Eindhoven University of Technology

MASTER

Predictive performance and discrimination in unbalanced classification

van der Zon, S.B.

Award date:
2016

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

Department of Mathematics and Computer Science
Eindhoven University of Technology

Master Thesis

Predictive Performance and Discrimination in Unbalanced Classification

Author
S.B. van der Zon
0877800

Graduation committee
prof.dr. M. Pechenizkiy (supervisor, DM)
dr. N. Zannone (assessor, SEC)
dr. A. Serebrenik (assessor, SET)

September 27, 2016

Abstract

In this thesis, we investigate two types of problems in classifier construction. First, we focus on the problem of class imbalance. We acknowledge that common accuracy metrics such as accuracy and precision are not appropriate to evaluate performance of a classifier for an imbalanced dataset, and therefore consider existing solutions including the ROC-curve and Kappa statistic, and extend the notion of the ROC-curve to a new single-value performance identifier "ROC-distance". We employ several methods that reduce class imbalance, using variable support for itemsets, and synthetically oversampling the minority class, using techniques that preserve the original characteristics of a dataset, while creating an equal amount of samples for both classes. Secondly, we focus on the problem of fairness and discrimination in classifiers. Classifiers that learn from socially sensitive data, can learn certain decisions that may be considered discriminatory, due to biased data or even discriminatory choices that are recorded, which may lead to structural discrimination by the classifier. We show methods that quantify and remove discrimination from the classifier. The influence of discrimination removal on classifier performance is evaluated and we propose a different sorting precedence than proposed by CBA2 (Classification Based on Associations), which deals better with discrimination removal than the original CBA2 method. We develop an association rule mining classifier. During the development of this classifier, we used Bayesian Networks to visualize decision characteristics of the classifier and acknowledge that this is an effective way of analysing certain properties of a classifier, tuning it for more optimal performance, and understanding of the dataset and classifier in general. Finally we test, evaluate and optimize our classification system on a real, non-trivial to predict dataset including both imbalance and discriminatory aspects. And show that our extensions to the existing approaches improve their performance.

Acknowledgements

First, I would like to thank my thesis advisor dr. Mykola Pechenizkiy, whenever I was in doubt about results or the direction of the thesis, he was able to motivate me and open my eyes to different interesting directions. dr. Pechenizkiy consistently helped me when I had questions, even during travelling or at late nights via email, which is greatly appreciated. Secondly, I would like to express my gratitude to the other experts on the committee, who were involved in the evaluation of this thesis: dr. Nicola Zannone and dr. Alexander Serebrenik. Without their valuable input, the validation of this thesis could not have been conducted successfully. I would also like to thank my fellow master student Bob Giesbers for countless helpful discussions. And my father dr. Ben van der Zon for reading my thesis and providing valuable feedback. Furthermore, I want to thank my brothers and friends, both at home and at the University, for accompanying me during most valued coffee breaks and times of relaxation, which gave me the energy to passionately work on my thesis. Finally, I express my greatest gratitude to my parents who gave me unfailing support and encouraged me throughout the years of my study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Classification and automated decision making have become very popular topics, and are utilized in many domains. It is used to recommend interesting items, make predictions, classify samples, propose matches between items, and more. In this thesis we investigate classifier performance in different settings for socially sensitive environments. The different settings that we study, and their relations, consist of two parts.

In part one, we investigate the problem of training a classifier on a dataset with an imbalanced class. A class is imbalanced when the attribute that we want to predict is not distributed equally among all samples in the dataset (e.g. 1 positive sample for each 100 samples instead of 50 positive samples for each 100 samples). This problem is well known in various disciplines including anomaly detection, fraud detection, medical diagnosis and other settings that involve a minority class that is to be predicted. For most machine learning algorithms it is much easier to build a classifier when the number of positive samples equals the number of negative samples. Take for example a dataset with 1000 records, of which 10 transactions are fraudulent. First, the machine learning algorithm has far less data to learn the characteristics of the the fraudulent transactions (minority class). This problem can be addressed by over-/under-sampling the data to create equal classes (i.e. creating additional artificial samples). Secondly, most performance measures include the number of true/false predictions to measure the quality of the classifier, according to this measure, always predicting false would yield 99% good predictions in the above mentioned example. This would not be a desired classifier, since it essentially only uses the distribution of the class to do its predictions (i.e. always predicting the bigger class). This problem can be addressed by using different measures that assign more value to true positive predictions than true negative predictions.

We propose and build a classification approach that uses association rule mining as the basis for our classifier. Association rule mining is the discovery of association relationships ($if \rightarrow then$)

among a set of items in a dataset, and has become an important data mining technique due to the descriptive and easily understandable nature of the rules. Although association rule mining was originally introduced by [1] to extract associations from market basket data, many more applications have risen for this technique (e.g. recommender systems, medical diagnosis and loan approval [22]). Association rule mining can be used to extract rules from a (historical) dataset, which are then used to build a classifier which can classify (or predict) new samples.

In part two, we introduce the notion of discrimination to our classifier framework, to establish the link to socially sensitive data. Discrimination is defined as a different treatment (decision), based on ones participation in a group, class or category, instead of individual properties. When socially sensitive data are involved, and human choices are recorded, the problem of discrimination arises. If discriminatory choices are recorded, we might teach a classifier this discriminating behaviour which would result in structural discrimination [19]. Take for example the case of credit approval at a bank. If we consider all historical decisions regarding credit approval, we often find discriminatory choices hidden in the data (e.g. in the case of credit approval, certain racial minorities or women are disadvantaged). If we teach a classifier from these examples, it may become a lot harder for certain minorities to get a loan at this bank. For this reason, it is important to be able to quantify the degree discrimination and act upon this. We discuss techniques to quantify and remove discrimination from a classifier, and investigate to what extend the accuracy of our classifier is affected by this.

One of the major tasks of an analyst is to find patterns in a dataset which may explain certain specific decisions. Association rules are intuitive to read, and therefore make a good basis for a tool that can help understanding a dataset. We developed a visualization tool using a Bayesian Network representation for a subset of associations rules (used by the classifier) that can help the analyst in accomplishing this task. The main problem encountered here was in the great number of association rules that were used to build the classifier. A Bayesian Network displaying a huge amount of edges is often cluttered and does not reveal any useful information, we therefore used criteria to select an interesting set to display using the Bayesian Network. This visual representation helps clarifying the choices that a classifier makes, but can also help understanding a dataset in general.

We investigate the influence of class imbalance and discrimination on classifier performance, and propose practical solutions to these problems. We show metrics such as the Kappa statistic, ROC-curve, and our proposed ROC-distance which are suitable for evaluating classifiers under these conditions. And propose methods to minimize their negative effects, including a weighted support measure that enables mining of an equal amount of itemsets for both classes and we extend the existing SMOTE [2] over-sampling technique to deal with categorical values. It is shown that discrimination removal affects classifier performance. We propose a technique

based on choosing rules by their predictive dependence, which is shown to minimize this performance decrease. We also construct a set of classifiers and test them on a non-trivial to predict dataset, where we show its performance on various aspects. Furthermore, we propose an additional pruning technique that can be added to the Apriori algorithm for performance improvement in case only class itemsets are desired.

This thesis is organized as follows. In Chapter 2, we motivate our problem, divide it into subproblems and briefly review all components and their connections to each other. In Chapter 3, we discuss the current state of the art and related work. In Chapter 4, we discuss classification in general and the problem of class imbalance where we propose techniques to enhance our classifier and performance measurements that are best suited class imbalance. In Chapter 5, we introduce Association Rules, their most important measures and the algorithms used to mine them. In Chapter 5.6, we show how Bayesian Networks can help understanding data and classifiers. In Chapter 6, we show results of experiments regarding the previously discussed topics on the speed dating dataset from [21]. In Chapter 7, discrimination with its related definitions and measures are discussed, and chapter 8 discusses the results of experiments regarding our improved classification approach on the speed dating dataset. Finally, Chapter 9 and 10 conclude with a discussion, conclusion and proposals for future work.

Chapter 2

Problem Statement

In the area of data mining, the main goal is to find patterns in data, such that we can better understand, analyse, and make decisions upon these data. In this thesis we restrict ourselves to classification; specifically automated decision making in a socially sensitive environment. Automated decision making is utilized in domains such as banking, insurance, match making, recommending interesting items, etc. The main objective is to learn from historical data; in order to predict future samples.

Many of such applications are used to predict an attribute that summarizes a subgroup of the population (e.g. fraud detection, loan approval and other anomaly detection applications). This is where the problem of class imbalance is introduced; it is often hard to teach a classifier based on an imbalanced attribute [2]. Intuitively this makes sense; because of the lack of positive samples in contrast to negative samples, less characteristics can be learned about the negative samples, resulting in less accurate predictions for this minority group. This problem is addressed in Chapter 4, where we show that common classifier performance measures such as accuracy and precision are inappropriate, since they assume equal classes. In this chapter, We also look into the various possibilities to solve this problem and propose solutions that minimize the effects of class imbalance by extending existing methods.

The problem of classification fairness, specifically discrimination, arises when the aim is to predict certain attributes about these minority groups in a socially sensitive dataset (e.g. banking, insurance and other datasets containing socially sensitive attributes such as race, gender, or various attributes about minority groups). There are many reasons for a classifier to yield unfair predictions regarding discrimination. First, many of these datasets consider human choices, which are known to contain discrimination, if such a dataset is used to train a classifier, the classifier will structurally discriminate these groups. Secondly, when a certain choice is made for a minority group, whether preferred or not, this does not incidentally mean

that everyone from this subgroup would like this treatment based on for example his/her ethnicity. Finally, biased data or incorrect labels can also lead to discrimination. This problem is addressed in Chapter 7, where we discuss methods to quantify the degree of discrimination of a classifier, and show that discrimination removal leads to predictive performance decrease.

During analysis of a classifier, it is often good to know what decision procedure is used by a classifier, as this can help understanding and improving the classifier, but also in learning general patterns/properties about the data. We propose a technique that uses Bayesian Networks to visualize our classifier.

In this thesis we investigate the effects of removing class imbalance (i.e. balancing the classes) and removing discrimination, while optimizing classifier performance. Also we look into techniques that help visualizing the decision procedure of a classifier. We do so by addressing the following research questions.

1. How does class imbalance influence classifier performance, and how can this influence be minimized? (Chapters 4-6)
2. How does discrimination removal influence classifier performance, and how can this influence be minimized? (Chapters 7-8)
3. What is a good way to visualize the decision procedure of a classifier? (Chapter 5.6)

In order to address these questions, a classification system is developed and evaluated on a real dataset.

Chapter 3

Background and Related Work

In this chapter, we briefly review the background and related work pertaining to the problems that are studied. First, we summarize popular classification methods which are later used to built a classifier, secondly, we summarize methods that deal with class imbalance, and finally we summarize literature on quantifying and removing discrimination from classifiers.

3.1 Classification methods

There are various classification methods available with many criteria for selecting appropriate methods to solve a classification problem, where the most important criteria are: is the domain continuous, discrete or categorical?; what is the simplest method that is able to solve the problem (complexity)?; are the attributes distributed linearly?; performance and number of dimensions; etc. We briefly discuss the most popular techniques and motivate our choice for classification association rules. The formal definition of classification is shown in Definition 1.

Definition 1. ***Classification*** is the problem of predicting to which category a new sample observation belongs, by extracting useful characteristics from a set of similar samples, who's categories are known.

Logistic regression [10] deals very well with linear continuous domains, but not with discrete/categorical domains. For domains that are non-linear, regularization methods exists and are relatively easy to implement. Furthermore, we get a probabilistic representation of the classification, which allows for ranking and understanding of the output. The existing model that is built can easily be updated in the future, but is mainly restricted to predicted a binary attribute.

Support Vector Machines (SVM) [9] are also restricted to continuous data and predicting a binary attribute. Output and results are incomprehensible in contrast to logistic regression.

SVMs can deal easily with a high number of dimensions and using an appropriate kernel, they can work well with non-linear data.

Naive Bayes (NB) [16] performs very well compared to its simplicity. NB however assumes independent attributes, but even when this assumption doesn't hold, NB often still performs surprisingly well. For scarce datasets, NB often isn't a good choice since it needs quite a lot of samples to be well-trained for good predictions.

Bayesian Networks [6] have the advantage that due to their rich structure, an expert can influence the decision process in order to get more efficient results. However for problems that have huge decision spaces, this is not suitable.

Decision Trees [20] work very well with different types of data (continuous/discrete/categorical and non-linear) and results are easy to interpret. Decision Trees are however sensitive to over-fitting and may get stuck in local minima (ensembles such as random forests and boosted trees can be used to prevent this). When the number of dimensions gets to high, Decision Trees may fail.

Nearest Neighbours [4] works well for a low number of dimensions and when a small number of samples needs to be classified. If a large number of samples needs to be classified, other methods are a better choice, since Nearest Neighbours requires all training data in order to do a classification, in contrast to most methods that build a model/function.

Neural Networks [26] gained quite some popularity over the last years and there are lots of libraries and implementations available. If implemented correctly they can perform well, however, picking the correct settings is difficult. Neural networks require a lot of training samples and converge slowly. The output and classifier is incomprehensible and not suited if one is interested in understanding the problem besides the classification.

Association Rules [1] are popular for their intuitively readable nature, which allows for exploring the data and investigating, for example, discrimination (Chapter 7). The most computationally intensive step in building an Association Rule classifier is mining the database's frequent itemsets, this problem scales in the number of dimensions and their domains. Association rules require categorical attributes, so continuous values should be discretized. Various useful applications such as market basket analysis, medical diagnosis, bio-medical literature, protein sequences, census data, logistic regression, fraud detection in web, CRM of credit card business etc. are discussed in [22]. Discretization methods that enable efficient mining of association rules in continuous domains are proposed by [24]. Association Rules and their

conversion to a classifier are discussed more thoroughly in Chapter 5.

3.2 Class Imbalance

The problem of class imbalance is explained by [11], also they show that class imbalance affects all types of classifiers (such as decision trees, Neural Networks and Support Vector Machines) and propose several basic re-sampling and cost-modifying methods. A classification system based on association rules is developed and discussed by [15], while their aim is to solve the problem of class imbalance using a cost-modifying method for the support of itemsets (i.e. enabling them to mine an equal amount of itemsets on both classes). An over-sampling approach, that creates artificial samples "between" existing samples is proposed by [2]. By doing so, they try to preserve the original characteristics as much as possible, while creating balanced classes. This paper also shows useful measurements for evaluating the performance of imbalanced classification.

3.3 Discrimination awareness

Discrimination in machine learning is becoming a more important research area, as predictions based on socially sensitive, historic data are becoming very popular. When these historical data are used to train classification algorithms, discriminative behaviour may be learnt as a side effect of biases or incompleteness of the data, or even discriminatory choices that are recorded. The formal definition of discrimination is shown in Definition 2.

Definition 2. ***Discrimination*** is defined as basing a treatment for people, on belonging to a certain group, such as race, gender and religion, instead of basing this treatment on properties of this individual.

The problem of discrimination is motivated by [13], where it is shown blind use of classifiers can result in discriminative classifier behaviour, they analyse these concepts on a crime suspect dataset. The fact that discrimination removal comes at the cost of reducing prediction accuracy is shown by [12], they also propose preprocessing techniques such as massaging and reweighing the dataset. However, [27], which focusses on establishing a normalization factor that enables comparison of different classifiers in a discriminative environment, mentions that the effects of massaging are not very precise and should be revised. The problem of explainable discrimination is discussed by [28] and [14], where they show that simply removing all discrimination from a classifier may introduce reverse discrimination, because some attributes that may seem discriminative can actually be explained (explainable discrimination). Extensions of the lift measure that are able quantify discrimination are established by [18] and

[23], enabling automatic discovery of discriminating association rules. Three classical machine learning algorithms (adaptive boosting, support vector machines, and logistic regression) are studied by [5], aiming to maintain high accuracy, while minimizing discrimination. Finally, [3], discusses data mining techniques, discrimination in general and also summarizes a number of the above mentioned papers.

Chapter 4

Classification and Class Imbalance

Classification is the process of building a model to predict unknown values. In this chapter we discuss several classification methods. Also we discuss methods that compensate for class imbalance (i.e. learning a classifier when one of the classes has less samples than the other). The results of these methods are then summarized in chapter 6. Classification is a supervised learning approach, this means that (in contrast to for example clustering), the labels of the training data are known on beforehand and a classifier is trained to specifically predict those labels.

4.1 Class imbalance

A dataset is imbalanced if the attributes to be classified are not distributed equally (i.e. there are significantly more samples for one classification category than the other). In most real-world datasets, the interesting anomalies are under-represented and the normal samples are over-represented. Also it is often the case that the cost of misclassifying an anomaly comes with much higher cost than misclassifying a normal sample. A few examples are airport screening, fraud detection but also recommending interesting advertisements; the cost of "correcting" a falsely classified normal sample is much lower in all these cases than "correcting" a falsely classified interesting sample.

In the following two sections, two classes of methods are explained that – in their own way – aim to represent the minority class with higher significance. First, we compare performance measures for classifiers and show why traditional measures such as classifier accuracy, precision and recall don't work, and propose better solutions. Secondly, we show two techniques that create artificial samples for the minority class. In Chapter 6, the results of these techniques are summarized.

4.2 Classifier performance (appropriate measures)

To understand why standard accuracy measures don't work for imbalanced classification problems, we first discuss some traditional performance measures and show why they are not appropriate. We then show measures that are appropriate and propose a new measure.

The most common way of quantifying the performance of machine learning algorithms is by using a confusion matrix as illustrated in Table 4.1 (for a 2 class problem). The columns represent the predicted class and the rows the actual class. TN (true negatives) and TP (true positives) represent the predictions that are correct. FN (false negatives) and FP (false positives) represent the predictions that are wrong.

	Predicted Negative	**Predicted Positive**
Actual Negative	TN	FP
Actual Positive	FN	TP

Table 4.1: Confusion Matrix

Classifier accuracy is defined as the number of correct predictions over all predictions (Definition 3). When we consider this performance measure for an imbalanced class classification problem it is easy to observe that the more unbalanced the class is, the easier it is to build a "good" classifier. For example, consider an insurance company dataset were detecting fraudulent claims is the goal. Suppose there is 1 fraudulent sample for each 100 samples; we could simply build a classifier that always predicts "non-fraudulent" and obtain an accuracy of 99%. Obviously this is not desired.

