MASTER IN BUSINESS ENGINEERING PROFESSIONAL FOCUS IN DATA SCIENCE

Benchmarking data augmentation techniques for credit scoring data

VAN HERREWEGHE, Lotte

Award date:
2022

Awarding institution:
University of Namur

Benchmarking data augmentation techniques for credit scoring data

Lotte Van Herreweghe

Directeur: Prof. I. Linden

Mémoire présenté
en vue de l'obtention du titre de
Master 120 en ingénieur de gestion, à finalité spécialisée
en data science

ANNEE ACADEMIQUE 2021-2022

Université de Namur, ASBL
Faculté des Sciences économiques, sociales et de gestion – Département des Sciences de gestion

Rempart de la Vierge 8, B-5000 Namur, Belgique, Tel. +32 [0]81 72 49 58/48 41

KU LEUVEN

FACULTY OF ECONOMICS AND BUSINESS

Benchmarking data augmentation techniques for credit scoring data

Master thesis

Lotte Van Herreweghe

Thesis submitted to obtain the degree

MASTER IN BUSINESS AND INFORMATION SYSTEMS ENGINEERING
Data Science

Promotor: Prof. Dr. Baesens Bart
Co-promotor: Prof. Dr. Linden Isabelle
Supervisor: Vanderschueren Toon

Academic year: 2020-2021

Contents

Leuven, 18/05/2022.

Abstract

Data augmentation has proved useful in training machine learning models for images or natural language processing. For tabular data, however, the existing data augmentation algorithms are much less numerous and well known. Nevertheless, most of the available data is tabular and existing DA techniques have already demonstrated that data augmentation can also improve the performance of classifiers here. Therefore, the purpose of this thesis is, on the one hand, to create a taxonomy of both mature and potential techniques. With the latter being techniques that are nowadays mainly used for other types of data, but have the potential to achieve good results on tabular data. On the other hand, the performance improvement offered by data augmentation is tested on credit scoring data. For this reason, 5 mature techniques are selected (insertion of noise, data swapping, SMOTE, CTGAN and PrivBayes), one from each category of the established taxonomy. Empirical results show that no algorithm consistently scores best. The classifier with which the DA technique is combined also has a major impact on performance. Moreover, a large variety of algorithms in terms of complexity is found. The most complex algorithms turn out to require a lot of time, processing power and understanding of the algorithm. Depending on the purpose for which DA is used, it may be permissible to use the extra time and computing power. But if, in the future, a company were to use DA by default on its data, for example, there are other alternatives that require fewer resources. Although no clear winning strategy is found, this thesis provides gainful insights in which techniques and models to combine when making predictions on credit scoring data. Furthermore, a clear taxonomy that can be consulted in need for an overview of existing DA techniques has been created and suggestions for research into other techniques have been done.

Acknowledgements

First of all, I would like to thank my promoter for giving me the opportunity to immerse myself in this subject. My supervisor, for the biweekly meetings in which I was always given good advice. My co-promoter, for the warm welcome in Namur. Fien, for being my Beta reader and putting the dots after the Dr. And last but not least, my family and friends for their continuous support.

Chapter 1

Literature study

1.1 What is data augmentation?

The term data augmentation originates with Tanner and Wong (1987). Nowadays, the definition is much broader and data augmentation refers to strategies for increasing the diversity of training examples without explicitly collecting new data (Feng et al., 2021). Generally speaking, there are two kinds of data augmentation. One way to inflate a dataset, is to take an instance from it, alter that instance slightly in one or multiple ways and add it to the original dataset. Modifying an existing instance can be label-preserving or not. The other option is to create synthetic samples by learning the distribution of the original dataset and then sampling from that distribution.
Furthermore, a distinction needs to be made between online and offline augmentation. Offline augmentation means the data are augmented and then stored to be loaded again when the model is trained. Online augmentation refers to augmenting the data as a new batch of original data is loaded to be used to train the model (Shorten, Khoshgoftaar & Furht, 2021). Both have advantages and disadvantages. Offline augmentation will likely be faster at training time, but online augmentation is typically more powerful.
Apart from increasing the performance of machine learning and deep learning models, data augmentation can also be used to tackle class imbalance, avoid overfitting or for privacy reasons.
While data augmentation is almost standard when deep learning algorithms are used, as is the case in a computer vision or natural language processing environment, data augmentation is not yet often applied when it comes to tabular data. However, the vast majority of available data is tabular and the possibilities are also promising there.

1.2 Why data augmentation?

In recent years, machine learning has seen enormous growth (Sestino & De Mauro, 2021), helped by the availability of much more computing power and data. However, not all classifiers are robust, resulting in a bad performance when they are deployed on real-life data that is just a little bit different than the training data. This problem can be solved

by providing the algorithm with more data (Carmon, Raghunathan, Schmidt, Liang & Duchi, 2022), but in some cases it is impossible or expensive to obtain additional data. That is where data augmentation becomes useful. Rebuffi et al. (2021) show that data augmentation improves the robustness of classifiers and thus improves performance. For this reason, data augmentation is often used for deep learning and machine learning models. The next paragraphs will go into further detail about the contexts in which data augmentation can be particularly useful.

1.2.1 Class imbalance and different costs

Class imbalance is the main reason why data augmentation is used on tabular data today. It is said that two classes are imbalanced, when one of them, the majority class, counts far more examples than the other, the minority class. There are numerous examples of class imbalance in real-life problems. The number of spam emails one receives is typically far lower than the number of wanted emails. Of all MRI scans that are taken, a lot of the patients will be healthy, while only a small number of patients is really ill. A dataset containing transactions will only contain a few fraudulent ones and an event log might only contain a few error events.

This causes problems because most data mining techniques do not handle this imbalance well. The reasons are multiple and have already been listed by Weiss (2004). Weiss indicates that not all evaluation metrics can handle class imbalance equally well, that the divide-and-conquer approach often used by data mining algorithms is also not robust against class imbalance and that the presence of noise can also be a disruptive factor. A brief summary will be given below, but for a more extensive overview listing examples and different kinds of solutions, we refer the reader to the original work.

Firstly, not all evaluation metrics tackle class imbalance very well. One of the most used metrics is classification accuracy for example. Classification accuracy measures how many examples were classified correctly. But this evaluation metric is biased towards the majority class (Hossin & Sulaiman, 2015). A classification accuracy of 50% means that 50% of the examples were classified correctly. This is the accuracy expected of a random classifier. Thus, when a classifier achieves a 99% accuracy, it seems to be doing very well. Only given class imbalance, it might be the case that the majority class covers 99% of the examples and the classifier is always predicting the majority class. The question is then: is this classifier really better than one with a lower accuracy that does predict the minority class sometimes?

It is also worth noting, that the minority class is often the one of interest and higher costs will be allocated to misclassifying a minority class example than a majority class example. This is the case for credit scoring data.

A second problem may be that the data mining algorithm employs a divide-and-conquer approach, meaning that the original problem is split up into smaller problems (Friedman, Kohavi & Yun, 1996). As a consequence, the few minority examples that are already present are split up among different partitions. This process leads to data fragmentation and this causes troubles since regularities can only be found when enough data are present.

The last reason that will be highlighted in this paper is the classifier's behaviour towards noise. Most classifiers are taught to generalize well to avoid fitting noisy data (Rivera, 2017). For this reason, small disjuncts are often eliminated because they are seen as insignificant. Given class imbalance, it is very hard to determine which small disjuncts are significant and which are not. As a consequence, both might be discarded, throwing away useful information in the process. Moreover, when the minority class counts few examples in the absolute sense and not only relative to the majority class, noise affects the ability of a classifier to learn the class concept of the minority class properly. Given that there are not many examples, the classifier will be much more likely to fit the noisy ones as well.

The next reason for data augmentation links seamlessly with this.

1.2.2 Avoid overfitting

The goal of training a classifier is to learn the pattern underlying the data (Ying, 2019). Noise can, however, obscure this pattern. The classifier will then learn the noise instead of the pattern. A typical symptom is that the classifier performs well on the training set, but not on the test set. This means it is overfitting. A good machine learning algorithm should be able to discern the useful data from the noisy data, but when the model is too complex or flexible it memorizes the noise instead of the underlying pattern. When choosing a certain model to fit the data, it is important to keep Occam's razor to always choose the simplest model in mind.

This does not mean that the simplest model should always be chosen, but that given the performance of the models, there is no point in using an overly complex model when a less complex model performs (almost) as well. This is part of the bias/variance tradeoff. On a spectrum with bias at one end (strong prior models or models with strong hypotheses) and variance at the other end (weak prior models or models that put forward few hypotheses), a point must be chosen between the two so that the model is neither underfitting (too strong bias) nor overfitting (too strong variance) (Briscoe & Feldman, 2010). The optimal point depends on the patterns to be learned.

Because model training is essentially a tuning of parameters, data augmentation is a possible solution to prevent overfitting (Ying, 2019; Zhong, Zheng, Kang, Li & Yang, 2020; Shorten & Koshgoftaar, 2019; Lee, Zaheer, Astrid & Lee, 2020). Well-tuned parameters implement a good balance between training accuracy and regularity which ensures overfitting and underfitting are avoided. The more examples a machine learning model has to learn from, the better its parameters will be tuned. But, getting more data is not always possible. That is where data augmentation comes in useful. Data augmentation renders it possible to enlarge a dataset without needing more data by changing existing examples or sampling from the learned distribution of the dataset.

There are other solutions for overfitting such as early stopping, pruning or regularisation. Readers can consult (Ying, 2019) for more information. However, data augmentation tackles the problem from the root: the training dataset. That is why in computer vision, where machine learning algorithms have to deal with various issues such as viewpoint, lighting, scale, background and more, data augmentation is often used

to handle overfitting (Shorten & Khoshgoftaar, 2019). In Natural Language Processing (NLP) high frequency numeric patterns in token embeddings or particular forms of language that do not generalize are often learned by the machine learning models (Shorten, Khoshgoftaar & Furht, 2021). Data augmentation can prohibit overfitting here as well.

1.2.3 Privacy reasons

While not the focus in this thesis, another reason to apply data augmentation could to protect the privacy of the subjects of the dataset. Medical datasets for example, can often not be shared because of privacy reasons. Data augmentation can then help to create an entirely new dataset based on the examples in the original one. Furthermore, data augmentation can protect the dataset from disclosure risk. Datasets are often de-identified using generalisation or randomisation. The generalised or randomised dataset is called the protected dataset. Disclosure risk means that an attacker would be able to use a protected dataset to derive information about instances in the original dataset (Domingo-Ferrer, 2009).

There are two main kinds of disclosure (Choi et al., 2018): attribute disclosure and presence disclosure. Attribute disclosure happens when an attacker can derive additional attributes about an instance, based on information they already have on that instance. Presence disclosure happens when an attacker is already in the possession of the original dataset and can derive whether instances from this original dataset are in the machine learning model's training set by observing it. This is also called a membership inference attack. However, data augmentation is only useful in this context when the mapping between the generated data and the original data is not one-to-one. Successful examples have been implemented already (Choi et al., 2018) and will be discussed in more detail later on in this paper.

For the reasons above, fully synthetic datasets are often created. This is not the focus of this thesis. Yet it is wise to include these applications: the techniques used to create these synthetic datasets can just as well be used to enrich datasets.

1.2.4 Next sections

The next section will outline the different techniques available for data augmentation. The rest of the literature review is divided into two parts. The first section contains techniques that sample new instances. A further subdivision was made into techniques that are already often used for tabular data and techniques that are not yet (often) used for tabular data, but are used for other kinds of data such as images or textual data. The first category includes Generative Adversarial Networks or GAN and Bayesian Networks. Variational autoencoders are discussed in the second category.

In the second main part of the literature study, the focus is on techniques that change existing instances. Here, too, the subdivision was made between techniques that are and are not often used with tabular data. In the first category, we find SMOTE and simple operations to slightly modify the data. In the second, adversarial attacks.

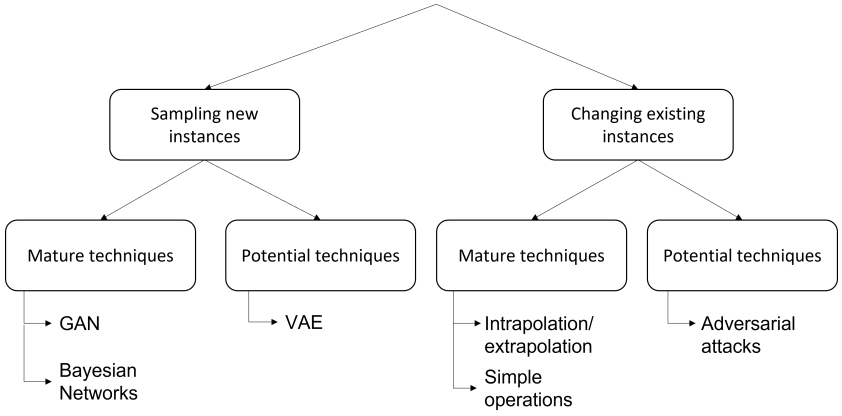

Figure 1.1: Taxonomy of the techniques that will be discussed in the next section.

1.3 Sampling new instances

1.3.1 Mature techniques

GAN

GAN stands for Generative Adversarial Networks. When working with GANs, two models are always trained: a generator and a discriminator. A generative model learns the distribution of the model and can tell how likely a given example is. It is used to generate new, realistic data instances. A discriminator tells how likely a label is given an example. (Dhariwal et al., 2020). In this context, the two have opposing objectives (Goodfellow et al., 2014). The generator wants to generate new instances that resemble the original ones as good as possible. These generated instances are then fed to the discriminator together with samples from the original dataset. The discriminator now has to try to distinguish fake and real instances (fake instances being the ones created by the generator). Thus, the generator wants to maximise the chance that the discriminator makes a mistake, while the discriminator itself wants to minimise this chance. This is actually a minimax two-player game. Both models will get better and better at their jobs as they receive feedback and there is only one unique and stable solution at the end: the generator discovers the data distribution and all instances are classified as having a 50% percent chance of being real or fake.

An analogy often made is that of a cop and counterfeiter. The counterfeiter will create false bills, which the cop will recognise. After learning why the cop recognises the bills, the counterfeiter creates better fake bills, until the cop can no longer distinguish if they are real or false.

GANs can be used to generate tabular data. DOPING, for instance, oversamples infrequent normal samples (rare normal events) to reduce false positive rates (Lim et al., 2018). The method is used in anomaly detection settings, where the ubiquity of false positives is one of the main challenges. DOPING is a form of unsupervised data

augmentation (UDA), a method of semi-supervised learning that reduces the need for labelled examples (Xie, Dai, Hovy, Luong & Le, 2020). The adversarial autoencoder (AAE) variant of GAN is used to transform the data distributions. The AAE is trained on the entire dataset without labels and makes use of general knowledge of the latent distribution to generate samples.

Another example is medGAN, a combination of GAN with an autoencoder to learn the distribution of discrete features of patient records (Choi et al., 2018). MedGAN, in contrast to DOPING, is used in a multilabel setting and introduces a new method to tackle mode collapse, a known problem when dealing with GAN, is also introduced in this paper: minibatch averaging. Mode collapse happens when the generator learns a specific and very plausible solution. If the next generation of discriminators do not learn to always reject this solution, the generator will start producing the same output over and over again, no matter the input it is given (Thanh-Tung & Tran, 2020). To resolve this, minibatch averaging takes the average of a minibatch of m samples and provides this to the discriminator. The discriminator can now calculate the distance from the samples it is given to the minibatch average. Thus, the generator is forced to generate diverse examples, otherwise it will be too easy for the discriminator to spot the fake samples (Choi et al., 2018).

