to predict past behavior of people, which seems not logical. Another reason might be the number of instances in the dataset with Cendris are not sufficient for "glmnet" to gain useful insights. Anyway, we can still find some surprising results, for instance, "MXauto" has a negative coefficient while "MXauleas" has a positive coefficient, which indicates that people who lease a car are more likely to cause payment problems than people who own a car.

6.1.2 Random forests

The "randomForest" package is chosen to fit tree based models, which provides an efficient implementation of random forests. Similar to the "glmnet" model, random forests algorithm also performs feature selection internally, this is also the reason why there are three models for each dataset as well.

Models	Accuracy	Recall	F1-score	AUC
rf1wo	69.9	39.0	17.3	55.8
rf2wo	82.8	16.9	13.7	52.7
rf3wo	89.6	6.5	9.1	51.7

Table 25: "RF" models on dataset without Cendris

Table 25 and 26 give us all measures of random forests models. If we only look at accuracy, we will consider model "rf3wo" and model "rf3w" as the best models, but it is necessary to check the other measures as well. Both model "rf3wo" and model "rf3w" have low recall and F1-score, it reveals that the predictions of them will always biased to the majority class, namely, the class labeled with "0". A good model can't accept such biased results. The aim of the problem is to find people

Models	Accuracy	Recall	F1-score	AUC
rf1w	61.5	42.9	15.3	53.0
rf2w	74.8	28.6	15.5	53.7
rf3w	85.5	3.6	3.8	48.2

Table 26: "RF" models on dataset with Cendris

who might cause payment problems in the future, therefore, recall must be taken into consideration. On the basis of this, model "rf1wo" and model "rf1w" should outperform other models. However, compared to the "glmnet" models, random forests doesn't improve the results as we expect.

6.1.3 K nearest neighbors

We choose K nearest neighbors algorithm in "class" package to fit models in this section. It requires explicitly feature selection for "KNN" models, so there are 6 models for each dataset. As defined previously, "fs" means the model is built with explicit feature selection by filter based variable importance.

Models	Accuracy	Recall	F1-score	AUC
knn1wo	71.1	33.8	16.0	54.1
knn2wo	81.4	23.4	17.0	54.9
knn3wo	87.4	6.5	7.7	50.5
knn1fswo	74.0	29.9	15.7	53.9
knn2fswo	79.7	19.5	13.5	52.2
knn3fswo	90.0	3.4	5.9	50.7

Table 27: "KNN" models on dataset without Cendris

Models	Accuracy	Recall	F1-score	AUC
knn1w	78.8	35.7	21.5	59.2
knn2w	78.0	21.4	13.6	52.2
knn3w	84.6	14.3	13.1	52.6
knn1fsw	74.2	39.3	19.8	58.3
knn2fsw	78.6	17.9	11.9	50.9
knn3fsw	84.4	7.14	6.9	49.2

Table 28: "KNN" models on dataset with Cendris

Table 27 and 28 show us that "KNN" algorithm works better on the dataset with Cendris. And so far, "knn1w" should be the best model we have seen with respect to F1-score and AUC. Another interesting observation is that feature selection can't help us to improve the results of KNN. One reason is because we use filter based variable importance method to select best features for all algorithms, but these features might not be suitable for "KNN". Therefore, if we would like to improve it, we can conduct a model based variable importance method directly on "KNN" algorithm and then use the most predictive features to rebuild models.

6.1.4 Naive bayes

For Bayesian models, "naivebayes" package is used. The "naive_bayes" function in this package accepts numeric matrix or dataframe with categorical or numeric values as predictors. Normally, numeric predictors are handled by assuming that they follow Gaussian distribution, given the class label. Alternatively, kernel density estimation (KDE), which is a non-parametric way to estimate the probability density function of a random variable, can be used to estimate the class-conditional distributions of numeric features.