Definition 3. ***Accuracy*** is the number of predictions that are in agreement with the ground truth divided by the total number of predictions.

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{total population}}$$

For this reason, we should use measures that take into account the distribution of the class. We consider a number of popular measures in the remainder of this section and discuss why one is, or is not applicable for measuring the performance of a classifier for an imbalanced class.

4.2.1 Precision and recall

Precision and recall are also two broadly used measures for quantifying classifier performance. Precision resembles the portion of *relevant selected* items out of all *selected* items (Definition 4; the ratio of truth predictions over all predictions). And recall resembles the portion of *relevant selected* items out of all *relevant* items (Definition 5; the ratio of positives that are predicted).

Definition 4. ***Precision*** measures how many of the predicted positives are actually true positives.

$$\text{precision} = \frac{|\{\text{actual positives}\} \cap \{\text{predicted positives}\}|}{\{\text{predicted positives}\}} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Definition 5. ***Recall*** measures how many of the actual positives we managed to recall.

$$\text{recall} = \frac{|\{\text{actual positives}\} \cap \{\text{predicted positives}\}|}{\{\text{actual positives}\}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Note that precision and recall on themselves are not very meaningful, since one could create a classifier with high precision / low recall (e.g. make one correct prediction, predict the rest negative) and vice versa (e.g. predict everything positive), hence we conclude that both should by high as an indication of a good classifier. We can see that this method gives a distorted view of performance when the class is not distributed equally. Consider again the $1 : 100$ distributed fraudulent dataset example. When we would predict all positives right (1% of the data), and make 3% mistakes (FP), precision would be 0.25, even though this may be a very good classifier. Recall is actually independent of the majority attribute (TN and FP) and considers only the actual positives, so could be a good measure, but as noted before, recall alone is not enough. We make use of recall in a successful measure later (ROC-Curve).

4.2.2 F1-score

The F_1-score is a summarized version of precision and recall and can be seen as a weighted average between the two. The F_1-score (Definition 6) is at most 1 (in the best case) and at least 0.

Definition 6. ***F_1-score*** summarizes precision and recall in one number, the F_1-score can be at most $max(\text{precision}, \text{recall})$, and the further the two are apart, the lower the F_1-score will

be.

$$\text{F}_1\text{-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Note that the F_1-score is the harmonic mean of precision and recall, which results in at most the average of the two, and the more they differ, the lower the value. Still, this method does not take into account the distribution of the class. Which results in a score that is penalized too much by the relatively low precision of an imbalanced classification task.

4.2.3 Kappa statistic

The Kappa statistic is a performance metric that compares an observed accuracy with an expected accuracy. The Kappa statistic is often used to compare the performance of different raters (when all raters agree, Kappa=1, the more disagreement, the lower the statistic). In evaluation of machine learning algorithms, one of the raters is chosen to be the ground truth, and the other the classifier. The observed accuracy is simply the accuracy of the classifier (number of truth predictions, false and positive, divided by all predictions). And the expected accuracy is calculated based on the marginal frequencies of the positives and the negatives, thus takes into account the distribution of the class (via the ground-truth rater), which makes this metric suitable for measuring performance of classifiers for imbalanced datasets. The Kappa statistic is defined in Definition 7, and is one of the metrics used in our experiments.

Definition 7. ***Kappa statistic*** - The Kappa statistic is used to measure the agreement among a number of different raters. In machine learning, one rater is the ground truth, and the other one is the classifier. The more agreement, the higher the Kappa statistic (up to 1) the less agreement, the lower the Kappa statistic.

$$\text{Kappa} = \frac{\text{observed accuracy (Definition 3)} - \text{expected accuracy}}{1 - \text{expected accuracy}}, \text{ where}$$
$$\text{expected accuracy} = \frac{\text{marginal frequency positives} + \text{marginal frequency negatives}}{\text{total population}}.$$

4.2.4 ROC space

The Receiver Operating Characteristic (ROC) curve is a graphical plot that is used to illustrate the performance of classifiers with various settings regarding their discriminative power. The x-axis of the plot represents the rate of false-positives (FPR) and the y-axis represents the

rate of true-positives (TPR, also known as *recall*) for a classifier. The curve is constructed by plotting classifiers with various sensitivity thresholds (Figure 4.1). An optimal classifier performance has a 1 true-positives rate and a 0 false-positives rate ($(0, 1)$ in the plot). A random classifier is positioned at the diagonal defined by the function TPR = FPR. Note that any classifier that consistently scores below this line could be inverted to obtain a good predictor. Since this metric takes into account rates that depend on the distribution of the class instead of absolute values, it is a good metric for classifiers that predict unbalanced classes.

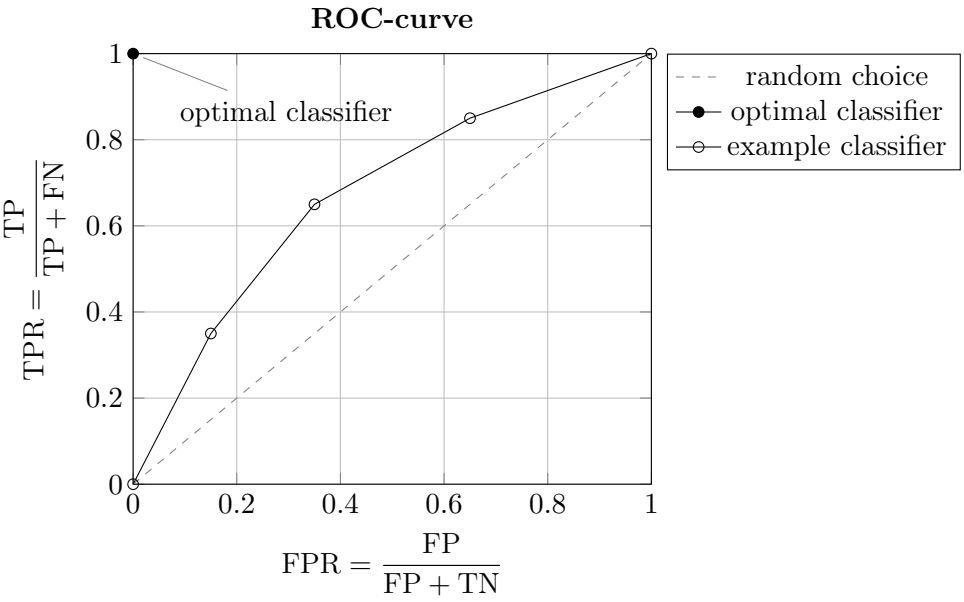


Figure 4.1: ROC Space, 5 different classifiers are evaluated (black solid line). The black dot in the upper-left corner indicates the optimal classifier, which predicts all positives correctly and predict no negatives wrong. The dashed diagonal grey line represents random guesses, all points that lie on this line are "just as good as flipping a coin".

However, in most practical scenarios it is desired to express the performance of a classifier in a single number, instead of a position on a plane. For this reason we propose the *ROC-distance* metric (Definition 8), which measures the distance from the classifier on the ROC plane to the top-left corner. An example is illustrated in Figure 4.2. This metric is also used to evaluate our experiments later on. First an acceptable subset of classifiers is selected, based on the summarized value provided by the ROC-distance metric, which can then be compared in the ROC-plane.

Definition 8. ***ROC-distance*** is defined as the distance from the sample the top-left corner

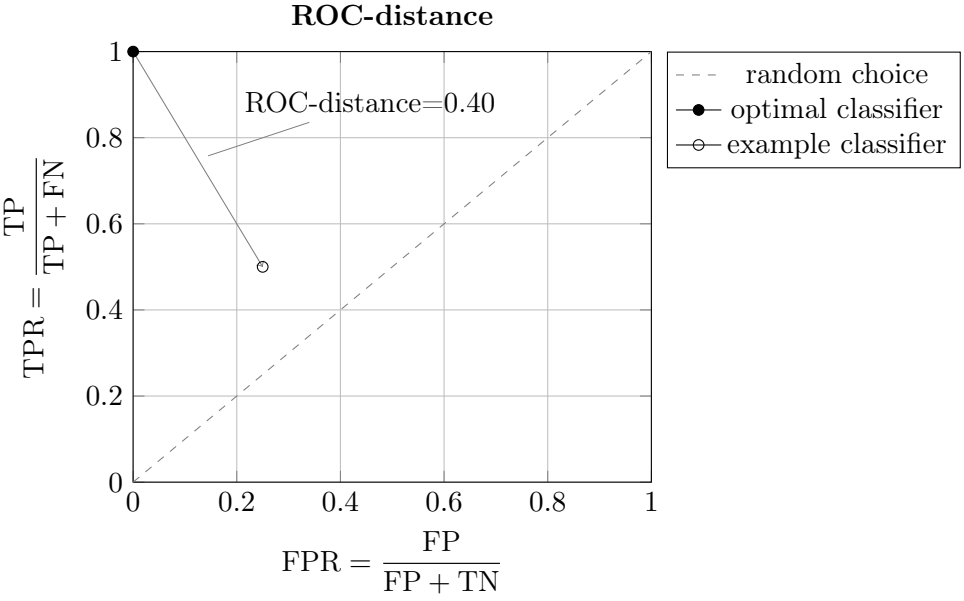


Figure 4.2: The ROC-distance of an example classifier $(0.25, 0.5)$.

$$\text{ROC-distance} = \frac{\sqrt{0.25^2 + (1 - 0.5)^2}}{\sqrt{2}} = 0.40$$

$(0, 1)$ in the ROC space.

$$\text{ROC-distance} = \frac{\sqrt{\text{FPR}^2 + (1 - \text{TPR})^2}}{\sqrt{2}}.$$

Note that this is simply the Pythagorean theorem, normalized over the maximum distance from $(0, 1)$ to $(1, 0)$ $(\sqrt{2})$.

4.3 Oversampling

Under- and over-sampling are both techniques used to change the distribution of a class which is over- or under-represented respectively. Under-sampling removes samples from the majority class, ideally aiming at destroying as less information as possible. Over-sampling artificially creates samples for the minority group, ideally aiming at preserving the characteristics of the dataset. We have combined two techniques in order to find an optimal solution. The first technique that we employ is SMOTE [2] (Synthetic Minority Over-sampling Technique), which randomly creates new (minority) samples "between" existing (minority) samples. The second technique CBA2 (Classification Based on Association) [15] is more tailored to our final approach that uses Association Rules (chapter 5) and aims at finding an equal amount of itemsets for both classes; in this way an equal amount of rules is created for both the minority and the majority class.

4.3.1 SMOTE

SMOTE is an over-sampling technique that creates synthetic samples for the minority class, while preserving its characteristics. First a SMOTE value n is defined. This value determines how many times the minority class is over-sampled (e.g. when we have class imbalance $1:10$, setting $n = 9$ will create 900% extra minority class samples, resulting in a new distribution of $1:1$). This technique tries to preserve the characteristics of the dataset by creating n synthetic minority samples for each sample *between* the sample itself and randomly selected samples from its k nearest neighbour. By modifying k, the variety of synthetic samples can be controlled; the lower the k, the less variety (only close samples are used), the higher the k, the more variety (further samples are also used). This is illustrated in Algorithm 1.

Chawla et al. [2] only consider numeric attributes, where the difference between a minority sample and a neighbour can easily be calculated (line 16 of Algorithm 1). Since we also want to employ this technique to categorical values, we propose to choose one of the categories at random. Note however that this is only possible for attributes that are not correlated/derived (e.g. when attribute a_{relStat} = single mother, which depicts relationship status equals *single mother*, we don't want attribute $a_{\text{gender}} = male$). Derived attributes can often easily be re-derived. For correlated attributes other heuristics should be considered, which take into account this correlation (e.g. suppose attributes a_{job} = engineer and a_{gender} = male, are correlated, we want our synthetic samples to keep reflecting this correlation). The results of this technique are summarized in Chapter 6.

Algorithm 1 SMOTE

```
1: function SMOTE(T, n, k)                ▷ Array T minority class samples, SMOTE n%,
2:                                                                  ▷ k nearest neighbour.
3:     if (n mod 1) ≠ 0 then               ▷ We only use a portion of all minority samples.
4:         shuffle T
5:     S = ∅                                                        ▷ The synthetic samples.
6:     for i = 0 to (n · |T|) do                          ▷ Create t · n synthetic samples.
7:         S = ∪ Synthetic(T[i])
8:     Answer: s                          ▷ |T| · n synthetic minority samples, based on T.
9:
10: function SYNTHETIC(t, k)               ▷ Basis sample t, k number of nearest neighbours.
11:     N = compute k nearest neighbors(t, k)
12:     nn = N[rand(0, k − 1)]             ▷ Random nearest neighbour out of the k nearest.
13:     s = ∅                                                        ▷ The synthetic sample.
14:     for a = 1 to |t| do                                ▷ For each attribute of base sample t.
15:         d = t[a] − nn[a]               ▷ Difference between sample t and nearest neighbour nn.
16:         r = rand(0, 1)
17:         s[a] = t[a] + r · d            ▷ Save the synthetic value to the synthetic sample s.
18:     Answer: S          ▷ Synthetic minority sample based on t and its k nearest neighbours.
```

4.3.2 CBA2

CBA2 [15] is a method used to create a rule based classifier, that deals with class imbalance and selects the rules that are most likely to yield the correct prediction. In this paragraph we focus on the creation of *balanced* itemset only. In Chapter 5, the selection of rules for classification will be discussed. Roughly speaking, when we have an equal amount of itemset (Definition 11 in Chapter 5), for each class-item, then an equal amount of association rules (Definition 11) for each class-item can be created. Obviously, when dealing with an imbalanced class, the itemsets will also follow this distribution, resulting in less itemsets for the minority class, and more itemsets for the majority class, thus finding only a few rules for the minority class. In order to solve this problem Liu et al. [15] propose a technique that uses a variable support. The support of an itemset, is simply its occurrence count throughout the database, divided by the number of transactions in the database (Definition 9).

Definition 9. ***Support*** - Let I be an itemset and let $t \in D$ be the transactions of a database. The support of the itemset with respect to D is defined as the number of transactions that

contain I, divided by the number of transactions in D. More formally:

$$\text{support}(I) = \frac{|\{t \in D \mid t \cap I \neq \emptyset\}|}{|D|}, \text{ where } I \text{ is an itemset.}$$

Note that the support of an empty itemset is 1.0, and the more specific an itemset is the lower the support (i.e. let I be an itemset and $I' \subseteq I$ then $\text{support}(I') \geq \text{support}(I)$). This is due to the downward-closure property of the support of itemsets.

Definition 10. ***Weighted support*** - Let I be an itemset, let $t \in D$ be the transactions of a database and let C be the set of class-items. The weighted support of the itemset with respect to C and D is defined as the original support, multiplied by the support of the class-item in I, if I does not contain a class-item, then the lowest support is used. More formally:

$$\text{wSupport}(I, C) = \text{support}(I) \cdot \min(\text{support}(C \cap I), \text{lowestSupport}(I))$$

By using this variable support, a lower minimum support threshold is used for the minority itemsets (an itemset that contains a minority class-item). And a higher minimum support threshold is used for the majority itemsets. More specifically the new *weighted* support is defined as the support of the class-item (if the itemset has one) times the support of the itemset (Definition 10). Using this support, we extract an approximately equal amount of itemset for both classes, thus enabling us to create an equal amount of rules for both classes. The following chapter (Chapter 5) discusses how to use these itemsets to construct a classifier. In Chapter 6, the results of this technique are summarized.

Chapter 5

Association Rules

To investigate the effects of class imbalance, and later discrimination, a classification system is built. Our classification approach uses association rule mining as its basis. Association rule mining is a technique that is used to find relations between attributes in large databases. An association rule ($X \rightarrow Y$) is an implication consisting of two parts: the *antecedent* (X) and the *consequent* (Y). The antecedent is the prerequisite that has to be true in order to infer the consequent with a certain confidence. The first step in finding association rules, is finding the frequent itemsets in the database, from which the association rules can be constructed, this can be done using algorithms such as Apriori [1], Eclat [25], FP-growth [8] and others [7]. Our approach uses Apriori because of its relatively simple implementation and good performance. This Chapter is organized as follows: The necessary definitions are explained, the most important quality measures of association rules are discussed, methods to extract frequent itemsets are shown, followed by methods to extract rules from the itemsets. Finally we propose a classifier visualization technique using Bayesian Networks, which can help in understanding the classifier.

5.1 Definitions

In order to define an association rule, we start with its smallest building blocks: items. An item is defined as an attribute-value pair, for example consider a database with 2 attributes (columns); gender and age, with domain(gender) = {male, female} and domain(age) = {[0 − 20], [21 − 40], > 40}. gender = male would be a valid item. More items can now make up an itemset, for example {[gender=male], [age=[21 − 40]]}, this itemset represents the intersection of the items (males with an age between 21 and 40). We only consider disjoint attributes, so an itemset with multiple items for one attribute would not have any meaning (a person cannot be a male and a female at the same time). An association rule consists of two itemsets; the *antecedent* and the *consequent* itemset, intuitively an association rule can be read as follows: if the antecedent holds, then the consequent can be inferred. Definition 11 formally defines items, itemsets and association rules.

Definition 11. ***Association Rule*** - Let $I = i_1, 1_2, ..., 1_n$ be a set of n attribute-value pairs called *items*. And let $D = t_1, t2_2, ..., t_m$ be a set of transactions called the *database*. Each transaction in D consists of some items in I. An association rule is defined as an implication of the form $X \rightarrow Y$, where $X, Y \subset I$ and $X \cap Y = \emptyset$, X and Y are two sets of items, also known as *itemsets*. X is called the *antecedent* and Y the *consequent*.

Consider the example mentioned before. A valid rule could be $X \rightarrow Y$, where X = {gender=male} and Y = {age=[0 − 20]} (if a sample is male, then with a certain confidence his age is between 0 − 20). In the next section, we discuss measures about the quality of rules.