Badu-Marf, Farooq and Paterson (2020) present the Composite Travel Generative Adversarial Network for learning the joint distribution of tabular travel attributes and sequential trip chain locations. The last feature makes it unique in this list.

TGAN (Xu & Veeramachaneni, 2018) is used for generating relational tables containing discrete and continuous variables and does not focus on learning a specific type of data or a specific class, unlike the previous GAN. Effectively generating discrete variables is done by putting noise and Kullback-Leibler divergence into the loss function. On the other hand, to deal with the multimodal distributions of continuous data, the numerical variables are clustered. CTGAN (Xu, 2020) builds upon TGAN and is trained by re-sampling the data and uses reversible data transformations. This way it can address common problems with tabular data such as mixed data types, gaussian or multimodal distributions or high dimensionality of the data. To address data imbalance in columns of the table, a conditional generator is used. CTGAN also works with a critic instead of a discriminator.

And lastly Table-GAN (Park et al., 2018) adds some extra elements to the classic GAN architecture: a third neural network is added and instead of only minimising the 'original loss', there is also an information loss and classification loss function. The third neural network is a classifier that will increase the semantic integrity of the records, meaning that the record and the label it gets should be compatible. The information loss function makes sure the mean and standard deviation of the distributions match and the classification loss function maintains the semantic integrity.

Bayesian Networks

A Bayesian network is a directed acyclic graph, which means that it is a specific type of graph that contains no cycles and has only directed edges between nodes (Stephenson,

2000). The directed edges indicate which variables depend on others. In a Bayesian network, the joint probability of all variables is equal to the product of probabilities of each variable given the value of its parent(s).

PrivBayes (Zhang, Cormode, Procopiuc, Srivastava & Xiao, 2017) is used to anonymise data. It first creates a Bayesian network that allows to model the correlations among the features of the data and to approximate the distribution of the data. Next, noise is injected to ensure differential privacy. Leaving out the noise, PrivBayes just learns the distribution of the data and then samples instances from it and thus, could certainly be used as a data augmentation technique.

Kaur et al. (2020) try to create synthetic datasets of health data by using Bayesian networks and compare the results with those of medBGAN (Baowaly, Lin, Liu & Chen, 2019). Their model can handle both numerical and categorical features and the authors find that their method outperforms medBGAN. Young, Graham and Penny (2009) on the other hand, try to create synthetical data from institutional care datasets. Sun and Erath (2015) focus on population synthesis given microsamples of this population and the complementary marginal information on the features and Gogoshin, Branciamore and Rodin (2020) focus on synthetic biological data. They use Bayesion networks to reflect the underlying networks of biological relations.

In other words, there are many possible scenarios in which Bayesian networks can be used.

1.3.2 Potential techniques

VAE

A VAE or variational autoencoder (Kingma & Welling, 2014) is a special type of autoencoder. An autoencoder consists of an encoder that compresses the input data to a latent space and a decoder that reconstructs the original data from the latent space as good as possible. What is different for a variational autoencoder is that it also tries to find a distribution in the latent space.

Variational autoencoders can be used for a wide range of applications, including generating images (Razavi, van den Oord & Vinyals, 2019; Gulrajani et al., 2016; Lu & Xu, 2018; Pu et al., 2016) or text classification where the VAE also learns to generate small sentences (Xu, Sun, Deng & Tan, 2016). Variational autoencoders can also be combined with GAN (Li, Huang, Luo, Zhang & Zha, 2021) or be used for speaker verification (Wu, Wang, Qian & Yu).

Although they have proven to be very successful in a very broad range of problems, variational autoencoders have not yet been used much to augment tabular data. Only recently the technique has been picked up to augment traffic data (Huang et al., 2019; Boquet, Morell, Serrano & Vicario, 2020; Islam, Abdel-Aty, Cai & Yuan, 2021), sales data of properties (Lee, 2021), or basic demographic data (Li, Tai & Huang, 2019).

1.4 Changing existing instances

1.4.1 Mature techniques

Interpolating

SMOTE was introduced by Chawla, Bowyer, Hall and Kegelmeyer in 2002 and has been a standard in research on data augmentation for tabular data ever since. SMOTE stands for Synthetic Minority Over-sampling Technique. Previous to SMOTE, oversampling often happened through replacement. SMOTE will take each minority example and look at its k nearest neighbours. Depending on the oversampling rate, one or more nearest neighbours are selected. Then, the difference between the feature vector in question and its nearest neighbour is taken. This difference is multiplied by a random number between zero and one and then added again to the feature vector in question. Thus, the decision region of the minority class becomes more general which improves the performance of decision tree classifiers.

Since 2002, many extensions have been proposed to alter SMOTE for specific problems such as multi-label settings (Charte, Rivera, del Jesus & Herrera, 2015), combined SMOTE with other techniques such as SVMs or boosting (Liang, Jiang, Li, Xue, & Wang, 2020; Chawla, Lazarevic, Hall & Bowyer, 2003) or just tweaked the algorithm to, for example, stop using noisy minority examples to generate synthetic samples from (Batista, Prati & Monard, 2004). For further information, the reader is referred to the 15-year anniversary article of SMOTE (Fernandez, Garcia, Herrera & Chawla, 2018) in which the original authors provide an extensive overview of all existing techniques.

Also making use of minority over-sampling is SIMO (Piri, Delen & Liu, 2018) or a synthetic informative minority over-sampling algorithm. SIMO first partitions the dataset into test and training dataset with the same imbalance as the original dataset. Next, they train a support vector machine (SVM) and select the minority examples that are closest to the SVM border as informative minority examples. Then, the nearest neighbours, only informative minority examples, of these examples are taken. The selected informative minority example will now be interpolated with a number of randomly chosen nearest neighbours to create synthetic examples. These examples are added to the dataset and a new SVM is generated. The steps are repeated until the imbalance gap is gone. As such, SIMO tackles one of the major drawbacks of SMOTE: overgeneralisation. SMOTE does not consider that majority examples might be closest located to the minority examples when generating synthetic examples. This might lead to overlapping between classes.

ADASYN stands for Adaptive Synthetic Sampling Approach (He, Bai, Garcia & Li, 2008) for Synthetic Learning and also extends SMOTE by distinguishing minority class examples that are easier or harder to learn. More synthetic examples are then generated for the latter. It does so by using a density distribution to automatically determine the number of synthetic samples that have to be generated for each minority data example. In this way the classifier decision boundary is more focused on hard to learn examples. This improves learning performance.

Simple Augmentations

Many augmentations that are simple for images, such as just rotating images, deleting random parts or adding noise are not possible or difficult to implement. There are a few that can be used nonetheless.

Similar to computer vision, where random parts of an image can be erased, the blanking technique can be used for tabular data. Vincent, Larochelle, Bengio & Manzagol (2008) worked with denoising autoencoders to fill in zeros for values that have to be masked. The only problem with this is that a zero actually might have some meaning. It would be better to use a value that does not have a meaning (Marais, 2019).

Another possibility is to add noise to the data. One possibility is to use the EZS noise method (Evans, Zayatz & Slanta, 1996). When using EZS, a single pure noise factor is generated for each record. This can, for example, be a multiplier. Furthermore, each record has a weight. The perturbed values are then calculated with the noise factor and the weight of the record (Massell, Zayatz & Funk, 2006). The MCF approach on the other hand allows you to choose from which distribution the noise should be sampled: a Gaussian distribution, Poisson distribution or Laplace distribution to transform values (van der Maaten, Chen, Tyree & Weinberger, 2014). It is also possible to create a completely synthetic dataset generated by posterior predictive models (Zayatz, 2007). The original data is used to develop a model for a certain variable and then that model is used to impute for each record a value for that variable.

It is also possible to combine adding noise and blanking values (Flossmann & Lechner, 2006). The technique was used to limit data disclosure, but the data still needed some processing afterwards to reduce bias due to measurement errors.

Mixing images has also proven to be a successful technique to augment images. Tabular data can be swapped or shuffled. Swapping implies that the attribute values are exchanged whereas with shuffling, the record i takes the attribute value of record j, which takes the attribute value of record k and so on (Muralidhar, Sarathy & Dandekar, 2006). Kosar and Scott (2018) introduced the hybrid bootstrap, another swap noise method that works in a similar manner to dropout but resamples values from other training points instead of replacing them by zeros. They ensure the corrupted values are valid attribute values. Mixup (Zhang, Cisse, Dauphin & Lopez-Paz, 2018) can be applied to images and tabular data alike and makes convex combinations of records.

Lastly, SubTab is a technique that the authors describe as similar to cropping for images (Ucar, Hajiramezanali & Edwards, 2021). The algorithm learns from a multi-view representation of the data by dividing the variables into subsets. Each subset is considered to be another view.

1.4.2 Potential techniques

Adversarial attacks

Adversarial examples are inputted to cause machine learning models to make mistakes. Changing a few pixels in images may lead a classifier to misclassify while the image does

not seem any different to the human eye. Adding a few "good words" to an email or misspelling "bad words" might lead a spam email to get past the filters.

Altering tabular data in this manner is more complex than just interchanging pixels and also easier to detect if there are no coherence constraints. Since all pixels have values between 0 and 255, this is not an issue in the image domain. Finally, an expert will not look at all the data when trying to classify an instance, but rather at a subset he deems important (Ballet et al., 2019). Thus adversarial attacks have not been used much for tabular data yet.

Ballet et al. (2019) tried to create adversarial examples to be able to get a loan. They defined their algorithm as an optimisation problem whose minimum is found using a gradient descent approach. Furthermore, they took the importance of features into account. Perturbing features that were more likely to be checked is penalised.

PermuteAttack on the other hand (Hashemi & Fathi, 2020) uses a gradient-free genetic algorithm to generate adversarial credit scorecards. Random features of the instances are permuted by choosing another possible value for the feature that is present in the training set. Perturbed instances are then scored with the help of a fitness function, the higher the obtained score, the likelier that the instance will be part of the next generation. The authors also propose a post-processing algorithm to ensure that the generated examples are realistic.

Cartella et al. apply adversarial attack on fraud detection (2021). They also make use of a generic algorithm, but implement a similarity function, which can not be higher than a certain threshold, to make sure the generated examples are realistic. The authors also introduce editability constraints. Since there are certain personal details the financial institutions always have at their disposal, these features can not be edited. Again, permuting important features is penalised.

Lastly, Miot (2021) introduces adversarial trading. The author first creates a model to predict if the market will move up and down. Next, the adversarial samples are generated using the fast gradient sign method (FGSM) and lastly an adversarial agent is created to make sure the generated examples are realistic. The author came to the conclusion that the adversarial examples cause a different outcome, but the benefits do not go to the agent who introduces them.

Chapter 2

Experiments

In this chapter, the data augmentation techniques will be tested on credit scoring data datasets. The goal is to find out which data augmentation techniques work best for credit scoring datasets and with which models they should be combined to obtain the best results. In the first part, the method used is explained: the datasets, data augmentation methods, models and metrics that were used. In the second part, the results for all data augmentation techniques, which are listed from very simple to state-of-the-art, are explained.

2.1 Method

2.1.1 Data

Three credit scoring datasets were used. The first dataset contains 5960 records and 12 features, the second contains 18918 records and because the attribute Days_late was deleted, eventually 18 features. Days_late was dropped because it correlated directly with the target variable in the dataset. In this dataset, it was assumed that if a person was more than 45 days late paying, they would default. The third dataset contains 30000 records and 23 features. The datasets were all cleansed in the following way: first, the duplicates were dropped. Furthermore, records where more than half of the features had missing values were also dropped. For the rest of the missing values, the mean was imputed in the case of numeric variables and the most frequent answer in the case of categorical variables. The possibilities for categorical data were also all put in lower case. Afterwards, weight of evidence encoding was applied to the categorical variables and the numerical variables were standardised.

2.1.2 Data augmentation techniques

Five data augmentation techniques will be tested, namely the inserting of noise, data swapping, SMOTE, CTGAN and PrivBayes. This means that all types of techniques that are currently used for tabular data are represented. The first two are simple aug-

mentation techniques, SMOTE belongs in the category of extropolating/intrapolating, CTGAN is a GAN and PrivBayes is a Bayesian network.

2.1.3 Models

The DA techniques will be combined with six models. First, three basic models: logistic regression, decision trees and KNN. These models are very different from each other and are often used together because they can give very different results (Bichler & Kiss, 2004; Shah, Patel, Sanghvi & Shah, 2020; Kulyukin, Mukherjee & Amlathe, 2018). Furthermore, XGBoost, random forests and multi-layer perceptron neural networks were also implemented because these techniques are more state-of-the-art than the previous three. The XGBoost classifier was implemented through the xgboost package in Python, the other classifiers were part of the sklearn package.

2.1.4 Metrics

The results will be compared using the accuracy, F1 and AUC metrics. All metrics are calculated as the mean after stratified 5-fold cross validation.

2.2 Results

2.2.1 Inserting noise

To insert noise, the following formula was used: the user inputs a value between zero and one for the parameter alpha. For each column, this parameter alpha is multiplied by the standard deviation of the column and a randomly generated vector that is the same length as the column and has values between zero and one. This result is then added to the original column. Higher values for the parameter alpha will result in more noise being added.
In order to find out which value for alpha works best, a loop was implemented that calculates the results of the three metrics for each value of alpha between zero and one, with step size of 0.01.
The rest of this section will look more closely at how each classifier reacts to noise insertion and whether the same results are achieved for all datasets. At the end, general conclusions are drawn about the usefulness of noise insertion.

Logistic Regression

Here (see figure 2.2) it can be seen that the values for the accuracy and the AUC measures remain more or less the same. The values for F1 do show a slight decrease. A possible explanation for the fairly stable course of the curves may be that the parameters of a logistic regression curve are estimated on the basis of the maximum likelihood estimator (MLE). This takes into account the presence of noise in the data.

Decision trees

In the first data set in particular, there is a clear increase in accuracy (see figure 2.3) when alpha is between 0.15 and 0.20. The AUC curve is also at its highest point then, even though the increase there is much smaller. The F1 measure has a small dip at the beginning, but rises briefly around the same values for alpha, after which it drops again. The accuracy continues to rise, but as both the values for the AUC and the F1 measure fall as alpha increases, alpha is best kept between 0.15 and 0.20.
In the other two datasets, we see much less effect. For dataset 2, it can still be said that there is a small increase in the F1 measure, again for alpha between 0.15 and 0.20. But dataset 3 is very stable.

KNN

Adding noise does not help a KNN classifier (see figure 2.4). Although the accuracy remains stable, the F1 and AUC values over all datasets decrease to a greater or lesser extent as more noise is added. The KNN classifier does not appear to be robust against noise and it is logical that this is therefore most evident in the smallest dataset (dataset 1).

XGBoost

The first dataset (Figure 2.5) clearly shows the same trend as the decision trees (Figure 2.3). For alpha values between 0.15 and 0.20 the accuracy, F1 and AUC increase. In datasets two and three, however, there was already only a slight increase noticeable for decision trees, but this has now completely disappeared. However, there is no harm in adding noise to the data since the measures are very stable, independent of alpha.

