Auditing Algorithmic Fairness with Unsupervised Bias Discovery

Selma Muhammad

City of Amsterdam

VU VRIJE UNIVERSITEIT AMSTERDAM

Auditing Algorithmic Fairness with Unsupervised Bias Discovery

A thesis submitted to the Vrije Universiteit in partial fulfillment of the requirements for the degree of

Master of Science in Artificial Intelligence

by

Selma Muhammad

2578081

To be defended in public on the 18th of August 2021

Supervisors

Academic Supervisor	dr. Emma Beauxis-Aussalet	Ethical Computing
Internship Supervisor	Linda van de Fliert	Public Tech @ City of Amsterdam
Second Reader	dr. Michael Cochez	Knowledge Representation and Reasoning

Audiatur et altera pars

May the other side also be heard

Medea,
Seneca

Preface

Although writing a master's thesis is mainly a solitary task, I wouldn't be here today without the help of my brilliant circle of friends, family, colleagues and supervisors.

First, I would like to thank Emma and Linda. Emma, thank you for introducing me to this interesting topic. You've made me so enthusiastic about fairness in AI and you were always so helpful, understanding and kind. Your curiosity was very contagious and you made me overcome obstacles I never thought I could overcome. Thank you for guiding me so well these past months.
Linda, I'm so thankful for all your help, advice and for always giving me some clarity whenever I felt lost. You've done everything in your power to make my internship at the City of Amsterdam a pleasant and memorable experience, in which you absolutely succeeded. I had such a great time! To Daniël, Petra and Iva, thank you for bringing me joy and support during the past months. I appreciate you immensely!

When zooming out a bit on the past months, I see the faces of my beloved friends and family. Hannah, Lynn, Kimberley, Shiemaa, Marit and Simone: thank you for being the best friends I could wish for. You guys are wonderful.
And to my brothers Ammar and Ameer: you've paved the road for me ever since I was a toddler crawling after you and demanding Donald Ducks and candy. You showed me what hard work and perseverance meant by your acts, which has inspired me throughout my whole life.

To my fiancé Amir, thank you for making the mundane moments beautiful and the beautiful moments even more special. You've been such a great and loving support these past months. From our long calls, evening walks, eating *zure matten*, shopping for our lovely future home, to proofreading my thesis: I've enjoyed every single minute of it. The future looks so exciting with you by my side.

Lastly and most importantly, I would like to thank *mama* and *baba*, for the fruit platters quietly slided to me during meetings and for the gentle motivational speeches during our dinners. For your contagious love for education and for your ability to always, always believe in me. I'm so lucky that you guys are not only my parents, but also my best friends. Believe me, I have many words, but I couldn't even come close to describing how much I love you. Should I achieve successes in my life, I hope you know that all of that is due to your love and support.

And to you, the reader of this thesis: thank you for taking out time to read this work. I hope you enjoy it.

Selma Muhammad
Lisserbroek, August 6th, 2021

Abstract

The presence of undesired bias in algorithms can lead to systematic discrimination against individuals and marginalized groups. While many fairness methods have been developed to mitigate this bias, they often require that the protected groups are specified beforehand. In this study, we introduced the Hierarchical Bias-Aware Clustering (HBAC) algorithm to locate clusters of bias from the results of classification algorithms. The advantages of HBAC are that it is model-agnostic and does not require any *a priori* knowledge about the dataset. We experimented with three different clustering algorithms in HBAC to find the model with the best performance in finding a high discrimination bias. Additionally, we included the errors as a clustering attribute and experimented with several error scaling factors. The evaluation of the HBAC algorithms consisted of visual and statistical analysis, after which we conducted an indirect evaluation of each clustering algorithm. The results indicate that HBAC-KMeans found the highest discrimination bias in real-world datasets, while HBAC-DBSCAN performed best on synthetic data. During the indirect evaluation, we found that HBAC-KMeans satisfied most of the desired properties, such as scalability and meaningfulness of the found clusters. From the error scaling experiment, we established that a scaling factor between 0.5 and 1 found the highest discrimination bias for HBAC-KMeans and HBAC-DBSCAN. Based on the results, we can conclude that HBAC shows promising results for discovering bias and describing the discriminated clusters in classification algorithms, but future work is required to deploy this fairness assessment tool on a larger scale.

Contents

Chapter 1

Introduction

Humans are biased, and make errors. Humans also build Artificial Intelligence (AI) technologies in which human errors and bias can further propagate, be it in the data that is collected, judgment errors, or in misinterpreting the problem and context for which the technology was intended to solve. In many cases, these errors have harmful effects on individuals, subpopulations and the entire society. Although much research has been conducted to eliminate errors in AI systems, there exists no technology or model that is entirely error proof: regardless, these computer systems and machines remain imperfect. However, it is possible to use these errors as a *means* to gain useful insights about how a model works, or to examine which individuals or subpopulations are particularly vulnerable to the harmful effects of a system. Systematic *error patterns* are a strong indicator of the presence of a harmful bias in an AI system. When a model systematically produces more errors for certain individuals or groups of persons, we can speak of *discrimination* (Buolamwini, 2018).

Such discrimination can be caused by multiple factors, which are often related to the data that is fed to the algorithm. For example, the data can be unrepresentative of reality, or it can contain existing prejudices or sensitive information (e.g gender or ethnicity), which causes the technology to wrongly base its recommendations, predictions and other output on (Buolamwini, 2018; Agrawal, 2019). Additionally, AI systems can be intersectionally biased when they are trained on data that contains sensitive information about individuals who belong to *multiple* protected groups, such as having a dark skin tone and being a female. This intersectionality amplifies the systematical discrimination that these vulnerable individuals experience from AI systems, as the individuals now are disadvantaged in multiple ways by the same system. Thus, to address this intersectional bias that causes many vulnerable individuals to be systematically discriminated against, more research is needed to develop fairness assessment methods that target this issue of intersectionality. We contribute to such research endeavours by experimenting with bias detection methods that consider all potential attributes may define a pattern of error, rather than a single protected attribute.

In the following sections, we will first explore the concepts of AI ethics, bias and fairness in section 1.1. In section 1.2, we discuss the problem of algorithmic fairness within the Public Domain. Finally, in sections 1.3 and 1.4, we provide a detailed account of the problem and the corresponding research questions that we investigate in this study.

1.1 AI Ethics and Fairness

Due to the proliferation of AI technologies, many organizations, governments, and industries have slowly recognised the need for ethical AI to ensure that the deployed AI-based systems are human-oriented, rights-respecting and socially beneficial. Consequentially, these institutions have endorsed a wide range of ethical principles, guidelines and governance frameworks to steer the development and deployment of AI technologies in the right direction. The Berkman Klein Center (2020) published a report in which these ethical principles across the world were summarised into the following eight themes: 1) privacy, 2) accountability, 3) safety and security, 4) transparency and explainability, 5) fairness and non-discrimination, 6) human control and agency, 7) professional responsibility and 8) promotion of human values. However, although defining these guidelines and principles provides organisations with clearer insights about the desired properties of the AI technologies, the significant gap between the theory of AI ethics and the practical design of AI systems is becoming increasingly evident (Morley et al., 2018; Kuziemski & Misuraca, 2020). This gap can be explained by the abundance of either too flexible tools, leading to *ethics washing*, or too strict methods, which are often non-adaptive to dynamic contexts. Consequentially, individuals, groups and the society as a whole are insufficiently protected by the proposed AI governance mechanisms (Morley et al., 2018).

For this reason, there is an increasing need for practical instruments that enable AI practitioners to better understand, evaluate and mitigate the potential harms of AI systems. Fjeld et al. (2020) proposed that tools to increase the transparency of AI systems could be a starting point for evaluating AI models, as individuals can then examine which criteria, objectives and logic the models used to arrive at their decisions. In effect, individuals could then determine whether this decision was made based on biased and/or discriminatory grounds. However, while transparency enables developers to better understand their models, transparency does not offer a method to assess whether the data, criteria and objectives contain *undesired biases*, while these can cause detrimental effects to individuals, groups and society.

When an algorithm contains undesired bias, its predictions, recommendations or classifications are skewed towards certain individuals or groups, hereby leading to an unequal treatment of the persons and groups impacted by the AI system. In many cases, this unequal treatment holds that the AI technology is systematically discriminating against an individual or groups of persons while favouring other persons and groups, for example by systematically producing more errors for a certain group when compared to other groups (Silberg & Manyika, 2019; Veale & Binns, 2017). Though each AI system needs bias to distinguish data points are from others based on predictive characteristics in order to generate the predictions, this bias becomes harmful when the model's output are based on *protected* or *sensitive attributes*, such as gender, ethnicity, sexual orientation, age and disability. These traits or characteristics involve sensitive information on which it is often legally prohibited to discriminate on (Veale & Binns, 2017).

The detection, evaluation and mitigation of undesired bias falls under the principle of *fairness and non-discrimination*, which articulates that *"bias in AI- in the training data, technical design choices, or the technology's deployment, should be mitigated to prevent discriminatory impacts*" (Fjeld et al., 2020). Understanding how AI technologies potentially carry unfair bias will be increasingly crucial in the future, as the models have the potential to significantly scale discrimination and to discriminate in unforeseen ways, hereby affecting individuals, corporations, organizations and economies worldwide. Additionally, when

algorithms are fair, they are likely to reduce the effect of prevailing *human biases*, such as gender and age biases. For example, a study found that an automated financial underwriting system was particularly benefiting historically underserved applicants, hereby diminishing the racial bias (Gates, Perry, & Zorn, 2002; Silberg & Manyika, 2019).

Since systematic discrimination by an AI technology is particularly harmful when this system is widely used for matters that affect people's daily lives, it is essential that the AI systems in the public domain are thoroughly evaluated for presence of undesired bias. According to the Universal Declaration of Human Rights, *"everyone has the right to equal access to public service in his country"* (The United Nations, 1948), which implies that AI systems used by Public Services must ensure that their technologies treat all citizens equally, regardless of their ethnicity, age, gender or other personal characteristics. For this reason, the scope of this research is narrowed down to AI technologies used by Public Services.

1.2 AI in the Public Domain

The use of Artificial Intelligence systems has increased substantially in the Public Domain, which involves governments, municipalities, housing amenities and voluntary organizations (Elsevier, 2018). AI technologies are embraced by both corporations and the public sector due to their efficiency, cost-effectiveness and performance when compared to manual labour, which in turn can increase the confidence and satisfaction of citizens with their services (de Sousa, de Melo, Bermejo, Farias, & Gomes, 2019). The deployed AI systems affect various aspects of life, such as in healthcare (e.g., predicting the onset of an illness or recommending suitable treatment for a disease), social services (e.g., for benefit allocation at job loss or retirement) or infrastructure (e.g, for predictive maintenance).

In order for all citizens to gain access to the benefits of AI technologies, these systems should be designed to facilitate this broad access. This broad access implies that AI models produce fair decisions, without placing disadvantaged people at even more unfavorable positions (Fjeld et al., 2020). In recent years, many problems have been identified with AI models deployed by public services, often with drastically negative consequences, of which the Childcare Allowance Affair and COMPAS are illuminating examples.

The Childcare Allowance Affair

In December 2020, the Parliamentary Interrogation Committee on Childcare Allowance presented an extensive report on the Child Allowance Affair to the Dutch cabinet (Van Aalst et al., 2021). The report explained into detail how over ten thousand families were wrongly accused of committing child welfare fraud, resulting in significant financial debts, marital problems, mental illnesses, and many children growing up in poverty. A substantial cause of this Affair was the use of a discriminatory algorithm, whose objective it was to assign risk scores to childcare benefit applicants about the likelihood of them committing fraud to receive the allowances. It turned out that these models used discriminatory demographic attributes, such as the presence of double nationalities, to assign a risk profile to these parents (Autoriteit Persoonsgegevens, 2018). Families with double nationalities were then extensively researched in other datasets, which led to more accusations of fraud, as putting individuals or groups under a magnifying glass often results in finding more "suspicious" activities that may indicate fraud. The Authoriteit Persoonsgegevens claimed that the Belastingdienst (Tax Authorities) was profiling citizens

by using their nationality as indicator to support decision-making processes around allocating social allowances. Under the European data protection legislation, nationality is considered to be a special category of personal data which cannot be processed unless there is an appropriate lawful basis provided, as nationality can indirectly reveal information about a person's racial or ethnical background which are protected attributes (Autoriteit Persoonsgegevens, 2018). According to the Authoriteit Persoonsgegevens, this lawful basis for using nationality as an indicator was not sufficient.

Overall, the deployed fraud classification algorithm did not perform well on several occasions:

- Sensitive feature(s) were used, such as nationality, which can indirectly reveal information about protected attributes;
- There was a lack of human oversight on the decisions produced by the algorithm;
- There was lack of understanding the consequences of using the algorithm;
- There was a lack of insight in the objective and trade-offs of the algorithm;
- Little attention was paid to the possible biases that the algorithm can create or amplify;
- The employees of the Belastingdienst did not know which indicators were used to assign a risk score to the applicants of child allowance.

From the Child Allowance Affair, it became evident that the discriminatory algorithm was the result of many processes that were executed incorrectly. Developers, civil servants, and others involved were ignorant or neglectful of the potentially discriminatory factors in the collected data and the model. When the model became deployed to classify parents as potential 'frauds', the involved civil servants and other employees did not have a clear understanding of how the model worked, nor did they evaluate the model's uncertainty when assigning the fraud risk profiles to families (Geiger, 2021). Additionally, the incriminated families were not given access to information about how the model arrived at its decision. In short, the lack of transparency, interpretability, and fairness in the algorithm largely contributed to the harmful consequences that were observed in the Childcare Allowance Affair.

Consequentially, many governmental authorities are currently reconsidering their practices regarding their deployment of algorithms to avoid a new Affair from occurring. One of the initiatives was the *algorithm quality and ethics assessment framework* developed by the Dutch Court of Audit (Algemene Rekenkamer, 2021). However, as we discussed in section 1.1, many of these assessment frameworks lack practical instruments that can be used to develop a non-discriminatory algorithm. The bias discovery methods we investigate in this thesis can contribute to the collection of practical instruments, as they have the potential to detect such systematic patterns of error and bias before the systems are deployed on a wide scale.

COMPAS

In 2016, ProPublica published a report on a recidivism-risk algorithm named *Correctional Offender Management Profiling for Alternative Sanctions* (COMPAS) which was widely used in criminal justice courtrooms of the United States to predict which of the defendants were likely to commit another crime. The journalists discovered that the COMPAS algorithm assigned almost twice as much higher risk scores to black defendants when compared to white defendants with similar backgrounds (Angwin, Larson, Mattu, & Kirchner, 2016).

The analysis conducted by ProPublica revealed how the racial disparities were amplified by the deployment of COMPAS and how the algorithm led to an increase of the already high incarceration rate of black people in the United States. It was later discovered that it had not been investigated whether race affected COMPAS's risk scores, as the main focus had primarily been on ensuring that the dataset and the risk scores were representative of the New York population.

The Child Allowance Affair and COMPAS are just a few of many examples where an algorithm disadvantaged vulnerable individuals and groups, hereby perpetuating the systematical discrimination in a society. The Public Sector faces the challenge of properly evaluating high-impact algorithms for undesirable bias. This challenge occurs at all levels of the Public Domain: from the government to educational institutions to municipalities. In this thesis, we will investigate these challenges for the will investigate these challenges for the municipality of Amsterdam, which holds an unique position within the Netherlands as the largest municipality of the nation. Moreover, the municipality of Amsterdam often sets an example for smaller and foreign municipalities in terms of technological innovation.

Ethical AI at the City of Amsterdam

The main task of the Dutch municipalities is to carry out tasks that directly affect citizens, such as maintaining and cleaning the streets and pavements, paying social benefits, collecting household waste and issuing passports or permits (Government of the Netherlands, 2018). Many of these responsibilities are increasingly involving algorithms that improve the speed and quality of these complex processes. Although many algorithms are still considered to be "simple" calculation models, there is an increasing demand for more algorithms that support civil servants in their tasks, such as in supporting the decision-making process in allocating benefits, detecting unemployment fraud and assisting in urban planning (*Algorithmic accountability for the public sector | Ada Lovelace Institute*, n.d.). For many of these tasks, more complex algorithms are required, but these often come at the price of an increased opacity. This increasing demand for algorithms that can solve complicated tasks can affect the lives of thousands to millions of citizens. For this reason, we must ensure that the deployed algorithms perform well, produce understandable and traceable output and do not harm individuals or marginalised groups.