5.2 Measures

There are a number of aspects to consider when choosing a good association rule, for example, one might be interested in knowing how often a rule can be used to predict some item, or the certainty of this prediction. In order to quantify these intuitive measures, the *support*, *confidence* and *lift* (Definitions 12, 13 and 14 respectively) of a rule are defined. The *support* measures how applicable a rule is to the dataset (for which portion of the data does it apply), the *confidence* measures the likelihood of an association rule to yield the correct prediction, given that its antecedent is satisfied and lift measures the degree of dependence between the antecedent and the consequent. Table 5.1 illustrates these concepts with some examples.

Definition 12. ***Support (AR)*** measures the chance that we might encounter this rule in the database. Let X and Y be itemsets. Then,

$$\text{support}(X \to Y) = \frac{|X \cup Y|}{|D|} = \text{support}(X \cup Y).$$

Definition 13. ***Confidence*** measures the chance that given the antecedent, the consequent itemset can be inferred. Let X and Y be itemsets. Then,

$$\text{confidence}(X \to Y) = \frac{\text{support}(X \to Y)}{\text{support}(X)}.$$

To put this in a statistical notation, the confidence can be interpreted as the conditional probability $P(E_Y|E_x)$.

Definition 14. ***Lift*** measures the degree of dependence between the antecedent and the consequent. Let X and Y be itemsets. Then,

$$\text{lift}(X \to Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X) \cdot \text{support}(Y)}.$$

Note that when the lift would be equal to 1, the antecedent and the consequent occur independently from each other, i.e. X would not contain any information about Y. When $lift > 1$ a greater association is implied, this helps selecting rules for classification.

#	**A**	**B**	**C**
1	a	1	x
2	a	1	x
3	a	1	x
4	a	1	x
5	a	1	y
6	a	1	y
7	b	1	z
8	b	1	z
9	c	2	x
10	c	3	x

Table 5.1: A database D containing 10 transactions with attributes A, B and C, with $dom(A) = \{a, b, c\}$, $dom(B) = \{1, 2, 3\}$ and $dom(C) = \{x, y, z\}$. Lets consider the following two example rules to illustrate the concepts above, $r_1 = [A{=}a, B{=}1] \Rightarrow [C{=}x]$ and $r_2 = [A{=}a, B{=}1] \Rightarrow [C{=}y]$.

$$\begin{aligned}
\text{support}(r_1) &= \frac{|[A{=}a, B{=}1, C{=}x]|}{|D|} = \frac{4}{10}, \\
\text{support}(r_2) &= \frac{|[A{=}a, B{=}1, C{=}y]|}{|D|} = \frac{2}{10}, \\
\text{confidence}(r_1) &= \frac{\text{support}(r_1)}{\text{support}([A{=}a, B{=}1])} = \frac{2}{3}, \\
\text{confidence}(r_2) &= \frac{\text{support}(r_2)}{\text{support}([A{=}a, B{=}1])} = \frac{1}{3}, \\
\text{lift}(r_1) &= \frac{\text{support}(r_1)}{\text{support}([A{=}a, B{=}1]) \cdot \text{support}([C{=}x])} = \frac{10}{9} \text{ and} \\
\text{lift}(r_1) &= \frac{\text{support}(r_2)}{\text{support}([A{=}a, B{=}1]) \cdot \text{support}([C{=}y])} = \frac{5}{3}.
\end{aligned}$$

This example shows that though we are more confident about a prediction coming from r_1, r_2 has a higher lift and therefore its antecedent and consequent have higher dependence on each other.

5.3 Mining Itemsets

There are vairous algorithms that mine itemsets. The most popular algorithms are Apriori [1], Eclat [25] and FP-growth [8]. Eclat stores a list for each item containing the ids of the transactions that have this item, then in the next step, it can intersect each of these lists in order to create lists of size 2, it continues this until the intersections yield empty sets. The main drawback of this method is the huge amount of memory these lists take. FP-growth scans trough the database while building a tree representation, that can efficiently be used to lookup the frequent itemsets. Other algorithms with more specific solutions are discussed by [7]. We choose Apriori because of its relative simple implementation and good performance.

5.3.1 Apriori

Our implementation uses the Apriori algorithm to obtain all itemsets from a database that satisfy a minimum support. The algorithm does this in an efficient way; it first find the size-1 itemsets that satisfy the minimum support. Then, for any size-2 itemset to satisfy the minimum support, the size 1 itemsets out of which this size 2 itemset is made of, should be present in the previous step. This principle holds for all size-k itemsets, due to the downwards closure property of the itemset' support; a size-k itemset can only satisfy the minimum support if *all* its size $k-1$ subitemsets satisfy the minimum support. By using this assumption, the algorithm can very efficiently prune the search space. A more detailed description is given in algorithm 2.

5.3.2 Apriori improvement

When the Apriori algorithm has completed, a huge amount of itemsets is generated, however, only a portion of these itemsets is actually useful to obtain association rules; namely the ones that contain the class-item. We propose an extra pruning step in each cycle of the Apriori algorithm that eliminates itemsets that will never evolve into an itemset containing the class-item. Other algorithms that mine only class association rules exist [17], and are worth comparing against this method, this is however not within our scope.

For an itemset to satisfy the minimum support *and* the criterion that it must eventually possibly contain the class-item, the following two claims can be made:

1) (used by Apriori) all of the itemset's sub-itemsets must satisfy the minimum support,

2) (our proposal) if itemset $I \in K$, where K is all size-k itemsets, does not contain any of the class items $c \in C$, then for at least one of the class-items c, all subsets of $I \cup \{c\}$

Algorithm 2 Apriori

1: **function** APRIORI(D, minsup)
2: $L_1 = \{\text{large 1-itemsets}\}$
3: **for** $k = 2; L_{k-1} \neq \emptyset; k{+}{+}$ **do**
4: $C_k = \{(p, q) \in (L_{k-1} \times L_{k-1}) \mid |p \cap q| = 1 \wedge |\text{attrs}(p_{k-1}) \cap \text{attrs}(q)| = 1\}$*
5: **for** $(p, q) \in C_k$ **do**
6: $c = p \cup q$ ▷ c is now a candidate with size exactly k.
7: $s_k = \text{subsets}(c, k-1)$ ▷ Get all $k-1$ subsets.
8: **if** $s_k \cap L_{k-1} = \emptyset$ **then** ▷ If one of the subsets of the candidate does not
9: **continue** ▷ occur in previous k-itemsets, move on to the next
10: **if** $\text{support}_D(c) < \text{minsup}$ **then** ▷ (due to downward closure property).
11: **continue**
12: $L_k = L_k \bigcup \{c\}$
13: Answer: L
14: * Take the Cartesian product of the $k-1$ itemsets, such that only the tuples that together have exactly one new item that is not from any attribute that already exists in either one of them.

with size k *must* be in K. Moreover, we want to know for itemset I, not containing a class-item, if it can eventually evaluate into an itemset that will contain a class-item. For this to be true, all subsets of $I \cup \{c\}$ must be in K, for at least one $c \in C$.

This extra pruning step is described in more detail in Algorithm 3.

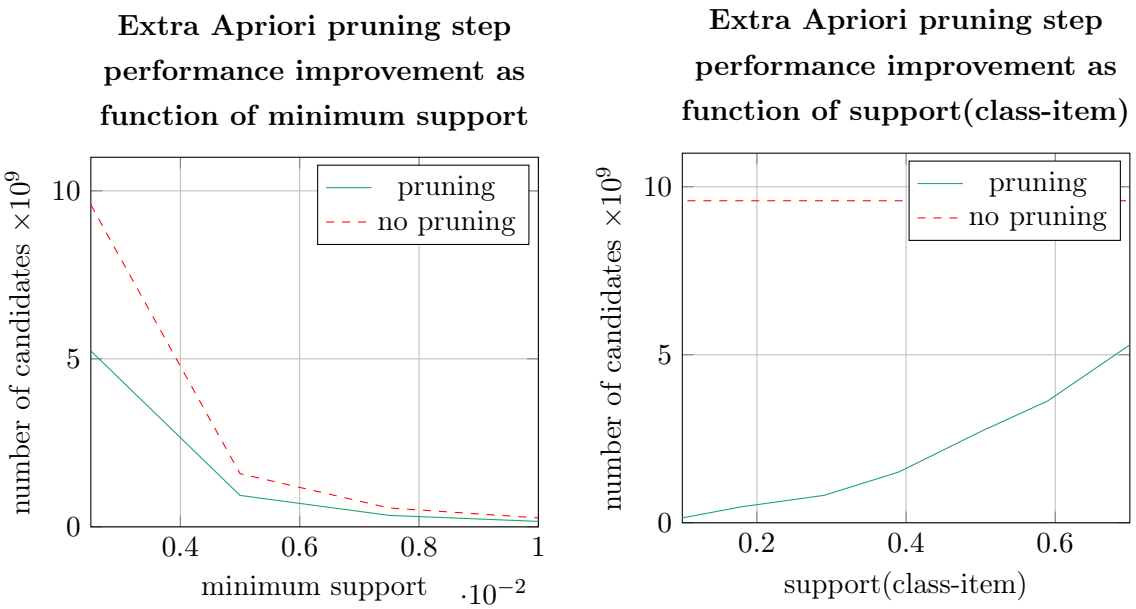
During our evaluation of this method, we notice that mining the itemsets for each class-item separately (afterwards combining the results) yields even faster results. For this reason, we propose that this method is most efficiently employed by mining the itemsets for each class-item individually. In Figure 5.1a, we show the substantial performance improvement that our pruning step achieved. We do so by showing the performance improvement as a function of the minimum support. It can be observed that the lower the minimum support is, the greater the improvement. In Figure 5.1b, we show the impact of the support of the class-item, note that performance improvement improves exponentially with the support of the class-item. From this experiment, we conclude that the more sparse an attribute is (the more items it contains, thus the lower the support), the greater the performance improvement is.

Furthermore, we want to emphasize that this method is worth further investigation, it may also yield better results when one wants to obtain *all* itemsets. This could simply be done by running the Apriori algorithm with our extra pruning step for each item in the dataset, and finally merging the results. Another advantage is that the itemsets for each item can be

Algorithm 3 Extra Apriori pruning step

```
1: function PRUNE(K, C)                          ▷ Set of all size-k itemsets K and
2:                                               ▷ set of all class-items C.
3:     for I ∈ K do
4:         keep = false                          ▷ We need to find a reason to keep I.
5:         for c ∈ C do                          ▷ For each class-item
6:             if (c ∩ I) ≠ ∅ then               ▷ If I contains a class-item,
7:                 keep = true                   ▷ keep I by definition.
8:                 break                         ▷ No need to look for more reasons to keep I.
9:             if I has attribute c.attribute then   ▷ Already contains same attribute,
10:                continue                      ▷ with different value, continue to next class-item.
11:            S = k-1 subsets of (I ∪ {c})      ▷ Construct all itemsets that need to be checked.
12:            if |S ∩ K| = (k + 1) then         ▷ If all exist,
13:                keep = true                   ▷ keep I.
14:                break                         ▷ No need to look for more reasons to keep I.
15:        if !keep then
16:            K = K \ I
17:    Answer: K
18:    // K now only contains itemsets that contain the class item, or might in the future.
```

mined in parallel. Note that this will create duplicate itemsets (thus wasted time), but still may be faster due to the exponential improvement of our candidate pruning.



(a) The performance improvement shown as a function of the minimum support. The performance improves exponentially as the minimum support decreases. In this experiment, all itemset are mined for both class-items.

(b) The performance improvement shown as a function of the support of the class-item. The performance improves exponentially as its support decreases. In this experiment, all itemsets are mined for one single class-item, with a varying support. All itemsets are mined with minimum support 0.0025

Figure 5.1: Figures (a) and (b) show the performance improvement of the extra Apriori pruning step as a function of the minimum support and the class-item's support respectively. We see that in both cases the performance improvement is substantial (even exponential). Note that when an attribute is really sparse, i.e. containing a lot of low support items, this technique is able to mine the itemsets for the attribute even more efficiently.

5.4 Extracting Association Rules

In order to extract association rules from the frequent itemsets, a class-attribute is first defined, which will be the consequent of each rule. When this attribute is defined, we simply scan over all frequent itemsets while converting into rules the itemsets that satisfy two conditions: 1) having the class-attribute and 2) satisfying the minimum confidence. This is illustrated in Algorithm 4. Note that this is just one iteration over all frequent itemsets, since both supports in line 10 are known already. We now have a set of all association rules that satisfy a minimum confidence and prediction class (consequent).

Algorithm 4 Extract Association Rules

1: **function** ExtractARs($I, C,$ minConf) ▷ Set I consisting of all itemsets,
2: ▷ set C consisting of the class items,
3: ▷ and minimum confidence threshold minConf.
4: $R = \emptyset$ ▷ The rules
5: **for** $i \in I$ **do** ▷ for each itemset i in I
6: $Y = I \cap C$ ▷ What is the consequent?
7: **if** $Y = \emptyset$ **then** ▷ If consequent is empty, continue to next itemset.
8: **continue**
9: $X = I \setminus Y$ ▷ What is the antecedent?
10: confidence $= \frac{support(I)}{support(X)}$ ▷ both supports known already, instant look-up
11: **if** confidence $<$ minConf **then** ▷ If confidence threshold is not satisfied,
12: **continue** ▷ continue to next itemset.
13: $R = R \bigcup \{\text{rule}(X, Y, \text{confidence})\}$
14: Answer: R

5.5 Building and using a classifier

When all rules for a certain class are found, a classifier is constructed. There are many ways to use the rules to predict new samples. CBA [15] uses an approach that selects the most confident rules first. When a new sample is offered to the classifier for prediction, it will iterate over its rule base to see if there is a rule that matches this new sample. When such a rule is found, it is used to predict the sample. When no rules are found, we use the class's distribution to predict the new sample. The exact sorting order is defined by Definition 15.

Definition 15. ***CBA precedence*** - Given two rules r_i and r_j, r_i precedes r_j if

1) the confidence of r_i is greater than that of r_j, or

2) their confidences are the same, but support(r_i) > support(r_j), or

3) both their confidences and supports are the same, but r_i is generated earlier than r_j.

Algorithm 5 illustrates the algorithm used to predict samples. The chance of over-fitting when using high confident rules is higher, because larger rules (i.e. more items) are more likely to have a higher confidence, we therefore propose a sorting based on the lift (dependence of the antecedent and consequent) of the rule, this is evaluated in Chapter 6.

Algorithm 5 Predict using ARs

```
1:  function PREDICT(t, R, C,t)
2:      R = sort(R)                                   ▷ According to precedence
3:      found = false
4:      for r ∈ R do                                  ▷ for each rule r in R
5:          if R.antecedent ⊈ t then                  ▷ If rule cannot be used for this sample,
6:              continue                              ▷ continue to next rule.
7:          found = true
8:          prediction = r.consequent
9:          confidence = confidence(r)
10:     if !found then                                ▷ If no rule was found, use distribution of C
11:         prediction = getMaxSupportItemset(C)      ▷ Predict the itemset that occurs most.
12:         confidence = support(prediction)
13:     if prediction ≠ c_t then
14:         confidence* = −1
15:     Answer: confidence                            ▷ How confident we are that this sample belongs to c_t.
16:                                                   ▷ Negative confidence if classified not c_t.
```

5.6 Bayesian Networks

In our approach, Bayesian networks are used to visualize certain characteristics of the classifier. This can aid the analyst greatly in the search for classifier flaws and understanding the classifier or the data in general. A Bayesian network is a graph representation of random variables (vertices), connected by their dependencies (edges). The vertices in the graph represent the antecedent and the consequent itemsets of the rules, and the edges represent the implication of the rule. The strength of an edge represents the confidence of the rule, and the colour of an edge encodes the degree of discrimination (discussed in Chapter 7). Furthermore we use the vertices' size to encode the support of an itemset.

Since the amount of rules that is mined by the algorithms is huge, a selection of interesting rules should be made. The analyst can determine the view:

1) the most general rules (high support),

2) the most, confident rules (high confidence),

3) the most discriminating rules, or

4) the most frequently used rules in classification.

Our implementation first selects an interesting subset based on one of these criteria, and then recursively "grows" the graph's branches to a certain maximum user specified depth. This is illustrated in Algorithm 6.

5.6.1 Bayesian Network examples

To illustrate the concept explained, we show two examples (both using the dataset explained in Section 6.1.1). First we show in Figure 5.2, a Bayesian network containing the most general (high support) rules. We show the top 5 rules that predict false, and the top 5 that predict true. Next we show in Figure 5.3, a Bayesian network containing the most confident rules. Again both top 5's are shown. Furthermore, the Bayesian Networks are used in the results section to visualize the classifiers that are obtained in the experiments.