Random Forest

With random forests (Figure 2.6), again more or less the same trends can be seen as with decision trees and XGBoost. Here, however, it is more difficult to say whether the data augmentation technique was successful. In dataset 1 we again see an increase in all metrics, although the necessary value of alpha is slightly higher than in decision trees: approximately between 0.20 and 0.25 (instead of between 0.15 and 0.20). In the third dataset, the insertion of noise has neither a positive nor a negative effect. But in the second dataset, the F1 value has already fallen significantly when alpha reaches 0.20. The AUC value has also fallen there, but to a much lesser extent.

Multilayer Perceptron

The introduction of noise does not improve the performance of the MLP classifier on any of the datasets (Figure 2.7). From the moment alpha takes on a value greater than 0.25, the F1 value actually decreases substantially across all datasets. The AUC, although

also decreasing, is more stable, as is the accuracy. This might be the case because backpropagation deals well with noisy data.

General conclusions

All models are quite robust, even when a lot of noise is added. Only the F1 metric suffers more from a lot of noise and dares to drop sharply. Given that there is imbalance in the datasets, a stable AUC means the predictions for majority class (here: people who will not default) are robust, while a deteriorating F1 means the predictions for the minority class (here: the people who will default) are not robust. Furthermore, especially decision trees, XGBoost and random forest classifiers benefit from adding a little noise. This is logical since both XGBoost and random forests are also based on decision trees.The effect is always largest on the smallest dataset. In general, the best value for alpha can be found between 0.15 and 0.25, depending on the classifier and dataset.

BEST ALPHA VALUES WHEN INSERTING NOISE			
Model	Dataset 1	Dataset 2	Dataset 3
Logistic Regression	0	0	0
Decision Tree	0.15	0.19	0.05
KNN	0	0	0.02
XGBoost	0.20	0.01	0.54
Random Forest	0.24	0	0
MLP	0.1	0.04	0

Figure 2.1: The alpha values that generated the best results according to classifier and dataset.

Figure 2.2: The accuracy, F1 and AUC values after noise insertion for all possible values of alpha as predicted by a logistic regression classifier on dataset 1, 2 and 3.

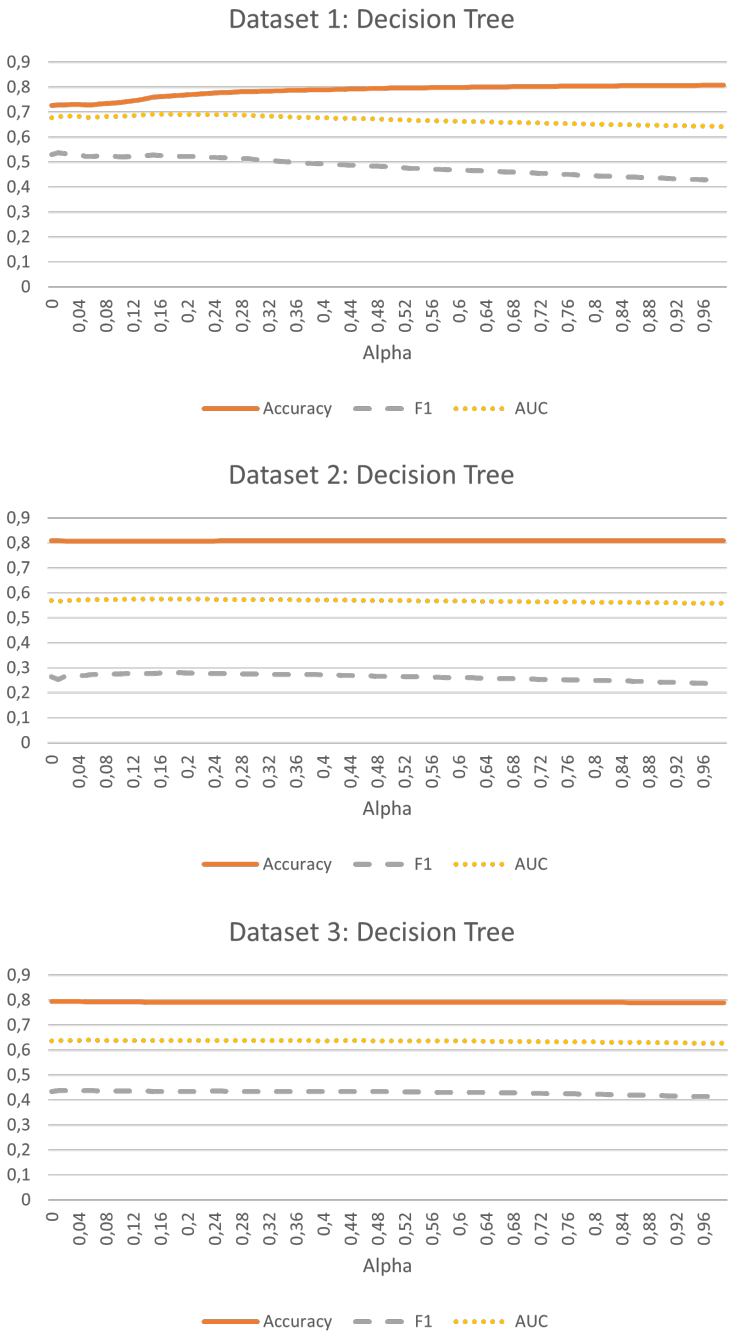

Figure 2.3: The accuracy, F1 and AUC values after noise insertion for all possible values of alpha as predicted by a decision tree classifier on dataset 1, 2 and 3.

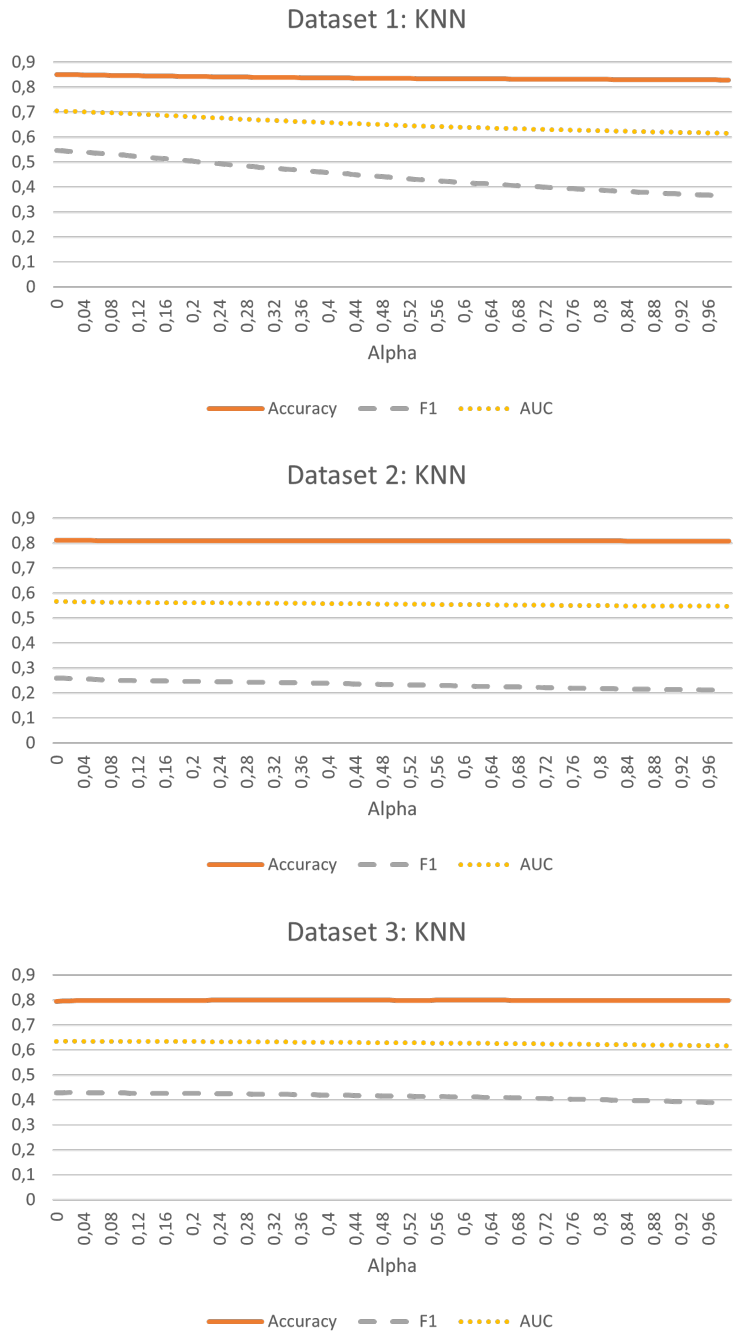

Figure 2.4: The accuracy, F1 and AUC values after noise insertion for all possible values of alpha as predicted by a k-nearest neighbour classifier on dataset 1, 2 and 3.

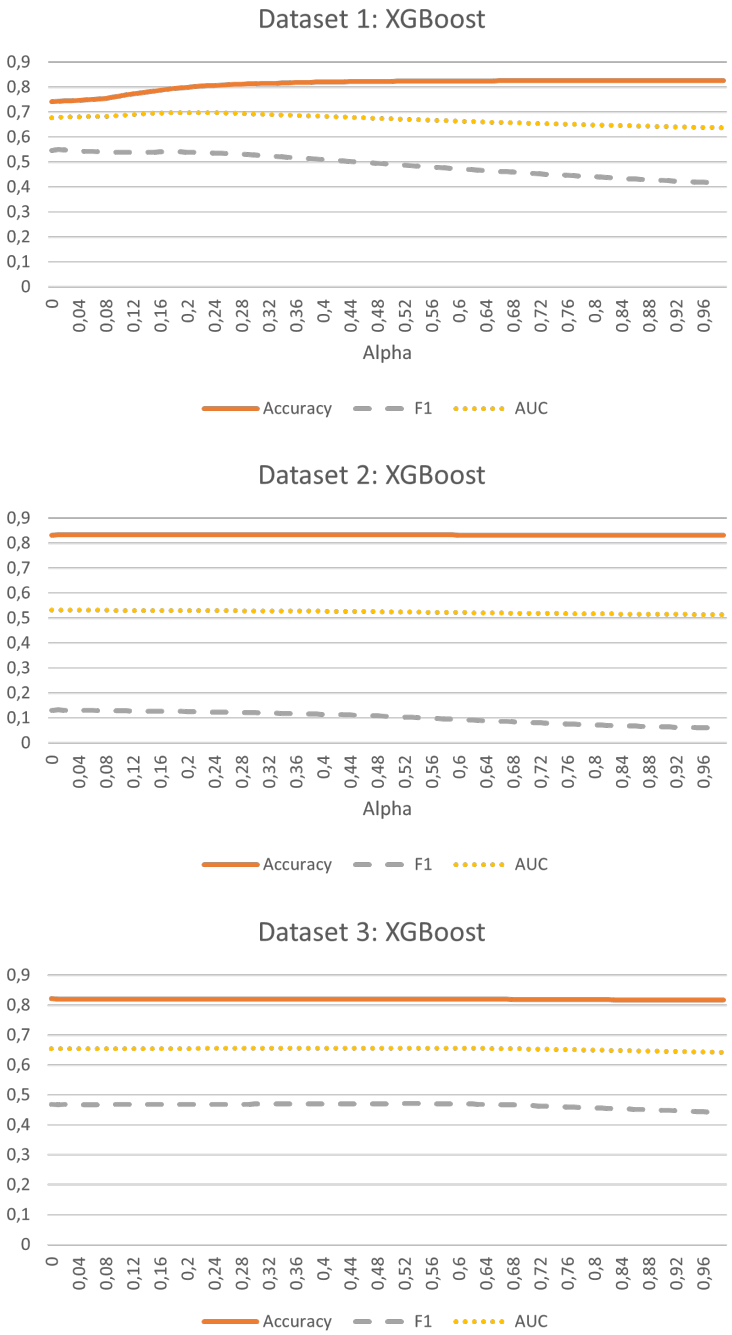

Figure 2.5: The accuracy, F1 and AUC values after noise insertion for all possible values of alpha as predicted by an XGBoost classifier on dataset 1, 2 and 3.

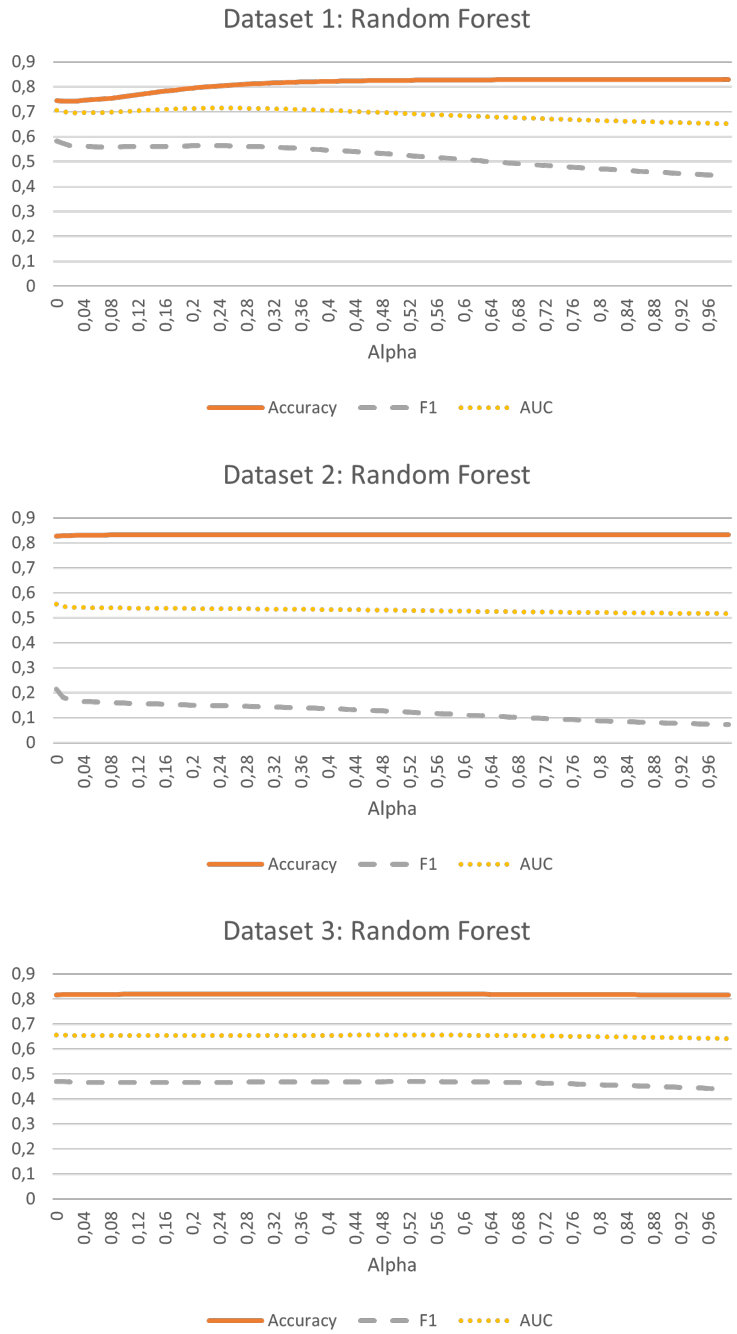

Figure 2.6: The accuracy, F1 and AUC values after noise insertion for all possible values of alpha as predicted by a random forest classifier on dataset 1, 2 and 3.