At the City of Amsterdam, AI systems are used for a wide range of tasks, such as for detecting waste, calculating the worth of a house and assigning risk scores to persons who rent out their homes to tourists. To ensure that algorithms are not biased, some qualitative assessment instruments have been developed, such as audits and impact assessments, and, in the case of Amsterdam, an Algorithm Register (Gemeente Amsterdam, 2021), but few evidence-based technical, quantitative instruments are available to the Municipality. This thesis aims to provide such an instrument to the Municipality.

1.3 Problem statement

In the abundance of guidelines and assessment frameworks, the Public Domain is in need of practical fairness instruments that can be widely used to discover and mitigate undesired bias. The Childcare Allowance Affair and COMPAS have illustrated the profound and harmful consequences of deploying a discriminatory algorithm on vulnerable individuals and groups. Despite the audits and ethical principles that have found their

way in the development of AI technologies, it remains difficult to find more hidden forms of bias, which are for example caused by a combination of attributes that reveal sensitive information or by proxy variables. Additionally, we observed in sections 1.1 and 1.2 that these algorithms are deployed in many domains and for a wide range of purposes. Therefore, this practical fairness assessment instrument should be applicable to a broad range of algorithms.

Thus, in this thesis, we contribute to filling this gap by proposing a model-agnostic unsupervised bias discovery algorithm that uses patterns of errors to find and describe undesired bias in classification algorithms. A model-agnostic algorithm is independent of the underlying machine learning model, which makes this algorithm more flexible and adaptive to a wide range of models. With this technical fairness assessment tool, developers and policy makers of the municipality of Amsterdam or other governmental departments can:

- Identify which groups are potentially discriminated against by the classification algorithm;
- Describe the potentially discriminated groups through visualisation techniques and statistical tests;
- Use the information about the discriminated groups to investigate how these persons are affected by the algorithm's decisions;
- Become more transparent to citizens about how classification algorithms are tested for discriminatory effects and bias.

With our proposed method, we aim to discover groups of persons for which the classifier is discriminating against by producing more errors on this group when compared to others. However, the mitigation and prevention of undesired bias is beyond the scope of this study.

1.4 Research Questions

This study primarily focuses on the use of unsupervised methods to find bias, with the aim to generate important insights into the applicability of these methods to discover hidden or complex patterns. Unsupervised learning models require no *a priori* knowledge about the data that is used, such as the vulnerable groups or protected attributes. This allows us to identify unexpected bias caused by, for example, intersectionality or proxy variables.

Thus, we propose a clustering technique to detect bias in classification algorithms based on clustering the errors, i.e clusters with a higher classification error density, after which we attempt to describe the highly biased clusters. This novel technique is inspired on the work of Misztal-Radecka and Indurkhya (2021), who developed a bias-aware hierarchical clustering model to automatically detect discriminated groups of users from recommendation algorithms.

Therefore, in this thesis, we address the following research questions:

1. How can we use clustering algorithms to automatically detect clusters of error?
2. What kind of clustering algorithm is most suitable to automatically detect clusters of error?

3. How can the clusters of error be interpreted by public servants and other stakeholders and assessors?
4. How can such clustering algorithms for bias discovery complement the existing fairness assessment tools?

1.5 Thesis Outline

This thesis is divided into seven chapters. In chapter 2, we discuss the State of the Art methods to detect bias in algorithms. The chapter ends with a review the concept of fairness in the ethical AI domain. Chapter 3 provides an introduction on all the main concepts related to clustering. In chapter 4, we cover the extensive methodology of the Bias-Aware Clustering experiment. Finally, in chapter 5, we analyse the results, after which we provide answers to our research questions and reflect on the executed and future work in chapters 6 and 7.

Chapter 2

Related Work

This chapter discusses the current State of the Art methods to address algorithmic fairness by finding undesired bias in data and algorithms. First, a brief explanation is provided on classification errors in 2.1. Then, we proceed with an introduction to the field of algorithmic fairness in section 2.2. In sections 2.3, 2.4 and 2.5, we discuss methods of detecting bias during pre-processing, algorithm optimization and post-processing, respectively. Finally, in section 2.6, we dive deeper into the current challenges of discovering undesired bias, hereby introducing the Bias-Aware Clustering as a new fairness assessment instrument.

2.1 Classification Errors

The main task of classification algorithms is to predict a categorical value, such as predicting the presence of a tumour based on a the pixels of CT-scan or classifying the tone of a word given its textual context. The classifier can either yield binary predictions (*yes* or *no*), or it can classify into more than two classes, which is named *multiclass classification*. The dataset of a classifier contains both the independent variables (which are the attributes) and the dependent variable (which is the target that we want to predict).

There are various evaluation metrics available to assess the performance of classifiers. These evaluation metrics are based on the classifier's performance on the test dataset, which contains unseen instances that the classifier has not been trained on before. Generally, the confusion matrix contains key information about the errors that a model has produced, which is why it is often the first step in evaluating a model's performance. The confusion matrix comprises four classes which indicate the number of correctly and incorrectly classified instances:

- True Positives (TP): Instances correctly classified as Positive;
- False Positives (FP): Instances wrongly classified as Positive, while they are Negative;
- True Negatives (TN): Instances correctly classified as Negative;
- False Negatives (FN): Instances wrongly classified as Negatives, while they are Positive.

Figure 2.1 displays a confusion matrix of a binary classifier.

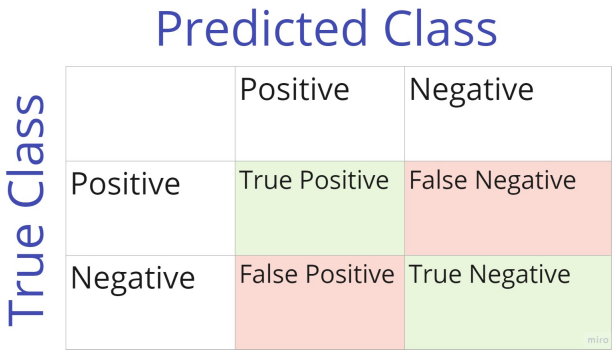


Figure 2.1: A Confusion Matrix

From the four classes of the confusion matrix, we can calculate the evaluation metrics Accuracy, Precision, Recall, F1-score, Specificity and AUC (*Area under the ROC Curve*), which are the most commonly used evaluation metrics for algorithms. The formulas of these metrics are provided in Appendix A.

Although this confusion matrix provides developers with insights about the model's performance and which type of errors (e.g False Positives or False Negatives), more metrics are required to discover whether the classifier systematically produces more errors for a group of persons: the general metrics often reveal limited information about these imbalances. Moreover, besides finding the biases, another research scope is to find the *cause* of these biases. The development of these anti-discrimination and fairness instruments falls under the scope of *algorithmic fairness*, our domain of interest which we will now further explore.

2.2 Algorithmic Fairness

Fairness has not only been engaging AI researchers, but it has a long history in philosophy and psychology as well, as defining what it means to be "fair" or preventing discrimination is a topic that concerns many domains. The concept of fairness relates to "*impartial and just treatment or behaviour without favouritism or discrimination*" (Oxford English Dictionary, 2012). Within the field of AI, algorithmic fairness is defined as "*the absence of any prejudice or favouritism toward an individual or a group based on their inherent or acquired characteristics.*" (Mehrabi, Morstatter, Saxena, Lerman, & Galstyan, 2019). These traits or characteristics involve sensitive information on which it is often legally prohibited to discriminate on, such as age, disabilities, gender, sexual orientation, ethnicity (Veale & Binns, 2017). Fairness issues can arise from various reasons, such as the indirect or direct presence of sensitive attributes that drive the algorithm's decision-making process (Barocas, Hardt, & Narayanan, 2019), dataset imbalances that under-represent vulnerable and marginalised groups (Buolamwini, 2018), and due to technical settings, such as wrongly set parameters or a faulty formalization of human constructs to "computer language" (Köchling & Wehner, 2020). These issues can occur at multiple stages during the development and deployment of an algorithm. In Figure 2.2, we have displayed the main domains in which undesired bias can occur.

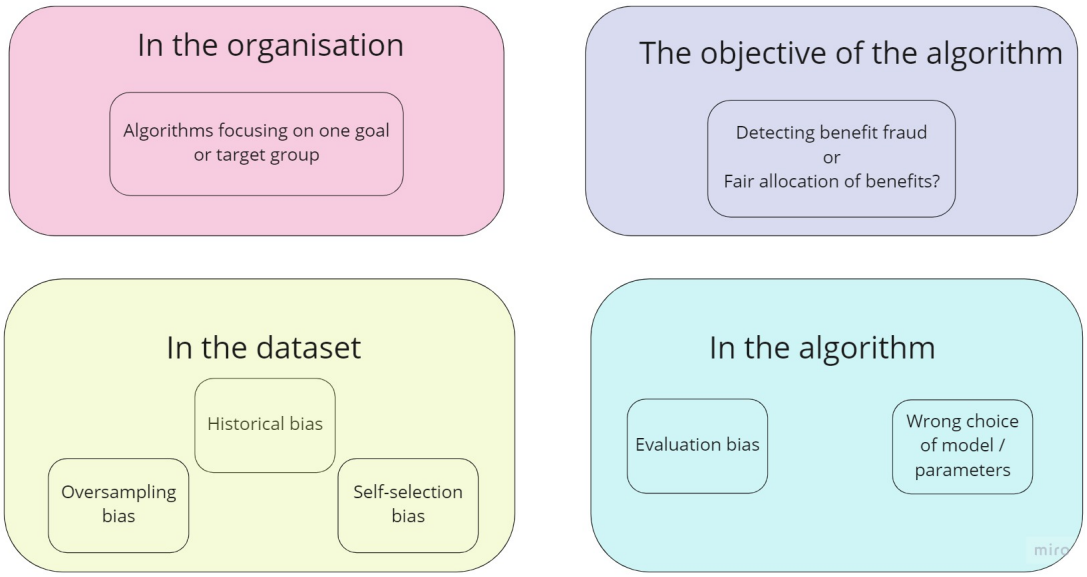


Figure 2.2: The sources of undesired bias. We used the example of the Child Allowance Affair.

Many of the current studies on fairness are presenting methods to detect, mitigate and prevent undesired bias in algorithms. The fairness instruments can be applied in the algorithm's lifecycle during dataset pre-processing, algorithm optimization, and during post-processing the results of an algorithm, which we will now turn to.

2.3 Pre-Processing

Preprocessing the data involves modifying the input data on which the algorithm is trained or evaluated such that the underlying discrimination is removed, which drives the algorithm to produce fair outcomes (Mehrabi et al., 2019; Friedler et al., 2019). A first step of implementing fairness measures can consist of removing sensitive features from the data, but Kamishima (2012) and Barocas et al. (2016) reported that removing direct sensitive attributes such as race or sexual orientation still leaves proxy variables that indirectly refer to the removed sensitive information. Moreover, removing sensitive attributes can result in a substantial accuracy loss (Barocas & Selbst, 2016). Better results are achieved by reweighing, resampling or transforming the training and the test data. In a study conducted by Dwork et al.(2012), the training data was transformed to minimize the dependencies between the sensitive attributes and the class/target labels. Other approaches involve changing the value of sensitive attributes and class labels of the instances in the dataset (Zafar, Valera, Rodriguez, & Gummadi, 2017). The main advantage of addressing fairness during pre-processing is that the bias in the data is then prevented from further propagating into the algorithm and its outcomes. However, as many models require slightly different input data to optimise their performance, pre-processing the dataset to minimise discrimination can lead to accuracy loss (Friedler et al., 2019).

2.4 Algorithm Optimization

Besides pre-processing, fairness measures can also be applied during algorithm optimization. This approach is often algorithm-dependent, as most optimization techniques apply fairness constraints on the parameters of the algorithms. Friedler et al. (2019) compared the effects of the fairness optimization techniques, which involved the *Prejudice Remover Regularizer* introduced by Kamishima et al. (2012) and the *Two Naive Bayes* approach by Calders et al. (2010). The Prejudice Remover Regularizer works by adding regularization terms on prediction algorithms with discriminative probabilistic models, such as logistic regression, to enforce the classifier's independence from sensitive attributes. However, the results showed that this method is sensitive to getting stuck in local minima, thereby not arriving at the global solution. The approach of Calders et al. (2010) comprises splitting the dataset into separate sets based on the sensitive attribute, where each set contains one particular value of a sensitive attribute. For instance, a dataset with the feature *gender* can be split into a dataset with only males and another with only females. Each model is trained separately, and the results are balanced afterwards. Although this method is intuitively appealing, it requires a large number of models when there are multiple sensitive attributes present in the dataset. Moreover, the *Two Naive Bayes* algorithm is difficult to apply on models with numerical values as sensitive attributes.

2.5 Post-Processing & Fairness Assessment

Finally, the fairness of algorithms can be detected and mitigated during the post-processing stage, which is after the model is applied to the test data and its performance is measured. The standard performance metrics, such as precision and recall, often reveal information about the general performance of the algorithm. But these metrics synthesize the overall performance, and disregard the model's results on specific individuals or marginalized groups. Fairness metrics are deployed to further investigate whether the model produced fair outcomes on the level of groups or individuals. However, although these metrics are often used during the post-processing stage, they can be applied to improve the entire algorithmic lifecycle to ensure that it does not contain undesired bias.

According to Mehrabi et al. (2019), these fairness metrics are often based on:

1. Statistical measures usually obtained from a confusion matrix;
2. The fairness and bias definitions that are adopted.
3. On which level fairness is measured: on individual, group or subgroup level. We will further explore fairness on the level of individuals and on group-level.

Individual Fairness

Individual fairness relates to assigning similar predictions to similar individuals. This approach on fairness requires a task-specific similarity metric which defines for each individual to what extent he/she is similar to another individual. This similarity metric is formalized as the *Lipschitz condition* on the classifier, which states that if the distance between two individuals is small, the classifier's predictions for these individuals should also be similar (Kim, Reingold, & Rothblum, 2018). The Lipschitz condition requires that for any two individuals x, y that are at distance $d(x, y) \in [0, 1]$, the statistical distance between $M(x)$ and $M(y)$ is at most $d(x, y)$. To evaluate whether an algorithm produces similar predictions for similar individuals, well-established evaluation metrics such as

Fairness through Awareness (Dwork et al., 2012) and *Counterfactual Fairness* (Kusner, Loftus, Russell, & Silva, 2017) are used. However, in this thesis, individual fairness lies beyond the scope of our interest.

Group Fairness

According to the group fairness definition, an algorithm is considered to produce fair predictions when both the advantaged group and the vulnerable, disadvantaged group are treated equally (Verma & Rubin, 2018). Various methods are available to assess whether the algorithm achieves group fairness. For group fairness, *Equality of Opportunity* (Hardt, Price, & Srebro, 2016), *Statistical Parity* (Zafar et al., 2017) and Predictive Rate Parity (Chouldechova, 2017) are well-established evaluation metrics.

The Equality of Opportunity definition, also named *Equalized Odds*, states that an algorithm is fair when a person originating from either the advantageous or the disadvantaged group has equal probability of being assigned a true positive prediction (Mehrabi et al., 2019). Thus, to satisfy this constraint, both groups are required to have equal true positive rates.

The Statistical Parity constraint, also named *Independence* or *Demographic Parity*, however, asserts that the likelihood of a positive outcome should be the same for a person from the advantageous and disadvantaged group. For instance, when deploying an algorithm to predict which applicants should be hired, both male and females should have equal probability of being hired. This definition requires that the vulnerable and advantaged groups are predefined, so that their positive outcome probabilities can be compared with one another (Corbett-Davies & Goel, 2018).

According to the Predictive Rate Parity definition, also named the *Sufficiency* criterion, an algorithm is fair when the precision rates are equal for the advantaged and disadvantaged group in the dataset. For instance, the probability of being hired by a company and being classified as a suitable applicant should be equal (Verma & Rubin, 2018; Chouldechova, 2017).