As shown in figure 8, we plot the density distributions of a numeric variable "age" for the model "nb1wo" in table 29. The red solid line represents the distribution for the negative class while the green dashed line corresponds to the positive class. Because the "usekernel" parameter is set as "True" in this model, so the distributions don't obey Gaussian distribution and the smoothing parameter for KDE, namely, the bandwidth is also reported along with the class label.

Models	Accuracy	Recall	F1-score	AUC
nb1wo	51.2	67.5	18.3	58.7
nb2wo	53.0	58.4	16.8	55.5
nb3wo	53.0	53.2	15.5	53.1
nb1fswo	51.1	62.3	17.1	56.2
nb2fswo	53.0	61.0	17.4	56.7
nb3fswo	51.4	50.6	14.4	51.0

Table 29: "NB" models on dataset without Cendris

Models	Accuracy	Recall	F1-score	AUC
nb1w	66.1	35.7	14.6	52.2
nb2w	68.4	28.6	12.8	50.2
nb3w	75.7	21.4	12.5	50.9
nb1fsw	67.8	35.7	15.3	53.2
nb2fsw	70.7	28.6	13.7	51.5
nb3fsw	78.3	21.4	13.8	52.4

Table 30: "NB" models on dataset with Cendris

Based on table 29, it is surprising to see that naive bayes algorithm produces the best recall (67.5%) so far on the dataset without Cendris, but with a cost of low accuracy. Meanwhile, the AUC still remains around 59%. From table 30, it seems that the results are not promising on the dataset with Cendris, especially for AUC, because the best AUC has a percentage of only 53.2%.

6.1.5 Neural networks using model averaging

Neural network is another famous machine learning algorithm. In the experiments, we use "avNNet" package to fit ensemble averaging neural network models, which is supposed to be much more predictive than a single neural network.

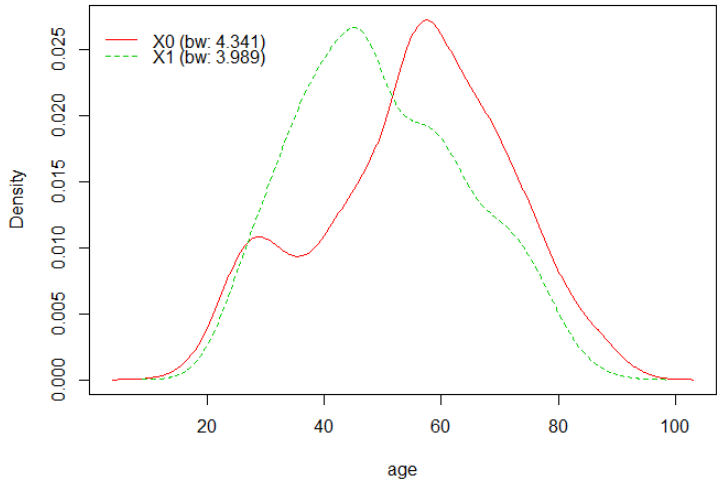

Figure 8: The densities of "age" for model "nb1wo"

Models	Accuracy	Recall	F1-score	AUC
nn1wo	59.1	53.2	17.4	56.4
nn2wo	64.5	49.3	18.3	57.6
nn3wo	72.3	42.9	20.0	58.9
nn1fswo	58.7	59.8	19.0	59.2
nn2fswo	55.8	59.7	18.0	57.6
nn3fswo	57.6	51.9	16.6	55.0

Table 31: "NN" models on dataset without Cendris

From table 31 and 32, we can see model "nn1fswo" yields the best AUC among all models on the dataset without Cendris, while model "nn3w" can be regarded as the best model on the dataset with Cendris. But if we take all measures into consideration, the model "nn3wo" might outperform all the others.

Now, let's summarize the results of all the models made by single algorithm. Generally speaking, Models of "glmnet" and "avNNet" outperform other algorithms in terms of the available datasets. If we only concern about AUC, then the model "knn1w" and the model "nn1fswo" models perform best. Furthermore, considering all measures, the model "glmnet3w" as well as the model "nn3wo" will win out.