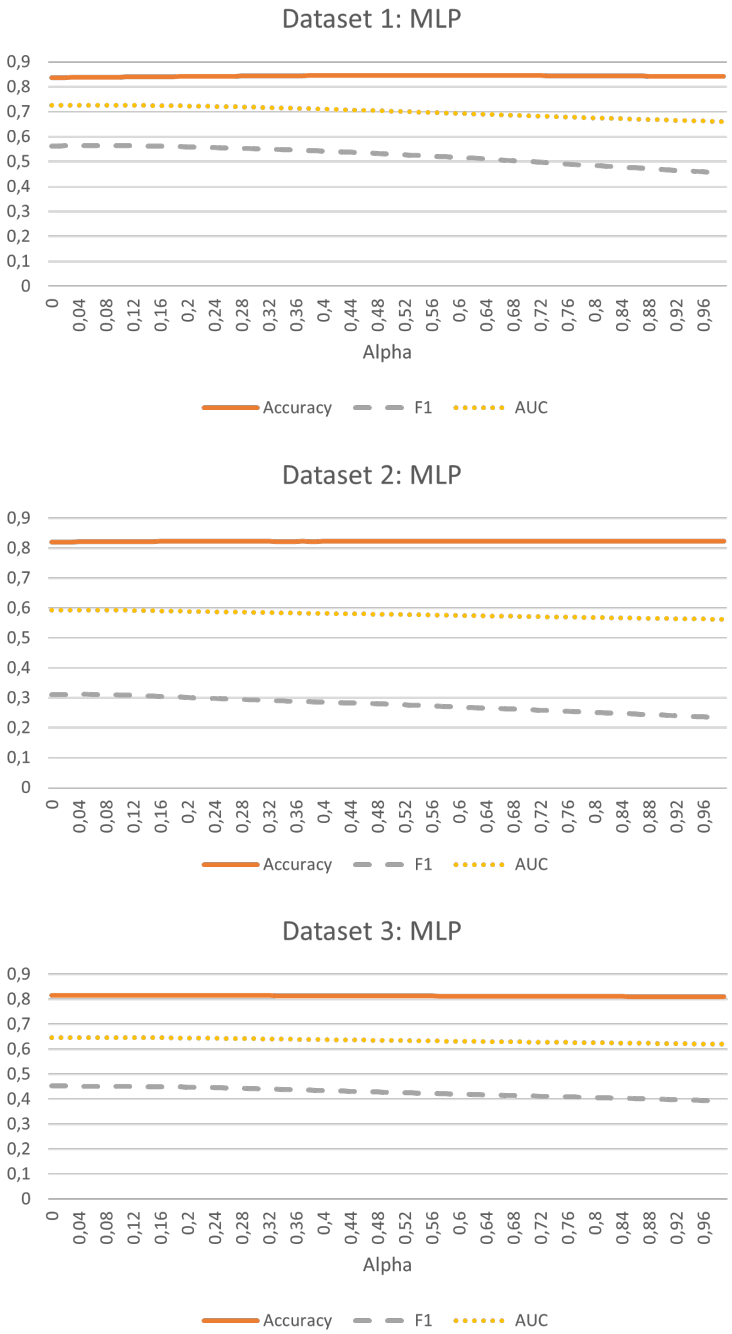

Figure 2.7: The accuracy, F1 and AUC values after noise insertion for all possible values of alpha as predicted by a multilayer perceptron classifier on dataset 1, 2 and 3.

2.2.2 Data Swapping

To swap the data, the user must first enter a value between zero and one. That is the percentage of values that will be swapped in each column. Then for each column two vectors are generated that consist of indexes. The numbers in the column with an index from the first column are replaced by the numbers that belong to the indexes from the second column. The number of columns in which data must be swapped can also be specified, but in this thesis the data augmentation technique was kept as simple as possible. Given that all categorical data have been converted using weight-of-evidence encoding, and that all variables have been standardised, it is possible to swap data in all columns.

Logistic Regression

While the accuracy and AUC remain quite stable even when all columns are completely shuffled, the same can not be said for the F1 measure. For the third dataset, it is only when more than 75% of the instances per column are swapped, that the F1 curve starts to decline. But for the other two datasets this is not the case, as can clearly be seen in the graphs (Figure 2.9). The F1 curves decline almost immediately.

Decision Trees

Unlike when noise is added, the decision trees experience no positive effect from swapping the data. The difference here could be due to the fact that when swapping data, the boundaries of the various values are not shifted. When, for example, noise is added to the highest column value of a class, a decision tree must take this into account. The data will, after the addition of noise, also contain values which were not previously present in the data set. This is not the case when the data is simply swapped. The decision tree classifiers are more robust than the logistic regression classifiers.

KNN

In the case of the KNN classifiers, we can see the same trends arise when instances are swapped as when noise is inserted. Again, KNN classifiers cannot handle the data augmentation. The reason for this could be that while the other techniques try to generalize trends across all instances, KNN tries to find the most alike instances. By perturbing feature values, other neighbours will be selected, which leads to an underperforming model.

XGBoost

XGBoost also does not benefit from data swapping data augmentation. The same tendencies can be seen here (Figure 2.12) as with decision trees. It is remarkable that in the first data set, the accuracy increases as soon as more than 65% of the instances are swapped.

Random Forest

Also for random forest classifiers, the same trends can be seen as for decision trees and XGBoost. The accuracy of dataset one increases very slightly from the beginning. With datasets two and three, however, this trend is not visible.

Multilayer Perceptron

The MLP classifiers are again very stable. Only the F1 curve is falling slightly.

General conclusions

None of the models perform better after the data swapping procedures. Nevertheless, most models are quite robust, even when swapping many instances. The graphs show that the first and thus the smallest dataset is most affected by data swapping. The third dataset on the other hand, is least affected by the perturbations of the data set. Moreover, the F1 measure is sensitive to data swapping, more so than the other two metrics. The table underneath shows the percentages of swapped data that obtained the best result per model and dataset. Most values are 0 (no swapped instances) or close to zero and even when the percentage is not equal to zero, the difference in performance is minimal.

BEST PERCENTAGE OF FEATURES TO SWAP			
Model	Dataset 1	Dataset 2	Dataset 3
Logistic Regression	0	0	0
Decision Tree	0.03	0	0.09
KNN	0	0	0
XGBoost	0.05	0	0.15
Random Forest	0	0	0.01
MLP	0	0.01	0

Figure 2.8: The percentage of swapped features that obtained the best performance per model and dataset.

Figure 2.9: The accuracy, F1 and AUC values after data swapping for all possible values of alpha as predicted by a logistic regression classifier on dataset 1, 2 and 3.

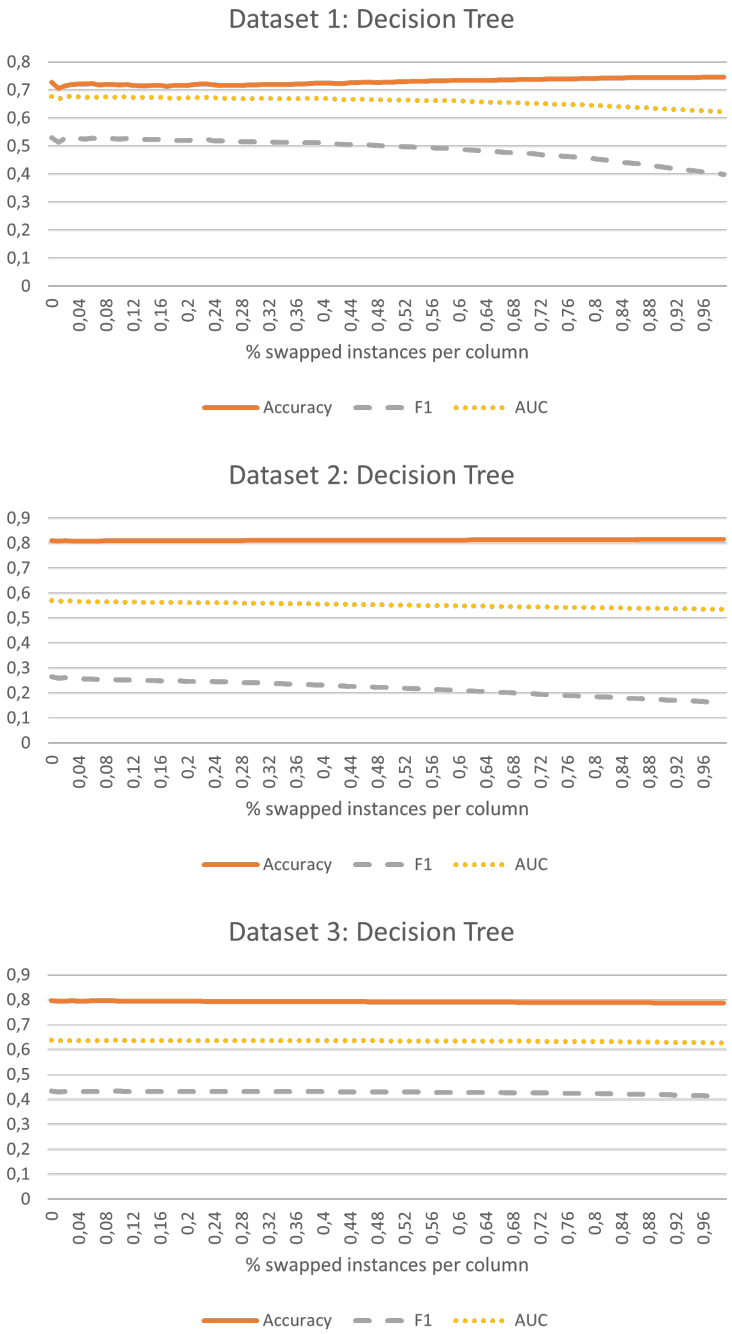

Figure 2.10: The accuracy, F1 and AUC values after data swapping for all possible values of alpha as predicted by a decision tree classifier on dataset 1, 2 and 3.

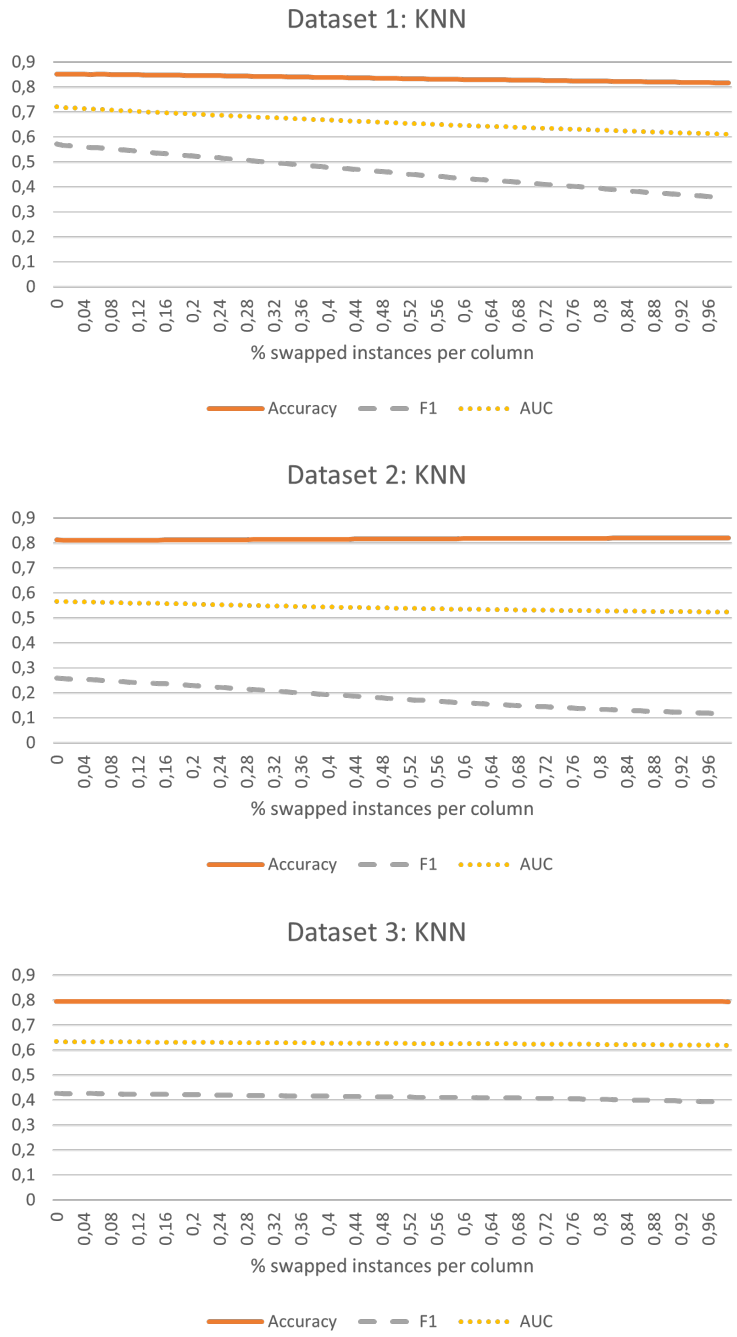

Figure 2.11: The accuracy, F1 and AUC values after data swapping for all possible values of alpha as predicted by a k-nearest neighbour classifier on dataset 1, 2 and 3.

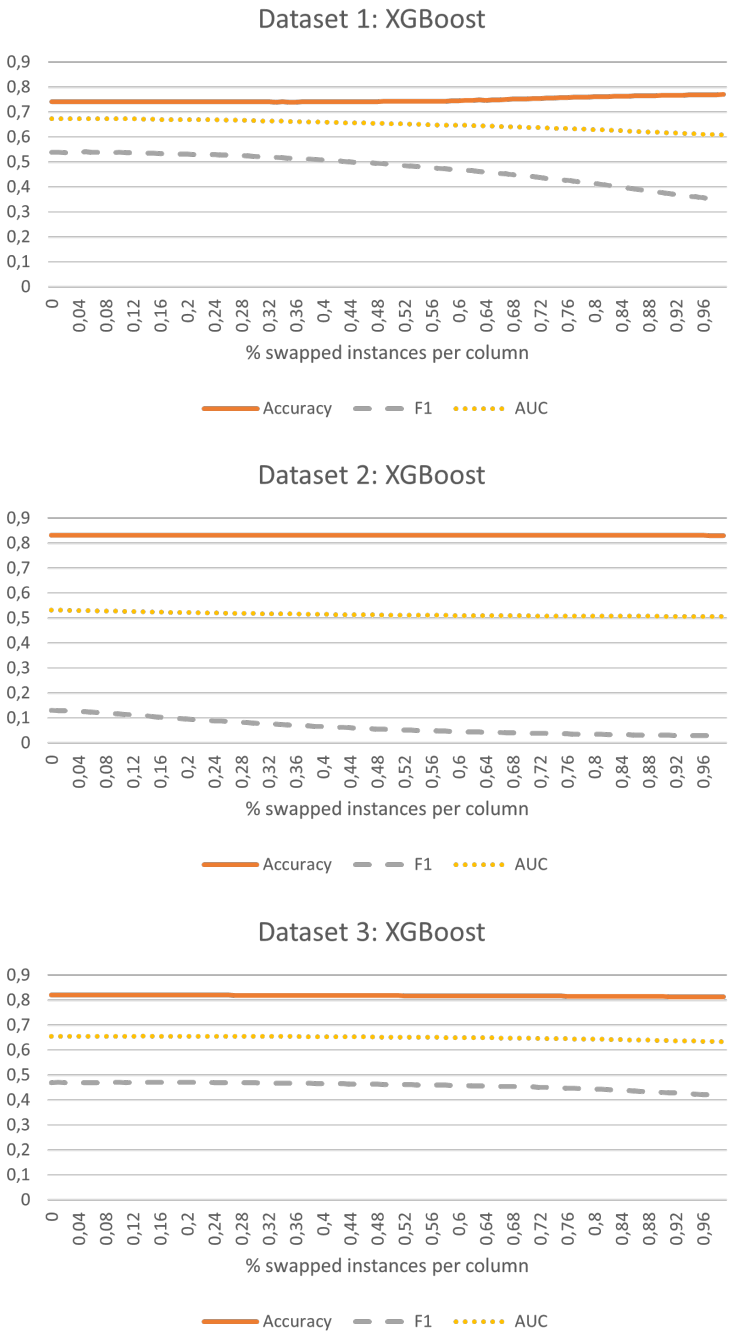

Figure 2.12: The accuracy, F1 and AUC values after data swapping for all possible values of alpha as predicted by an XGBoost classifier on dataset 1, 2 and 3.