Many of these group fairness metrics are applied during the post-processing stage. The main advantage of addressing fairness at this stage is that these fairness instruments are less intrusive than the pre-processing and algorithm optimization techniques, as they do not modify the input data or the model itself (Barredo Arrieta et al., 2020). Most of the post-processing methods, however, require that developers pre-define the sensitive attributes of the dataset to determine whether the algorithm produces its outcomes based on protected characteristics or is biased towards individuals or vulnerable groups. Yet, the sensitive information is often hidden in the dataset: there are multiple sources for bias, and most biases are not captured by single or recognisable attributes (Barredo Arrieta et al., 2020). This is primarily caused by the presence of *proxy variables* that cause *indirect bias*. Indirect bias refers to the phenomenon that even when the protected attributes are not considered directly by the model, the protected groups or individuals are still unjustly affected by the algorithm. This effect is caused by proxy variables, which are attributes that correlate strongly with the sensitive attributes. For instance, even when omitting the ethnicity attribute from the dataset, using geographical and other contextual variables may still reveal information about the individual's ethnicity.

One approach for handling indirect bias is to research how sensitive attributes are encoded by other attributes (Veale & Binns, 2017). Still, indirect bias remains difficult to address,

as the AI developers are faced with blind spots, hidden patterns and *unknown unknowns*, where they are unaware of their unconscious bias and therefore unable to retrieve how and where the harmful bias was included in the data or the algorithm (Hao, 2019). The Johari Window introduced by Luft and Ingham (1961) illustrates the gaps in self-awareness and awareness about others with which all humans are faced. These challenges are also faced by AI developers and policy makers who aim to reduce and prevent undesired bias.

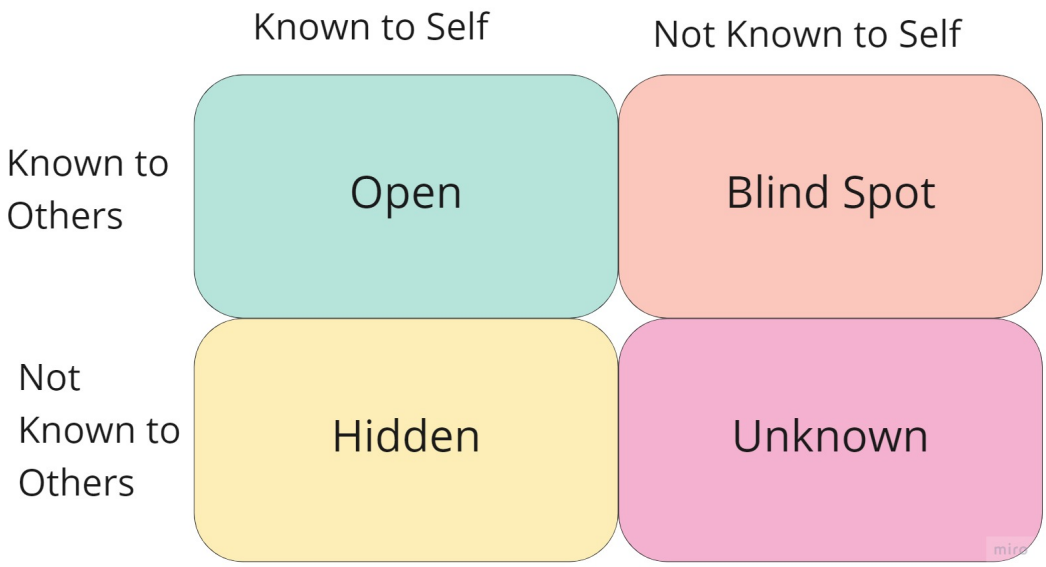


Figure 2.3: The Johari Window

Additionally, not only the sensitive attributes are often required, but *a priori* knowledge about vulnerable groups or individuals is often required for these fairness metrics, such as *"pregnant women"* or *"refugees"*. In many cases, developers need domain experts to identify these vulnerable groups, but often, these domain experts are not available or able to recognise all the combinations of attribute values that characterise each marginalised group in the dataset.

For the problems of finding bias while facing proxy variables, intersectional bias and other *unknown unknowns*, several studies have proposed to use unsupervised methods to discover bias in algorithms during the post-processing stage.

2.6 Unsupervised Learning with Clustering

Unsupervised learning entails the use of algorithms that discover hidden patterns or data groupings without the need for human intervention (IBM, 2020). These data groupings are formed through clustering algorithms, which we further explain in Chapter 3.

In the domain of algorithmic fairness, multiple studies have attempted to discover hidden bias using clustering algorithms. Nasiriani et al. (2019) proposed a method to detect possible discrimination with hierarchical clustering, where they aimed to identify individuals with similar characteristics but who are treated differently by the algorithm. Hence, the difference in treatment indicates a form of discrimination, which can be either positive

(favouring) or negative (harmful). The hierarchical clustering comprises two factors: it first selects the statistically relevant attributes to only consider explanatory attributes, hereby reducing any noise. In the second stage, a top-down hierarchical clustering algorithm is applied to automatically form clusters that are homogeneously labelled, which means that the majority of the instances within that cluster have the same decision value. For each cluster, the level of discrimination is the difference in the ratio between the positive and negative decisions. A positive ratio for a cluster means that this cluster is positively discriminated against by the algorithm, since it receives more positive than negative decision labels. Although this method allows underrepresented or favoured groups to be automatically detected, determining whether a group is discriminated still requires that the attributes are pre-identified. Thus, discrimination based on indirect sensitive information is not captured.

To address the limitations of the top-down hierarchical clustering method, Mistzal-Radecka and Indurkhya (2021) introduced the *Bias-Aware Hierarchical K-Means Clustering* (BAH-KM) algorithm, which is a similar method to identify for which groups of persons a recommender algorithm is underperforming. This model works by automatically forming and splitting clusters with the K-Means clustering algorithm based on the negative bias. This negative bias indicates the difference between the performance of a cluster and the performance of all the other clusters. The higher the deviation is, the more it indicates that the model is either favouring or discriminating against this particular cluster of data points. The choice of performance metric can be changed, which means that the BAH-KM is suitable for a wide range of algorithms. For instance, applying the BAH-KM on the results of a regression algorithm requires that the mean residuals is calculated for each cluster, whereas the BAH-KM applied on the results of a classification algorithm may comprise the True Positive Rate or Accuracy as metric. Aside from the model-agnostic properties of the BAH-KM, this method also does not require any *a priori* knowledge about the sensitive attributes in the dataset, since the discriminated groups are automatically formed with the partitioning mechanism of the clustering algorithm.

Still, there are some drawbacks to this bias discovery method. First, BAH-KM does not find the favoured clusters, while this positive discrimination can also provide useful insights about the decision-making process of the original model. Furthermore, the BAH-KM uses K-Means to split one of the clusters into two new clusters when a high negative bias has been found. K-Means has the drawback that it requires a pre-specified number of clusters, whereas clustering algorithms such as DBSCAN or Mean-Shift are density-based clustering algorithms which do not require the number of clusters to be determined in advance. Finally, BAH-KM has only been applied on recommender algorithms, and its ability to detect discriminated groups from classification or regression algorithms has not been examined yet.

Considering the drawbacks of BAH-KM, we propose the unsupervised bias detection algorithm *Hierarchical Bias-Aware Clustering* (HBAC), in which we modify the BAH-KM as follows:

- We apply HBAC on the results of a classification algorithm instead of recommendation algorithms. Thus, we investigate how this unsupervised bias detection method functions on a different type of dataset, where we also have access to the ground truth labels;
- We include the errors of the classifier as one of the attributes on which HBAC is applied. These errors are scaled differently than the other features, which we will

also experiment;
- A different metric is used to calculate the negative bias. Instead of the Normalized Discounted Cumulative Gain, which is primarily a measure to evaluate the ranking quality of a recommendation algorithm, we now use Accuracy as evaluation metric, which is more suitable for evaluating classification algorithms;
- The Bias-Aware Clustering algorithm will be implemented with K-Means, DBSCAN and Mean-Shift to evaluate which of these algorithms performs best in identifying highly discriminated clusters.

In conclusion, HBAC complements the current post-processing fairness instruments by its ability to automatically detect discriminated groups that consist of persons whom the classifier is discriminating against. Unlike the top-down hierarchical clustering proposed by Nasiriani et al. (2019), HBAC does not require any knowledge about the sensitive information hidden in the dataset. As for the BAH-KM introduced by Misztal-Radecka and Indurkhya. (2021), our method sets out to extend this method with the aforementioned modifications, so that it becomes usable for a wider range of algorithms. Finally, with HBAC, developers can gain more insight about the hidden biases of their algorithms and assess the biases with different fairness definitions, as HBAC allows different performance metrics to define the bias.

Chapter 3

Clustering Algorithms

This chapter addresses the relevant concepts that are required to understand the clustering analysis experiment conducted in this study. We first provide a general introduction on unsupervised methods in section 3.1 before we delve into the domain of clustering algorithms in section 3.2.

3.1 Unsupervised Learning

Contrarily to supervised learning, where an algorithm is trained to abstract patterns using labelled examples, unsupervised models learn patterns from data without any examples. Generally, unsupervised learning methods are used for dimensionality reduction and cluster analysis. Dimensionality reduction is used for many applications where the used data has a high dimensionality, such as for object recognition and speech processing. The aim of dimensionality reduction methods, such as Principal Component Analysis and factor analysis, is to create a meaningful representation of the data with fewer dimensions (Van Der Maaten, Postma, & Van Den Herik, 2009). This meaningful and reduced data representation can then be used, among many other purposes, for classification and visualisation algorithms. While we will use Principal Component Analysis for our experiment, our main scope of interest is the other type of unsupervised learning methods, which is clustering.

3.2 Clustering Properties

The main objective of clustering is to find a natural grouping among data points. Clustering algorithms organise data into groups or *clusters* such that there is a *intracluster-similarity* and a low *inter-cluster similarity*. *Intracluster-similarity* refers to data points within the same group being similar to each other, while a high *inter-cluster similarity* implies that the groups should be sufficiently distinguishable (Yeomans & Xu, 2020). Figure 3.1 shows the relationship between intra-cluster and inter-cluster similarity.

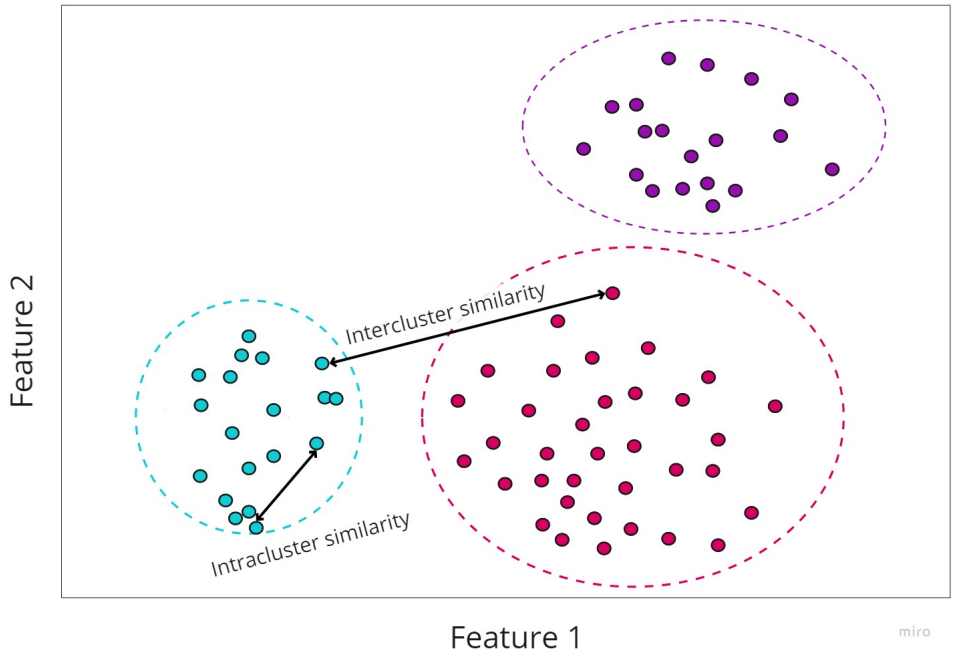


Figure 3.1: The relationship between intracluster and intercluster similarity

Clustering algorithms generally contain a proximity measure, an algorithm to form the clusters, and an evaluation criterion.

3.2.1 Distance Metrics

A **proximity measure**, also referred to as *distance metric* is required to measure the similarity or the dissimilarity between the data points. The Euclidean distance metric is most used to calculate the distance between data. Below, the formula of the Euclidean Distance is provided.

$$d_{euc}(x,y) = \sqrt{\sum_{i=1}^{n} (\{x_i - y_i)\}^2} \tag{3.1}$$

Besides Euclidean distance, the Manhattan distance metric is also commonly used as a proximity measure in clustering algorithms. This distance metric can be formalised as follows:

$$d_{manh}(x,y) = \sum_{i=1}^{m} \left| x_i - y_i \right| \tag{3.2}$$

Figure 3.2 displays the differences between the Euclidean and Manhattan distance metrics.

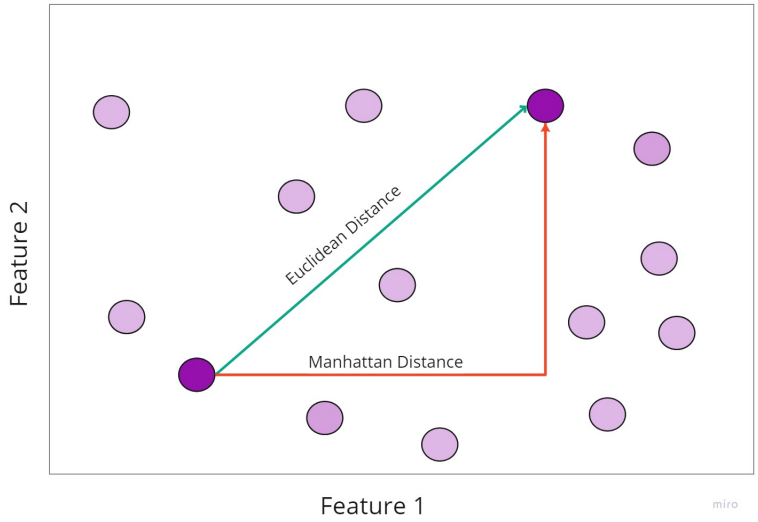


Figure 3.2: The Euclidean and Manhattan distance metrics

Other than the distance metrics, clustering also requires an **algorithm** that computes the clusters. A wide range of possible clustering algorithms is available to group similar data points into compact clusters and to isolate the clusters from each other. Clustering algorithms can be categorised into *hierarchical*, *partitional* and *density-based* algorithms (Serra & Tagliaferri, 2018).

3.3 Hierarchical Clustering

With hierarchical clustering, a hierarchical relationship is constructed among the data to group instances together(Xu & Tian, 2015). This hierarchical relationship can be either agglomerative (*bottom-up*), where all instances are a separate cluster and eventually merge into larger clusters, or it can divisive (*top-down*), where all data points are placed in a single cluster and are iteratively divided in smaller clusters (Serra & Tagliaferri, 2018). The main advantage of these methods is that they provide an interpretable clustering visualisation named a dendrogram, which is displayed in Figure 3.3. However, hierarchical clustering is often computationally expensive and sensitive to noise and outliers (Arvai, 2020).

Figure 3.3: Agglomerative and Divisive Clustering on a Dendrogram

3.4 Partitional Clustering

As for partitional clustering algorithms: these work by constructing a cluster around a datapoint that is considered the center. The number of clusters, which is the number of *center datapoints*, needs to pre-specified with these techniques. The main advantage of partitional clustering methods is that they work well for spherically-shaped clusters and their high scalability to complex and high-dimensional data. Nevertheless, these methods are less suited to form clusters with complex shapes or varying sizes: with partitional clustering, only evenly sized and spherical clusters can be constructed. The pre-specification of the number of clusters and the sensitivity of these methods to outliers and noise are also disadvantageous (Xu & Tian, 2015).

K-Means and K-Medoids are the most well-established partitional clustering methods. With K-Means, first, number k is selected, which defines the number of datapoints that are the centers of clusters, from now on referred to as *centroids*. Each other datapoint is then assigned to the closest centroid, after which the centroid is recalculated to ensure that it is the mean of the datapoints belonging to it. This process is repeated until it achieves the stopping criterion. Often, the stopping criterion involves *convergence*, where the centroids remain the same during recalculation and no data points are assigned to different clusters (Arvai, 2020). Figure 3.4 illustrates the KMeans clustering model.