Algorithm 6 Grow Branches

```
1: Construct base graph G, this is the interesting top subset of rules we want to know more
   about. Itemsets are represented by nodes, and the implications are represented by edges.
2: let (V, E) ∈ G be the vertices and edges of graph G.
3: for e ∈ E do
4:     GrowBranches(e.in, G, d)
5: function GrowBranches(v, G, d)          ▷ Vertex v, that we grow, base graph G,
6:                                                          ▷ depth d we want it to grow.
7:     let (V, E) ∈ G be the vertices and edges of graph G.
8:     R = extract rules that have consequent v
9:     for r ∈ R do
10:        v_1 = r.antecedent
11:        v_2 = r.consequent
12:        if v_1 ∉ V then
13:            V.add(v_1)
14:            GrowBranches(v_1, G, d − 1)
15:        if v_2 ∉ V then
16:            V.add(v_1)
17:        if e ∉ E then
18:            V.add(v_1)
```

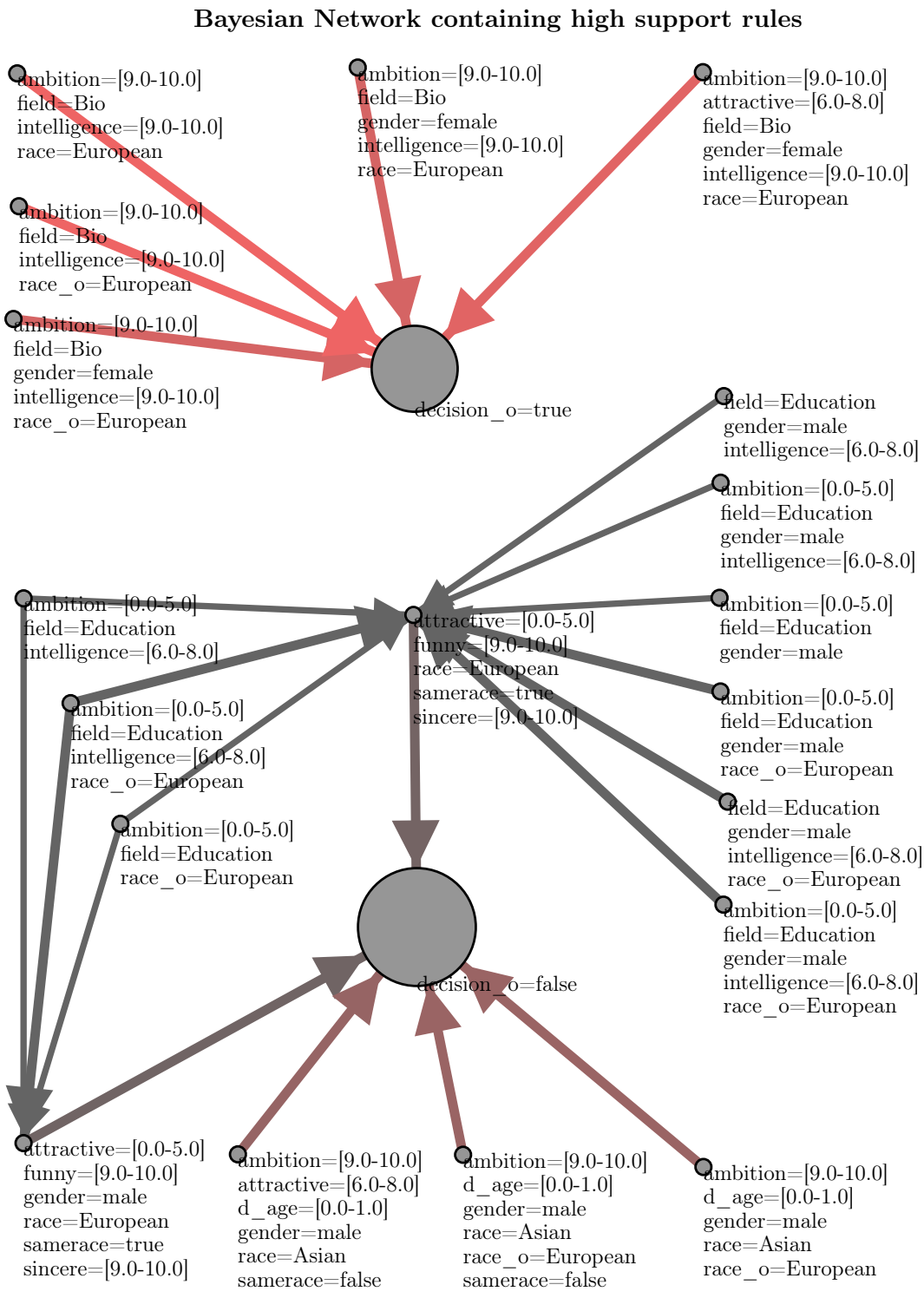


Figure 5.2: Bayesian network containing the most general rules, mined from the speed dating dataset (discussed in Chapter 6.1.1). Take for example the bottom-left rule, which represents the association between European males who do not find themselves attractive, are funny, are sincere, date someone that is also European and not being liked on a speed date.

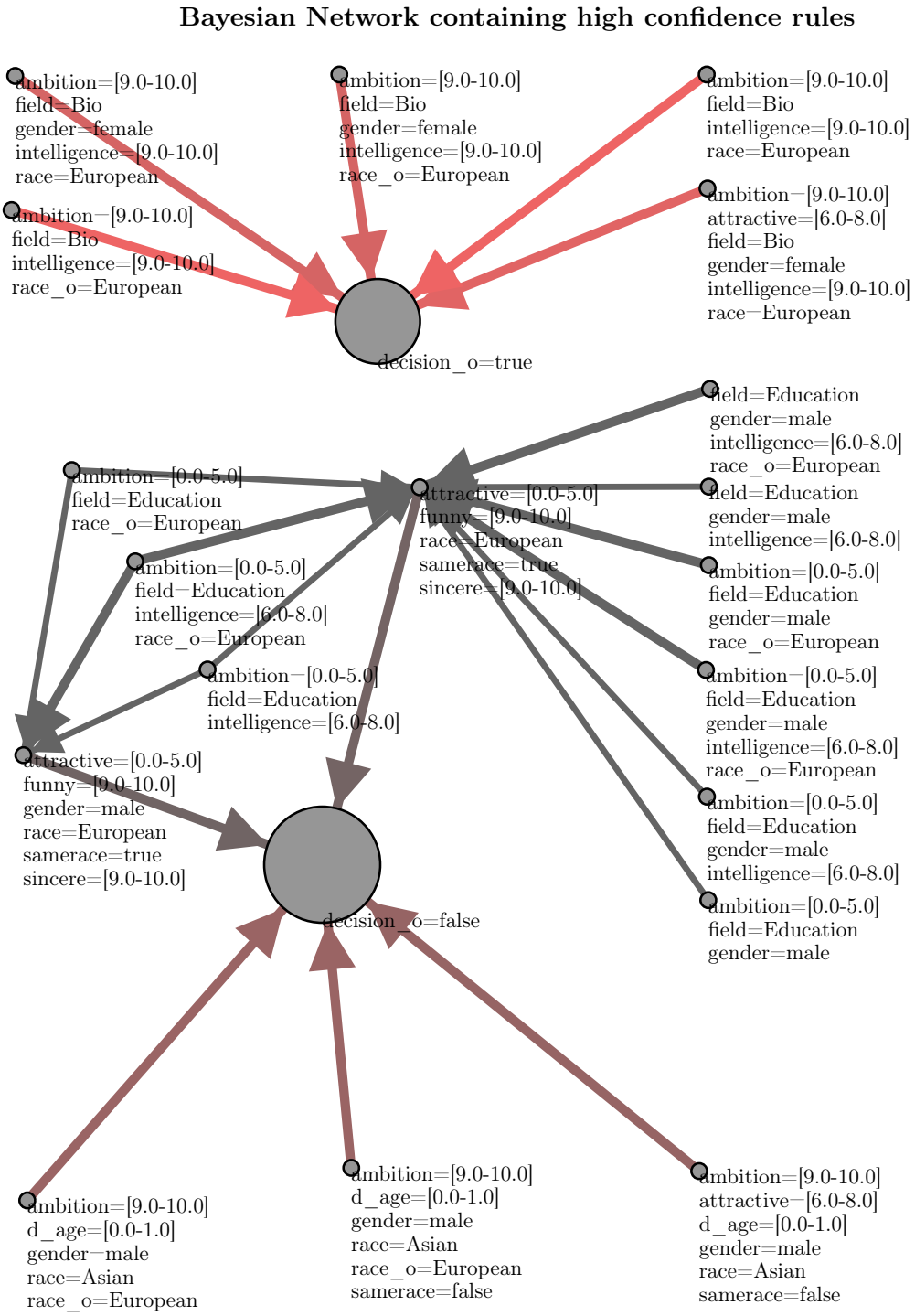


Figure 5.3: Bayesian network containing the most confident rules, mined from the speed dating dataset (discussed in Chapter 6.1.1). The red rules represent discriminating choices, for example the top-right rule, which represents the association between ambitious, intelligent, European people that study a biology related field and being liked on a speed date. In Chapter 7, we explain how discrimination is quantified.

Chapter 6

Experiments on Class Imbalance

In this section, we investigate the performance of classifiers with different settings regarding class imbalance. We discuss experiments with different SMOTE values, static/variable minimum support and different minimum support and confidence thresholds. We use the Speed Dating Experiment dataset [21] for our experiments, this dataset is suited because of its unbalanced class attribute (whether a person likes the other or not 1.3:1). The classifier that is constructed can be used to predict whether a speed date participant is likely to be a success or not, based on attributes such as interests, race, field of study, and more.

6.1 Setup

6.1.1 Dataset

We have chosen to do our experiments on this dataset because it exhibits both class imbalance and possible discrimination (discussed later). This dataset was collected from participants in experimental speed dating events from 2002-2004. During the events, the attendees would have a four minute "first date" with every other participant of the opposite sex. At the end of their four minutes, participants were asked if they would like to see their date again. They were also asked to rate their date on six attributes: attractiveness, sincerity, intelligence, fun, ambition, and shared Interests. No pre-selection was done. Within this setting we have chosen to build a classifier that will predict how likely it is to be liked by your partner, given some attributes. The attributes that we included are described in Table 6.1. The class-attribute for which the classifier is built, is *decision_ o* (decision of partner at night of event). Our dataset contains 8000 samples. In the experiments, 4000 samples are used for training the classifier, and 4000 samples are used for testing.

6.1.2 Discretization/Preprocessing

Our association rule mining approach requires discretization of numeric attributes for two reasons. First, due to the huge amount of attribute-value pairs that a numeric domain is made out of, there will be a really high number of frequent itemsets. Secondly, due to this sparsity of the itemsets, we only find a reasonable amount of itemsets when the minimum support is set very low, which will in turn result in small meaningless itemsets. We have chosen to discretize the numerical attributes by observing a histogram for each attribute, which allowed us to choose interesting discretization bins. Intuitively, for discretizing over 4 bins, we took the first quartile for the first bin, the second quartile for the second bin, etc. Figure 6.1 shows the bins that were chosen for *rate yourself* attributes. By applying this strategy to all attributes, the search space is significantly reduced, enabling efficient mining of meaningful itemsets.

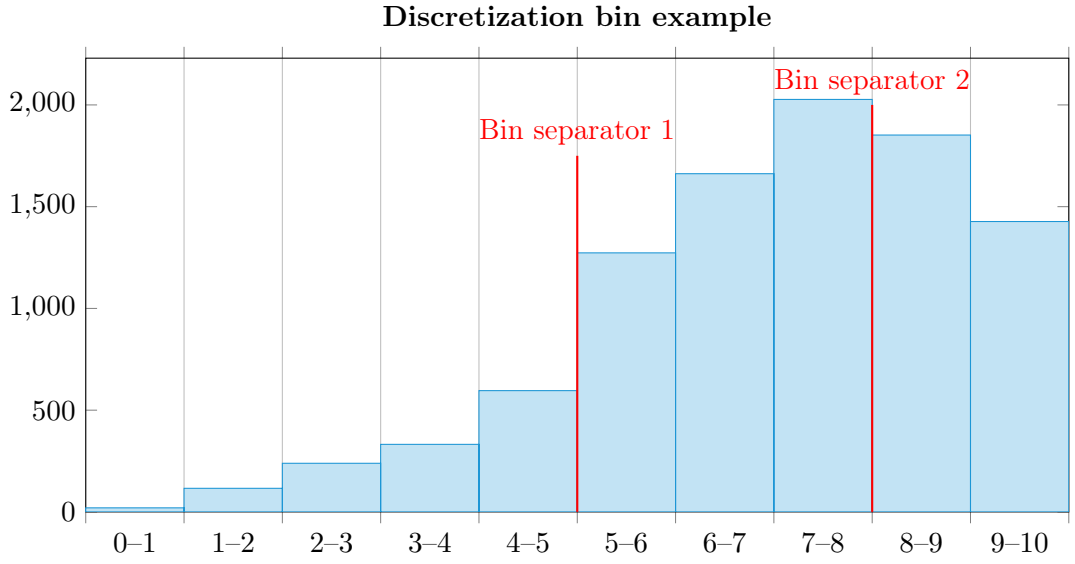


Figure 6.1: The distribution of the *rate yourself* attributes. We have chosen $bin_1 = [0 - 5]$, $bin_2 = [6 - 8]$ and $bin_3 = [9 - 10]$.

name	description	values
gender	Gender of self.	male/female
d_age	Difference in age.	[0-37]
race	Race of partner.	Black/African American / European/Caucasian-American / Latino/Hispanic American / Asian/Pacific Islander/Asian-American / Other
race_o	Race of self.	same as above
samerace	Whether the two persons have the same race or not.	0 = no, 1 = yes
field	Field of study.	Anthropology / Architecture / Art / Bio / Business / Chemistry / Climate / Communications / Earth / Education / Engineering / Finance / GSAS / Health / History / International / Journalism / Languages / Law / Marketing / Math / Media / Medical / Neuro / Other / Philosophy / Physics / Politics / Psychology / Religion / Social
	Rate yourself	
attractive	attractiveness	[1-10]
sincere	sincerity	[1-10]
intelligence	intelligence	[1-10]
funny	being funny	[1-10]
ambition	ambition	[1-10]
interests_correlate	Correlation between participant's and partner's ratings of interests.	[-1,1]
decision_o	Decision of partner at night of event.	true = like, false = not.

Table 6.1: Description of Speed Dating Experiment dataset attributes. decision_o is the imbalanced class that we want to predict.

6.2 Experiments

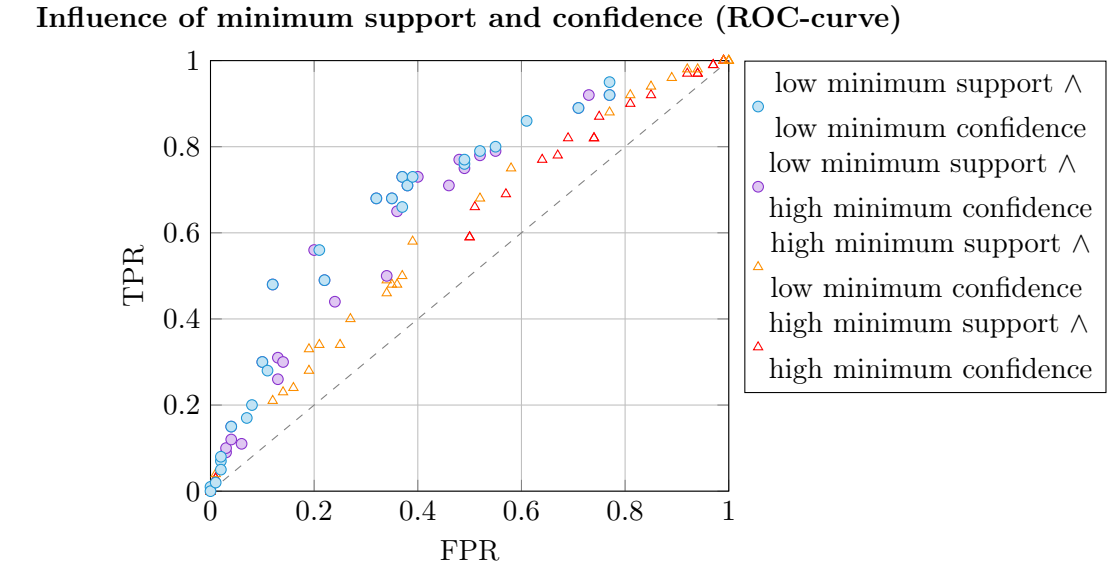
6.2.1 Experiment parameters

During the experiments, the performance of the classifiers is evaluated by adjusting a number of parameters. We show the total population of classifiers in each experiment, which consists of all possible permutations of the parameters below. We then inspect the contribution of each parameter by fixing certain settings, which can be observed more closely in a plot.

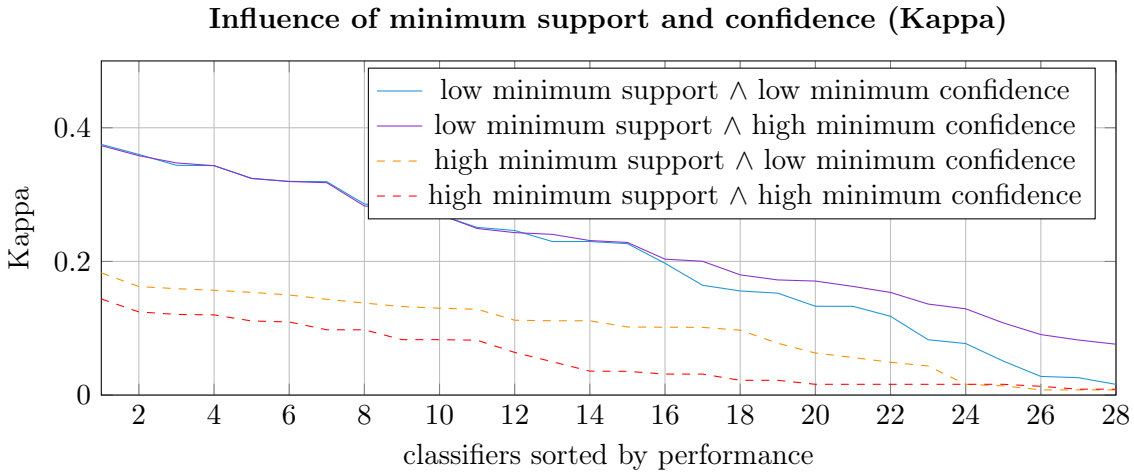
$$\begin{aligned}
\text{SMOTE} &= \{0, 0.3, 0.7, 0.85, 1.0, 1.5, 2.0\} \text{ (over-sampling)},\\
\text{weighted support} &= \{\text{true}, \text{false}\} \text{ (changing weight of minority/majority class)},\\
\text{minimum support} &= \{0.0025, 0.005, 0.0075, 0.01, 0.02, 0.03, 0.04\},\\
\text{minimum confidence} &= \{0.3, 0.4, 0.5, 0.6, 0.7\},\\
\text{sort classifier on lift} &= \{\text{true}, \text{false}\} \text{ (default CBA2 sort is: confidence, support)},\\
\text{total} &= 980 \text{ classifiers}.
\end{aligned}$$

6.2.2 Minimum support and confidence

The most fundamental parameters for classification using association rules are the minimum support and confidence thresholds. We investigate the effects of these parameters, by observing 4 subgroups of classifiers; 1) low minimum support, low minimum confidence, 2) low minimum support, high minimum confidence, 3) high minimum support low confidence, and finally, 4) high minimum support, high minimum confidence. These subgroups of classifiers are evaluated in Figure 6.2a using the ROC-plane and in Figure 6.2b using the Kappa statistic. We observe that classifiers with lower minimum support perform significantly better than classifiers with a higher support threshold. It is also interesting to observe that for a decreased minimum support, the minimum confidence does not impact performance any more. This is due to the CBA2 sorting precedence, which select the high confidence rules anyway (i.e. the low-confident rules are not used frequently). Hence we conclude that, if possible, a low support threshold is the best choice, if this is not possible (due to search space explosion), the confidence threshold should also be taken into account.