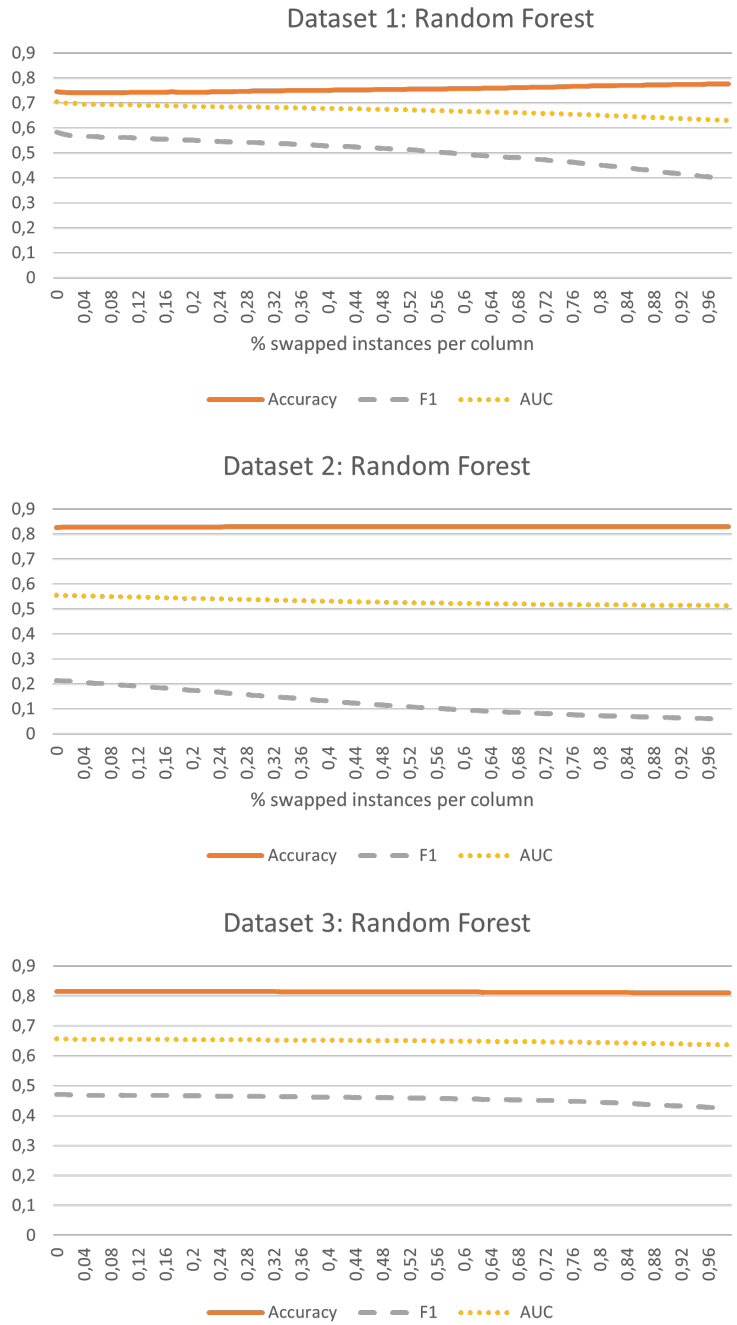

Figure 2.13: The accuracy, F1 and AUC values after data swapping for all possible values of alpha as predicted by a random forest classifier on dataset 1, 2 and 3.

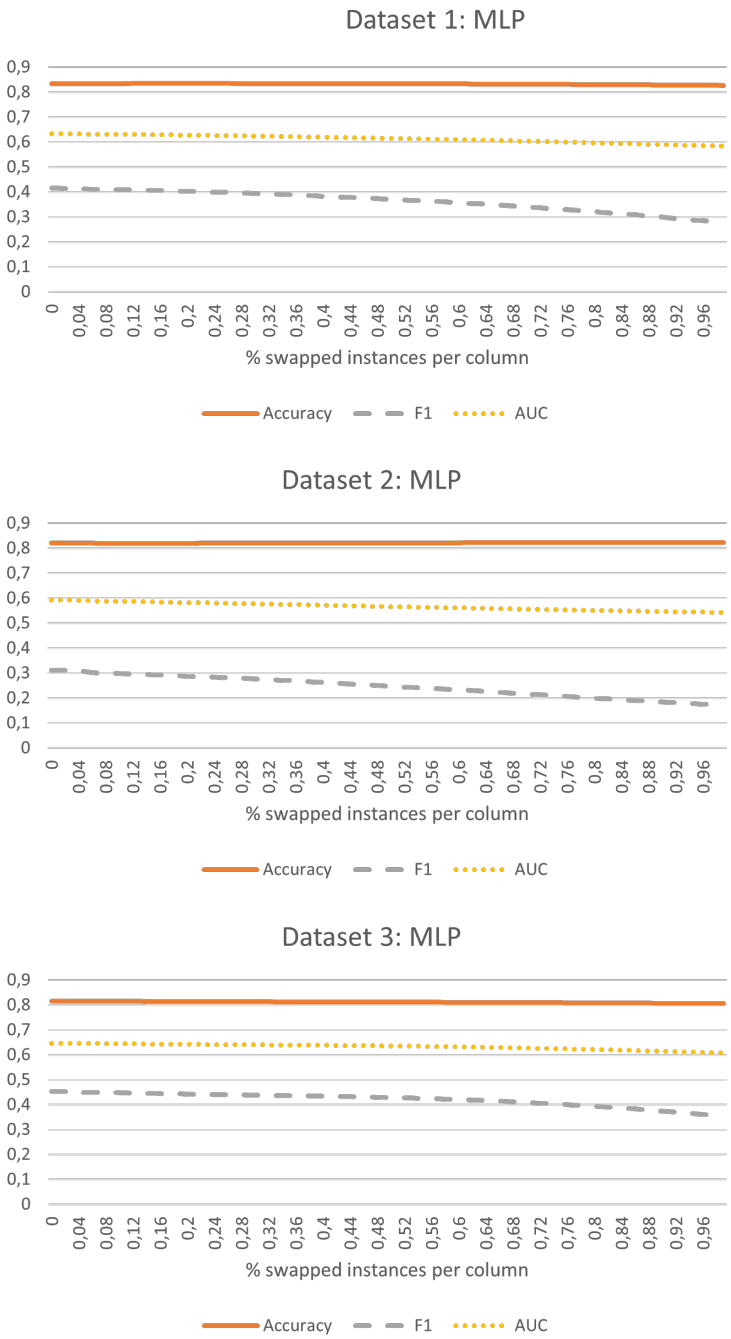

Figure 2.14: The accuracy, F1 and AUC values after data swapping for all possible values of alpha as predicted by a multilayer perceptron classifier on dataset 1, 2 and 3.

2.2.3 SMOTE

What is SMOTE again?

SMOTE stands for Synthetic Minority Over-sampling Technique (Chawla, Bowyer, Hall & Kegelmeyer, 2002). SMOTE will take each minority example and look at its k nearest neighbours. One or more of these neighbours are then selected depending on the over-sampling rate. Then the difference between the chosen minority example and its nearest neighbour is taken, multiplied by a random number between 0 and 1 and added to the chosen example. In this way, the decision region of the minority class becomes more general.

Implementation

SMOTE was imported from the imbalanced learn (imblearn) over_sampling library. The algorithm is an implementation of the original article by Chawla, Bowyer, Hall and Kegelmeyer. It is possible to oversample only the minority class, but also to oversample all classes. Since, in this case, the setting of the parameter made no difference, only the results for minority class over-sampling are shown below.

How to read the results?

Results are shown in the tables below. For each dataset, two columns are shown. One containing the performance values of the models when no data augmentation is applied and one containing the performance values of the models after applying data augmentation. When the second value is lower than the first, it is put between brackets. Furthermore, this value is put in bold when the difference, positive or negative, in performance before and after data augmentation is larger than 5%. Since all values are the mean of the results after stratified 5-fold cross validation, an ANOVA, with alpha equal to 0.05, was performed to test the significance of the differences. When the difference is significant, these values are written in italic and underlined. Tables showing results after applying CTGAN and PrivBayes are constructed in the same way.

Results

When comparing the three metrics, it is immediately noticeable that the performance after SMOTE is much lower with regard to accuracy and much higher for the F1 and AUC metrics. The accuracy is lower for all datasets and all models after the application of SMOTE than before. The F1 and AUC values, however, are almost always higher after data augmentation.
For dataset 1, the accuracy decreases the least dramatically after applying SMOTE, but the F1 and AUC values also increase the least compared to the other datasets. Relatively speaking, SMOTE is most successful on the second dataset. The difference between the F1 and AUC metrics before and after data augmentation is greatest for the second dataset. SMOTE was also successfully applied on the third dataset, but the

differences are not as large and/or significant as for dataset 2. This can also be seen on the graphs in the appendix (see figure A.1).
What can be seen even more clearly on these graphs is that the largest differences in performance, both in a negative sense for the accuracy and in a positive sense for F1 and AUC, are achieved with the logistic regression and MLP classifiers. These models performed poorly before data augmentation was applied, but are among the better models after data augmentation. It makes sense that both models behave in the same way since, in the abstract, MLP can also be seen as a regression model with many intermediate steps. XGBoost also gets quite good results, but the difference in performance there depends a bit more from dataset to dataset. Finally, the KNN classifier performs very well for the first dataset, but not for the others. This is logical since KNN performs best on small datasets.

SMOTE : accuracy

Model	Dataset 1		Dataset 2		Dataset 3	
	Without DA	With DA	Without DA	With DA	Without DA	With DA
Logistic Regression	0,833741	***(0,764406)***	0,831307	***(0,659441)***	0,809678	***(0,679927)***
Decision Tree	0,727339	(0,685374)	0,809316	***(0,758353)***	0,796062	***(0,727482)***
KNN	0,852633	(0,820392)	0,810848	*(0,697029)*	0,795428	***(0,678191)***
XGBoost	0,740687	(0,729066)	0,832417	***(0,752167)***	0,821225	*(0,789288)*
Random Forest	0,744328	(0,730108)	0,826866	(0,812434)	0,815685	*(0,794794)*
MLP	0,833048	**(0,756439)**	0,832258	*(0,697401)*	0,815852	***(0,752178)***

SMOTE : F1

Model	Dataset 1		Dataset 2		Dataset 3	
	Without DA	With DA	Without DA	With DA	Without DA	With DA
Logistic Regression	0,363554	***0,497629***	0,085017	***0,424255***	0,35434	***0,472256***
Decision Tree	0,529902	(0,505227)	0,264381	***0,338358***	0,433687	0,449096
KNN	0,571273	0,613233	0,263193	***0,366092***	0,427678	0,44503
XGBoost	0,539208	**0,581844**	0,130967	***0,423764***	0,469158	***0,530593***
Random Forest	0,582282	(0,569138)	0,213791	***0,327838***	0,470623	0,501411
MLP	0,414962	***0,513856***	0,164856	***0,451543***	0,452441	***0,522497***

SMOTE : AUC

Model	Dataset 1		Dataset 2		Dataset 3	
	Without DA	With DA	Without DA	With DA	Without DA	With DA
Logistic Regression	0,610142	***0,701505***	0,518974	***0,690587***	0,604462	***0,667514***
Decision Tree	0,676846	0,700657	0,570145	*0,603391*	0,637506	0,646867
KNN	0,720794	**0,777491**	0,568223	***0,624823***	0,634561	0,644157
XGBoost	0,67391	**0,734447**	0,531429	***0,667084***	0,65512	*0,699291*
Random Forest	0,704954	0,718829	0,555141	*0,598839*	0,656314	0,677075
MLP	0,633151	***0,716714***	0,540397	***0,706355***	0,646919	***0,701486***

Figure 2.15: The accuracy, F1 and AUC values for all datasets as predicted by the models on the datasets with and without SMOTE data augmentation. Legend: (...): negative difference i.e. the models perform worse after data augmentation - **Bold**: the difference in performance before and after data augmentation is larger than 5% - *underlined and in italics*: the difference is found to be significant after performing an ANOVA with alpha equal to 0.05.

2.2.4 CTGAN

What is CTGAN again?

CTGAN is a Generative Adversarial Network. As explained before, two models are always trained when working with GANs, a generator to learn the distribution of the model and a discriminator to discern whether a given instance belongs to the original dataset or was created by the generator (Goodfellow et al., 2014). CTGAN (Xu, 2020), that was based on the previous work TGAN (Xu & Veeramachaneni, 2018) works with a conditional generator to address data imbalance in columns of the table and a critic instead of a discriminator that just tries to make the output bigger for real instances than for fake instances, without using the 0.5 threshold.

Implementation

The installed CTGAN implementation is part of *The Synthetic Data Vault Project*, a project from DataCebo to which Xu contributed. The code for this repository can be found on GitHub and the repository can easily be installed using a pip command.

500 epochs

What is immediately noticeable is that the performance after data augmentation with CTGAN is almost always lower than before. Upon closer inspection, it can be seen that especially the F1 values drop significantly. This is where most of the strong negative differences can be found, which also often turn out to be significant after the ANOVA test. The AUC values are also considerably lower after data augmentation, although the largest differences can mostly be seen for the first data set. It was also more difficult for the GAN to get good results on the first dataset since the values for the metrics were the highest here anyway.
Relatively speaking, the accuracy is less affected. This suggests that CTGAN can replicate certain classes very well and other classes less so. In this case, it could be further investigated whether there are large differences in accuracy between the different classes.
As far as the accuracy is concerned, the best results are obtained with the first data set as opposed to the F1 and AUC measures. The difference here is even positive for half of the models: the decision tree, XGBoost and random forest models. The latter two are based on decision trees and it is therefore logical that these models perform in the same line.
However, since no line can be drawn in the results of the models across the datasets, it is too early to make any confident statements about the differences in performance between the models. While the first dataset scores well on accuracy and poorly on F1 and AUC, the results of the second and third datasets are much more aligned across the three metrics. The second scores very evenly and although never good, the results after data augmentation are almost never more than 5% or significantly worse. In the third dataset, the results fluctuate a bit more across the models, with the models based on decision trees doing poorer here.