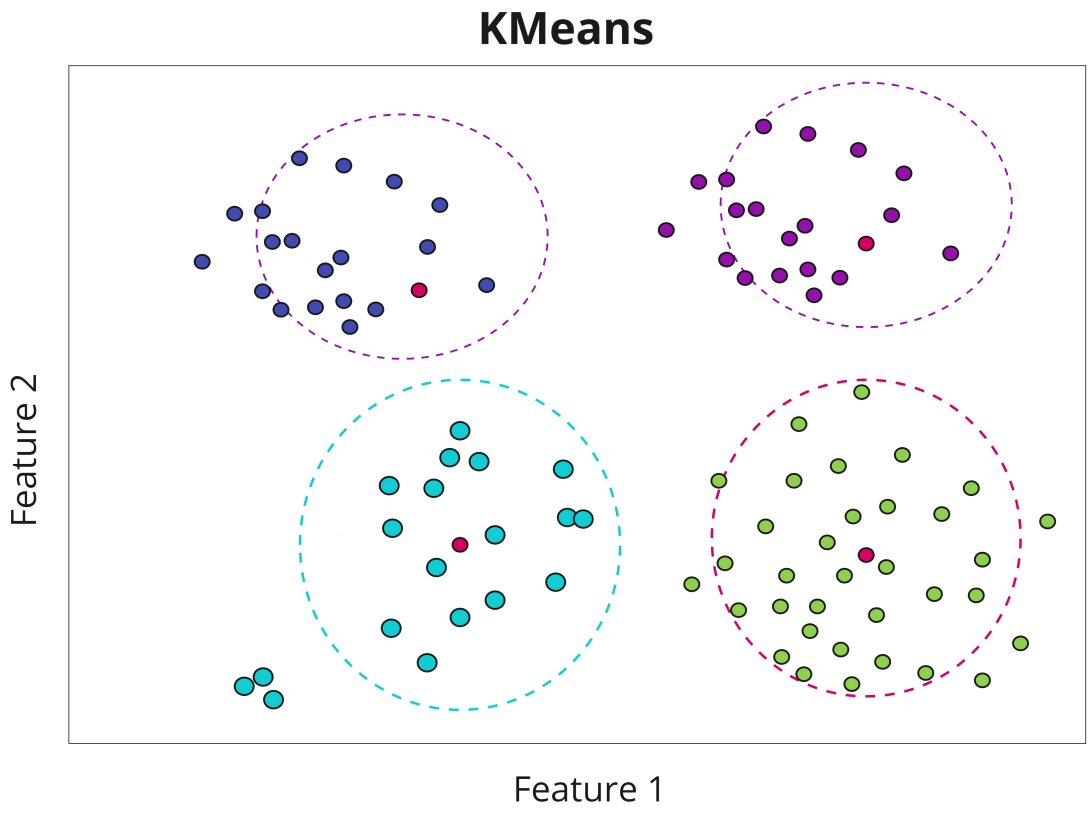


Figure 3.4: The KMeans model visualised

3.5 Density-based Clustering

Clustering methods belonging to this category try to form clusters from dense regions in the feature space. These dense regions are separated by regions with a lower density, which acts as a boundary between the clusters. Examples of density-based clustering methods are DBSCAN and MeanShift.

DBSCAN

(*Density-Based Spatial Clustering of Applications with Noise*) (DBSCAN) works by finding areas in the feature space which satisfy a minimum density and which are separated by lower-density areas. This minimum density level estimation is calculated based on a threshold for the number of neighbours named *minPts* and a radius ε, which determine for a given dataset what a dense area looks like. For any datapoint, if this point is surrounded by more than *minPts* within radius ε, it is considered a *core point* (Schubert, Sander, Ester, Kriegel, & Xu, 2017). All datapoints within the ε radius of a core point belong to the same cluster as this core point and are *density connected*. If these neighbours are also core points, the neighborhoods become transitively included. Non-core points in a cluster are named *border points*. Finally, if a datapoint is not density reachable from any core point, it is considered an outlier. Figure 3.5 shows an illustration of DBSCAN model.

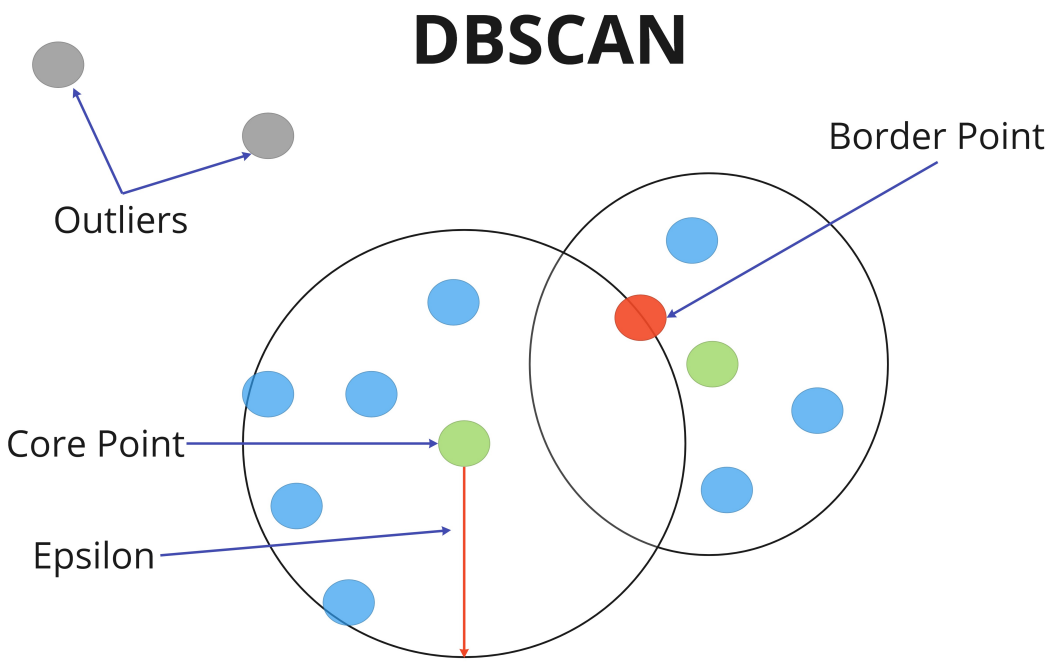


Figure 3.5: The DBSCAN model visualised

Compared with partitional clustering models such as K-Means, DBSCAN has multiple advantages: it does not require the numbers of clusters to be prespecified, and it creates clusters of arbitrarily shape and size (Scikit-learn, n.d.). While this eliminates the need for domain experts to indicate the number of clusters, DBSCAN remains relatively difficult to optimize due to the ϵ and *MinPts* that specify which dense region DBSCAN will consider as a cluster.

MeanShift

Besides DBSCAN, MeanShift is also a well-established density-based clustering method. This type of clustering is mainly used to "*discover blobs in a smooth density of samples*" (Scikit-learn, n.d.). The clustering algorithm assumes that there is a probability density function from which the data is sampled and where dense regions in the feature space correspond to the maxima of the underlying distribution (Derpanis, 2005). The clustering technique attempts to place centroids (similar to those in KMeans) on the maxima of this density function.

MeanShift has a single parameter named the *bandwidth* which is illustrated in Figure 3.6. This parameter represents the kernel or *window* that is shifted iteratively over the datapoints to find denser regions. The shift is guided by the mean shift vector, points towards the direction of the most dense regions within each window. During each iteration, the kernel is shifted to the centroid of the datapoints within the bandwidth. The higher this bandwidth is, the more datapoints it considers when looking for dense regions, which results in large clusters. On the other hand, if we use an extremely small bandwidth, each datapoint will have its own cluster.

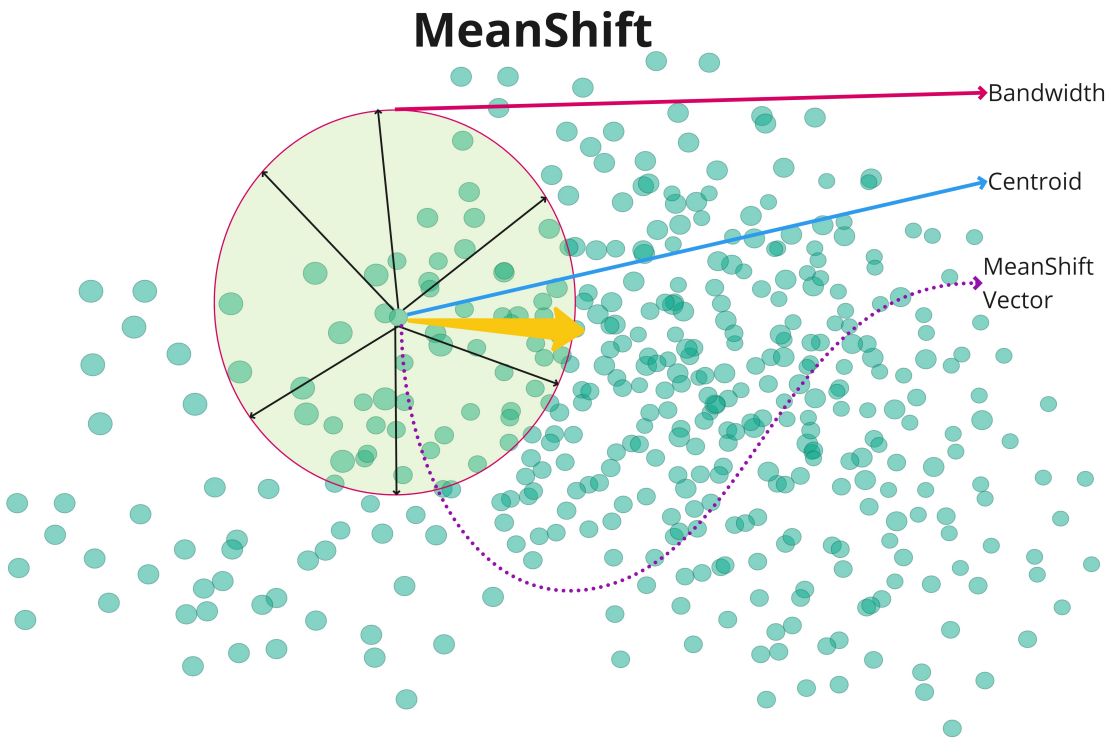


Figure 3.6: A visualisation of MeanShift

Similar to DBSCAN, MeanShift does not require the numbers of clusters to be specified beforehand, and it can form clusters with more shapes and different sizes when compared with KMeans. However, this algorithm does not work well with high-dimensional data.

3.6 Cluster Evaluation

Finally, a **criterion function** is required to evaluate the meaningfulness and usefulness of the found clusters. Meaningful clusters contribute to expanding domain knowledge, while useful clusters primarily provide developers with insights about the groups in their dataset, for example for business customer segmentation (Arvai, 2020). Since clustering techniques fall under the umbrella of unsupervised techniques, it is more difficult to evaluate their performance. To recall, the main objective of clustering methods is to achieve a high intra-cluster similarity and low inter-cluster similarity. By means of *internal*, *external*, *manual* and *indirect* evaluation methods, it is possible to evaluate the clustering quality.

Internal evaluation refers to summarizing the clustering into a single quality score. These methods are used to evaluate whether the clustering objective of maximizing intra-cluster similarity and minimizing the inter-cluster similarity is achieved. With external evaluation, the clustering results are compared with a 'ground truth' model. Finally, with manual evaluation, a domain expert judges the quality and with an indirect evaluation, the usability of the clustered data is assessed based on its intended application (Feldman & Sanger, 2006).

Within the scope of the research conducted in this study, we will predominantly focus

on indirect evaluation methods. Additionally, since the clustering models are applied on a specific use case -which is *identifying discriminated clusters based on clustering the errors of classification algorithms-*, further analysing the usability of the found clusters can provide informative insights. Thus, with indirect evaluation, we establish properties and characteristics related to the use case to evaluate the performance of the clustering methods.

Chapter 4

Methodology

In this chapter, we discuss the methodological approach of the Hierarchical Bias-Aware Clustering (HBAC) experiment conducted in this study to discover bias in the output of classification algorithms. As mentioned in Section 2.6, HBAC is an adaptation of the unsupervised bias detection method conducted by Misztal-Radecka and Indurkhya (2021), which aims to discover potentially discriminated groups of similar users in recommender systems.

We begin by formally defining the problem and addressing the fairness and bias issues in section 4.1. The general HBAC pipeline is then laid out in section 4.2. In section 4.3, we provide further information about including errors as a clustering feature. Finally, we give an overview of the evaluation in section 4.4 and we lay out the experimental set-up in section 4.5.

4.1 Defining Group Fairness and Bias

First, let us define a set of N persons p where each person has k attributes $x_1, \cdots, x_k \in X_p$, such as *age*, *nationality* or *income*. A person's corresponding attributes X_p are referred to as a *person vector*. Furthermore, we have a classification algorithm $f(X_p) = \hat{y}_p$ that returns predictions $\hat{y}_p$ for a person p based on his attributes X_p. For any person p, a prediction might be either accurate, which means that there is no difference between the ground truth label y_p and prediction $\hat{y}_p$, or the prediction can be inaccurate, hereby indicating an error ($\hat{y}_p \neq y_p$). Finally, with evaluation metric $\bar{M}(A(p))$, we can calculate the average errors of an algorithm A in generating predictions for all persons $p \in P$.

For this study, we work on binary classification problems, and we choose an error metric $\bar{M}$ that considers False Positives (FP) and False Negatives (FN) as errors. The value of this error metric and increases as the number of misclassifications increases. Our metric is equivalent to 1-Accuracy.

$$\bar{M} = \sum_{p=1}^{1/N} \left| \hat{y}_p - y_p \right| \tag{4.1}$$

The error metric M_G for a group G of N_G persons can be calculated as follows:

$$\bar{M}_G = \sum_{p=1}^{1/N} \left| \hat{y}_p - y_p \right| \quad \forall p \in G \tag{4.2}$$

We can compute the error metric $\bar{M}$ for the remaining N persons outside of group G as:

$$\bar{M} = \sum_{p=1}^{1/N} \left| \hat{y}_p - y_p \right| \quad \forall p \notin G \tag{4.3}$$

This metric M can be adapted to the task algorithm A tries to solve (e.g classification or regression) and its context (in some models, a False Positive is preferred over a False Negative, which can affect which metric is most applicable for the model.) When assessing the fairness of a classification algorithm, the *independence criterion* should be satisfied. According to this criterion, algorithm A should perform equally well for all groups of persons:

$$M \perp P_G,\ P_G \in P \quad \forall p \in G \tag{4.4}$$

Following from this independence criterion, if there exists a group of similar persons $P_G \subset P$ for which metric M_G is substantially lower than the average metric for the rest of the persons M, then this group is discriminated against by algorithm A based on metric M. Note that with some other error metrics (e.g FP rate), a lower metric M means a favouring bias rather than a discriminating bias.

4.1.1 Defining Discrimination and Favouring Bias

The degree of discrimination can be measured as the difference between metric M for group G and metric M for all other data points. More formally, we adhere to the following definition for calculating the bias:

$$Bias_G = \bar{M}_G - \bar{M} \tag{4.5}$$

In our case, discriminating bias occurs when $Bias_G < 0$ and the favouring bias when $Bias_G > 0$. A negative $bias_G$ indicates discrimination, as algorithm A is producing more errors for this group G when compared to all the other groups in the dataset. On the contrary, when $Bias_G$ has a high positive value, then it means that A is favouring this cluster G to other clusters, as the algorithm performs relatively better for this group than for all the other groups. Furthermore, the more extreme the bias is, meaning that it is either very low or high, the more it indicates a presence of bias. To select the high biases in both directions, the absolute bias $\left| M_G - M_{\setminus G} \right|$ can be investigated. Preferably, the classifier should have a null or close to zero absolute bias, which indicates that the errors are uniformly distributed on all identified groups.

4.2 The Clustering Pipeline

To evaluate whether an algorithm A is discriminating against a single or multiple group(s) of persons, we apply the Hierarchical Bias-Aware Clustering (HBAC) on the results of

algorithm A. The HBAC pipeline comprises three consecutive stages: first, the original dataset is pre-processed. In the second stage, the clustering algorithm is applied on the data. Finally, the characteristics of the biased clusters are analysed.