(a) Classifiers grouped by minimum support and confidence, evaluated in the ROC-plane.



(b) Classifiers grouped by minimum support and confidence, evaluated using the Kappa statistic.

Figure 6.2: In Figure (a), we observe all classifiers with minimum support$=\{0.0025, 0.0400\}$ and minimum confidence$=\{0.3, 0.7\}$. An immediate observation is that classifiers with a lower minimum support perform better. When this threshold is low enough, the minimum confidence can roughly be ignored, since it doesn't impact performance any more.

6.2.3 Weighted support

In this experiment we illustrate the effects of the weighted support measure (discussed in Subsection 4.3.2). The weighted support is used to create an equal amount of itemsets for both the minority and the majority class. This results in an equal amount of association rules

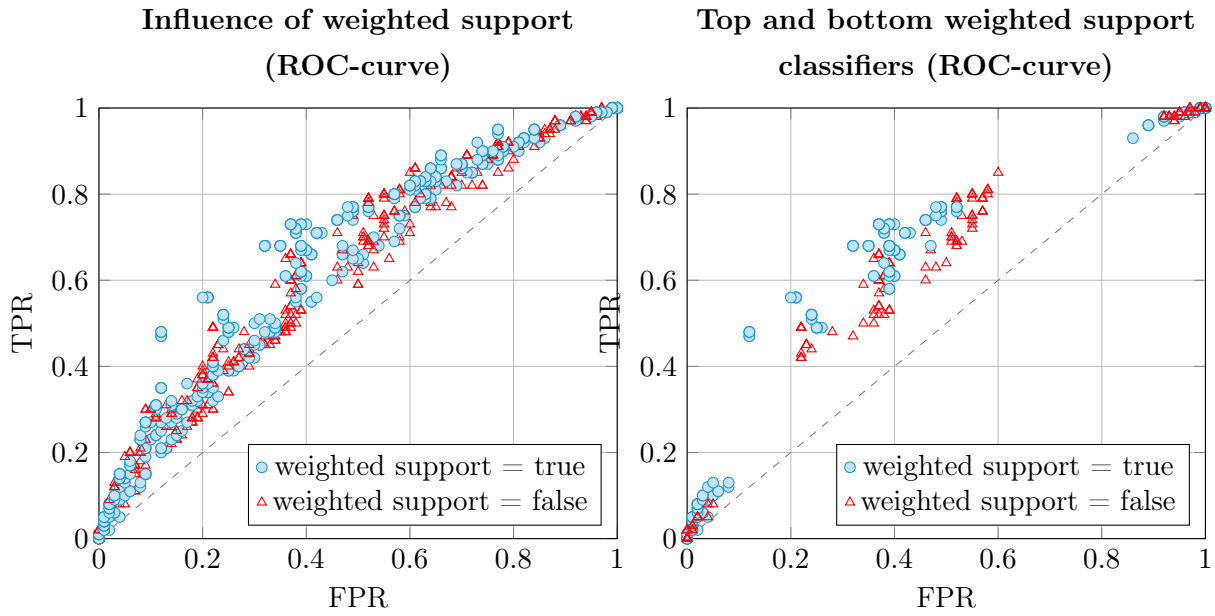
for both class-items, which should result in better classification. Figure 6.3a and 6.3b show the difference between the set of classifiers that make use of the weighted support (blue) vs. the set of classifiers that don't (red) evaluated in the ROC-plane. It is clear that the the weighted support classifiers perform consistently better. In Figure 6.3c the Kappa statistic is used to evaluate the two subsets of classifiers, again the weighted support classifiers score consistently better.

6.2.4 SMOTE

In this experiment we illustrate the effects of the SMOTE over-sampling method (discussed in Subsection 4.3.1). This oversampling technique is used to balance a dataset, i.e. create an equal amount of samples for the minority and majority class. Our dataset has a class imbalance of 1.3:1 samples for decision_o=false/true, which is not a lot, but we can still show that performance increases as the SMOTE-value is chosen correctly. At a certain point, when the amount of SMOTE becomes too high, performance decreases. This happens for two reasons: 1) we create too many synthetic samples, and the class becomes unbalanced again, and 2) the original characteristics will fade due to high number of synthetic samples. When no oversampling is done (i.e. SMOTE = 0.0), we get various results that are spread across the ROC-curve (not bad though). When we add the correct amount of oversampling (SMOTE = 0.3), we observe more samples towards the centre of the curve, which means a higher TPR and lower FPR. When that class is imbalanced again by oversampling too much, performance decreases. In the Kappa statistic plot, we also see this significant improvement for SMOTE = 0.3.

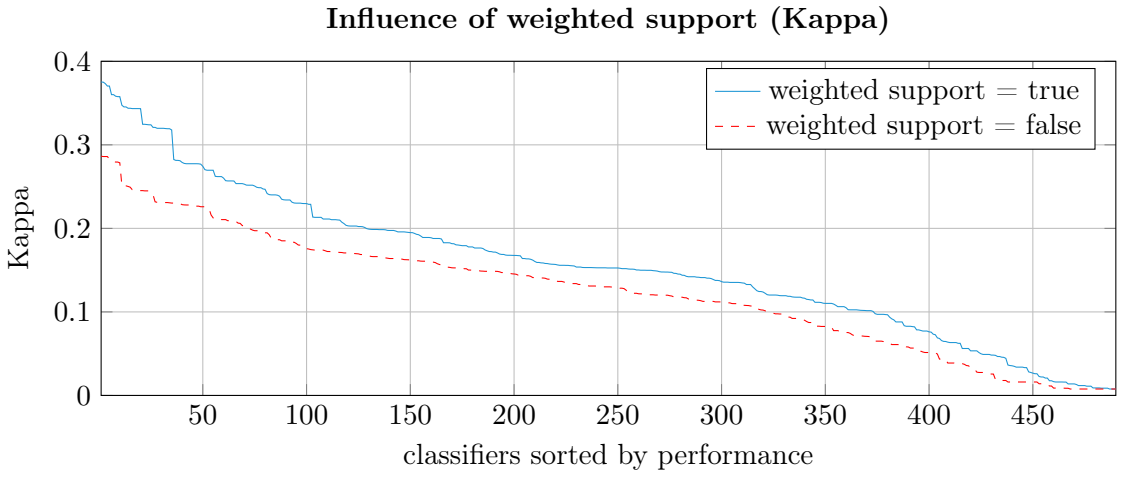
6.2.5 Weighted support vs. SMOTE

In this experiment we investigate which of the two methods is more effective in dealing with class imbalance (weighted support from CBA2, or SMOTE). Also we want to see how the two methods stack together. Figures 6.5a and 6.5b show 4 lines (evaluated using the ROC-distance metric and Kappa statistic respectively); 2 for weighted support, and 2 for a normal support both no SMOTE and optimal SMOTE. We observe that stacking both methods yields optimal performance, and that SMOTE contributes more to the good results than the weighted support (comparing the red and blue line with the orange line representing normal support and no SMOTE). This makes sense since the more SMOTE balances the class, the less work is left for the weighted support.



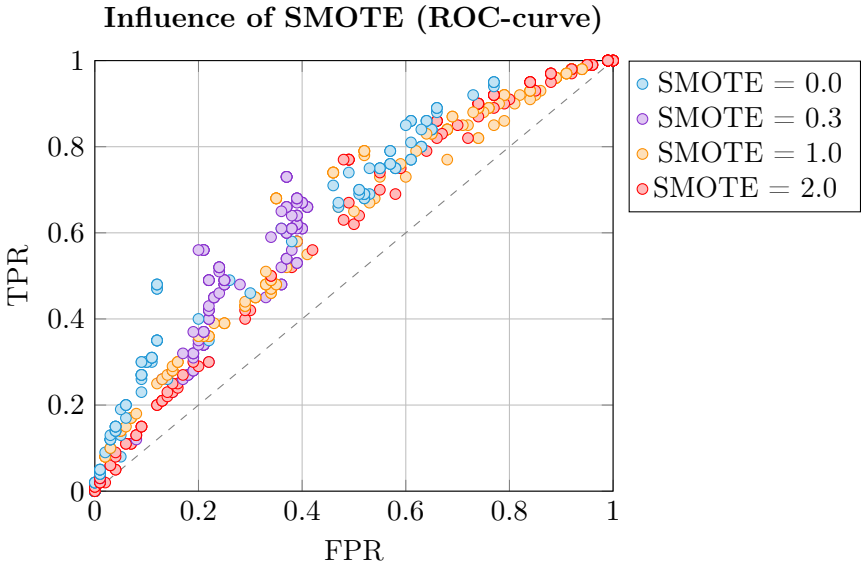
(a) Classifiers grouped by weighted lift = true/false, evaluated in the ROC-plane.

(b) Top and bottom 100 classifiers grouped by weighted lift = true/false, evaluated in the ROC-plane.

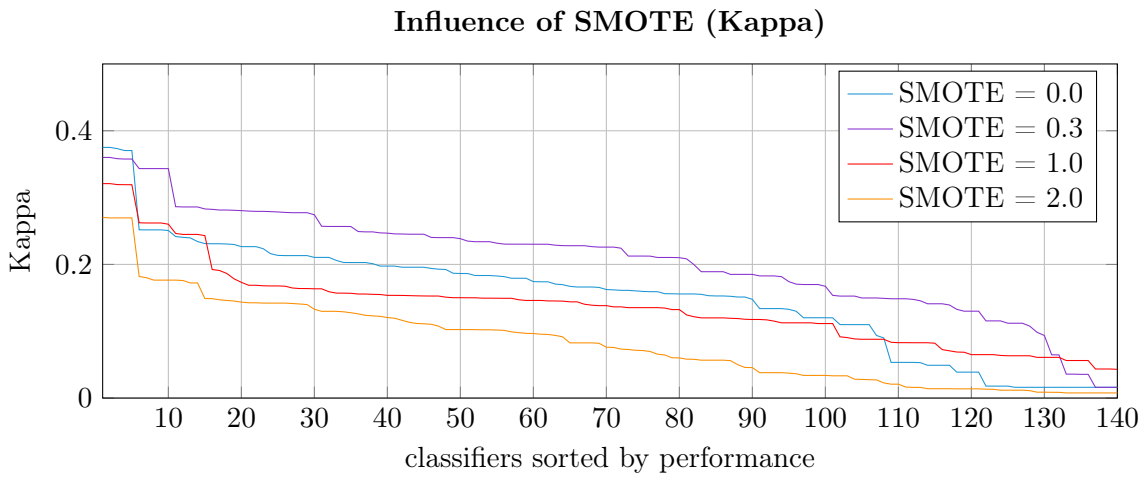


(c) Classifiers grouped by weighted lift = true/false. Evaluated using the Kappa statistic.

Figure 6.3: In Figure (a), we observe all classifiers (all settings), it is clear from this plot that various classifiers with a weighted support (based on the distribution of the class), can perform better. In Figure (b), the top and bottom 100 (lowest/highest ROC-distance, Definition 8) from the two different weighted support settings is shown. It is clear that as well in the best as the worst performers, the weighted support is a valuable addition. Figure (c) shows the same samples, but now tested with the Kappa statistic instead of the ROC-curve. It can be seen that according to the Kappa statistic also the classifiers that use a weighted support perform better. The performance improvement is not substantial, but it is consistent.

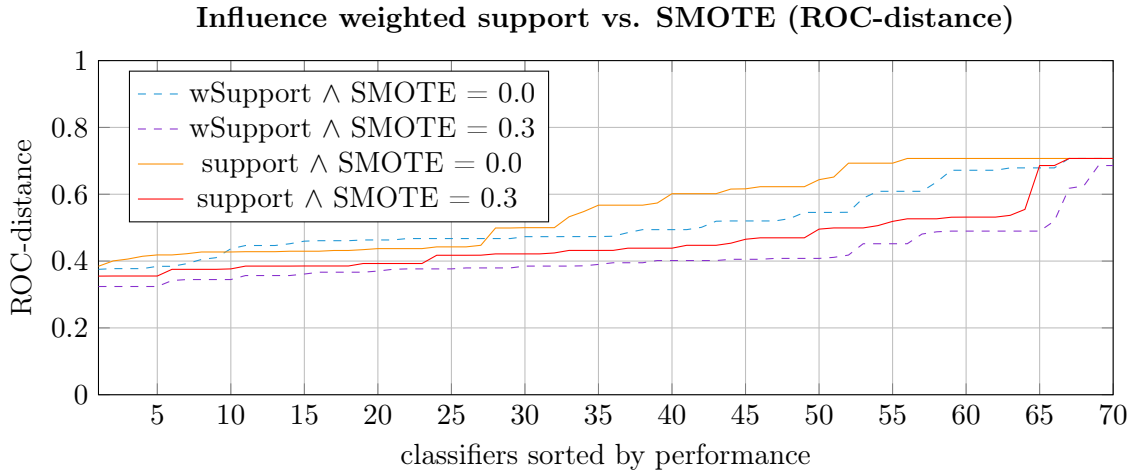


(a) Classifiers grouped by SMOTE, evaluated in the ROC-plane.

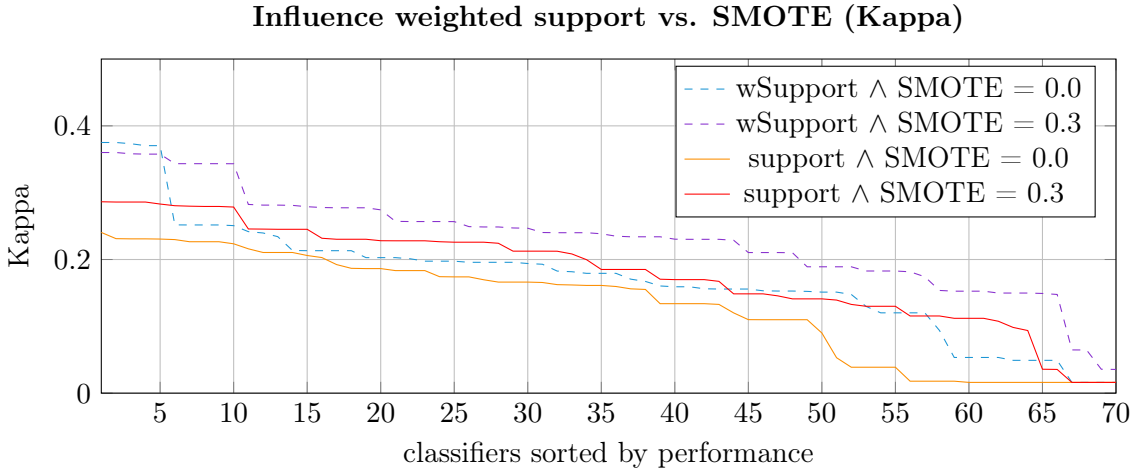


(b) Classifiers grouped by SMOTE, evaluated using the Kappa statistic.

Figure 6.4: In Figure (a), we observe all classifiers with SMOTE $= \{0.0, 0.3, 1.0, 2.0\}$. The optimal classifiers can be found at SMOTE $= 0.3$, the other classifiers perform worse, because of their class imbalance (either in favour or against the original minority class-item).



(a) Classifiers grouped by support = {weighted, normal} and SMOTE = {0.0, 0.3}, evaluated using the ROC-distance metric.



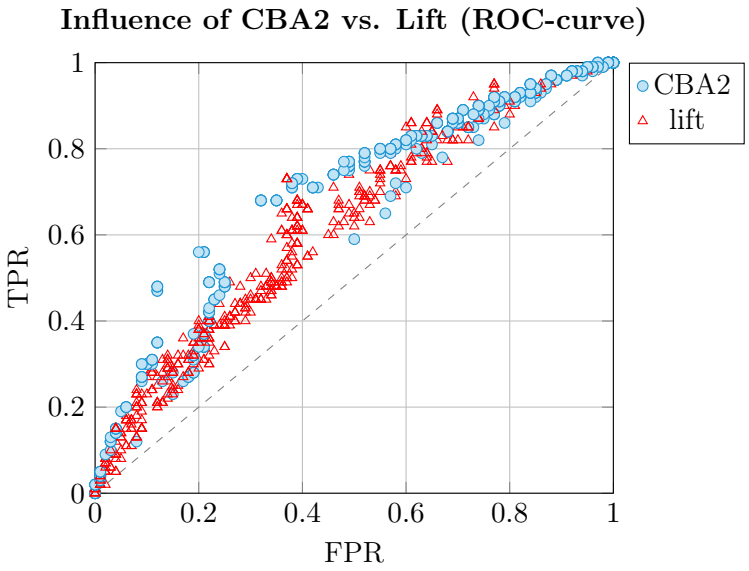
(b) Classifiers grouped by support = {weighted, normal} and SMOTE = {0.0, 0.3}, evaluated using the Kappa statistic.

Figure 6.5: We observe that the combination of SMOTE and weighted support yields the best results. SMOTE contributes more to this result than the weighted support.

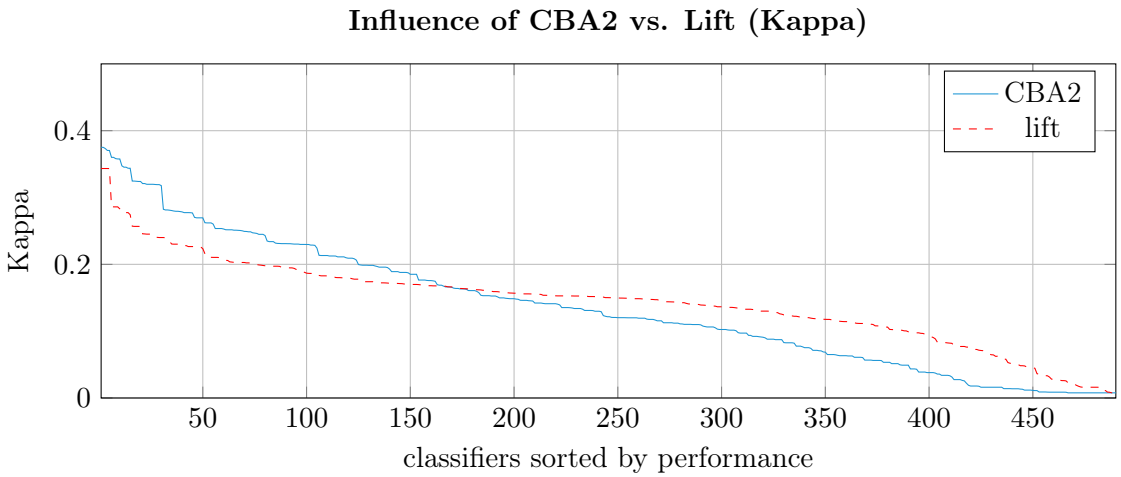
6.2.6 CBA2 vs. lift sorting

A rule based classifier consists of a set of rules. The question in the construction of such a classifier is "which rules should be used?". By sorting the rules on confidence (as CBA2 suggests), we make sure that the prediction we do encounters the most confident rules first, if a rule matches the sample to be predicted, it is used. We investigate if another sorting order could improve classification under certain settings. Figures 6.6a and 6.6b show two subgroups of classifiers; the classifiers sorted according to CBA2 (confidence) and the classifiers sorted

according to lift. We see that the difference in classification is not substantial, however, the CBA2 method performs better, but the lift method more consistent (which is visualized in the Kappa statistic plot).