CTGAN 500 : accuracy

Model	Dataset 1		Dataset 2		Dataset 3	
	Without DA	With DA	Without DA	With DA	Without DA	With DA
Logistic Regression	0,832701	(0,820042)	0,831835	(0,826707)	0,810189	(0,798988)
Decision Tree	0,727339	0,770978	0,809316	(0,8026)	0,794121	*(0,753378)*
KNN	0,851247	***(0,800448)***	0,81661	(0,807199)	0,795586	(0,772953)
XGBoost	0,742074	0,79837	0,833104	(0,831835)	0,820193	*(0,798254)*
Random Forest	0,744328	0,798196	0,826866	0,828875	0,816758	*(0,79442)*
MLP	0,833221	(0,824549)	0,832258	(0,826866)	0,816892	(0,797922)

CTGAN 500 : F1

Model	Dataset 1		Dataset 2		Dataset 3	
	Without DA	With DA	Without DA	With DA	Without DA	With DA
Logistic Regression	0,397325	**(0,343271)**	0,164479	(0,15159)	0,358506	(0,404033)
Decision Tree	0,529902	***(0,275808)***	0,264381	(0,22617)	0,441702	***(0,383419)***
KNN	0,547654	***(0,342638)***	0,237954	(0,19656)	0,422608	(0,407351)
XGBoost	0,546553	***(0,239161)***	0,164602	(0,12925)	0,464416	(0,416247)
Random Forest	0,582282	***(0,19612)***	0,213791	**(0,13303)**	0,466201	**(0,405058)**
MLP	0,414544	(0,37131)	0,167482	(0,15077)	0,458435	(0,424154)

CTGAN 500 : AUC

Model	Dataset 1		Dataset 2		Dataset 3	
	Without DA	With DA	Without DA	With DA	Without DA	With DA
Logistic Regression	0,625977	(0,606486)	0,540142	(0,53744)	0,60611	(0,625344)
Decision Tree	0,676846	**(0,566342)**	0,570145	(0,553092)	0,64176	*(0,609714)*
KNN	0,705048	***(0,603536)***	0,560021	(0,545806)	0,632021	(0,624384)
XGBoost	0,677734	***(0,56156)***	0,54152	(0,531833)	0,653	(0,631887)
Random Forest	0,704954	***(0,546896)***	0,555141	(0,532287)	0,654249	(0,626352)
MLP	0,632928	(0,619543)	0,541142	(0,536914)	0,649854	(0,635182)

Figure 2.16: The accuracy, F1 and AUC values for all datasets as predicted by the models on the datasets with and without CTGAN data augmentation with 500 epochs. Legend: (...): negative difference i.e. the models perform worse after data augmentation - **Bold**: the difference in performance before and after data augmentation is larger than 5% - *underlined and in italics*: the difference is found to be significant after performing an ANOVA with alpha equal to 0.05.

1000 epochs

The results after data augmentation using CTGAN with 1000 epochs are somewhat surprising. The results for the first and second datasets are on average higher, as might be expected, but the results for the third dataset are lower than for 500 epochs. For the first two datasets, the results are in the same line as above. Dataset 1 scores particularly well on accuracy and less so on F1 and AUC, although the results are not as dramatic as for CTGAN with 500 epochs. In the second dataset, only the scores on accuracy are slightly lower, but the F1 and AUC values are clearly higher than above. With dataset 3, an opposite trend is visible: the accuracy is slightly higher while the F1 and AUC measurements are a lot lower.
After data augmentation, the decision tree, XGBoost and random forest classifier again give good results in terms of accuracy for dataset 1. However, these results cannot be

extended across datasets or metrics. In dataset 2, the random forest and MLP classifiers perform slightly less with respect to F1 and AUC; in the third dataset, the MLP classifier together with the KNN classifier performs slightly better. Which classifiers are better or worse in combination with CTGAN therefore strongly depends on the dataset used.

CTGAN 1000 : accuracy

Model	Dataset 1		Dataset 2		Dataset 3	
	Without DA	With DA	Without DA	With DA	Without DA	With DA
Logistic Regression	0,833741	(0,822124)	0,832205	*(0,806935)*	0,809678	*(0,792792)*
Decision Tree	0,727339	**0,817962**	0,809316	*(0,768026)*	0,796062	*(0,767929)*
KNN	0,852633	***(0,793697)***	0,812223	*(0,779498)*	0,795428	(0,781445)
XGBoost	0,740687	**0,830097**	0,832417	*(0,809156)*	0,821225	*(0,800367)*
Random Forest	0,744328	**0,828363**	0,826866	(0,809473)	0,815685	*(0,796663)*
MLP	0,833048	(0,82316)	0,820206	(0,793508)	0,815852	(0,802002)

CTGAN 1000 : F1

Model	Dataset 1		Dataset 2		Dataset 3	
	Without DA	With DA	Without DA	With DA	Without DA	With DA
Logistic Regression	0,363554	**(0,261513)**	0,15752	**0,2255**	0,35434	**(0,22933)**
Decision Tree	0,529902	**(0,41832)**	0,264381	0,29437	0,433687	***(0,318554)***
KNN	0,571273	***(0,388854)***	0,259978	0,2868	0,427678	**(0,342219)**
XGBoost	0,539208	**(0,418081)**	0,130967	**0,20521**	0,469158	***(0,318125)***
Random Forest	0,582282	**(0,379136)**	0,213791	(0,21326)	0,470623	***(0,290587)***
MLP	0,414962	**(0,363868)**	0,310667	(0,3039)	0,452441	**(0,33501)**

CTGAN 1000 : AUC

Model	Dataset 1		Dataset 2		Dataset 3	
	Without DA	With DA	Without DA	With DA	Without DA	With DA
Logistic Regression	0,610142	(0,574394)	0,538254	0,559555	0,604462	(0,56371)
Decision Tree	0,676846	(0,640605)	0,570145	0,579077	0,637506	***(0,586025)***
KNN	0,720794	***(0,621852)***	0,567312	0,575306	0,634561	(0,59832)
XGBoost	0,67391	(0,64317)	0,531429	0,551197	0,65512	***(0,595946)***
Random Forest	0,704954	**(0,624222)**	0,555141	(0,552506)	0,656314	***(0,586225)***
MLP	0,633151	(0,616034)	0,592964	(0,588081)	0,646919	(0,602503)

Figure 2.17: The accuracy, F1 and AUC values for all datasets as predicted by the models on the datasets with and without CTGAN data augmentation with 1000 epochs. Legend: (...): negative difference i.e. the models perform worse after data augmentation - **Bold**: the difference in performance before and after data augmentation is larger than 5% - *underlined and in italics*: the difference is found to be significant after performing an ANOVA with alpha equal to 0.05.

Conclusion

In this case, the use of CTGAN data augmentation yields lower performance far more often than higher performance. Moreover, it is impossible to estimate in advance which combination of models and metrics will best respond to the data augmentation as this is different for each dataset. Here, of course, a maximum of 1000 epochs were run. Perhaps the results would improve with a higher number of epochs. But that would require a lot of processing power and/or time. Furthermore, only the number of epochs was adjusted,

the algorithm itself was not changed. Tuning the algorithm to the datasets could also be a possibility for improvement.

2.2.5 PrivBayes

What is PrivBayes again?

PrivBayes (Zhang, Cormode, Procopiue, Srivastava & Xiao, 2017) creates a Bayesian network that is used to model the correlations among the features and approximate the distribution of the data. Normally, noise is injected afterwards to ensure differential privacy. But, since the goal of this thesis is not to secure the data for privacy reasons, but merely to augment the data, no noise will be added.

Implementation

To implement PrivBayes, the DataSynthesizer package was used (Ping, Stoyanovich & Hoye, 2017). Their package works in three different modes: random mode, independent attribute mode and correlated attribute mode. The last mode implements PrivBayes. The implementation of this package can be found on GitHub as well. Two parameters could be set: epsilon, to define how much noise should be inserted and the degree of the Bayesian network, to define how many parents each attribute has. Epsilon was put to zero so that no noise was added. The degree of the Bayesian network was put to two because of constraints concerning computing power.

Results

PrivBayes does not offer good results. The values of the metrics are almost always lower after applying PrivBayes. The largest differences can be seen in the first data set. There the difference in accuracy is actually positive a number of times. Before data augmentation, the decision tree, XGBoost, and random forest classifiers did not get very good results compared to the other three, but after applying PrivBayes, the results of the six models are much more in line. However, this trend does not extend to the F1 and AUC values. There, the differences are always negative and the logistic regression, decision tree and MLP classifier decrease the least and thus obtain the best results. In the second and third dataset, the results after data augmentation are always just below the original results and more or less the same trend line is followed (see also figure A.4). The differences are smallest for the third dataset. This is also the largest dataset. It is possible that PrivBayes still needs quite a lot of data to be trained properly. But the purpose of data augmentation is precisely to augment small data sets in a realistic yet cheap way. However, only as many instances were sampled after PrivBayes as were in the original dataset. So, it could be tested whether adding sampled instances to the original dataset or just using larger datasets gives better results for this technique.
Looking at the various metrics, it turns out that the F1 measure suffers the most. On average, the classifiers perform worse on F1 after data augmentation, across all datasets, compared to the other two measures.

PrivBayes : accuracy

Model	Dataset 1		Dataset 2		Dataset 3	
	Without DA	With DA	Without DA	With DA	Without DA	With DA
Logistic Regression	0,833741	*(0,814844)*	0,832153	(0,829826)	0,809678	(0,804939)
Decision Tree	0,727339	**0,803749**	0,809316	(0,799641)	0,796062	(0,777807)
KNN	0,852633	*(0,8138)*	0,812223	(0,803341)	0,795428	(0,794594)
XGBoost	0,740687	**0,821953**	0,832417	*(0,828611)*	0,821225	(0,818321)
Random Forest	0,744328	**0,820219**	0,826866	(0,825491)	0,815685	(0,793492)
MLP	0,833048	(0,821257)	0,820206	(0,814072)	0,815852	(0,812148)

PrivBayes : F1

Model	Dataset 1		Dataset 2		Dataset 3	
	Without DA	With DA	Without DA	With DA	Without DA	With DA
Logistic Regression	0,363554	***(0,209747)***	0,157064	***(0,075647)***	0,35434	(0,32235)
Decision Tree	0,529902	***(0,372222)***	0,264381	***(0,192422)***	0,433687	(0,421217)
KNN	0,571273	***(0,312144)***	0,259978	***(0,138574)***	0,427678	***(0,352207)***
XGBoost	0,539208	***(0,326174)***	0,130967	***(0,040575)***	0,469158	(0,456055)
Random Forest	0,582282	***(0,334727)***	0,213791	***(0,048975)***	0,470623	(0,438886)
MLP	0,414962	***(0,30105)***	0,310667	***(0,14058)***	0,452441	(0,43512)

PrivBayes : AUC

Model	Dataset 1		Dataset 2		Dataset 3	
	Without DA	With DA	Without DA	With DA	Without DA	With DA
Logistic Regression	0,610142	***(0,555019)***	0,538098	*(0,516095)*	0,604462	(0,592079)
Decision Tree	0,676846	**(0,612292)**	0,570145	*(0,538016)*	0,637506	(0,630374)
KNN	0,720794	***(0,588096)***	0,567312	*(0,520756)*	0,634561	*(0,600715)*
XGBoost	0,67391	**(0,597131)**	0,531429	*(0,507421)*	0,65512	(0,64872)
Random Forest	0,704954	**(0,598384)**	0,555141	*(0,507527)*	0,656314	(0,640769)
MLP	0,633151	*(0,585836)*	0,592964	***(0,526102)***	0,646919	(0,638494)

Figure 2.18: The accuracy, F1 and AUC values for all datasets as predicted by the models on the datasets with and without PrivBayes data augmentation. Legend: (...): negative difference i.e. the models perform worse after data augmentation - **Bold**: the difference in performance before and after data augmentation is larger than 5% - *underlined and in italics*: the difference is found to be significant after performing an ANOVA with alpha equal to 0.05.

2.3 Conclusion

The two simplest techniques to implement and to understand were tested first in this section. There is one technique that certainly never works: data swapping. The performance was invariably lower after applying this technique. Adding noise to the data, on the other hand, could produce interesting results. Decision trees, XGBoost and random forests in particular benefited from this technique. However, the results depended on the dataset used, with the smallest dataset achieving the best results.
The best known data augmentation technique for tabular data proved why it is a standard within this research area. Although the accuracy is lower after applying SMOTE, the performance in terms of F1 and AUC is much higher. The classifiers that benefit most from SMOTE are the logistic regression and MLP classifiers. Using a logistic

Best combinations of models and DA techniques						
	Logistic Regression	Decision Tree	KNN	XGBoost	Random Forest	MLP
Noise	X	V*	X	V*	V*	X
Data Swapping	X	X	X	X	X	X
SMOTE	V	X	V*	V-	X	V
CTGAN	Indeterminate					
PrivBayes	Indeterminate					

Figure 2.19: Table showing the best combinations of DA techniques and models for credit scoring datasets. Legend: X: This combination does not work - V: This combination works - V*: This combination works optimally for small datasets - V-: This combination works, but is not optimal relative to the others - Indeterminate: No conclusions could be drawn based on the research done here.

regression model has the advantage of using a simple, easy and quickly implementable model, but the disadvantage is that this model is not as powerful as the others. So although these classifiers benefit greatly from the application of SMOTE, this does not automatically mean that they are the best performing. An MLP classifier on the other hand is very high-performing but also requires the most time and computing power to implement. So depending on the available capacity, it can also be useful to implement for example an XGBoost classifier. Relatively speaking, the results increase less, but this is a high-performance classifier that still requires less time and processing power than an MLP model. When working with small datasets, the KNN classifier also proved a valid option.
Although the latter two techniques are the most state-of-the-art, little benefit has been gained from them. With CTGAN, the reason might be that the algorithm was not run with enough epochs. From 500 to 1000 epochs, there was a noticeable improvement in the results. PrivBayes did best with the largest data set. From this we can cautiously conclude that this technique needs quite a lot of data to be trained, which somewhat contradicts the purpose of data augmentation. In any case, these techniques are more difficult to implement and tune and require a lot of time to run. Depending on the context and the company that wants to implement them, it might be better to go for a slightly simpler technique.

2.4 Discussion

Due to a lack of time and computing power, not all algorithms were implemented optimally. For example, PrivBayes could be tested with different degrees in the Bayesian Network and CTGAN could be run with even more epochs. Furthermore, analysing the results gave rise to other ideas: instead of using only the samples generated by the CTGAN or PrivBayes algorithm, these could also be added to the original dataset. The code of these algorithms could also be adapted to better fit specific (types of) datasets. Different techniques could also be combined: noise or SMOTE could be implemented first and a more complex technique afterwards. Furthermore, only techniques that are

already quite well-known within this research area have been implemented here, but other techniques such as VAE or adversarial attacks also offer possibilities that have not been explored here.