Figure 4.1 displays the HBAC pipeline which covers the dataset pre-processing and both stages of the experiment.

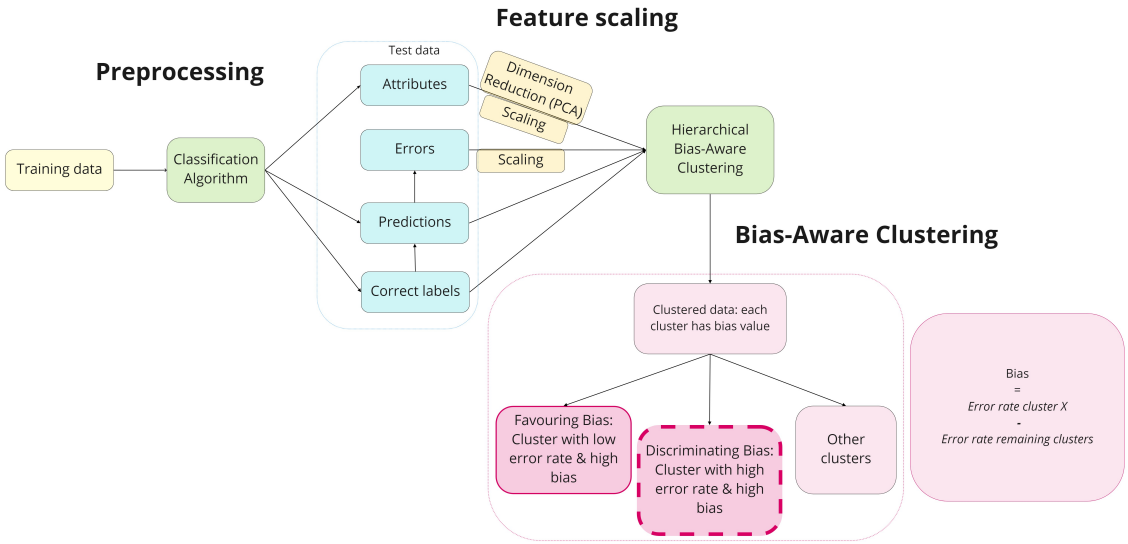


Figure 4.1: The HBAC bias pipeline

The goal of pre-processing the original dataset is to create a suitable data structure and content for the HBAC algorithm. This original dataset is used to train the classifier (explained in Section 4.2.2) and then to evaluate its performance on test data. The results of this classifier on the test data are further processed and then fed to the HBAC algorithm 4.2.5.

4.2.1 Pre-processing

The first step of the pipeline is pre-processing the dataset which will be used by the classifier to generate the predictions. The pre-processing includes investigating the dataset distribution and outliers, as well as imputing the missing values. Furthermore, we apply one-hot encodings to the categorical attributes, as the clustering component of the HBAC algorithm requires numerical data to calculate the distance between the data points. With one-hot encodings, categorical attributes are converted to numerical attributes by creating binary columns for each category. For example, the categorical attribute "*gender*", containing the categories "*male*", "*female*" and "*not-specified*", can be converted to the attributes in Table 4.1:

Table 4.1: One-hot encoded features

Instances	Gender-female	Gender-Male	Gender-Not-Specified
Mark	0	1	0
Melissa	1	0	0

4.2.2 Training the Classifier

We used 80% of the original dataset as training data and used the remaining 20% as the testset. If the original dataset contains less than 800 instances, a different train-test ratio might be used for the experiment, for example a 70-30 ratio. The type of classification algorithm is context-dependent, but in this case, we have chosen the Random Forest Classifier due to its versatility, and its robustness to outliers and non-linear data (Breiman, 2001).Additionally, Random Forest is less prone to overfitting when compared to other classification algorithms, such as Decision Trees and Support Vector Machines.

4.2.3 Generating the HBAC Dataset

After applying the classification algorithm on the dataset, we generate the dataset required for HBAC. Essentially, the HBAC dataset consists of the results of a classification algorithm obtained on the test dataset. These results are obtained by calculating the errors from the predictions and correct labels per instance. In our case, we use $\left|\hat{y}_p - y_p\right|$.
Thus, the HBAC dataset contains the following components:

- The instances of the test set with their corresponding attributes;
- The classification predictions;
- The ground truth labels;
- The classification error labels (0 for correct classification, 1 for misclassification).

Both the predictions and the ground truth labels are included to be able to further identify the type of errors, such as False Negatives and False Positives. All components are all fed to the HBAC in form of a tabular data structure. We exclude any other types of data structures from this experiment. Table 4.2 provides an example of the desired data structure.

Table 4.2: Desired HBAC dataset structure

Person	Age	Occupation-Plumber (one-hot encoded)	Predicted Class	True Class	Errors (in absolute values)
0	18	0	2	1	1
1	32	1	1	1	0
2	54	0	1	2	1

4.2.4 Feature Scaling

All the attributes of this dataset excluding the errors are rescaled using the MinMaxScaler from Sci-Kit Learn, such that each feature has a mean of zero and a standard deviation of one, meaning that they have the properties of a normal distribution. By standardizing the attributes of a dataset, the features are assigned an equal weight and scale, which is often required for clustering algorithms, since they use distance metrics which are prone to irregularities in the feature scales and sizes (Virmani, Taneja, & Malhotra, 2015).

The scaling of the error attribute impacts the results of HBAC and is a topic of research in itself. For instance, the clustering algorithm may isolate clusters of error faster or more accurately if the error attribute is given a larger range of values, which increases the distance between the correct and incorrect datapoints. The potential scaling approaches, and the rationale behind including the errors and subsequent scaling process are investigated in section 4.3.

4.2.5 HBAC Algorithm

The resulting dataset is then fed to the HBAC algorithm, of which the pseudo-code is provided in Algorithm 1.

Algorithm 1: Hierarchical Bias-Aware Clustering (HBAC)

```
Place the set of N instances in single cluster k;
Calculate error metric M_k;
for i=1 until max iterations do
    Split cluster k into n clusters k_i,...,k_n with Clustering Algorithm*;
    Calculate bias_{k_0}...bias_{k_n};
    if max(bias_{k_i},...,bias_{k_n}) > bias_k then
        Add these new clusters to the cluster list;
        Remove previous cluster k;
    end
    if New cluster has the highest variance(σ^2) in M and
      Size of new cluster ≥ minimal splittable cluster size and
      bias_{new cluster}>bias_{previous iteration} then
        Select new cluster as k;
    else
        Select random cluster from list
    end
end
```

During initialisation, all data points of the dataset are assigned to the same cluster k, for which we calculate the performance with error metric M. Then, we apply a clustering model, which is either K-Means, DBSCAN or Mean-Shift, to split this first cluster into new clusters. For each of the new clusters that result from k, we first calculate its error rate M_{k1}, M_{k2},M_{k3},..,M_{k4} and then the discriminating bias according to the definitions provided in section 4.1. To recall, the bias indicates the difference between the error metric for the selected cluster and the error metric for the remaining clusters.

Each of the clustering models has a single or multiple parameters that define the number of clusters in which cluster k is split. The approach for hyperparameter tuning and handling other specific properties of these clustering models is further explained in section 4.5.2.

After calculating the discriminating bias for each of the clusters, we compare these biases with each other to determine which of these cluster will be split into new clusters during the next iteration. The following criteria are used to select this new k:

- The cluster must have highest variance σ^2 of M when compared to the other clusters and the cluster of the previous iteration. This negative bias ($bias < 0$) indicates a discrimination bias, since it contains instances for which the algorithm produces relatively more errors when compared with other clusters.
- The cluster must contain sufficient instances to be considered as a *splittable* cluster. This constraint prevents that HBAC will only find small clusters with few instances, for which it is difficult to abstract a meaningful pattern. Hence, each cluster should be equal or exceed the minimum splittable cluster size. If the candidate cluster has fewer instances than the minimum cluster size, the HBAC considers a different cluster randomly until it finds a candidate cluster with an appropriate size.

The algorithm is terminated after it reaches the maximum iteration threshold or after no clusters are found that have a higher discrimination bias when compared to the clusters of the previous iteration.

4.2.6 Visualisation

Furthermore, as shown in Figure 4.1, the dimensions of the data are reduced using Principal Component Analysis. This form of dimension reduction, however, is only applied on the dataset to obtain the clustering visualisations. Thus, the HBAC is applied on all the dimensions of the dataset.

4.3 Errors as a Clustering Attribute

Errors in the dataset indicate for which persons the algorithm produced incorrect predictions. One of the contributions in this HBAC algorithm is the inclusion of the classification errors as a clustering attribute.

The advantage of including the errors is that they can guide the clustering algorithm to find more biased clusters. Using errors as a clustering feature adds a new dimension to the feature space, which is then used by the clustering algorithms to calculate the (Euclidean) distance between each data point. The distance between the data points on each dimension impacts which and how many clusters are eventually formed. Before scaling, the error dimension contains 1's for errors and o's for correct predictions, and we want to use this dimension to lead the algorithm into forming clusters with either too many (which is the discriminating bias) or just few errors (which might indicate the favouring bias).

However, the effect of adding the errors as attribute should be balanced to prevent the formation of clusters with either all or none of the errors, as this would yield a low performance variance in the clusters. This performance variance drives the bias-aware algorithm to split on the clusters with a high variance to eventually identify the highest discriminated clusters, which can also be small clusters within larger clusters. A too large effect of the errors as attributes would result in large clusters which cannot be split into smaller, possibly more biased groups. Additionally, as illustrated in Figure 4.2, we also want to prevent the clustering algorithm from isolating clusters that share similar characteristics but only differ slightly in the error distribution.

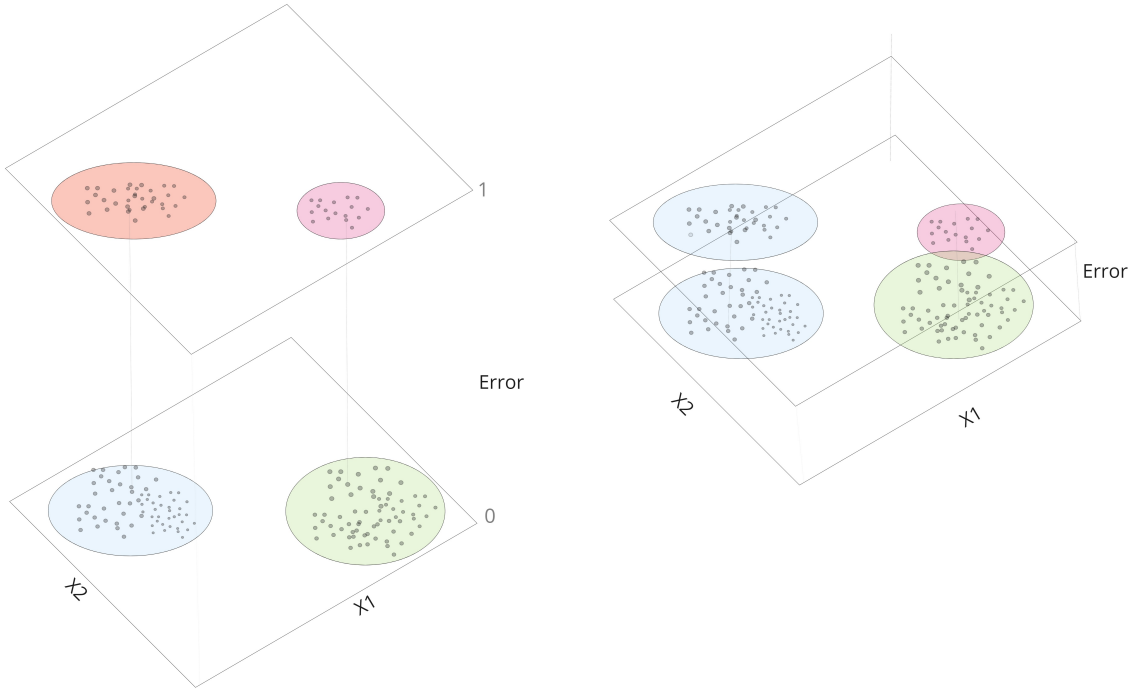


Figure 4.2: The effect of the error dimension on the formation of clusters

Thus, there is a trade-off between scaling or weighing the errors to guide the clustering process in finding biased clusters, while preventing too large and uninformative clusters. To account for this balance, we set the default scaling factor of the errors to 0.8 and we further experiment with higher and smaller error scaling factors, which we will further describe in Section 4.4.

4.4 Evaluation

To evaluate the effectiveness of the HBAC in finding biased clusters, we set up several experiments with each their own objective. All the experiments combined give insight into the performance and effectiveness of the HBAC bias detection method in finding discriminated groups.

Analysing Discriminated Clusters

The first goal is to find the cluster with the highest discrimination bias. We compare the ability of the HBAC-KMeans, HBAC-DBSCAN and HBAC-MeanShift in finding the cluster with the highest discriminating bias. We use multiple evaluation methods to analyse the discriminated results.

First, we construct a table containing the differences between the average unscaled values of the discriminated cluster and of the remaining cluster per feature. With this table, we can also visualise the differences in means through a parallel coordinate plot.In parallel coordinate plots, each data point is presented as an unbroken line segment where it passes parallel axes. Each of these axes represents a scaled feature and all axes are placed parallel to each other (Edsall, 2003). This visualisation technique is most useful for comparing the data points based on their different feature values. However, to compare the coordinates

of the clusters, they need to have the same scale. If the scales of the features vary, the differences between the features with smaller values will not be visible. Thus, the parallel coordinate plot will not be suitable for datasets containing varying scales.

Besides the parallel coordinate plot, we also compare the distributions of the discriminated versus remaining clusters through a kernel density estimation (KDE) plot using the Seaborn package.

Lastly, we use Welch's Two-Samples T-Test for Unequal Variances to examine whether the differences in means for each feature are statistically significant. We mainly use this statistical test to locate the features that require closer inspection: since the table with feature value differences contains unscaled values for each feature, we might otherwise overlook the features with small values.

Role of Error Scaling

Besides the default error scaling factor of 0.8, we also explored the role of the error feature in finding clusters with a high discrimination bias. We experimented with a scaling the errors with a range of 0-5 with intervals of 0.5 to observe the effects of this attribute on identifying biased clusters.

Indirect Evaluation

Finally, besides a quantitative and visual analysis of the found discriminated clusters, we also evaluated the properties of each of the three Bias-Aware Algorithms. The purpose of this analysis was to examine which of these models is most suitable for the intended application of identifying discriminated clusters.

The indirect clustering evaluation entailed analysing the HBAC algorithms on standard qualitative clustering and context-specific (bias-discovery) properties. For the standard qualitative clustering properties, we will examine for each algorithm how it performs in terms of scalability and robustness. Bias-discovery specific properties consist of interpretability, validity and meaningfulness of the obtained clusters. Furthermore, we evaluate the role of a domain expert to tune the clustering-specific parameters, as this fairness assessment instrument is deployed in multi-disciplinary AI teams that also involve non-technical employees who need to be able to understand these clusters.

Table 4.3: Indirect Evaluation properties

Property	**Explanation**
Scalability	Ability to work for both small and large datasets and low- or high-dimensional data. Additionally, we evaluate the clustering algorithm's reliance on the data distribution to find clusters.
Robustness	To what extent does the HBAC method yield different results for each run?
Interpretability & meaningfulness of found clusters	What information can be obtained from the discriminated clusters?
Parameter tuning complexity / sensitivity	How sensitive is the HBAC algorithm to the clustering parameter(s)?
Role of the domain expert	To what extent is the domain expert needed for the HBAC-model? What kind of information is required for the clustering algorithm?

4.5 Experimental Set-Up

4.5.1 Datasets

The HBAC algorithm was applied on four datasets in total: 1) German Credit Risk, 2) Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) Recidivism Risk, 3) Synt1 and 4) Synt2.