6.2 Ensemble methods

6.2.1 Averaging and weighted averaging

The naming method changes a little bit in this section. For instance, "accwo" indicates that we select the best model from each algorithm based on the accuracy on the dataset without Cendris. Particularly, "waucwo" means we set weights for each candidate model of "aucwo" and then create the weighted averaging model.

Models	Accuracy	Recall	F1-score	AUC
nn1w	64.9	32.1	13.0	50.0
nn2w	69.0	35.7	15.7	53.8
nn3w	71.3	39.3	18.2	56.7
nn1fsw	62.0	39.3	14.4	51.7
nn2fsw	68.1	28.6	12.7	50.0
nn3fsw	74.2	32.1	16.8	55.0

Table 32: "NN" models on dataset with Cendris

Models	M1	M2	M3	Accuracy	Recall	F1-score	AUC
accwo	rf3wo	knn3fswo	nn3wo	84.4	22.1	18.7	56.0
recwo	glmnet1wo	nb1wo	nn1fswo	52.7	65.8	18.2	59.1
f1wo	glmnet1wo	nb1wo	nn3wo	56.8	57.9	17.7	57.3
aucwo	glmnet1wo	nb1wo	nn1fswo	52.7	65.8	18.2	59.1
waucwo	0.175	0.65	0.175	52.0	68.4	18.6	59.9

Table 33: "Averaging ensemble" models on dataset without Cendris

According to table 33 and 34, if we take all the best models based on AUC, it will yield the best averaging model for both datasets, that is to say, 65.8% recall and 59.1% AUC for dataset without Cendris while 35.7% recall and 60.6% AUC for dataset with Cendris. Next, we adjust different weights for each candidate model (M1, M2, M3) and we put the best result in the bottom row of both tables. The good news is that we lift the AUC and recall to some extent. Now, we have 68.4% recall and 59.9% AUC for dataset without Cendris whilst 39.3% recall and 60.8% AUC for dataset with Cendris. In the latter model, "waucw", it still has an accuracy over 78% and the F1-score beyond 23%.

6.2.2 Majority voting

From above analysis, we know that "aucwo" and "aucw" can make the best predictions by averaging ensemble method. Therefore, we only perform majority voting on the components within these two models. In other words, as for the dataset without Cendris, we build a majority voting model using the model "glmnet1wo", "nb1wo" and "nn1fswo". At the same time, we make another majority model with the model "knn1w", "nn3w" and "glmnet3w" for the dataset with Cendris.

Table 35 shows that the majority voting model "mvwo" improves the accuracy, F1-score and AUC once again based on the model "waucwo", now the AUC for the dataset without Cendris comes to 60.5%. Whereas the majority model can't improve the weighted averaging model for the dataset with Cendris, see table 36, it decreases the AUC of "waucw" from 60.8% to 57.1%, and other measures also decline in some degree.

Models	M1	M2	M3	Accuracy	Recall	F1-score	AUC
accw	rf3w	knn3w	nb3fsw	86.6	10.7	11.5	52.0
recw	rf1w	nn3w	glmnet3w	71.0	42.9	19.4	58.2
f1w	knn1w	nn3w	glmnet3w	81.5	35.7	23.8	60.6
aucw	knn1w	nn3w	glmnet3w	81.5	35.7	23.8	60.6
waucw	0.2	0.4	0.4	78.8	39.3	23.2	60.8

Table 34: "Averaging ensemble" models on dataset with Cendris

Models	M1	M2	M3	Accuracy	Recall	F1-score	AUC
mvwo	glmnet1wo	nb1wo	nn1fswo	55.6	66.2	19.5	60.5