Appendix A

A.1 Appendix

A.1.1 Graphs SMOTE

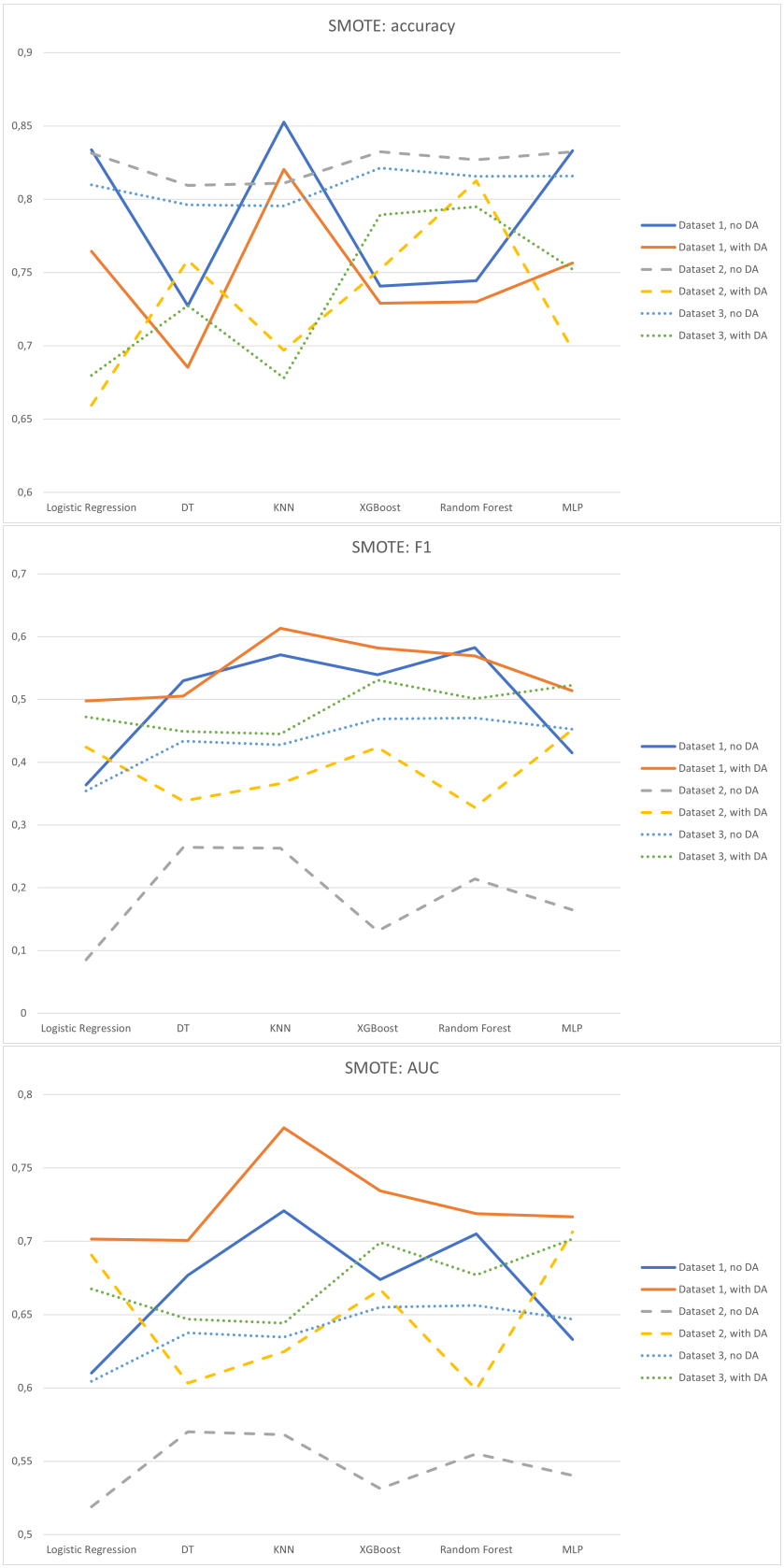

Figure A.1: The accuracy, F1 and AUC given for all models and datasets before and after applying SMOTE.

A.1.2 Graphs CTGAN

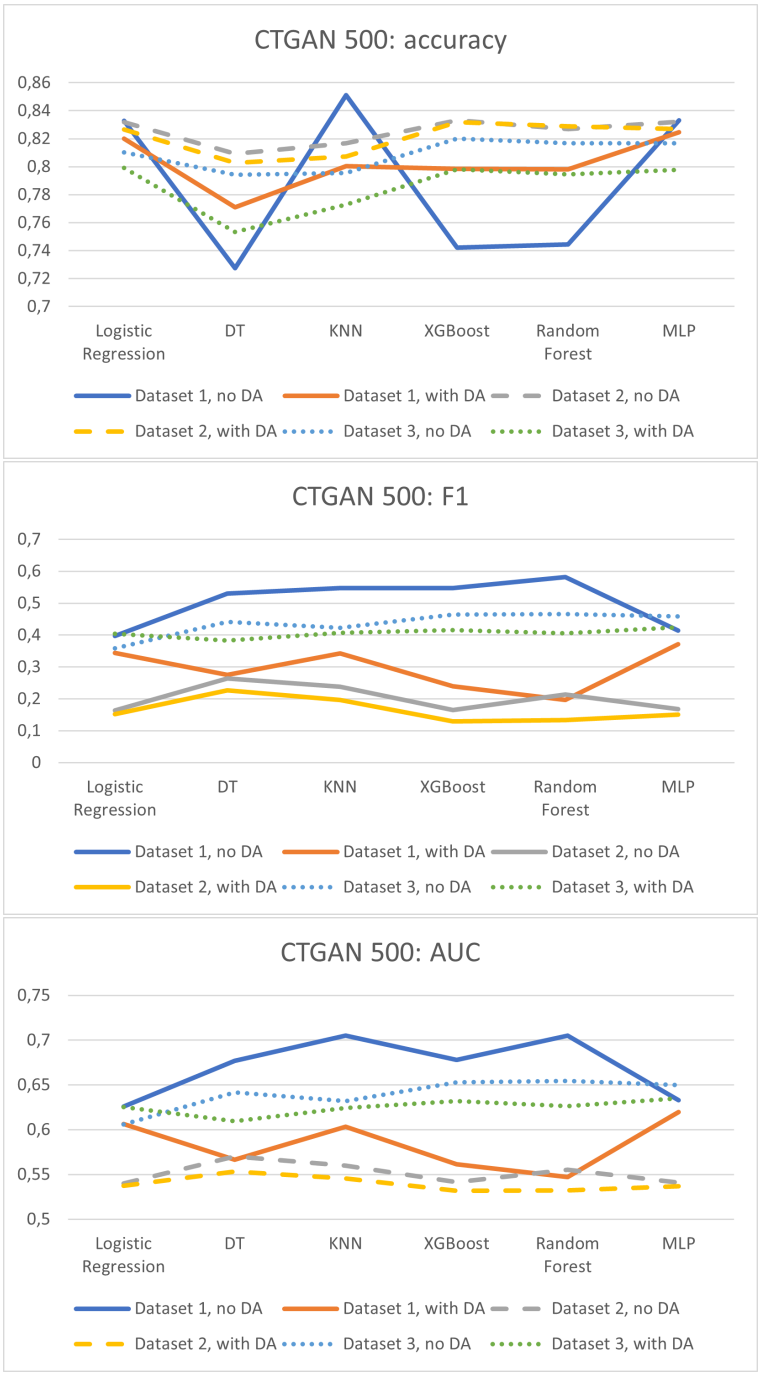

Figure A.2: The accuracy, F1 and AUC given for all models and datasets before and after applying CTGAN with 500 epochs.

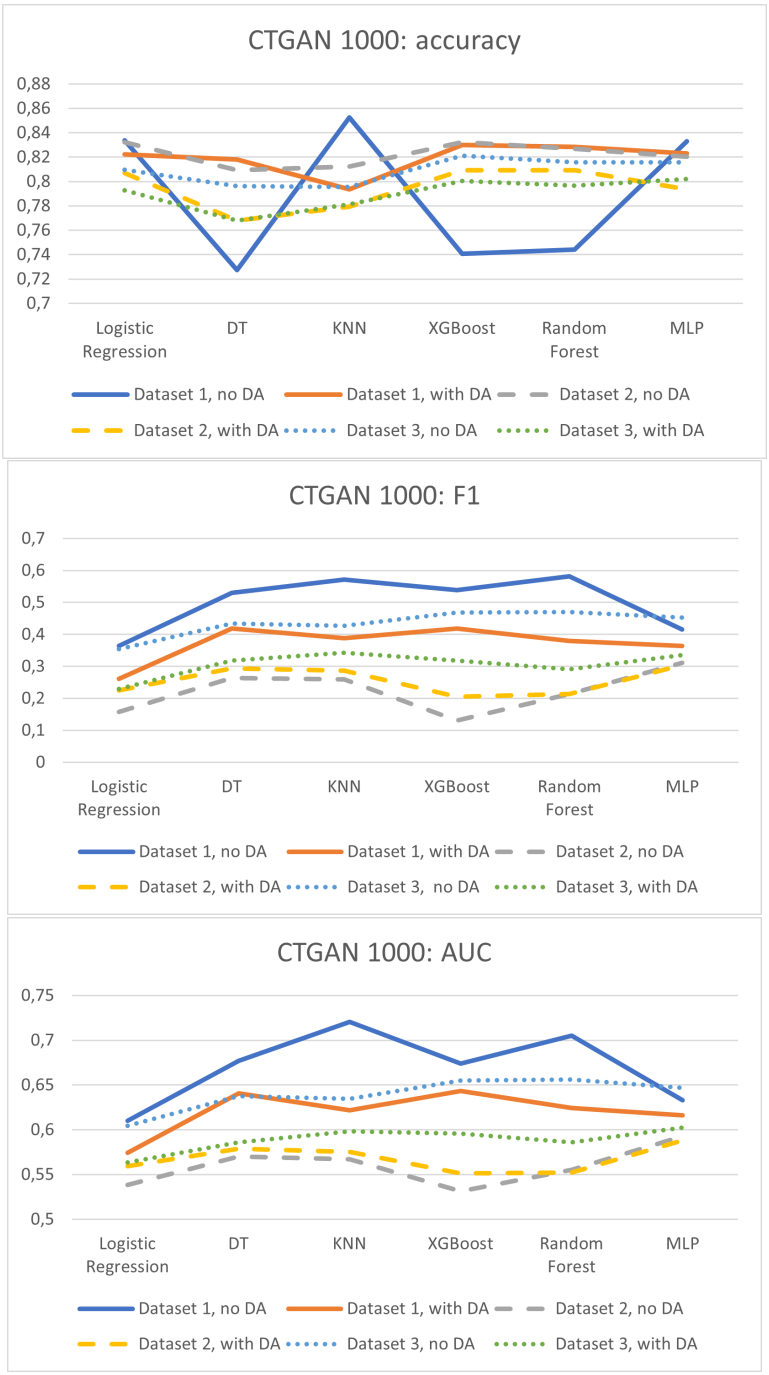

Figure A.3: The accuracy, F1 and AUC given for all models and datasets before and after applying CTGAN with 1000 epochs.

A.1.3 Graphs PrivBayes

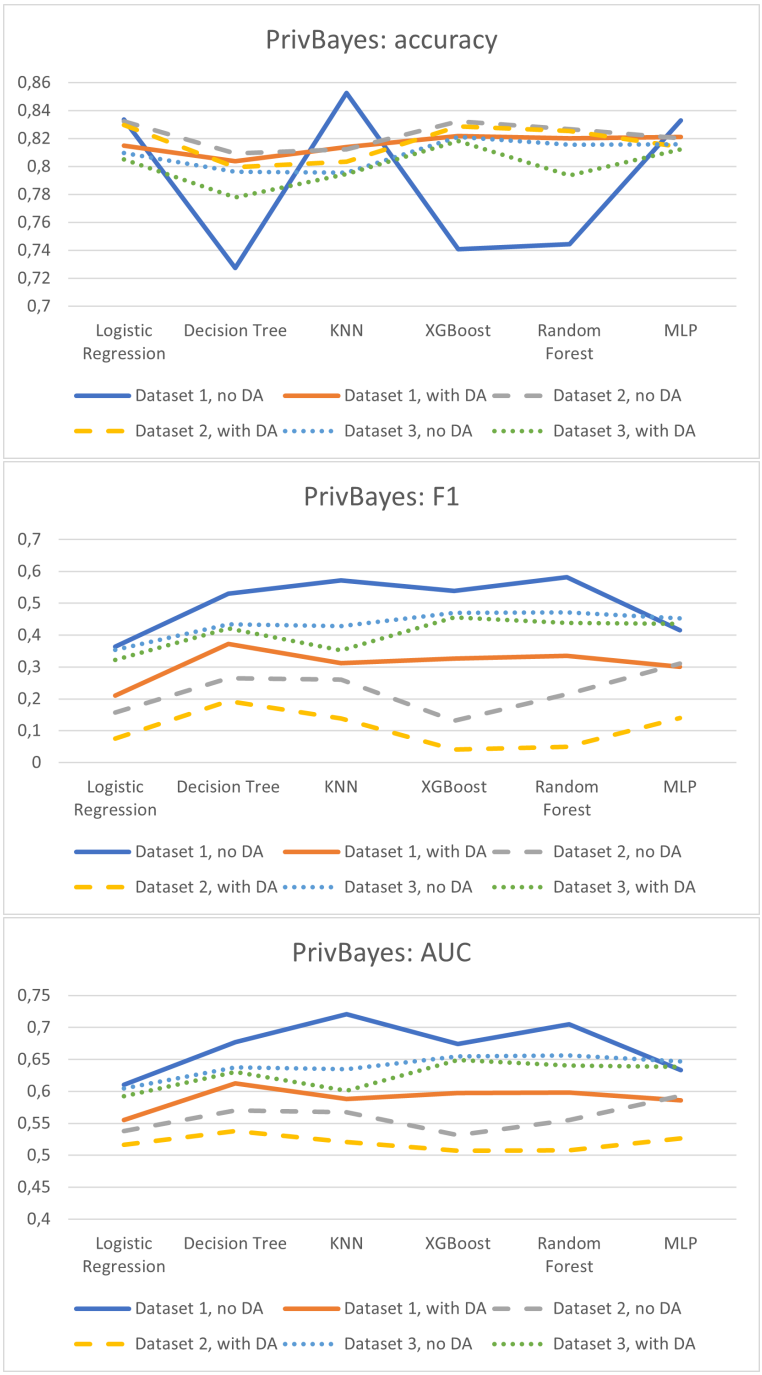

Figure A.4: The accuracy, F1 and AUC given for all models and datasets before and after applying PrivBayes.

Bibliography

Badu-Marfo, G., Farooq, B., & Paterson, Z. (2020). Composite travel generative adversarial networks for tabular and sequential population synthesis. arXiv:2004.06838 [cs, stat]. http://arxiv.org/abs/2004.06838

Ballet, V., Renard, X., Aigrain, J., Laugel, T., Frossard, P., & Detyniecki, M. (2019). Imperceptible adversarial attacks on tabular data. arXiv:1911.03274 [cs, stat]. http://arxiv.org/abs/1911.03274

Baowaly, M. K., Lin, C.-C., Liu, C.-L., & Chen, K.-T. (2019). Synthesizing electronic health records using improved generative adversarial networks. Journal of the American Medical Informatics Association, 26(3), 228–241. https://doi.org/10.1093/jamia/ocy142

Batista, G. E. A. P. A., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsletter, 6(1), 20–29. https://doi.org/10.1145/1007730.1007735

Bichler, M., & Kiss, C. (2004, december). A Comparison of Logistic Regression, k-Nearest Neighbor, and Decision Tree Induction for Campaign Management . AMCIS 2004 Proceedings. Americas Conference on Information Systems (AMCIS).