German Credit Risk

The German Credit Risk dataset is constructed by economics Professor Hans Hofmann and obtained from the UCI Machine Learning Repository (1994) and from Kaggle (2017). This dataset is representative for the intended use-case of HBAC, since it contains sensitive attributes (e.g *sex* and *age*) and the potential impact of discriminating groups of persons can have long-lasting harmful effects. The dataset contains information about 1000 individuals with 10 personal characteristics as features, and it is used to predict whether they have a bad credit risk. A bad credit risk implies that the model predicts that this individual will have difficulty repaying it to the bank, which can lead the bank to deny the loan application. There are 10 attributes in data, such as *Age*, *Credit Amount*, *Checking Amount* and *Sex*.

The train-test split ratio is set to 70% train and 30% testdata, which has led to 92 instances in the test dataset.

After applying the RandomForest classifier on the test data, we obtained an accuracy of 0.64. This relatively low score was achieved on purpose, as it allows us to inspect the errors more during the experiment of identifying discriminated clusters. The resulting dataset on which we will apply the HBAC experiment is depicted in Figure 4.3.

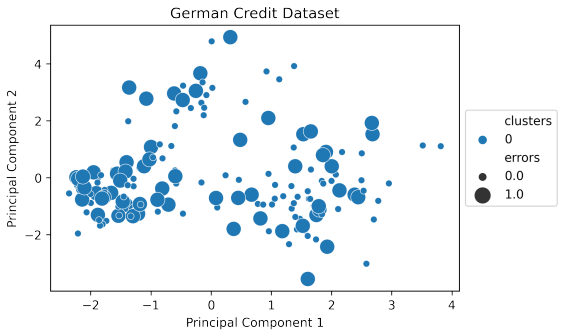


Figure 4.3: The German Credit dataset

COMPAS

COMPAS is a dataset initially researched by ProPublica (2016), and of which we use a simplified subset obtained from Kaggle (2017). The dataset contains information about defendants, and it is used to predict which of these defendants is likely to become a recidivist within two years. The dataset contains 6172 instances and ten features, including *gender*, *race*, *age* and *the number of prior crimes*. We use train-test ratio of 70-30, hereby providing us with 1852 instances for HBAC. Figure 4.5.1 shows a visualisation of the COMPAS dataset.

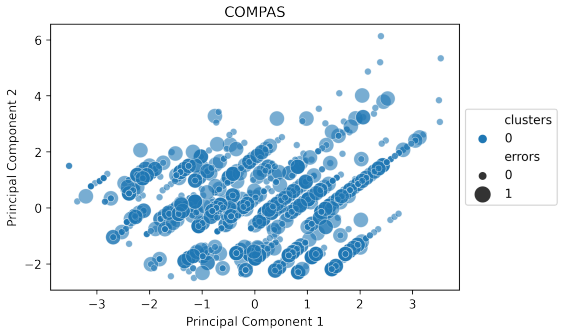


Figure 4.4: The COMPAS dataset

Theoretical Datasets Synt1 and Synt2

Finally, we created two synthetic datasets with different properties. These datasets supply possibilities to further observe the clustering behaviour of the HBAC models as they have more instances than the other datasets used in this study, and the distributions and errors of these instances can be manipulated. As mentioned in Chapter 3.5, the performance of the clustering models is often dependent on the shape and distribution of the dataset.

The main difference between Synt1 and Synt2 lies in the error distribution: Synt1 only contains errors which are uniformly distributed over all the generated classes, while Synt2 also contains error clusters. These error clusters allow us to investigate whether the HBAC algorithm can identify small but important clusters within larger clusters.

Each of the classes contains datapoints sampled from normal distributions of which the mean lies between -10 and 30 for both features and the standard deviation varies between 0-10. By increasing or decreasing the standard deviation, we can experiment with diverse levels of distribution densities.

Furthermore, we have uniformly distributed the errors over the classes with a probability of 0.22 for Synt1 and a probability of 0.12 for Synt2. For Synt2, we also added error clusters of which the mean was close to the class in which it was placed, and the standard deviation was <0.55. Around 70% of these error clusters were errors.

Overall, Synt1 contains 2111 instances of which 443 are errors. This yields an accuracy of 0.79. Synt2 has 2350 instances of which 790 are errors. Figures 4.5 and 4.6 display visualizations of Synt1 and Syn2., respectively. Since the synthetic datasets only have two features (besides the errors), we will not apply dimensionality reduction to generate the visualisations.

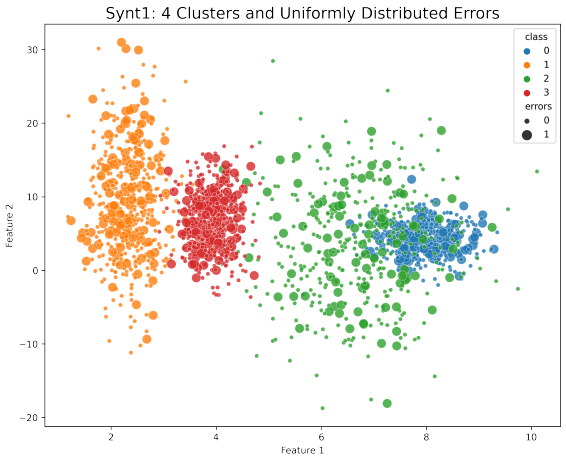


Figure 4.5: The Synt1 dataset

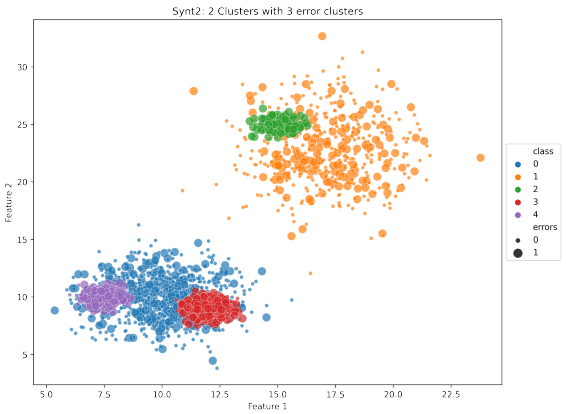


Figure 4.6: The Synt2 dataset

Table 4.4 summarizes the key characteristics of the datasets.

Table 4.4: Dataset Properties

	Wine	**German Credit**	**COMPAS**	**Synt1**	**Synt2**
Dataset size	178	1000	6172	2111	2350
Train set size	106	700 (70%)	4320 (70%)	-	-
Test set	72 (40%)	169	1852 (30%)	2111	2350
Target Variable	Type of wine	Bad credit risk	Recidivism within 2 years	/	/
Number of features	13	10 (before one-hot encoding)	11 (before one-hot encoding)	2	2
Accuracy (test set)	0.917	0.667	0.673	0.790	0.664
Additional properties	/	One-hot encoded features	One-hot encoded features	Uniformly distributed errors	Uniformly distributed errors + error clusters

4.5.2 Clustering Approaches

Three clustering techniques, consisting of K-Means, DBSCAN and Mean-Shift were applied in HBAC. For each of these algorithms, we used the same Euclidean distance metric to measure the distance of the datapoints, which is further explained in 3.2.1. In addition to the corresponding attributes, each clustering algorithms required specific parameter tuning.

K-Means

For K-Means Clustering, the number of clusters needs to be pre-specified to hierarchically split the first cluster and the subsequent clusters. We used the same approach as Misztal-Radecka and Indurkhya. (2021), who used k=2. Furthermore, to speed up the process of finding the optimal centroids that define the clusters, we used the KMeans++ initialisation parameter. With K-Means++, the initial clusters are selected after each other: the first centroid is chosen randomly, but each subsequent centroid is selected such that it has a maximum distance from the nearest centroid (Arthur & Vassilvitskii, 2007; Khosla, 2021). This initialization process ensures that the centroids are placed far from each other, which increases the likelihood that the centroids will be quickly found in different clusters.

DBSCAN

Two parameters required tuning: *epsilon* (ϵ) and *minimum number of samples* (*MinPts*). As mentioned in 3.5, these parameters are sensitive and relatively difficult to tune. The combination of ϵ and *MinPts* results a particular density, and the clustering algorithm only finds clusters at that or above this density. Thus, when the data contains variable densities, DBSCAN will not be able to identify these as clusters and will consider them as outliers.

We used the method proposed by Schubert et al. (2017) and Rahmah & Sitanggang (2016) to find the appropriate epsilon for each dataset. This approach involves using

a k-NearestNeighbours algorithm to calculate the distance from each data point to its neighbour, after which a K-distance graph can be constructed. Using the K-Distance graph, the distances of the datapoints to their neighbours are sorted in ascending order, and we select the ϵ as the value that corresponds to the maximum curvature of the distances.

However, since we iteratively split on a cluster, the ϵ should be also incrementally decreased to still identify the dense areas. Therefore, we iteratively decrease ϵ with a value ranging from 0.001 to 0.01.

Mean-Shift

For Mean-Shift, only a single parameter required tuning, which is the bandwidth. We used the bandwidth estimation function from Sci-Kit Learn to calculate the appropriate bandwidth for each dataset. Additionally, we iteratively decreased the bandwidth for some of the datasets with 0.1, as this Mean-Shift would otherwise be unable to identify changes in densities after we split a cluster into smaller clusters.

Chapter 5

Results

In this chapter, we evaluate the performance of the Hierarchical Bias-Aware Clustering algorithm, hereby answering research questions RQ1 and RQ2. In section 5.1, the clusters with the highest discriminating bias are analysed. Section 5.2 then discusses the results obtained from the error-scaling experiment. Finally, we conduct an indirect evaluation of the HBAC-KMeans, HBAC-DBSCAN and HBAC-Mean-Shift algorithms in Section 5.3 to investigate how suitable each of these algorithms based on their properties and performance for finding discriminated clusters.

5.1 Highest Discriminated Clusters

The primary objective of HBAC is to identify groups of instances with a high discriminating bias from the results of a classifier. After following the HBAC procedure as described in chapter 4, we analyse the clusters with the highest discrimination bias for each of the four datasets.

5.1.1 German Credit

For German Credit, one-hot encoded features were used to transform the categorical features into one-hot numeric features. Figures 5.1, 5.2 and 5.3 present the clusters found by HBAC-KMeans, HBAC-DBSCAN and HBAC-MeanShift, respectively. Apart from the similarity between Cluster 1 in HBAC-DBSCAN and cluster 1 in HBAC-MeanShift, the clustering algorithms identified fairly different clusters.

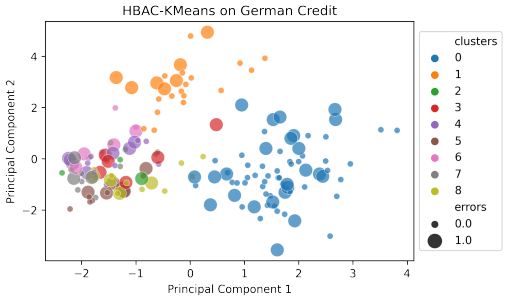


Figure 5.1: Clusters obtained with HBAC-KMeans

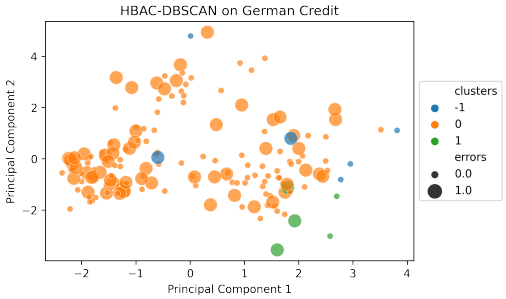


Figure 5.2: Clusters obtained with HBAC-DBSCAN

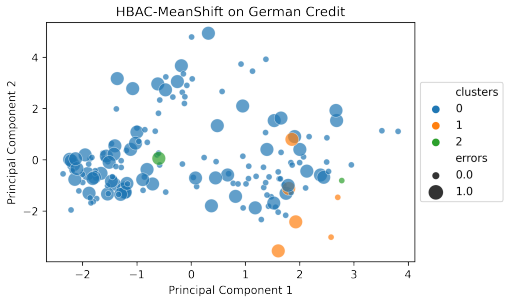


Figure 5.3: Clusters obtained with HBAC-MeanShift

HBAC-KMeans

Of the 9 identified clusters, cluster 3 contained the highest discriminating bias of -0.505. This cluster consisted of 7 instances of which 6 had errors. When comparing the means of the features for the discriminated versus the remaining clusters in table B.1, we observe that the discriminated cluster has instances that do not satisfy all the categories, e.g there are no adults or seniors amongst the 7 instances. This method of observing the absence or lower values for the one-hot encoded features can help to describe the patterns found in the discriminated cluster.

Furthermore, we used Welch's Two-Samples T-Test for Unequal Variances (with α of 0.05) to observe whether there were statistically significant differences in the means for the non-hot encoded features (which were *Job*, *Credit Amount* and *Duration*), but we found no significant results. Figure 5.4 describes the density distributions of these features. For the one-hot encoded features, we found significant differences in means when the discriminated cluster had no instances with positive values for a feature. Interestingly, the discriminated cluster had a much higher mean for *purpose-education* (0.86) when compared to the remaining clusters (0.02), which is also statistically significant (t=5.81, p=0.001). This information tells us that it is likely that the persons in the discriminated cluster have an occupation related to education.

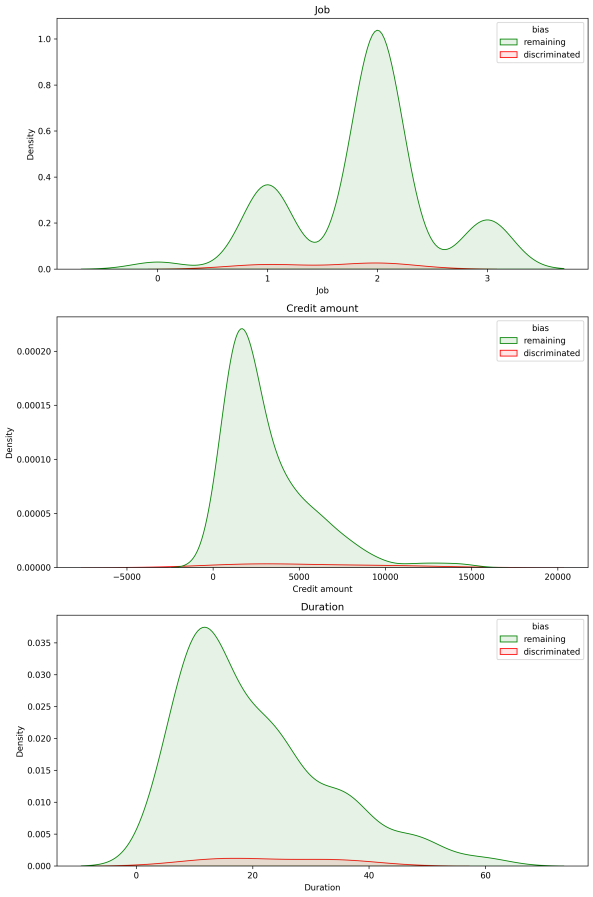


Figure 5.4: HBAC-KMeans Density Distributions in German Credit

HBAC-DBSCAN

For HBAC-DBSCAN, we used $MinPts = 3$ and $\epsilon = 6.5$, which was determined from the K-distance plot depicted in Figure B.1. Of the two clusters and the outliers, cluster 1 had the highest discrimination bias of -0.234. This cluster contained 5 instances of which 3 were misclassified. Contrarily to the results obtained from K-Means, we see in Figure 5.5 that the persons in this discriminated cluster had an average age of 65 years, which is substantially higher than the average age of the remaining clusters (34). The remaining results, which are provided in table B.2, show that this cluster also contained persons who are more often house owners. After conducting Welch's t-test, we found significant results for these features with t=22.35, p=6.46e-09 for *Age* and t=7.49, p=4.09e-12 for *House owner*.

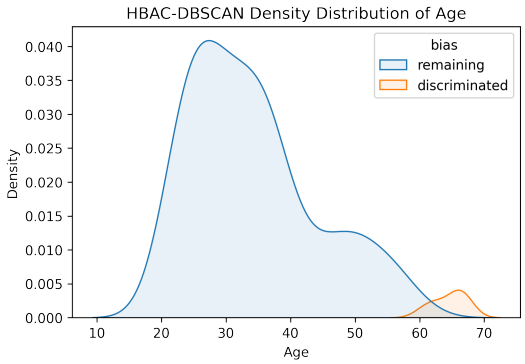


Figure 5.5: HBAC-DBSCAN Density Distribution for Age in German Credit

HBAC-MeanShift

Lastly, for HBAC-MeanShift we found the highest discrimination bias of -0.037 in cluster 1. This result was achieved using the estimation of the bandwidth, which amounted to 5.95, and by decreasing the minimal splittable and minimal acceptable cluster size to 1% of the dataset. Cluster 1 consisted of 6 instances of which 4 contained errors. After inspecting the differences in the means of the discriminated and remaining clusters in table B.3, we observed the same patterns as identified for HBAC-DBSCAN, particularly for the age of the person. Figure 5.6 displays the density distribution of this feature, which seems to be identical to Figure 5.5 of DBSCAN.