Table 35: "Majority voting" models on dataset without Cendris

Models	M1	M2	M3	Accuracy	Recall	F1-score	AUC
mvw	knn1w	nn3w	glmnet3w	75.1	35.7	18.9	57.1

Table 36: "Majority voting" models on dataset with Cendris

6.2.3 Stacking

Because the stacking technique we adopt can only handle train sets with the same size, and less SMOTE data in train set, more reliable results we will get. Thus, we use all the models fitted on the smallest SMOTE train set to build stacking models. In previous sections, we have seen that 5 different algorithms have been applied to fit models. Therefore, first of all, we fit a stacking ensemble model with all these 5 algorithms. Furthermore, we create more stacking ensemble models with only 4 algorithms for each, then there will be 5 different combinations in total. Here we illustrate the 5-algorithm stacking model and the best 4-algorithm stacking model in table 37 and 38 respectively.

Excitedly, we obtain the best models ("sfourwo" and "sallw") on both datasets with respect to AUC. Moreover, other measures are also acceptable with accuracy over 61%, recall more than 57% as well as F1-score bigger than 20%. In a word, we can use these two models as our final predictive models.

6.3 Subgroup discovery

In "subgroup.discovery" package, we can adjust the following parameters to get expected results:

- peeling.quantile: quantile to peel off for numerical variables
- min.support: minimal size of a box to be valid
- max.peel: maximal size of a peel, as a fraction. Defaults to 0.1
- train.fraction: train-test split fraction used in validation, defaults to 0.66
- max.boxes: maximum number of boxes, NA or leave out for no limit

Models	Bottom layer	Top layer	Accuracy	Recall	F1-score	AUC
sallwo	glmnet1wo rf1wo knn1wo nb1wo nn1fswo	Random forests	74.2	32.5	16.9	55.2
sfourwo	glmnet1wo rf1wo nb1wo nn1fswo	Naive bayes	61.9	59.7	20.3	60.9

Table 37: "Stacking ensemble" models on dataset without Cendris

Models	Bottom layer	Top layer	Accuracy	Recall	F1-score	AUC
sallw	glmnet1w rf1w knn1w nb1fsw nn1fsw	avNNet	67.0	57.1	21.9	62.5
sfourw	rf1w knn1w nb1fsw nn1fsw	avNNet	74.8	42.9	21.6	60.2

Table 38: "Stacking ensemble" models on dataset with Cendris

- quality.function: function to use for determining set quality, defaults to mean
- optimal.box: during validation, choose the box with the highest quality or a simpler box, two standard errors from the optimum

During the experiments, we set the value of peeling.quantile = 0.05, (the recommend range is 0.05-0.1 according to [26]) min.support = 0.1 and optimal.box = "2se", while we keep all the other arguments remain the defaults. By this setting, the algorithm will produce interesting rules contains a number of conditions that represent the potential group of people who are more likely to cause payment problems.

6.3.1 Rules

For both datasets, we first present all the boxes or covers generated by the subgroup discovery algorithm. Then we list all the conditions that constitute these rules. And in the end, the leftover set will be given, where the algorithm stops because relative support of rules in this subset is less than the "min.support" we have set.

1. **Dataset without Cendris**

   - Cover 1 (the first rule):
     - Cover set size: 2223
     - Cover set quality: 0.08
     - Box relative quality: 0.11 (1.37)
     - Box relative support: 0.51 (1135)
   - Conditions of cover 1:
     + 27 <= Age <= 56
     + Initial Rent >= 420
     + Gender != 'Unknown'
     + Maritial Status != {'Unmarried living together', 'Married(Partner deceased)'}
     + Type I != {'Semi-detached villa','Porch apartment','Duplex apartment', 'Studio'}
     + Type II != {'Senior housing','Senior apartment','Apartment with lift'}
   - Leftover (the remaining dataset except cover 1):
     - Cover set size: 1088
     - Cover set quality: 0.05

     Cover 1 is the first rule found by the algorithm, and only this rule is valid for the dataset without Cendris. Details of the results are displayed as above. The cover set size in cover 1, 2223, is the size of the original dataset and the cover set quality indicates the mean of the class label over all the instances in the original dataset. The box relative quality represents the quality of the subgroup which consists of the conditions in the first rule. We can see that the quality has been increased from 0.08 to 0.11 in this subgroup with 1135 instances.