(a) Classifiers grouped by classifier sorting according to CBA2 (confidence) or sorting on lift, evaluated in the ROC-plane.



(b) Classifiers grouped by classifier sorting according to CBA2 (confidence) or sorting on lift, evaluated using the Kappa statistic.

Figure 6.6: We observe that the CBA2 classifier (sorted on confidence) performs best, however, according to the Kappa plot, the lift method performs more consistent.

6.2.7 Best performing rules

In Figure 6.7 we show the top 10 rules (5 predicting true and 5 predicting false), that are used by the best classifier. The top 10 is defined by the rules that did most predictions correctly.

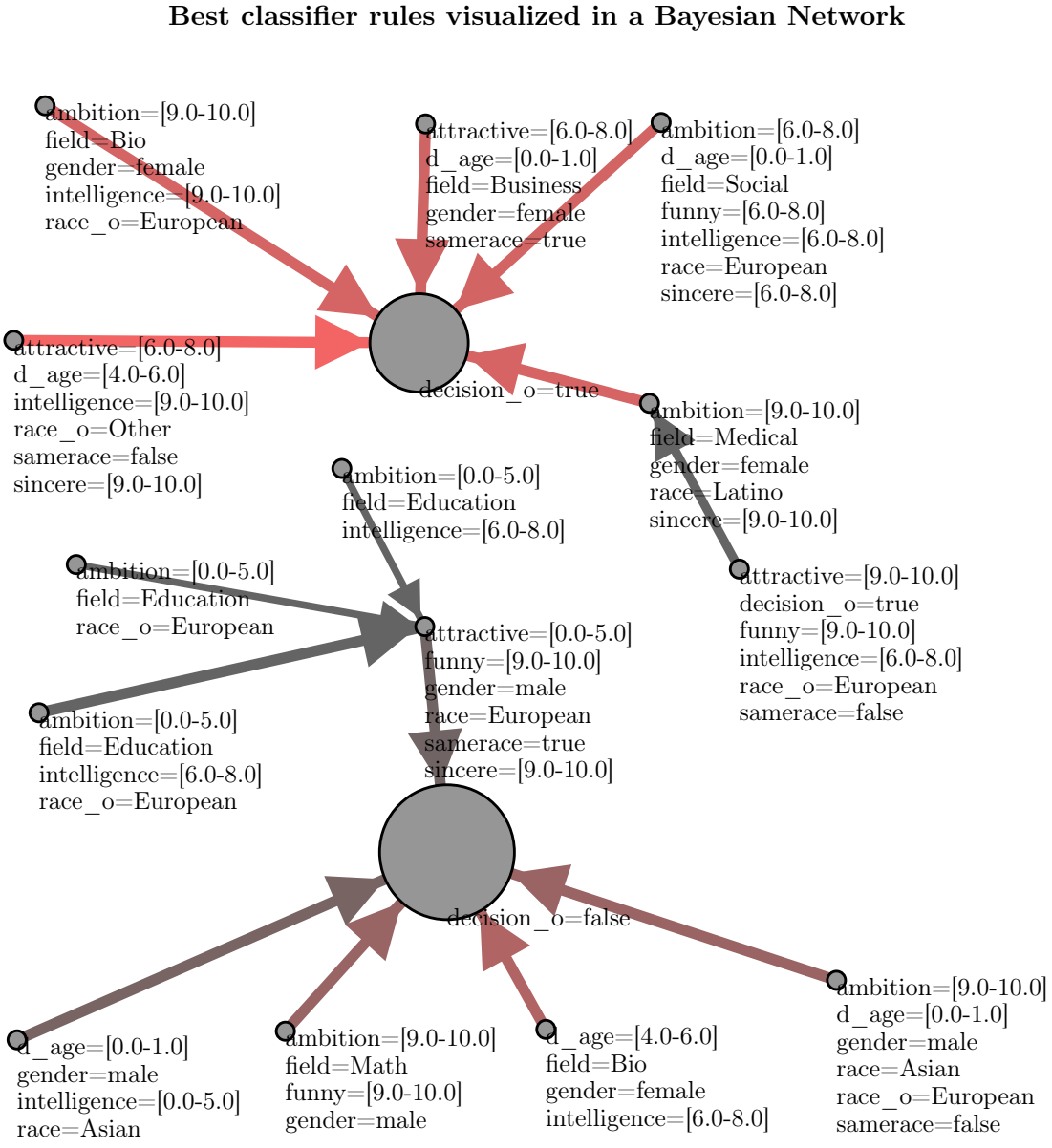


Figure 6.7: Bayesian network containing the best rules. Moreover, the rules that are used most frequently *and* yield the correct prediction. For example the top-left rule, which represent the association between ambitious, intelligent, European females that study a biology related field and being liked.

6.3 Conclusion on experiments

We have seen that the ROC-curve, ROC-distance and Kappa statistic are useful tools in the evaluation of classifiers for unbalanced classification. When constructing a classifier, one of the most influential parameters is the minimum support threshold; when set low enough minimum confidence can be neglected. Also we have shown that even though the class was not extremely unbalanced, methods such as SMOTE and the weighted support provide useful techniques in improving the classifier (SMOTE contributing the most). Finally, we proposed and tested another sorting precedence for the CBA2 classifier, which did not yield improvement, but neither performed much worse, it is however a more consistent method, so when various settings need to be changed for some reason, it may become the better choice (we see this when discrimination comes into play in the next chapter).

Chapter 7

Discrimination

Discrimination is giving a different (often less favourable) treatment to an individual, based on his or her participation in a group, class or category, instead of basing this treatment on individual properties. Since association rules are often mined on socially sensitive historical data (banking, insurance or market basket recommendation), these rules can capture choices that may be considered discriminative. When such rules are used to construct a classifier, certain groups will be systematically discriminated, which is of course highly undesirable. A fair rule is a rule that makes the same prediction for two individuals that have similar properties, but different sensitive attributes (e.g. a female with an income labelled moderate, should have the same chance of getting a loan at a bank, as a male with the same income). This claim is used to quantify discrimination of a rule in the next section. Several methods exist that remove discrimination from the dataset or the classifier [12].

The problem of discrimination is a complex problem, mainly due to the high number of dimensions of the data, which results in numerous contexts in which discrimination may happen. While a choice for a certain subgroup may not seem discriminatory, observing a subgroup of this subgroup may actually reveal discrimination. For example, when it seems that no women are discriminated at general, one may inspect the subset of elderly women to find that they are less likely to get a loan. In order to solve these problems, we need methods that can automatically check the entire classifier for discrimination. In this chapter we first introduce a formal, quantifiable definition of discrimination. We then show an approach that efficiently finds the rules that exceed a specified amount of discrimination. Finally, we show an approach that makes the classifier discrimination free, and show the top-5 discriminative rules that are found in the Speed Dating Experiment dataset [21].

7.1 Definition and quantifying discrimination

There is no standard definition of discrimination by legislation. We use the definition from [19]; the extended lift (elift, Definition 16). Which measures the confidence of a choice made for a discriminated group in contrast to the whole group (i.e. the same itemset without items that are defined to be potentially discriminative). Take for example credit approval, here it is desirable to know that women are treated the same as the rest of the population (men). Formulated in itemsets:

$$\begin{aligned}\text{confidence}([\text{gender=female}, \text{loan\_status}{=}x] \Rightarrow [\text{approve\_loan=false}]) \\ \approx \text{confidence}([\text{loan\_status}{=}x] \Rightarrow [\text{approve\_loan=false}]).\end{aligned}$$

It is important to realize that a computer cannot define which attributes are considered "discriminative"; a domain expert is always required to define these attributes, and the degree to which they are allowed to discriminate (i.e. base a decision on). Potentially discriminative itemsets (PD-itemsets) are defined as the attributes that are evaluated for discrimination. A rule is defined to be discriminating when its extended lift exceeds a domain expert specified threshold (e.g. the rule in the example above could have an elift of 1.5, meaning it is 1.5 times more likely that a women with loan_status=x would not be approved, compared to the entire population with loan_status=x). It is up to the domain expert to define whether an elift of 1.5 is discriminative or not.

Definition 16. ***elift*** - Let $A, B \rightarrow C$ be a classification rule with confidence$(B \rightarrow C) > 0$, where A is a PD-itemset, B is the context and C is the choice that is made. The extended lift of the rule is:

$$\text{elift}(A, B \rightarrow C) = \frac{\text{confidence}(A, B \rightarrow C)}{\text{confidence}(B \rightarrow C)}.$$

Intuitively this translates to: What is change that decision C is made for A, in context B, in contrast to the same decision for the entire population in context B.

For example, a rule $A, B \rightarrow C$, with A = [gender=female], B = [loan_status=x] and C = [approve_loan=false], with an extended lift of 3, means that being female increases 3 times the probability of not being approved, with respect to the average confidence of people with loan_status=x.

The elift measures the difference between a potentially discriminated group and the entire population. However, in some cases it is desired not to know the contrast between this group and everyone, but between this group and all *other* people. The selection lift (slift, Definition 17) quantifies this notion.

Definition 17. ***slift*** - Let $A, B \to C$ be a classification rule with confidence$(B \to C) > 0$, where A is a PD-itemset, B is the context and C is the choice that is made. The selection lift of the rule is:

$$\text{slift}(A, B \to C) = \frac{\text{confidence}(A, B \to C)}{\text{confidence}(\neg A, B \to C)}.$$

Finally the contrasted lift (clift, Definition 18) compares two specific subgroups.

Definition 18. Let $A, B \to C$ be a classification rule and v_1 and $v_2 \in dom(A)$, with confidence$(A = v_2, B \to C)$ minimal and > 0, where A, a PD-itemset, B the context and C the choice that is made. The contrasted lift of the rule is:

$$\text{clift}(A = v_1, B \to C) = \frac{\text{confidence}(A = v_1, B \to C)}{\text{confidence}(A = v_2, B \to C)}.$$

Note that either $elift \geq slift \geq clift$ or $elift \leq slift \leq clift$, i.e. each lift is a more/less specific variant of the other (depending on the consequent of the rule). For this reason, and the fact that clift requires more information (v_1 and v_2), we restrict ourselves to the notion of elift for automated discrimination discovery.

7.2 Finding discriminatory rules

In order to efficiently calculate the elift of each rule, the subsets of the itemset that were obtained by Apriori are used. By doing so, we only need to iterate over the rules once. This is illustrated in Algorithm 7.

Algorithm 7 Filter Discriminatory Association Rules

1: **function** FILTERELIFT(R, I_d, e) ▷ Set R consisting of all rules, set I_d consisting
2: ▷ of all PD-itemsets and elift threshold e.
3: **for** $r \in R$ **do** ▷ for each rule r in R
4: $a =$ largest subset of r.antecedent in D
5: $b = r$.antecedent $\setminus A$
6: $bc = b \cup r$.consequent
7: $\text{conf}_{abc} = \text{confidence}(r)$ ▷ Already known.
8: $\text{conf}_{bc} = \text{confidence}(bc)$ ▷ Found in $k-1$ itemsets.
9: $\text{elift} = \dfrac{\text{conf}_{abc}}{\text{conf}_{bc}}$
10: **if** elift $> e$ **then**
11: $R = R \setminus r$

The rules that exceed the maximum elift threshold are removed, resulting in a discrimination free set of rules that can be used to construct a classifier as explained in Section 5.5. In the next chapter we evaluate the notion of discrimination, and investigate which parameters can be used to compensate for this loss of rules. Also we display in a Bayesian Network, a subset of rules that are considered discriminative by this method.

Chapter 8

Experiments on Discrimination

In this section, we extend the previous experiments with the concept of discrimination. This is done by introducing a new parameter *maximum elift* (Definition 16, from Chapter 7); the degree to which a classifier is allowed to discriminate. When we lower this threshold, the classifier is not allowed to use all information (i.e. discriminative rules are removed), and is therefore expected to have a lower performance than the same classifier allowing for discrimination. In the following experiments, we show to what extend this threshold influences the classifier performance, and what parameters can be tuned to minimize this effect. The same dataset and attributes as in Chapter 6 are used.

8.1 Setup

We want to emphasize that the choice of what is discriminating should always be set by a domain expert; what is discriminating in one case, may not be discriminating in the other (e.g. discrimination in banking is totally different from discrimination in speed dating). In our experiment, it is important to keep in mind that even though some people might not want to date other people from certain minorities (which is there right), a classifier should not learn this (e.g. when African American people mainly have "successful speed-dates" with people of the same race, a classifier should not prevent this group of people from dating people of other races). In our experiments, we define the following attributes as discriminative (Table 8.1).

name	description
race	Race of partner.
race_o	Race of self.
samerace	Whether the two persons have the same race or not.
field	Field of study.

Table 8.1: Discriminating attributes selection for experiments.

8.2 Experiments

8.2.1 Experiment parameters

In the following experiments the degree of allowed discrimination is modified to investigate what the effect of throwing away this information (rules) has on classification. First, we investigate the effects of a decreasing elift threshold, in the experiments that follow, we show the relation/influence that parameters such as minimum support, minimum confidence and the sorting order (CBA2's confidence vs. lift) of the classifier have. The maximum elift parameter has the following settings.

$$\text{maximum elift} = \{1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5\}.$$

Which makes for 7 times the original amount of classifiers that we tested in Chapter 6 (now 6860 permutations). For most of the following experiments we choose to fix the other parameters to the optimal settings found before.

$$\begin{aligned} \text{SMOTE} &= 0.3, \\ \text{weighted support} &= \text{true} \\ \text{minimum support} &= 0.0025, \\ \text{minimum confidence} &= 0.3, \\ \text{sort classifier on lift} &= \text{false} \end{aligned}$$

8.2.2 Discriminative rule examples

The Bayesian network in Figure 8.1 shows the top 10 rules (5 predicting true and 5 predicting false), that are considered most discriminative.

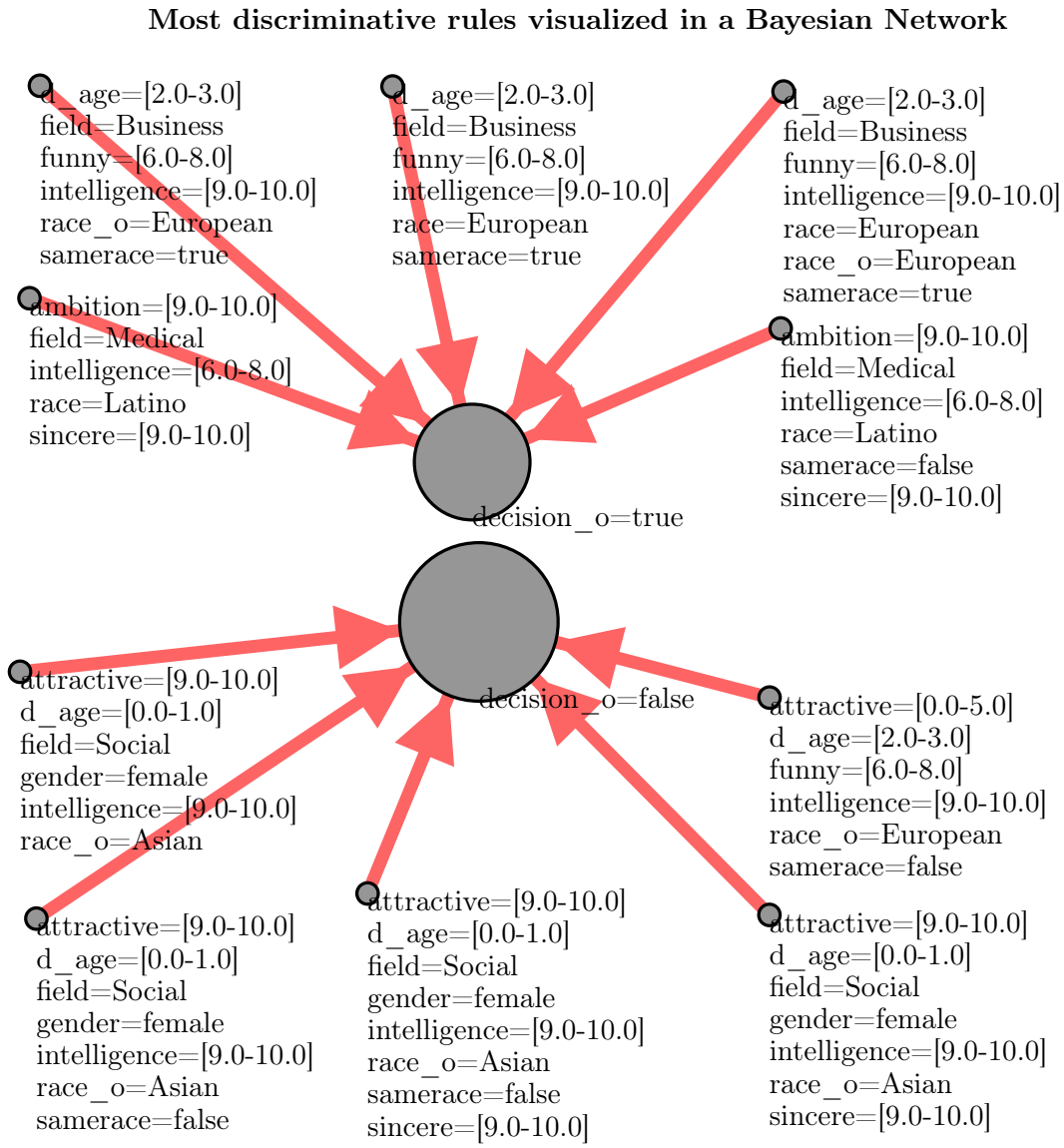
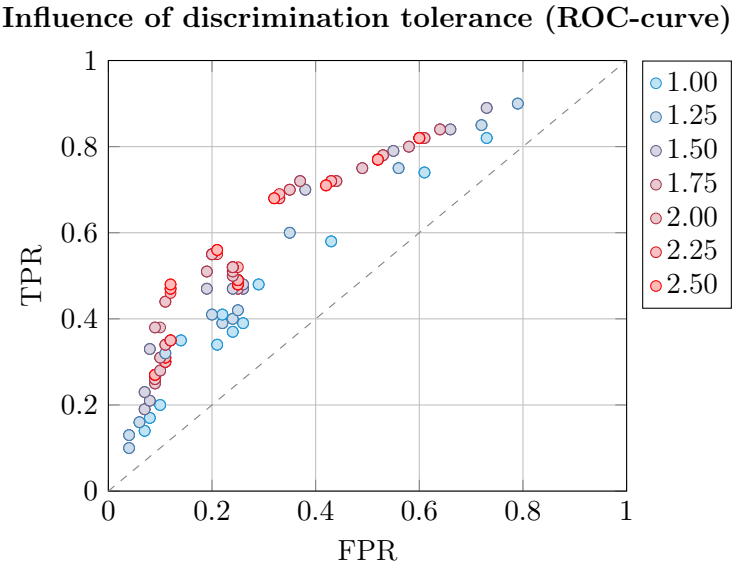


Figure 8.1: Bayesian network containing the most discriminative rules. The red edges represent the degree of discrimination. Take for example the bottom-right rule, which states that attractive, intelligent, sincere females with small age difference from their speed dating partner are generally not being liked by Asian males (race_**o**). Note that this rule is 2.5 times more likely for Asian people than for people from other races.