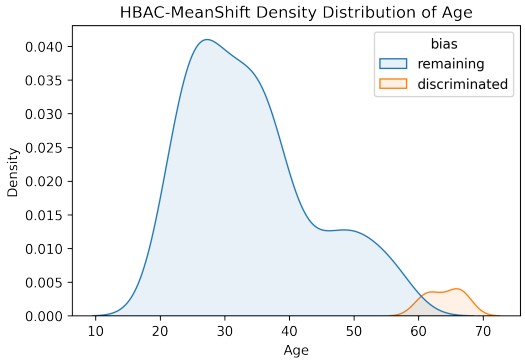


Figure 5.6: HBAC-MeanShift Density Distribution for Age in German Credit

Preliminary Conclusion

To conclude, we found a large difference between the discriminated clusters of KMeans, and DBSCAN or MeanShift. While the discriminated persons in KMeans on average had a younger age, the persons in DBSCAN and MeanShift all fell under the age category of seniors. Nevertheless, KMeans succeeded in finding the highest discrimination bias of -0.505.

5.1.2 COMPAS

Figures 5.7, 5.8 and 5.9 present the clusters found by HBAC-KMeans, HBAC-DBSCAN and HBAC-MeanShift, respectively. At first view, we observe that clusters 1 in DBSCAN

and MeanShift contain roughly the same instances.

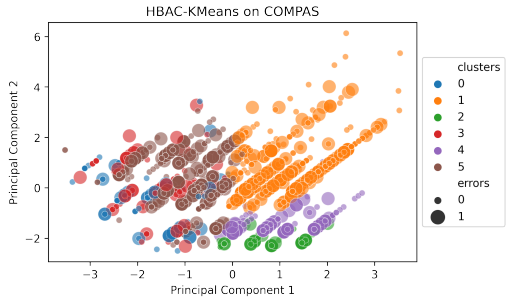


Figure 5.7: Clusters obtained with HBAC-KMeans

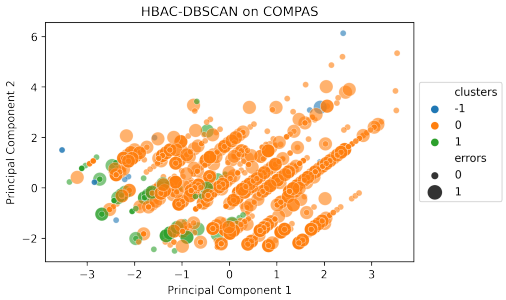


Figure 5.8: Clusters obtained with HBAC-DBSCAN

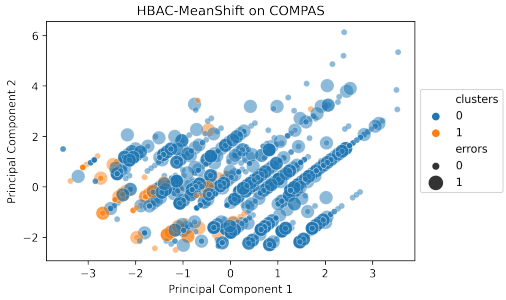


Figure 5.9: Clusters obtained with HBAC-MeanShift

HBAC-KMeans

Of the 6 clusters, we found the highest discriminating bias of -0.049 in cluster 2. This cluster contained 64 instances of which 24 had errors. From table B.4, and the parallel coordinate plot in Figure 5.10, we found the largest differences between the discriminated and other clusters for the features *Number of Priors*, *Age Below Twenty Five*, *Female* and *Misdemeanor*. Using this information, we would describe the discriminated cluster as containing females who are all under the age of twenty-five, of which most of them have an African-American ethnicity.

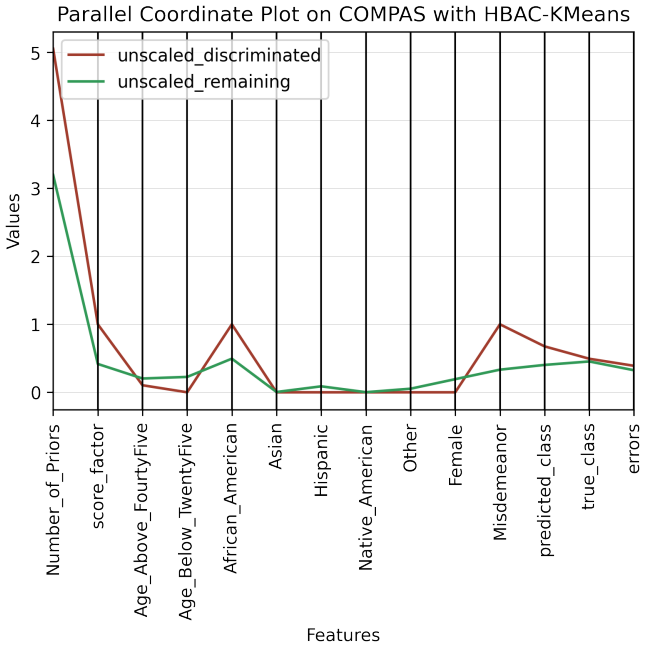


Figure 5.10: HBAC-KMeans Parallel Coordinate Plot for COMPAS

HBAC-DBSCAN

Since HBAC-DBSCAN did not find clusters for *eps=0.5* retrieved with a K-distance plot, we experimented with increasing the epsilon, and we eventually found the best results with *eps=4* and *minimum samples=2*. In total, 2 clusters and one outlier cluster were identified for COMPAS. Cluster 1 had the highest discriminating bias of -0.03, and it contained 93 instances of which one-third had errors. Furthermore, we use the parallel coordinate plot in Figure 5.11 and table B.5 to characterise the discriminated cluster. It can be concluded that this cluster is difficult to describe distinctively, as it both has a large size and the most important difference in cluster means can be found for *Other*, which has a non-descriptive name.

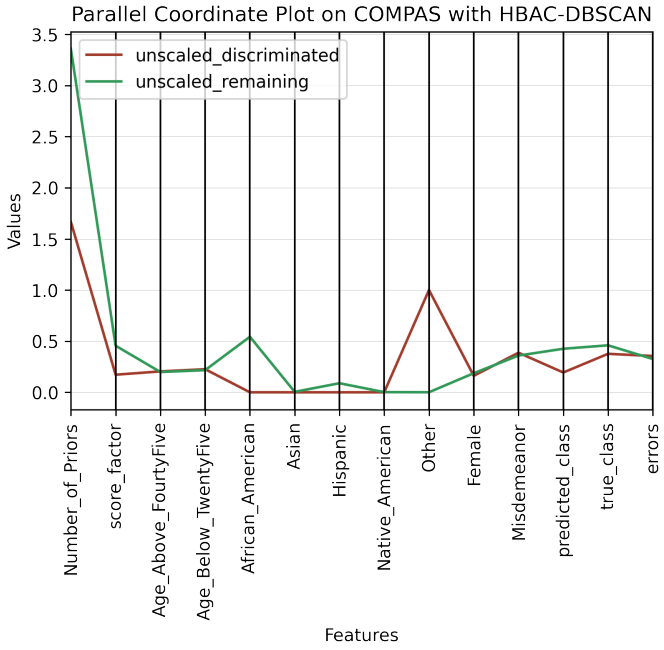


Figure 5.11: Parallel Coordinate Plot for COMPAS

HBAC-MeanShift

Finally, with MeanShift, we used the estimated bandwidth of 5.95, which partitioned the feature space in two clusters. The discriminated cluster, with a bias of -0.025, contained 94 instances of which 33 contained errors. Figure B.3 shows that what makes the discriminated cluster distinctive from the remaining clusters is again mainly due to the differences in means of features *Number of Priors* and *Other*.

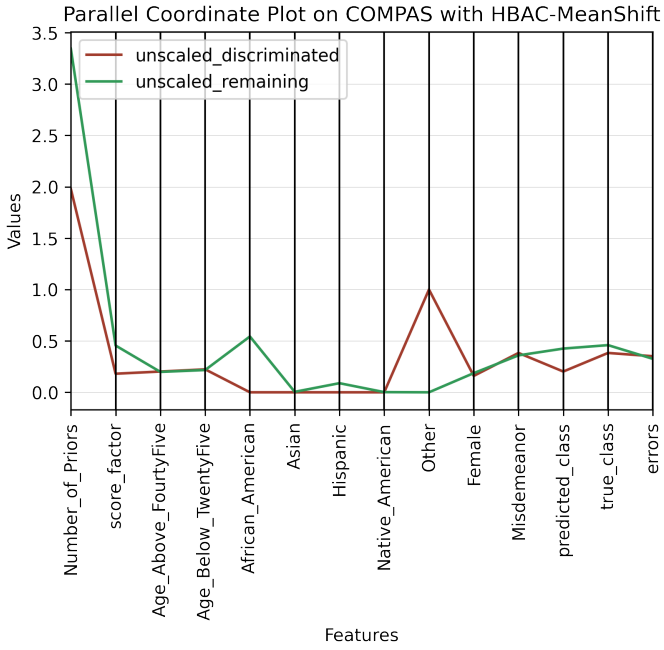


Figure 5.12: Parallel Coordinate plot HBAC-MeanShift

Preliminary Conclusion

To summarize, HBAC-KMeans found the highest discrimination bias of -0.049 when compared with HBAC-DBSCAN (-0.030) and HBAC-MeanShift (-0.025). Additionally, the discriminated cluster of KMeans was smaller than those of the other clustering algorithms, which allowed for a more refined examination of the differences between the discriminated and remaining clusters.

5.1.3 Synt1

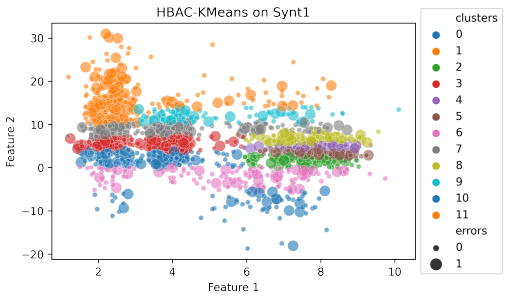


Figure 5.13: Clusters obtained with HBAC-KMeans

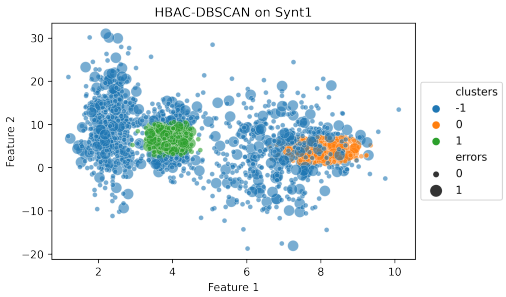


Figure 5.14: Clusters obtained with HBAC-DBSCAN

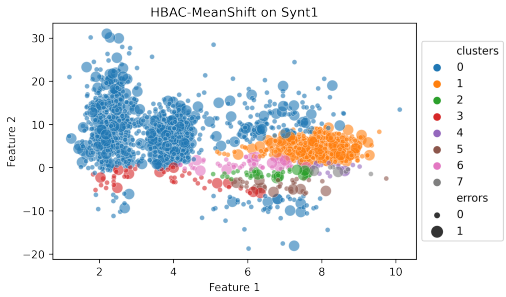
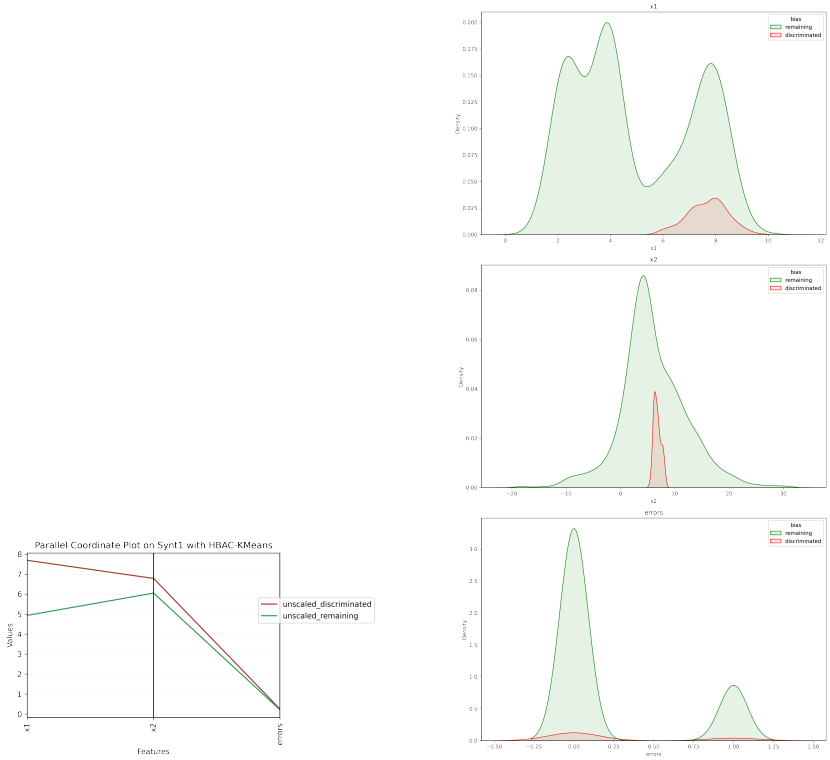


Figure 5.15: Clusters obtained with HBAC-MeanShift

HBAC-KMeans

The behaviour of KMeans's partitioning mechanism can be clearly observed from Figure 5.13, as it horizontally divided the feature space into clusters of similar size. From the 12 identified clusters, cluster 8 had the highest discrimination bias of -0.037. This cluster contained 139 instances of which 39 had misclassifications. The Figures in 5.16 show that the largest difference between the discriminated and remaining clusters can be found for feature X_1, which has a higher average of $\mu = 7.69$ when compared to the remaining clusters with $\mu = 4.94$. By conducting Welch's t-test, we find that this difference is statistically significant ($t = 33.98, p = 6.18e-116$).

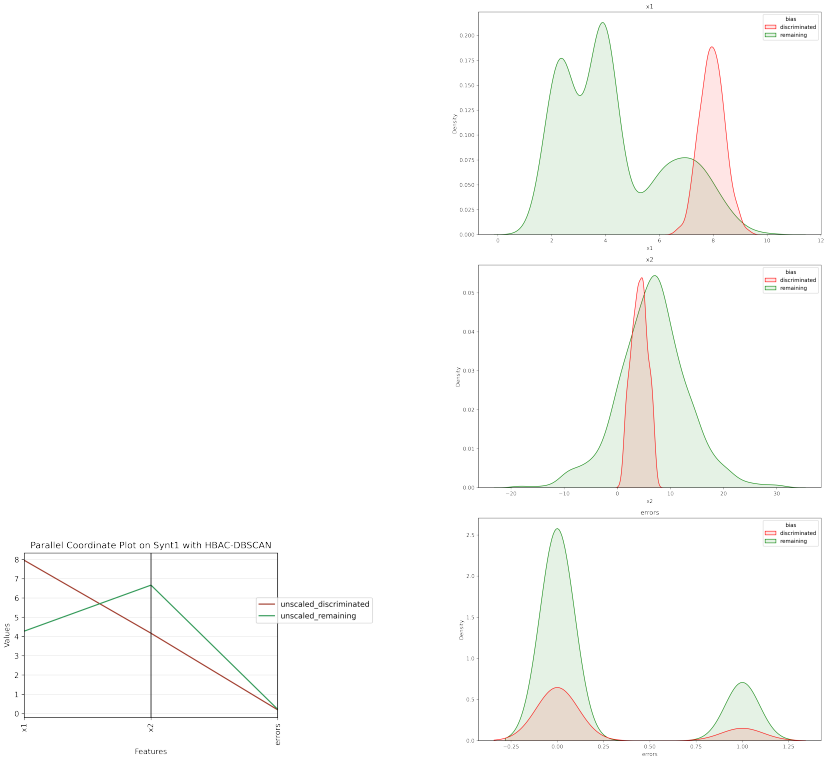


(a) Parallel Coordinate Plot (b) Distribution plots

Figure 5.16: The Discriminated and Remaining clusters retrieved from HBAC-KMeans

HBAC-DBSCAN

The highest bias was found in cluster 0, which amounted to 0.026. This result is peculiar, since this means that HBAC-DBSCAN did not manage to find discriminating bias in Synt1, as the other cluster had a bias of 0.029. When inspecting Table B.7 and Figure 5.18, we see that the discriminated cluster has an higher mean for X_1 and a lower mean for X_2 when compared to the remaining clusters. The density plot shows that, in particular, the discriminated cluster is distinctive by its values for X_1.