     Besides, we can find conditions of the rule are more interpretable compared with the logistic regression model, for instance, the first condition shows that people between 27 and 56 years old are more likely to cause payment problems, the reason is that people in this range mostly makes a living by themselves and are more likely to face the problem of losing jobs while they don't have enough allowances like the older. Another interesting condition summarizes that people who pay an initial rent more than 420 euro have more probability to cause payment problems. However, we should take care of how we interpret the results of subgroup discovery, actually, we should consider a tenant might be more problematic if this person meet all the conditions of the rule instead of only focus on one single condition, this might be the main difference of interpretation between a rule and a logistic regression model.

2. **Dataset with Cendris**

- Cover 1 (the first rule):
  - Cover set size: 809
  - Cover set quality: 0.08
  - Box relative quality: 0.15 (1.82)
  - Box relative support: 0.19 (150)
- Cover 2 (the second rule):
  - Cover set size: 659
  - Cover set quality: 0.07
  - Box relative quality: 0.09 (1.25)
  - Box relative support: 0.59 (387)
- Conditions of cover 1:
  + 28 <= Age <= 59
  + Marital Status != 'Married(Partner deceased)'
  + Useful Income Area <= 2
  + Number of rooms >= 2.95
  + Person type == {'Multicultural city residents','Decent neighborhood residents'}
  + Stage of life == {'Family without children <35 years old','Family with children, youngest 13+ years old','Single 55-69 years old','Family without children, 55-69 years old'}
  + Well-being class >= 3 (The worst class is 5)
  + Type I != {'Corner house', 'Semi-detached villa','Porch apartment','Gallery apartment'}
- Conditions of cover 2:
  + Gender != 'Unknown'
  + Leased car == 0
  + Person type == {'Concerned old people','Enterprising old people'}
  + Stage of life == {'Single 35-54 years','Family without children, 35-54 years old}
- Leftover (the remaining dataset except cover 1):
  - Cover set size: 257
  - Cover set quality: 0.04

  The subgroup discovery algorithm generates two rules for the dataset with Cendris. The first rule has a better quality but with a cost of low support while the second rule just improve the overall quality slightly with a better support.

  The conditions of both rules are combined together for a more general description of the problematic tenants, and surprisingly, the rules suggest almost the same range of age for problematic tenants. Additionally, it reveals that people whose useful income areas are less than 2 are more problematic.

6.3.2 Prediction

In table 39 and 40 we demonstrate the results of converted models built based on the the above rules. Nevertheless, the overall performance is not as good as the best models we already get from stacking ensemble models in section 6.2.3. But it is notable that the subgroup discovery algorithms can still generate models with recall more than 30%. Hence, we believe that if the algorithm can produce more representative rules then it will yield models with better results.

Models	Accuracy	Recall	F1-score	AUC
sbwo	77.5	31.2	18.3	56.4

Table 39: "Subgroup" models on dataset without Cendris

Models	Accuracy	Recall	F1-score	AUC
sbw	66.7	35.7	14.8	52.6

Table 40: "Subgroup" models on dataset with Cendris

7 Conclusion

In this thesis, we have compared five different machine learning algorithms: lasso and elastic-net regularized linear models, random forests, K nearest neighbors, naive bayes and neural network using model averaging. In this particular problem, "glmnet" and "avNNet" achieves the best overall performance for both datasets. Besides, we also investigate how feature selection can affect the prediction results. In some sense, feature selection can improve the results but it depends on which measure should be taken into consideration. Furthermore, different oversampling and undersampling rates of SMOTE have been tested while different kinds of ensemble methods have been created to improve the results as well. It seems that, sometimes, less oversampling works well on several algorithms, however, undersampling could also make sense, so we must make a trade off between them. In summary, it is always worthwhile to try both kinds of sampling techniques and find the best combination of rates for a specific dataset. Finally, a subgroup discovery algorithm is introduced to help us get more insights from the data, meanwhile, predictive models are built based on the rules generated from the subgroup discovery algorithm. In this way, a comparison analysis has also been done between machine learning and subgroup discovery.