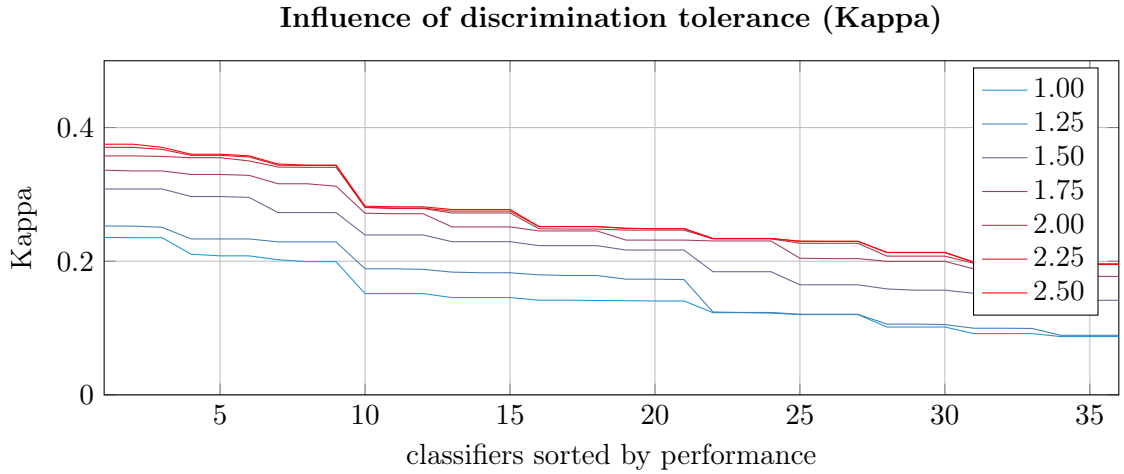
8.2.3 Maximum elift

In this experiment, we sketch the general effect that a decreasing tolerance for discrimination has on the classifier performance. Figure 8.2 shows this, where the maximum elift threshold is encoded by the colour (red for discrimination tolerance and blue for strict classification).

We see that after a certain maximum elift (2.0), performance converges. This is caused by the fact that there are no rules that discriminate more than elift = 2.5. The performance decrease for not allowing discriminating rules is significant. In the further experiments we investigate what parameters contribute to a good discrimination-free classifier.



(a) Classifiers colour coded by the maximum elift threshold (blue is low tolerance, red is high tolerance), evaluated in the ROC-plane.

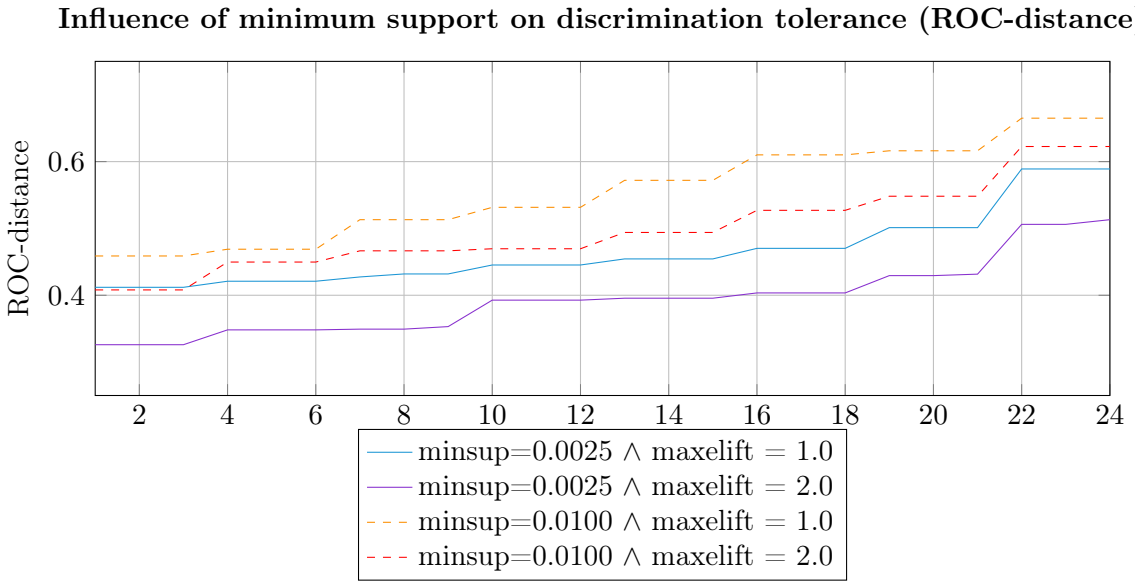


(b) Classifiers colour coded by the maximum elift threshold, evaluated using the Kappa statistic.

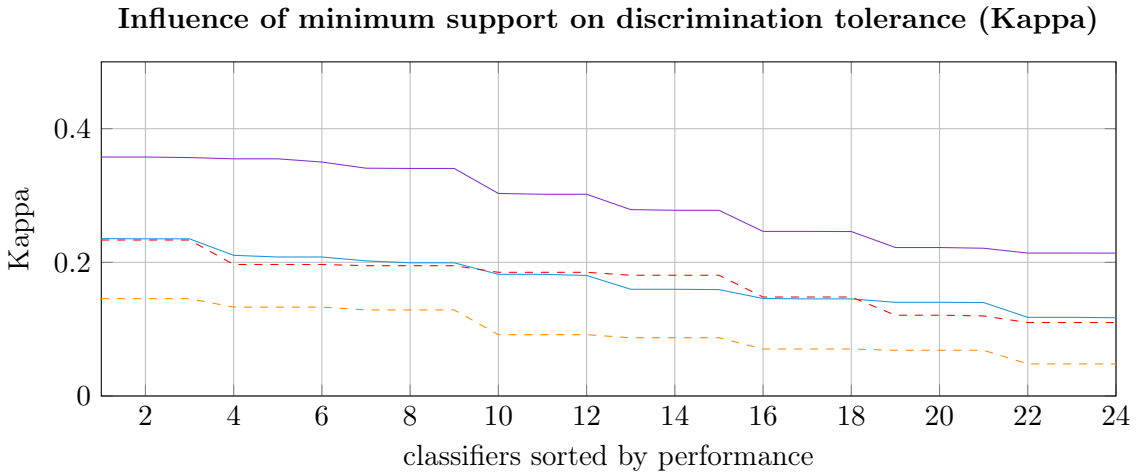
Figure 8.2: It is immediate to observe that disallowing discrimination has a significant impact on classifier performance.

8.2.4 Minimum support vs. maximum elift

In this experiment we investigate which classifiers are affected most by discrimination intolerance regarding minimum support. Figure 8.3 shows that both minimum support categories (high and low) suffer the same amount from decreasing discrimination tolerance. Thus, it is not possible to compensate for the discrimination threshold using the minimum support.



(a) Classifiers grouped in 4 categories regarding minimum support and maximum elift are shown, evaluated using the ROC-distance metric.

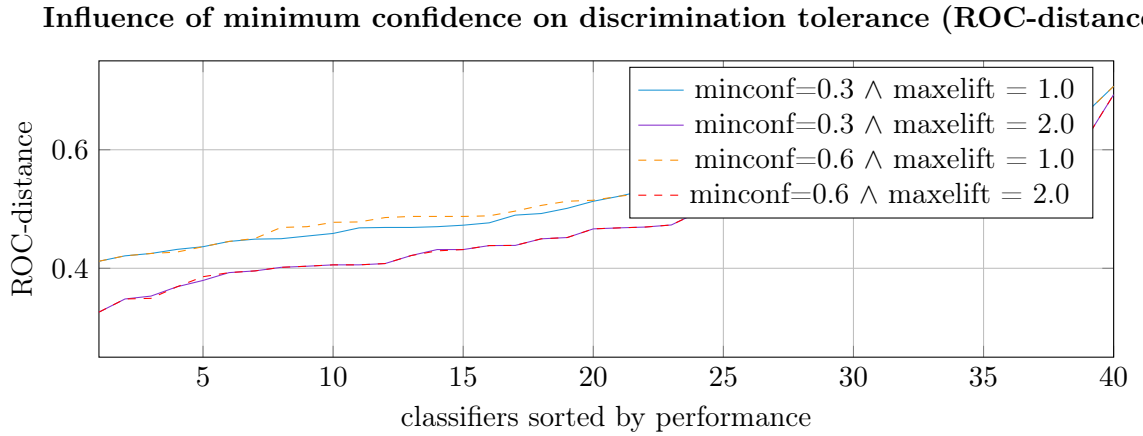


(b) Classifiers grouped in 4 categories regarding minimum support and maximum elift are shown, evaluated using the Kappa statistic.

Figure 8.3: We observe that both minimum support categories suffer just as much from the discrimination intolerance.

8.2.5 Minimum confidence vs. maximum elift

In this experiment we investigate which classifiers are affected most by discrimination intolerance regarding minimum confidence. We saw before (in Chapter 6) that confidence had no influence on the better 50% of classifiers, due to the confidence sorting order of the classifier. However, it is likely that by increasing the discrimination intolerance, confident rules are removed, so this is worth investigating. Figure 8.4 shows that both minimum confidence categories (high and low) suffer the same amount from decreasing discrimination tolerance. Thus, the confidence threshold still does not influence the classifier performance significantly.



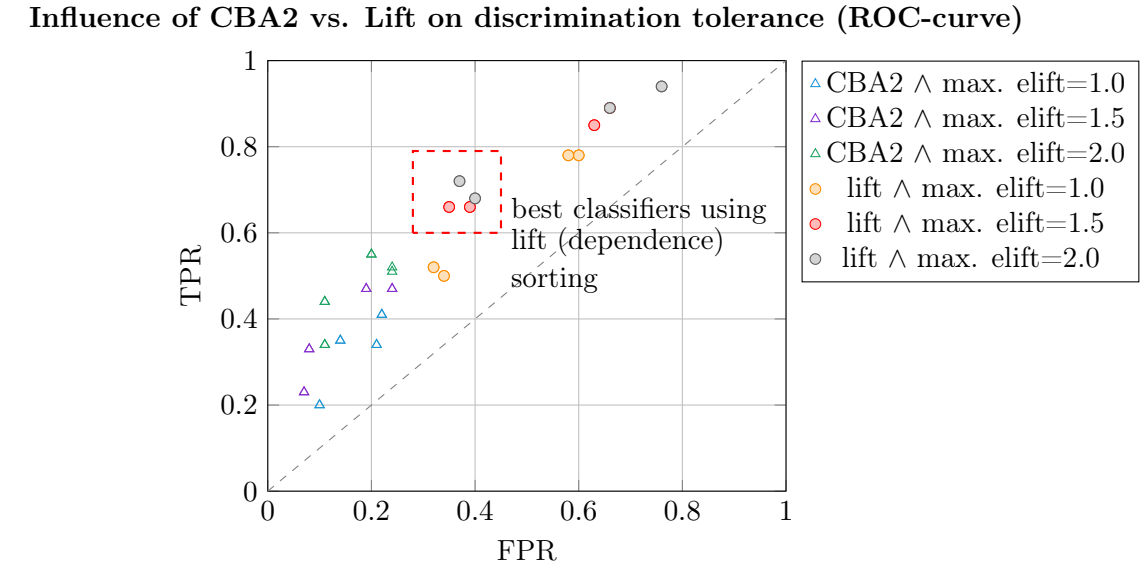
(a) Classifiers grouped in 4 categories regarding minimum confidence and maximum elift are shown, evaluated using the Kappa statistic.

(b) Classifiers grouped in 4 categories regarding minimum confidence and maximum elift are shown, evaluated using the Kappa statistic.

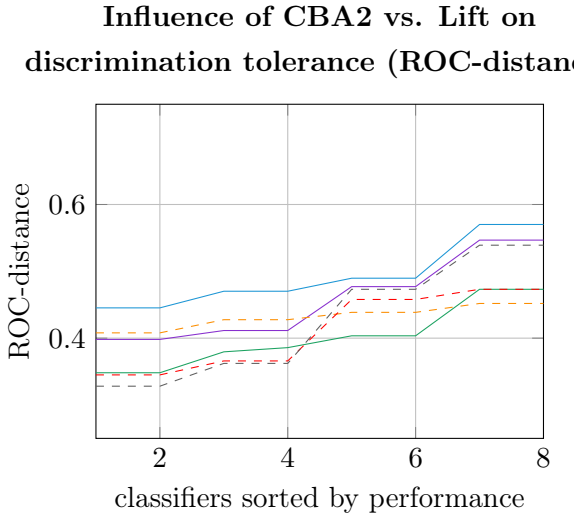
Figure 8.4: We observe that minimum confidence still doesn't affect classifier performance, also not when discrimination intolerance is increased.

8.2.6 CBA2/lift vs. maximum elift

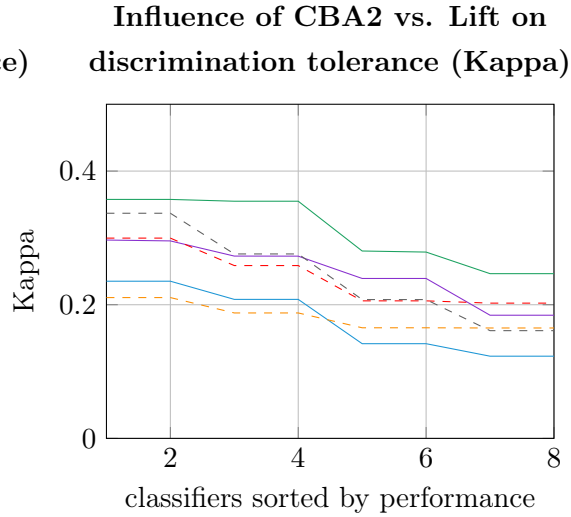
In the last experiment from Chapter 6, we show that when discrimination is not at stake, the default CBA2 sorting order performs best, however, in Figure 8.5 we observe that when discrimination comes into play, choosing the lift sorting order (antecedent/consequent dependence) for a classifier can perform significantly better according to the ROC-metric. The main reason for this is that discriminating rules are often rules with higher confidence, thus when sorting on confidence, we loose a significant portion of rules that are frequently used in classification. However, the ROC-metric and Kappa statistic disagree on this somewhat, because Kappa is more interested minimizing FPR while staying away from the "random choice" area (observed accuracy = expected accuracy), and ROC treats TPR and FPR more equally. What we observe in the Kappa plot (Figure 8.5c) is that classifiers using the lift sorting, perform more consistently (the higher the discrimination threshold, the flatter the line becomes). Comparing the blue and yellow classifiers, shows that the top 4 classifiers using CBA2 perform better, but the bottom 4 using lift perform better. When observing the ROC-plane, it is important to note that all classifiers using the lift sorting precedence are located towards the north-east, while the classifiers using the confidence sorting precedence are located towards the west (which is favoured by the Kappa statistic). Also note that when these samples are observed in the ROC-distance plot, from comparing the blue (no discrimination tolerance CBA2) and orange (no discrimination tolerance lift) lines clearly lie further from the optimal point $(0, 1)$. For the green (high tolerance CBA2) and grey (high tolerance lift) line, this difference already starts to get smaller. When we would increase the tolerance even more, we would see that CBA2 performs better again. We conclude that when less discrimination is tolerated, the lift sorting precedence yields better and more consistent classifiers, having higher true positive rate, but also a higher false positive rate. This is especially useful when a high TPR is favoured over a low FPR. Figure 8.6 shows a Bayesian network, containing the top 10 best performing rules that are used by the lift classifier.



(a) Classifiers grouped in 6 categories regarding classifier sorting and maximum elift are shown, evaluated in the ROC-plane.



(b) Classifiers grouped in 6 categories regarding classifier sorting and maximum elift are shown, evaluated using the ROC-distance metric.



(c) Classifiers grouped in 6 categories regarding classifier sorting and maximum elift are shown, evaluated using the Kappa statistic.

Figure 8.5: In Figure (a), we see that classifiers using the lift sorting precedence (dependence of antecedent/consequent) perform better (higher TPR. but also higher FPR). In Figure (b), this is shown more clearly, where we compare the actual distances to the optimal $(0, 1)$ point in the ROC-plane. In Figure (c), we show that the Kappa metric is more in favour of low FPR, thus, favouring the CBA2 classifiers. When however comparing the two strict (non-discriminative) classifiers (blue and orange), we see that only the top 50% performs better using CBA2.