Boquet, G., Morell, A., Serrano, J., & Vicario, J. L. (2020). A variational autoencoder solution for road traffic forecasting systems: Missing data imputation, dimension reduction, model selection and anomaly detection. Transportation Research Part C: Emerging Technologies, 115, 102622. https://doi.org/10.1016/j.trc.2020.102622

Briscoe, E., & Feldman, J. (2010). Conceptual complexity and the bias/variance tradeoff. Cognition, 118(1), 2–16. https://doi.org/10.1016/j.cognition.2010.10.004

Carmon, Y., Raghunathan, A., Schmidt, L., Liang, P., & Duchi, J. C. (2022). Unlabeled data improves adversarial robustness. arXiv:1905.13736 [cs, stat]. http://arxiv.org/abs/1905.13736

Cartella, F., Anunciacao, O., Funabiki, Y., Yamaguchi, D., Akishita, T., & Elshocht, O. (2021). Adversarial attacks for tabular data: Application to fraud detection and imbalanced data. arXiv:2101.08030 [cs]. http://arxiv.org/abs/2101.08030

Charte, F., Rivera, A. J., del Jesus, M. J., & Herrera, F. (2015). MLSMOTE: Approaching imbalanced multilabel learning through synthetic instance generation. Knowledge-Based Systems, 89, 385–397. https://doi.org/10.1016/j.knosys.2015.07.019

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321–357. https://doi.org/10.1613/jair.953

Chawla, N. V., Lazarevic, A., Hall, L. O., & Bowyer, K. W. (2003). Smoteboost: Improving prediction of the minority class in boosting. In N. Lavrač, D. Gamberger, L. Todorovski, & H. Blockeel (Red.), Knowledge Discovery in Databases: PKDD 2003 (pp. 107–119). Springer. https://doi.org/10.1007/978-3-540-39804-212

Choi, E., Biswal, S., Malin, B., Duke, J., Stewart, W. F., & Sun, J. (2018). Generating multi-label discrete patient records using generative adversarial networks. arXiv:1703.06490 [cs]. http://arxiv.org/abs/1703.06490

Data augmentation in python: Everything you need to know. (2020, oktober 20). Neptune.Ai. https://neptune.ai/blog/data-augmentation-in-python

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., & Sutskever, I. (2020). Jukebox: A generative model for music. arXiv:2005.00341 [cs, eess, stat]. http://arxiv.org/abs/2005.00341

Domingo-Ferrer, J. (2009). Disclosure risk. In L. Liu & M. T. Özsu (Red.), Encyclopedia of Database Systems (pp. 848–849). Springer US. https://doi.org/10.1007/978-0-387-39940-91506

Evans, T., Zayatz, L., & Slanta, J. (1996). Using Noise for Disclosure Limitation of Establishment Tabular Data. Annual Research Conference and Technology Interchange.

Feng, S. Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., & Hovy, E. (2021). A survey of data augmentation approaches for nlp. arXiv:2105.03075 [cs]. http://arxiv.org/abs/2105.03075

Fernandez, A., Garcia, S., Herrera, F., & Chawla, N. V. (2018). Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. Journal of Artificial Intelligence Research, 61, 863–905. https://doi.org/10.1613/jair.1.11192

Flossmann, A., & Nolte (Lechner), S. (2006). Combining blanking and noise addition as a data disclosure limitation method (SSRN Scholarly Paper ID 929429). Social Science Research Network. https://papers.ssrn.com/abstract=929429

Friedman, J. H., Kohavi, R., & Yun, Y. (1996). Lazy Decision Trees. AAAI-96 Proceedings. AAAI.

Gogoshin, G., Branciamore, S., & Rodin, A. S. (2020). Synthetic data generation with probabilistic Bayesian Networks [Preprint]. Systems Biology. https://doi.org/10.1101/2020.06.14.151084

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks. arXiv:1406.2661 [cs, stat]. http://arxiv.org/abs/1406.2661

Google Developers. (2020) Common problems — generative adversarial networks. Retrieved 12 november 2021, van https://developers.google.com/machine-learning/gan/problems

Google Developers. (2021) Background: What is a generative model? — generative adversarial networks. Retrieved 4 april 2022, van https://developers.google.com/machine-learning/gan/generative

Greene, W. H. (2003). Econometric analysis (5th ed). Prentice Hall.

Gulrajani, I., Kumar, K., Ahmed, F., Taiga, A. A., Visin, F., Vazquez, D., & Courville, A. (2016). Pixelvae: A latent variable model for natural images. arXiv:1611.05013 [cs]. http://arxiv.org/abs/1611.05013

Hashemi, M., & Fathi, A. (2020). Permuteattack: Counterfactual explanation of machine learning credit scorecards. arXiv:2008.10138 [cs, stat]. http://arxiv.org/abs/2008.10138

He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 1322–1328. https://doi.org/10.1109/IJCNN.2008.4633969

Hossin, M., & Sulaiman, M. N. (2015). A review on evaluation metrics for data classification evaluations. International Journal of Data Mining & Knowledge Management Process, 5(2), 01–11. https://doi.org/10.5121/ijdkp.2015.5201

Huang, D., Song, X., Fan, Z., Jiang, R., Shibasaki, R., Zhang, Y., Wang, H., & Kato, Y. (2019). A variational autoencoder based generative model of urban human mobility. 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), 425–430. https://doi.org/10.1109/MIPR.2019.00086

Islam, Z., Abdel-Aty, M., Cai, Q., & Yuan, J. (2021). Crash data augmentation using variational autoencoder. Accident Analysis & Prevention, 151, 105950. https://doi.org/10.1016/j.aap.2020.105950

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. arXiv:1312.6114 [cs, stat]. http://arxiv.org/abs/1312.6114

Kulyukin, V., Mukherjee, S., & Amlathe, P. (2018). Toward audio beehive monitoring: Deep learning vs. Standard machine learning in classifying beehive audio samples. Applied Sciences, 8(9), 1573. https://doi.org/10.3390/app8091573

Lee, C. (2021). Data augmentation using a variational autoencoder for estimating property prices. Property Management, 39(3), 408–418. https://doi.org/10.1108/PM-09-2020-0057

Lee, J.-H., Zaheer, M. Z., Astrid, M., & Lee, S.-I. (2020). Smoothmix: A simple yet effective data augmentation to train robust classifiers. 756–757.

Li, F., Huang, W., Luo, M., Zhang, P., & Zha, Y. (2021). A new VAE-GAN model to synthesize arterial spin labeling images from structural MRI. https://doi.org/https://doi.org/10.1016/j.displa.2021.102079

Li, S.-C., Tai, B.-C., & Huang, Y. (2019). Evaluating variational autoencoder as a private data release mechanism for tabular data. 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC), 198–1988. https://doi.org/10.1109/PRDC47002.2019.00050

Liang, X. W., Jiang, A. P., Li, T., Xue, Y. Y., & Wang, G. T. (2020). LR-SMOTE — An improved unbalanced data set oversampling based on K-means and SVM. Knowledge-Based Systems, 196, 105845. https://doi.org/10.1016/j.knosys.2020.105845

Lim, S. K., Loo, Y., Tran, N.-T., Cheung, N.-M., Roig, G., & Elovici, Y. (2018). Doping: Generative data augmentation for unsupervised anomaly detection with gan. arXiv:1808.07632 [cs, stat]. http://arxiv.org/abs/1808.07632

Lu, Y., & Xu, P. (2018). Anomaly detection for skin disease images using variational autoencoder. arXiv:1807.01349 [cs, stat]. http://arxiv.org/abs/1807.01349

Marais, J. A. (2019). Deep Learning for Tabular Data: An Exploratory Study. Stellenbosch University.

Massel, P., Zayatz, L., & Funk, J. (2006). Protecting the Confidentiality of Survey Tabular Data by Adding Noise to the Underlying Microdata: Application to the Commodity Flow Survey*. In Privacy in Statistical Databases (pp. 304–317).

Miot, A. (2021). Adversarial trading. arXiv:2101.03128 [cs, q-fin]. http://arxiv.org/abs/2101.03128

Muralidhar, K., Sarathy, R., & Dandekar, R. (2006). Why swap when you can shuffle? A comparison of the proximity swap and data shuffle for numeric data. In J. Domingo-Ferrer & L. Franconi (Red.), Privacy in Statistical Databases (pp. 164–176). Springer. https://doi.org/10.1007/1193024215

Overfitting in machine learning: What it is and how to prevent it. (2017, september 7). EliteDataScience. https://elitedatascience.com/overfitting-in-machine-learning

Park, N., Mohammadi, M., Gorde, K., Jajodia, S., Park, H., & Kim, Y. (2018). Data synthesis based on generative adversarial networks. Proceedings of the VLDB Endowment, 11(10), 1071–1083. https://doi.org/10.14778/3231751.3231757

Ping, H., Stoyanovich, J., & Howe, B. (2017). Datasynthesizer: Privacy-preserving synthetic datasets. Proceedings of the 29th International Conference on Scientific and Statistical Database Management, 1–5. https://doi.org/10.1145/3085504.3091117

Piri, S., Delen, D., & Liu, T. (2018). A synthetic informative minority over-sampling (Simo) algorithm leveraging support vector machine to enhance learning from imbalanced datasets. Decision Support Systems, 106, 15–29. https://doi.org/10.1016/j.dss.2017.11.006

Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., & Carin, L. (2016). Variational autoencoder for deep learning of images, labels and captions. arXiv:1609.08976 [cs, stat]. http://arxiv.org/abs/1609.08976

Raghuwanshi, B. S., & Shukla, S. (2020). SMOTE based class-specific extreme learning machine for imbalanced learning. Knowledge-Based Systems, 187, 104814. https://doi.org/10.1016/j.knosys.2019.06.022

Razavi, A., van den Oord, A., & Vinyals, O. (2019). Generating diverse high-fidelity images with vq-vae-2. arXiv:1906.00446 [cs, stat]. http://arxiv.org/abs/1906.00446

Rebuffi, S.-A., Gowal, S., Calian, D. A., Stimberg, F., Wiles, O., & Mann, T. (2021). Data augmentation can improve robustness. arXiv:2111.05328 [cs, stat]. http://arxiv.org/abs/2111.05328

Rivera, W. A. (2017). Noise reduction a priori synthetic over-sampling for class imbalanced data sets. Information Sciences, 408, 146–161. https://doi.org/10.1016/j.ins.2017.04.046

Sestino, A., & De Mauro, A. (2021). Leveraging artificial intelligence in business: Implications, applications and methods. Technology Analysis & Strategic Management, 1–14. https://doi.org/10.1080/09537325.2021.1883583

Shah, K., Patel, H., Sanghvi, D., & Shah, M. (2020). A comparative analysis of logistic regression, random forest and knn models for the text classification. Augmented Human Research, 5(1), 12. https://doi.org/10.1007/s41133-020-00032-0

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of Big Data, 6(1), 60. https://doi.org/10.1186/s40537-019-0197-0

Shorten, C., Khoshgoftaar, T. M., & Furht, B. (2021). Text data augmentation for deep learning. Journal of Big Data, 8(1), 101. https://doi.org/10.1186/s40537-021-00492-0

Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U., & Sutton, C. (2017). Veegan: Reducing mode collapse in gans using implicit variational learning. arXiv:1705.07761 [stat]. http://arxiv.org/abs/1705.07761

Stephenson, T. A. (Red.). (2000). An introduction to bayesian network theory and usage. IDIAP.

Sun, L., & Erath, A. (2015). A Bayesian network approach for population synthesis. Transportation Research Part C: Emerging Technologies, 61, 49–62. https://doi.org/10.1016/j.trc.2015.10.010

Tanner, M. A., & Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. Journal of the American Statistical Association, 82(398), 528–540. https://doi.org/10.1080/01621459.1987.10478458

Thanh-Tung, H., & Tran, T. (2020). On catastrophic forgetting and mode collapse in generative adversarial networks. arXiv:1807.04015 [cs, stat]. http://arxiv.org/abs/1807.04015

Ucar, T., Hajiramezanali, E., & Edwards, L. (2021). Subtab: Subsetting features of tabular data for self-supervised representation learning. arXiv:2110.04361 [cs, stat]. http://arxiv.org/abs/2110.04361

van der Maaten, L., Chen, M., Tyree, S., & Weinberger, K. (2014). Marginalizing corrupted features. arXiv:1402.7001 [cs]. http://arxiv.org/abs/1402.7001

van Dyk, D. A., & Meng, X.-L. (2001). The art of data augmentation. Journal of Computational and Graphical Statistics, 10(1), 1–50. https://www.jstor.org/stable/1391021

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. Proceedings of the 25th International Conference on Machine Learning - ICML '08, 1096–1103. https://doi.org/10.1145/1390156.1390294

Weiss, G. M. (2004). Mining with rarity: A unifying framework. ACM SIGKDD Explorations Newsletter, 6(1), 7–19. https://doi.org/10.1145/1007730.1007734

Wu, Z., Wang, S., Qian, Y., & Yu, K. (2019). Data augmentation using variational autoencoder for embedding based speaker verification. Interspeech 2019, 1163–1167. https://doi.org/10.21437/Interspeech.2019-2248

Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., & Le, Q. V. (2020). Unsupervised data augmentation for consistency training. arXiv:1904.12848 [cs, stat]. http://arxiv.org/abs/1904.12848

Xu, L. (2020). Synthesizing tabular data using conditional GAN [Thesis, Massachusetts Institute of Technology]. https://dspace.mit.edu/handle/1721.1/128349

Xu, L., & Veeramachaneni, K. (2018). Synthesizing tabular data using generative adversarial networks. arXiv:1811.11264 [cs, stat]. http://arxiv.org/abs/1811.11264

Xu, W., Sun, H., Deng, C., & Tan, Y. (2016). Variational autoencoders for semi-supervised text classification. arXiv:1603.02514 [cs]. http://arxiv.org/abs/1603.02514

Ying, X. (2019). An overview of overfitting and its solutions. Journal of Physics: Conference Series, 1168, 022022. https://doi.org/10.1088/1742-6596/1168/2/022022

Zayatz, L. (2007). New Implementations of Noise for Tabular Magnitude Data, Synthetic Tabular Frequency and Microdata, and a Remote Microdata Analysis System (Statistics 2007-17; RESEARCH REPORT SERIES). U.S. Census Bureau.

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2018). Mixup: Beyond empirical risk minimization. arXiv:1710.09412 [cs, stat]. http://arxiv.org/abs/1710.09412

Zhang, J., Cormode, G., Procopiuc, C. M., Srivastava, D., & Xiao, X. (2017). Privbayes: Private data release via bayesian networks. ACM Transactions on Database Systems, 42(4), 1–41. https://doi.org/10.1145/3134428

Zhong, Z., Zheng, L., Kang, G., Li, S., & Yang, Y. (2020). Random erasing data augmentation. Proceedings of the AAAI Conference on Artificial Intelligence, 34(07), 13001–13008. https://doi.org/10.1609/aaai.v34i07.7000

FACULTY OF BUSINESS AND ECONOMICS
Naamsestraat 69 bus 3500
3000 LEUVEN, BELGIË
tel. + 32 16 32 66 12
fax + 32 16 32 67 91
info@econ.kuleuven.be
www.econ.kuleuven.be

31010