Next, we answer all the research questions we presented in the introduction chapter. To end with, some future work will be discussed afterwards.

- *Is it possible to build a classification model to predict if a tenant will cause payment problems or not based mainly on the tenant's profile?*
  According to the experiments, we can find it is easy to fit a model that has high accuracy on an unbalanced dataset. But it is always biased to the majority class which is not expected by the customer because the minority class is of more interests. Moreover, we are using current data to predict the past behavior of tenants, so the features we find can not represent real situations of the tenants during the years that they caused payment problems. In the meantime, we also lack a lot of crucial features such as the working status of tenants at that time, namely, we are eager to know whether something changed or happened to the tenant when the payment problems occurred. In conclusion, we can build a number of machine learning models, but due to the poor quality of the original data, the best result (AUC of approximately 60%) is not good enough to be used in practice.

- *Is it possible to use subgroup discovery algorithm to find which kind of people are more likely to cause payment problems and then build a predictive model based on the most representative rules?*
  The subgroup discovery does find a subgroup of people who are more likely to cause payment problems for both datasets and the results are proven to be more interpretable compared with the machine learning models. And we can also successfully convert the mined rules to a predictive model by the method discussed in chapter 5. The models based on the discovered rules can't outperform stacking models, but they are as good as models made by single machine learning algorithms. As a result, it is recommended to apply

subgroup discovery to obtain more interpretable results along with the machine learning models.

- *Which supervised machine learning algorithm performs best with regard to this specific problem?*
  Random forests is assumed to produce the best results but it isn't for this particular problem. The "glmnet" and "avNNet" models generate the best models made by a single algorithm. Therefore, we can generalize that different algorithms could be suitable for different problems. In fact, there is no omnipotent algorithm in the world. It is wise to try as many algorithms as possible in order to find the most suitable one for a specific problem.

- *How can we interpret the results of both machine learning and subgroup discovery?*
  When it comes to the interpretability of machine learning models, one would like to consider models like logistic regression and linear regression in the first place. Because these models can be easily interpreted by the coefficient estimates. However, some other machine learning algorithms such as random forests and neural network works like a black box scheme, which means the results of these models are hard for people to explain. The rules of subgroup discovery are more straightforward and more interpretable to humans. The only thing we should be careful when we interpret the results of subgroup discovery is that we should interpret the rule with all conditions instead of one by one, because all of these conditions describe the whole picture of the subgroup while one single condition is not too meaningful. In summary, if more interpretable results are expected, subgroup discovery will be a better option, while machine learning dominates if the predictive ability is required.

By this thesis, some difficulties can also be summarized when people try to apply machine learning in practice. The first obstacle should be the data. Where can we find reliable data sources? How many rows does the dataset have? Is the quality of data good enough to fit a predictive model? After these three preliminary questions, we should also think about how we can manipulate the data and pre-process it in order to build workable models. Beyond this, we should also consider the methods of conducting feature engineering and model selection. More importantly, unbalanced distribution is a pretty common issue in real world data, so the last question is how can we conquer the unbalance of the real world datasets?

So far, a lot of work has been done in this thesis. But the story can be continued. For instance, in the experiments, we can try more algorithms like support vector machines, and adjust the coefficients of weighted averaging ensemble methods with more trials. In addition, we can also run feature selection by model based variable importance instead of filter based variable importance. Most importantly, we are confident that the results could break the bottleneck if we manage to get more data and find more predictive features.