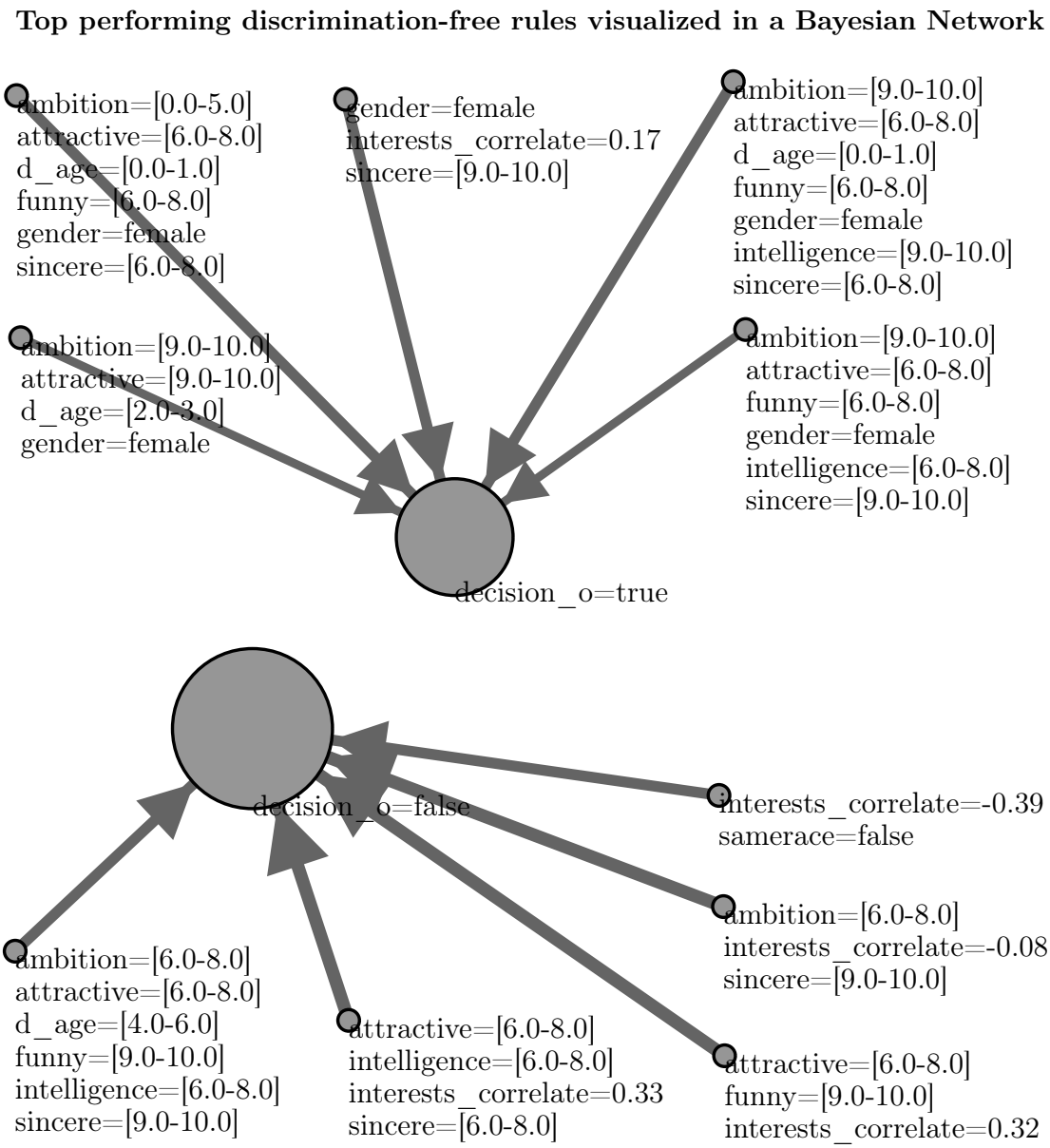


Figure 8.6: Bayesian network containing the top 10 discrimination free rules (5 predicting true and 5 predicting false). These rules are obtained using the lift sorting precedence. The top 10 is defined by the rules that do most predictions correctly.

8.3 Conclusion on experiments

We have seen that a decreasing tolerance for discrimination significantly affects classifier performance (negatively). The minimum support and minimum confidence thresholds do not have influence on this behaviour. This can be explained by the fact that discriminative rules generally have a higher confidence and do not depend on support, so by increasing or decreasing the minimum support, a new portion of discriminative rules is removed or added respectively. The minimum confidence of the classifier does not improve performance for the same reason as mentioned in Chapter 6, namely, the low confident rules that are added, are not used because of the sorting order. One remarkable finding is that changing the sorting precedence of the classifier from confidence (CBA2) to lift (our proposal) results in significantly better classifiers when the discrimination intolerance is increased, compared to the original sorting order. Moreover, classifiers using lift score better on the ROC-plane, tending to move more to the centre/north-east part (high TPR, slightly higher FPR) of the plane, while CBA2 classifiers stay in the south-west part (low TPR, low FPR) of the plane. The Kappa statistic however still favours the CBA2 method as it is more interested in minimizing FPR. We conclude that in the construction of a low-discrimination-tolerance-classifier, our lift proposal is a good solution. This parameter should still be accompanied by appropriate over-sampling and a sufficiently low enough minimum support threshold.

Chapter 9

Discussion

In this thesis the following main research questions are addressed:

1. How does class imbalance influence classifier performance, and how can this influence be minimized?
2. How does discrimination removal influence rule based classification performance, and how can this influence be minimized?
3. What is a good way to visualize the decision procedure of a classifier?

These questions are addressed by investigating the problems of class imbalance and discrimination and proposing techniques that minimize their effect. A classification system is developed that is used to evaluate these techniques; first the problem of class imbalance, and second the problem of discrimination in combination with class imbalance. We show that in both problem areas, progress is made on improving or combining the investigated methods. During evaluation of our methods, Bayesian Networks are efficiently used to visualize the decision logic of a classifier. The experiments are conducted using the speed dating dataset explained in Section 6.1.1.

9.1 Classification and Class Imbalance

In Chapters 4 and 6, the problem of class imbalance is explained and evaluated respectively. Two classes of methods are explained that aim at amplifying the minority class.

First, we discuss the CBA2 method which uses a weighted support, and enables Apriori to mine an equal amount of itemsets for both classes (the majority class and the minority class). This method is evaluated and we show that this provides consistently better classifiers (Figure

6.3). Furthermore, CBA2 proposes a sorting technique based on rule confidence for selecting rules that are likely to yield correct predictions. We propose to sort the rules on their lift (dependence between the rule's antecedent and consequent). However, this does not yield better results (Figure 6.6), but it does result in a set of more consistent classifiers. This method is tested again when the notion of discrimination is introduced.

Second, SMOTE is used to over-sample the data to create balanced classes. This approach is originally designed for continuous domains, on which it efficiently preserves the characteristics of the class that is to be over-sampled. We propose an extension of this technique that enables artificial creation of categorical attributes as well. And show that this technique consistently improves classifier performance (Figure 6.4). Furthermore, we also show the combined effect of SMOTE and CBA2 (Figure 6.5), where it can be observed that stacking these methods yields better results, but most success is accounted for by SMOTE. This makes sense since SMOTE creates equal classes, which minimizes the effect of weighted support.

In the experiments, various other parameters are investigated as well. We show that minimum support has the greatest influence on classifier performance and that minimum confidence does not make a significant difference (Figure 6.2). If the minimum support is too high, rules are not found for each sample, and some need to be guessed or predicted with low confidence rules. Lowering the minimum confidence threshold generally does not improve the classifier, since we first use all high confidence rules, which are by definition also contained in classifiers with a higher minimum confidence threshold. These are expected observations and give no new insights.

Finally, we motivate that common classifier evaluation techniques (such as classifier accuracy and precision) are inappropriate for imbalanced classification problems. For this reason other measurements are consulted: the Kappa statistic and the ROC-curve, both taking into account the distribution of the class. The Kappa statistic summarizes its information into one single value, whereas the ROC-curve identifies a classifier with a coordinate in a plane. This has the advantage that specific characteristic of the classifier are visible (TPR and FPR), but sometimes makes it hard to see which classifier is closer to the optimum. We therefore propose the ROC-distance, which is used to summarize results on the ROC-plane. We compare the ROC-curve and Kappa statistic and show that using both, gives good, yet slightly different insights on the results. The Kappa statistic is more concerned by minimizing FPR, whereas the ROC-curve by a balanced result. This is clearly illustrated in Figure 8.5.

9.2 Association Rules

A classification system based on association rules is constructed. A fundamental step in

obtaining association rules is mining the itemsets of the dataset. Apriori is used to do this. Since we are only interested in itemsets containing the class-items (which is only a percent of all itemsets are mined), we propose an extra pruning step for the algorithm. By throwing away itemsets that will not evolve to itemsets containing the class-item in the future, we significantly limit the exponential growth of Apriori's candidate generation. In Figure 5.1, it is shown that the mining time now depends on the minimum support threshold, and the supports of the class-items. The lower the minimum support, the greater the performance improvement (due to the exponential nature of Apriori), and a decreased support of the class-items also decreases runtime of the algorithm. We also found that mining the itemsets for the class-items individually results in better performance than mining the itemsets for both class-items at once, which may seem counter intuitive, since some itemsets will be mined twice (namely the ones that are undecided to contain a class-item in the future).

Furthermore, we propose a visualization technique using Bayesian Networks for classifiers. And show that it can efficiently be used to visualize classifier discrimination, high confident rules, general rules and rules that are used most often in classification.

9.3 Discrimination

In Chapters 7 and 8, the problem of discrimination is explained and evaluated respectively. First, measures that quantify discrimination are shown, these measures are used to remove discrimination from the rule base of the classifier. We emphasize that discrimination in learning from socially sensitive data is a problem, and show that its removal decreases classifier performance (Figure 8.2) as suggested by previous literature.

Various parameters of the classification system, such as minimum support, minimum confidence, and classifier sorting precedence are evaluated regarding discrimination. We show that the impact of minimum support and minimum confidence is negligible (Figures 8.3 and 8.4). This is explained by the fact that discriminative rules are generally higher confident rules, with all kind of supports (thus not related to support). By adding more low confident discrimination-free rules, since these added rules are simply not used, as an effect of the confidence sorting, and the classifier does not improve either. We however propose a sorting based on lift, which results in better classification when discriminative rules are removed. This is illustrated in Figure 8.5. We show that the performance improvement evaluated using the Kappa statistic is not significant, because the Kappa statistic values minimizing FPR more than maximizing TPR, but using the ROC metrics, we observe that a better classifier is indeed the result of our lift sorting proposal. Intuitively, this makes sense, since most of the discriminating rules have a high confidence (which is implied by the elift definition). When discrimination is removed, so are the high confident rules. The positive impact of our lift sorting

precedence increases as more of these discriminating, high confident rules are removed.

Finally, Bayesian Networks are used to illustrate several discriminating and non-discriminating rules in classifiers, in Figures 8.1 and 8.6 respectively. And are shown to be effective at visualizing certain properties of the classifier. Note however that a single Bayesian Network does not cover the entire decision space of the classifier and may give a biased view. It is therefore important that multiple aspects of the classifier are observed, using multiple Bayesian networks.

Chapter 10

Conclusion and Future Work

This thesis investigates several techniques to improve classification systems, that take into account class imbalance and discrimination, and proposes techniques that deal with these problems.

A survey on techniques that deal with problems in the area of class imbalance is given. First, it is shown that common measures to evaluate classifier performance, such as precision and classifier accuracy are not appropriate for measuring the performance of imbalanced classification. Techniques such as the Kappa statistic and the ROC-curve, which are more appropriate for this setting are discussed, and a new measurement, ROC-distance, is proposed, which summarizes information on the ROC-plane in a single value, and is easier for comparison. This distance is used to select an interesting subset, conforming a maximum distance (performance), and can then be evaluated for TPR and FPR. In future work, it may be interesting to assign weights to the axis used by the ROC-distance, in this way we can optimize for TPR/FPR specifically. Second, oversampling methods are investigated; SMOTE creates artificial samples and tries to preserve as much characteristics of the original dataset as possible, whereas CBA2 uses variable support that enables mining of an equal amount of itemsets for both classes. SMOTE originally only considers continuous domains, so a technique to extend this to categorical attributes is proposed, and successfully employed on our dataset containing categorical attributes. This technique may be interesting for further research as even better interpolation on categorical attributes could be done when certain heuristics about these attributes are collected. CBA2 originally constructs a classifier from association rules by sorting the rules on confidence, a sorting on lift (dependence between rule antecedent and consequent) is proposed, but does not seem to make an improvement in general classification (later we see that is does aid in discrimination-free classification). These techniques are evaluated, and it is shown that SMOTE and CBA2 both contribute to better classifier performance. When

combined, the performance increases even more, however SMOTE contributes most to this improvement.

A classification system based on association rules is built. A fundamental step in finding association rules is finding frequent itemsets, which Apriori is used for. An effective extra pruning step for the Apriori algorithm is proposed, which optimizes its performance when only the itemsets of the class-items are needed (just a percent of all itemsets). It is shown that performance relative to the algorithm without pruning increases as the minimum support threshold gets lower, as a natural effect of the exponential candidate generation used by Apriori. Also it is shown that using this extra step, the runtime of the Apriori algorithm depends on the support of the class-items (i.e. the lower the support, the lower the runtime, with a maximum of the original Apriori runtime). This method is worth further investigation, where a first step would be to see if intelligently combining results mined from all items individually yields better results than the original Apriori algorithm. Furthermore, a visualization technique that makes use of Bayesian Networks, to illustrate the characteristics of a classifier is proposed. This technique is used throughout the research and provides useful insights regarding the choices a classifier makes.

Finally, the problem of discrimination is shown and the extended lift measure is used to quantify discrimination and create certain degrees of discrimination-free classifiers. We acknowledge that discrimination removal (thus removal of useful information), decreases classifier performance, and show that our proposal of sorting the classifier on lift (dependence of the antecedent and the consequent of an association rule) yields better results than the original sorting order proposed by CBA2 (which sorts on confidence). Our improvement in contrast to the original CBA2 classification method yields better results as more discrimination is removed. Future work could be done to see if this method performs better for all discrimination removal techniques. Relating to the class imbalance techniques, SMOTE in combination with the elift (discrimination) measure could make for interesting future work. By combining these two techniques, an over-sampling technique for minimizing discrimination in the dataset could be realized (i.e. creating more "discriminating" samples for the non-discriminated portion of the dataset).

Bibliography

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Acm sigmod record*, volume 22, pages 207–216. ACM, 1993.

[2] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[3] Bart Custers. *Discrimination and privacy in the information society.* PhD thesis, Springer, 1866.

[4] Charles Elkan. Nearest neighbor classification. *elkan@cs.ucsd.edu, January*, 11:3, 2011.

[5] Benjamin Fish, Jeremy Kun, and Ádám D Lelkes. A confidence-based approach for balancing fairness and accuracy. *arXiv preprint arXiv:1601.05764*, 2016.

[6] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2-3):131–163, 1997.

[7] Gösta Grahne and Jianfei Zhu. Fast algorithms for frequent itemset mining using fp-trees. *IEEE transactions on knowledge and data engineering*, 17(10):1347–1362, 2005.

[8] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM Sigmod Record*, volume 29, pages 1–12. ACM, 2000.

[9] Marti A. Hearst, Susan T Dumais, Edgar Osman, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998.

[10] David W Hosmer and Stanley Lemeshow. Introduction to the logistic regression model. *Applied Logistic Regression, Second Edition*, pages 1–30, 2000.

[11] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

[12] Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1):1–33, 2012.

[13] Faisal Kamiran, Asim Karim, Sicco Verwer, and Heike Goudriaan. Classifying socially sensitive data without discrimination: an analysis of a crime suspect dataset. In *2012 IEEE 12th International Conference on Data Mining Workshops*, pages 370–377. IEEE, 2012.

[14] Faisal Kamiran, Indrė Žliobaitė, and Toon Calders. Quantifying explainable discrimination and removing illegal discrimination in automated decision making. *Knowledge and information systems*, 35(3):613–644, 2013.

[15] Bing Liu, Yiming Ma, and Ching-Kian Wong. Classification using association rules: weaknesses and enhancements. In *Data mining for scientific and engineering applications*, pages 591–605. Springer, 2001.

[16] Kevin P Murphy. Naive bayes classifiers. *University of British Columbia*, 2006.

[17] Loan TT Nguyen, Bay Vo, Tzung-Pei Hong, and Hoang Chi Thanh. Car-miner: An efficient algorithm for mining class-association rules. *Expert Systems with Applications*, 40(6):2305–2311, 2013.

[18] Dino Pedreschi, Salvatore Ruggieri, and Franco Turini. Measuring discrimination in socially-sensitive decision records. In *SDM*, pages 581–592. SIAM, 2009.

[19] Dino Pedreshi, Salvatore Ruggieri, and Franco Turini. Discrimination-aware data mining. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 560–568. ACM, 2008.

[20] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[21] Fisman R. and Iyengar S. Speed dating experiment, 2004. Kaggle, `https://www.kaggle.com/annavictoria/speed-dating-experiment`.

[22] Akash Rajak and Mahendra Kumar Gupta. Association rule mining-applications in various areas. In *Proceedings of International Conference on Data Management, Ghaziabad, India*, pages 3–7, 2008.

[23] Salvatore Ruggieri, Dino Pedreschi, and Franco Turini. Data mining for discrimination discovery. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(2):9, 2010.

[24] Marco Vannucci and Valentina Colla. Meaningful discretization of continuous features for association rules mining by means of a som. In *ESANN*, pages 489–494. Citeseer, 2004.

[25] Mohammed Javeed Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000.

[26] Guoqiang Peter Zhang. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462, 2000.

[27] Indre Zliobaite. On the relation between accuracy and fairness in binary classification. *arXiv preprint arXiv:1505.05723*, 2015.

[28] Indre Žliobaite, Faisal Kamiran, and Toon Calders. Handling conditional discrimination. In *2011 IEEE 11th International Conference on Data Mining*, pages 992–1001. IEEE, 2011.