References

[1] V. Gulshan, L. Peng, M. Coram, and et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*, 316(22):2402–2410, 2016.

[2] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. *ArXiv e-prints*, 2016.

[3] Y. Wu, M. Schuster, and et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *ArXiv e-prints*, 2016.

[4] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[5] Y. Kou, C. T. Lu, S. Sirwongwattana, and Y. P. Huang. Survey of fraud detection techniques. In *IEEE International Conference on Networking, Sensing and Control, 2004*, volume 2, pages 749–754 Vol.2, 2004.

[6] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 3(3):210–229, 1959.

[7] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

[8] L. Liu and M. T. Zsu. *Encyclopedia of Database Systems*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[9] A. Schneider, G. Hommel, and M. Blettner. Linear Regression Analysis. *Dtsch Arztebl International*, 107(44):776–782, 2010.

[10] S. H. Walker and D. B. Duncan. Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54:167–79, 1967.

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

[12] H. Hotelling. Analysis of a complex of statistical variables into principal components. *J. Educ. Psych.*, 24, 1933.

[13] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[14] T. G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 1–15, 2000.

[15] T. K. Ho. Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition - Volume 1*, page 278, 1995.

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

[17] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[18] F. Herrera and et al. An overview on subgroup discovery: Foundations and applications. *Knowl. Inf. Syst.*, 29(3):495–525, 2011.

[19] S. Helal. Subgroup discovery algorithms: A survey and empirical evaluation. *Journal of Computer Science and Technology*, 31(3):561–576, 2016.

[20] J. A. Nelder and R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3):370–384, 1972.

[21] B. R. Kowalski and C. F. Bender. Pattern recognition: a powerful approach to interpreting chemical data. *Journal of the American Chemical Society*, 94(16):5632–5639, 1972.

[22] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, 1996.

[23] V. T. Jack. Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of Clinical Epidemiology*, 49(11):1225 – 1231, 1996.

[24] W. Klösgen. Explora: A multipattern and multistrategy discovery assistant. pages 249–271, 1996.

[25] S. Wrobel. An algorithm for multi-relational discovery of subgroups. In *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 78–87, 1997.

[26] J. H. Friedman and N. I. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9(2):123–143, 1999.

[27] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas. Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2):111–117, 2006.

[28] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning*, 3:1157–1182, 2003.

[29] T. M. Phuong, Z. Lin, and R. B. Altman. Choosing snps using feature selection. In *2005 IEEE Computational Systems Bioinformatics Conference (CSB'05)*, pages 301–309, 2005.

[30] K. W. Bowyer, N. V. Chawla, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *CoRR*, abs/1106.1813, 2011.

Appendix I

Features	Descriptions
postcode	6-positie postcode
huisnummer	Huisnummer
MXleeftijd	Leeftijdsklasse
MXsentype	Ouderensegmentatie
MXctht	Consumententype
MXlevfas	Levensfase
MXmanvrw	Man en/of vrouw aanwezig
MXnvolw	Aantal volwassenen in het huishouden
MXopleid	Opleidingsniveau
MXwelstand	Welstandsklasse
MXinkomen	Gezinsinkomen
MXbrink	Besteedbare inkomensruimte
MXtwinkkw	Aantal Inkomens in het huishouden
MXprijs	Prijs boven kwaliteit
MXsockl	Sociale Klasse
MXchartype	Charitype
MXmedia	Mediatype
MXkwdb	Leest kwaliteitsdagbladen
MXpopdb	Leest populaire dagbladen
MXpostord	Postordergevoeligheidsindicator
MXDuurzaam	Duurzaamheidsindicator
MXactie	Actiegevoeligheid
MXcc	Bezit Creditcard
MXadopter	Adoptertypologie
MXauto	Bezit auto
MXauleas	Bezit Leaseauto

Table 41: Cendris data reference table

Features	Possible values (integer)	Range of actual values
MXleeftijd	1 - 14	18 - 85+
MXsentype	0 - 4	5 categories
MXctht	1 - 11	11 categories
MXlevfas	1 - 11	11 categories
MXmanvrw	1 - 3	3 categories
MXnvolw	1 - 2	1 or 2
MXopleid	1 - 7	7 categories
MXwelstand	1 - 5	5 categories
MXinkomen	1 - 5	5 categories
MXbrink	1 - 4	4 categories
MXtwinkkw	0 - 1	1 or 2(and more)
MXprijs	1 - 2	2 categories
MXsockl	1 - 5	5 categories
MXchartype	1 - 9	9 categories
MXmedia	1 - 5	5 categories
MXkwdb	0 - 1	Yes or No
MXpopdb	0 - 1	Yes or No
MXpostord	0 - 2	Low/medium/high
MXduurzaam	0 - 2	Low/medium/high
MXactie	0 - 4	5 categories
MXcc	0 - 1	Yes or No
MXadopter	1 - 5	5 categories
MXauto	0 - 1	Yes or No
MXauleas	0 - 1	Yes or No

Table 42: Possible values for features from Cendris data

Appendix II

		Actual class	
		Positive	Negative
Predicted class	Positive	44	345
	Negative	33	529

Table 43: Confusion matrix of model "glmnet1wo"

		Actual class	
		Positive	Negative
Predicted class	Positive	30	239
	Negative	47	635

Table 44: Confusion matrix of model "rf1wo"

		Actual class	
		Positive	Negative
Predicted class	Positive	18	118
	Negative	59	756

Table 45: Confusion matrix of model "knn2wo "

		Actual class	
		Positive	Negative
Predicted class	Positive	52	439
	Negative	25	435

Table 46: Confusion matrix of model "nb1wo"

		Actual class	
		Positive	Negative
Predicted class	Positive	46	362
	Negative	31	512

Table 47: Confusion matrix of model "nn1fswo"

		Actual class	
		Positive	Negative
Predicted class	Positive	50	422
	Negative	26	450

Table 48: Confusion matrix of model "aucwo"

		Actual class	
		Positive	Negative
Predicted class	Positive	52	431
	Negative	24	441

Table 49: Confusion matrix of model "waucwo"

		Actual class	
		Positive	Negative
Predicted class	Positive	51	396
	Negative	26	478

Table 50: Confusion matrix of model "mvwo"

		Actual class	
		Positive	Negative
Predicted class	Positive	46	331
	Negative	31	543

Table 51: Confusion matrix of model "sfourwo"

		Actual class	
		Positive	Negative
Predicted class	Positive	13	90
	Negative	15	227

Table 52: Confusion matrix of model "glmnet3w"

		Actual class	
		Positive	Negative
Predicted class	Positive	12	117
	Negative	16	200

Table 53: Confusion matrix of model "rf1w"

		Actual class	
		Positive	Negative
Predicted class	Positive	10	55
	Negative	18	262

Table 54: Confusion matrix of model "knn1w"

		Actual class	
		Positive	Negative
Predicted class	Positive	10	93
	Negative	18	224

Table 55: Confusion matrix of model "nb1fsw"

		Actual class	
		Positive	Negative
Predicted class	Positive	11	82
	Negative	17	235

Table 56: Confusion matrix of model "nn3w"

		Actual class	
		Positive	Negative
Predicted class	Positive	10	46
	Negative	18	271

Table 57: Confusion matrix of model "aucw"

		Actual class	
		Positive	Negative
Predicted class	Positive	11	56
	Negative	17	261

Table 58: Confusion matrix of model "waucw"

		Actual class	
		Positive	Negative
Predicted class	Positive	12	102
	Negative	16	215

Table 59: Confusion matrix of model "sallw"