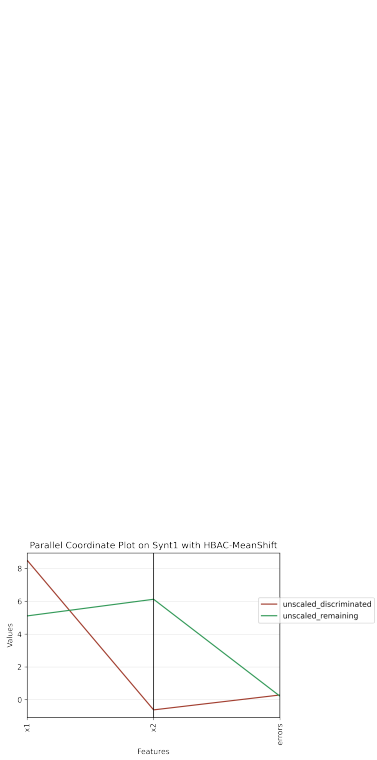


(a) Parallel Coordinate Plot (b) Distribution plots

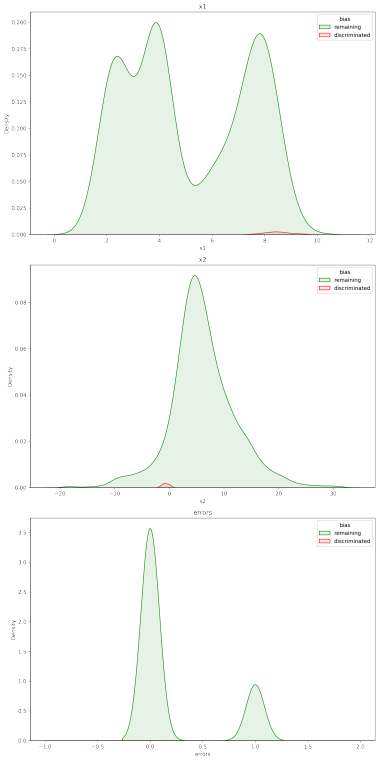
Figure 5.17: Analysis of Synt1 clusters obtained from HBAC-DBSCAN

HBAC-MeanShift

With the estimated bandwidth of 5.104, MeanShift identified only one cluster. However, when we decreased the bandwidth to 4.15 and iteratively 0.15, we found 8 clusters of which cluster 7 had the highest discrimination bias of -0.076. When compared to the discriminated cluster of KMeans and DBSCAN, cluster 7 was relatively small: it contained 7 instances of which 2 had errors.



(a) Parallel Coordinate Plot



(b) Distribution plots

Figure 5.18: The Discriminated and Remaining clusters retrieved from HBAC-MeanShift

Preliminary Conclusion

It was difficult for HBAC-DBSCAN and HBAC-MeanShift to find discriminating bias, as DBSCAN found no discriminating bias and MeanShift required rigorous tuning before finding clusters. MeanShift succeeded in finding the highest discrimination bias of -0.076, but it contained only 7 instances. Therefore, the discriminated cluster found by HBAC-KMeans seems more informative. Alltogether, this dataset allowed us to observe the behaviour of the clustering algorithms when errors are uniformly distributed.

5.1.4 Synt2

Finally, Figures 5.19, 5.20 and 5.21 present the results of the dataset Synt2. Interestingly, the error clusters on the bottom left were better recognized by the bias-aware algorithms than the error cluster on the upper right.

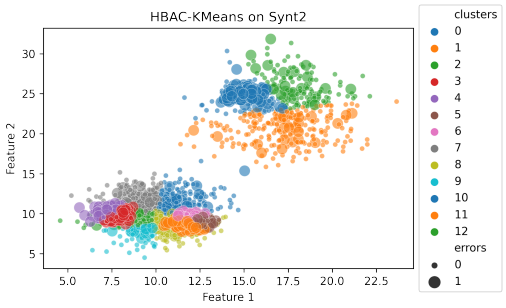


Figure 5.19: HBAC-KMeans clusters

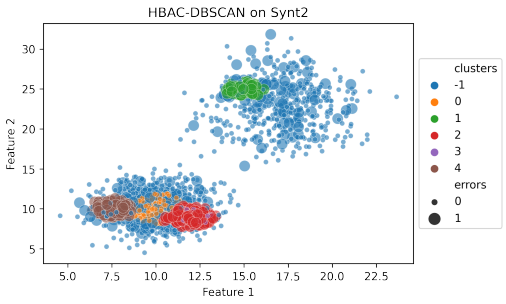


Figure 5.20: HBAC-DBSCAN clusters

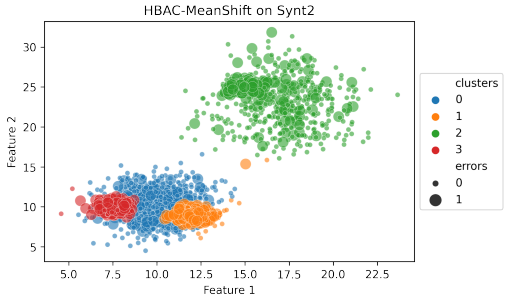
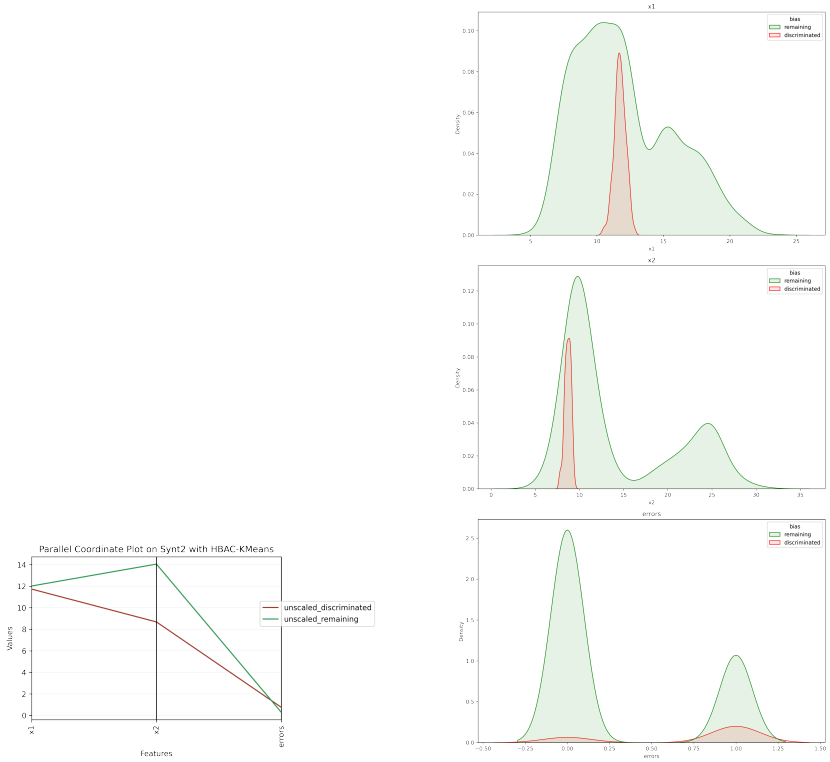


Figure 5.21: HBAC-MeanShift clusters

HBAC-KMeans

Of the three error clusters, two were identified by this algorithm. Cluster 1, which contained 226 instances of which 171 had errors, had the highest discrimination bias of -0.465. This cluster contained 226 instances of which 171 were errors. Based on the Figures in 5.22 and Table B.9, we see that the largest difference between the discriminated and remaining clusters is caused by X_2. Welch's t-test confirms this hypothesis, as there is a statistically significant difference for this feature ($t = -37.52, p = 2.91e-239$).



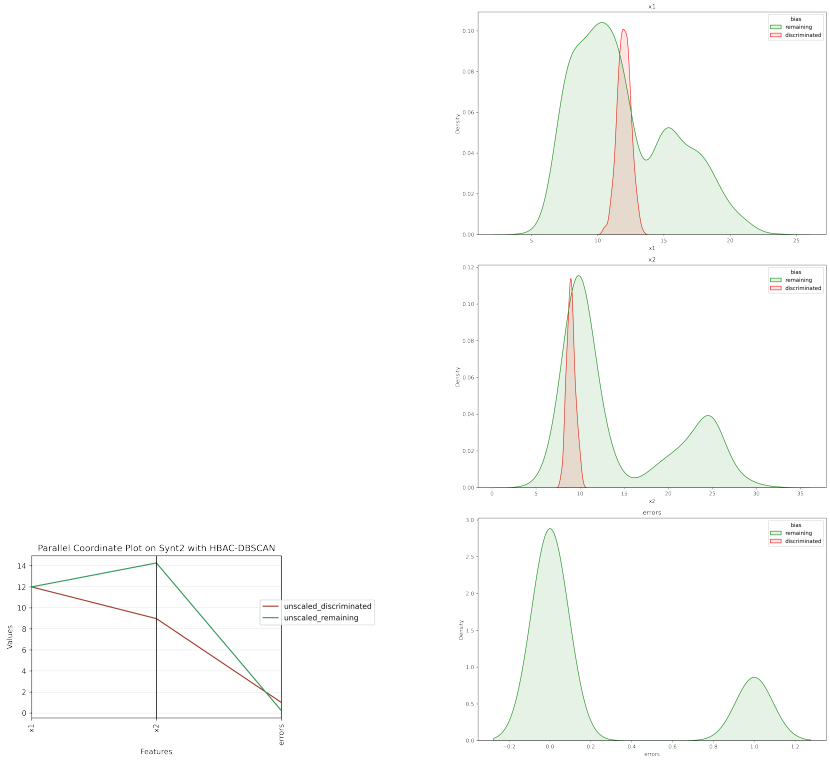
(a) Parallel Coordinate Plot (b) Distribution plots

Figure 5.22: Analysing the Discriminated and Remaining clusters retrieved from HBAC-KMeans

HBAC-DBSCAN

This clustering algorithm succeeded in finding the three error clusters. It found the highest discrimination bias of -0.76 in cluster 2. This cluster contained 324 instances of which 324 had misclassifications. The Figures in 5.23 show that the discriminated cluster contained lower values for X_2, while there were few differences for X_1. Welch's t-test confirmed our observations, as there is no significant difference for X_1 ($t = -0.02, p = 0.98$) but there was a substantial result for X_2 with ($t = -35.59, p = 1.60e - 218$).

e-218



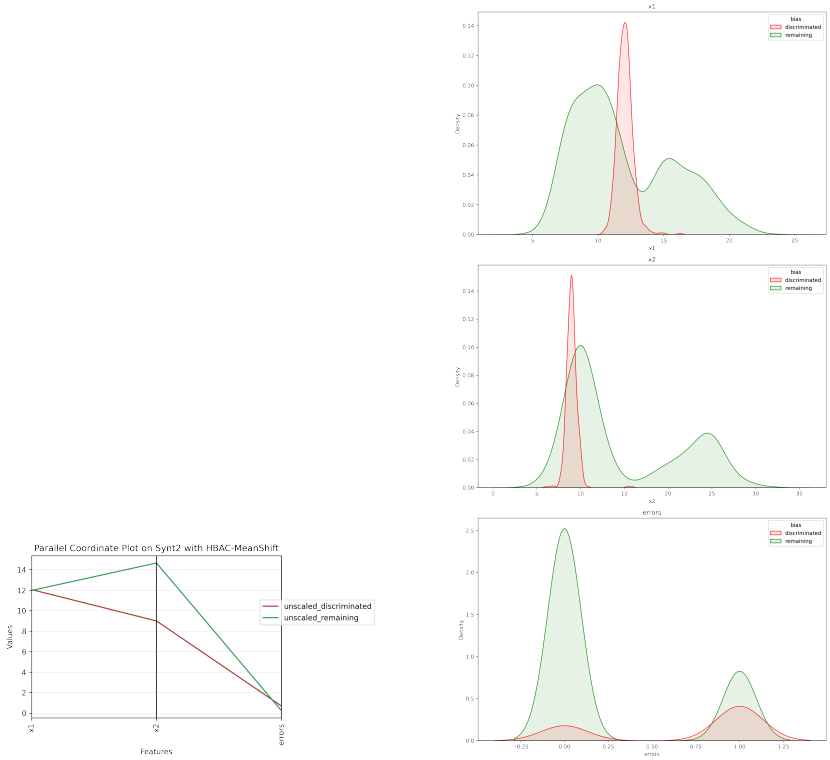
(a) Parallel Coordinate Plot (b) Distribution plots

Figure 5.23: Analysing the Discriminated and Remaining clusters retrieved from HBAC-DBSCAN

HBAC-MeanShift

Finally, we conducted several experiments with HBAC-MeanShift to find the cluster with the highest discrimination bias. With the estimated bandwidth of 5.296, the feature space was divided into two clusters, which are displayed in Figure B.10. Of these clusters, cluster 0 had the highest discrimination bias of -0.085. This group of instances, however, is uninformative: it contains 1702 instances, which is half of the dataset. Moreover, the error clusters were not identified, which led us to refine the bandwidth by decreasing the value to 3. The resulting plot shown in Figure 5.21 indicates that the algorithm performed better: first, it managed to find two of the three error clusters, and, second, it identified a discrimination bias of -0.451 for cluster 1.

When visually inspecting the parallel coordinate and distribution plots of the features displayed in Figure 5.24, we see that the largest difference can be found for the second feature, where the discriminated cluster has a narrower dispersion when compared with the remaining cluster. This is in accordance with how the error cluster was originally created (described in section 4.5.1, as its standard deviation was set to be smaller than the cluster it was placed in.



(a) Parallel Coordinate Plot (b) Distribution plots

Figure 5.24: Analysing the Discriminated and Remaining clusters retrieved from HBAC-MeanShift

Preliminary Conclusion

Contrarily to the previous results, HBAC-DBSCAN identified the cluster with the highest discrimination bias (-0.770) when compared to HBAC-KMeans (-0.465) and HBAC-MeanShift (-0.085). This result shows the applicability of DSCAN when a dataset contains clearly distinguishable data distributions.

5.2 Influence of Error Scaling

We experimented with scaling the error feature by applying HBAC-KMeans and HBAC-DBSCAN on the Synt2 dataset with error scaling factors ranging from 0 to 5. In the Figures below and in Appendix C, we see the results of increasing the influence of the error feature on the clustering space.

At scaling factor of 0 (displayed in Figure 5.25 and 5.28), the errors were not included as feature, which led the HBAC algorithm to only cluster based on the other two features. Even without the error feature, HBAC was able to locate two of the error clusters.

When increasing the scaling factor, we observe that more clusters are formed within the larger cluster, hereby dividing this large cluster into clusters containing either only errors, or no errors. Though the error feature is useful to find clusters of errors, the effects of the other features on the clustering space are negligible. Therefore, a too scaling factor, which we observe in Figures 5.27 and 5.30, yields uninformative clusters as they are now

primarily defined based whether they have errors or not, instead of also including the other features.

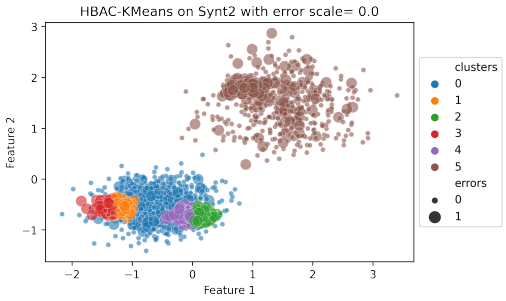


Figure 5.25: error scale 0.0

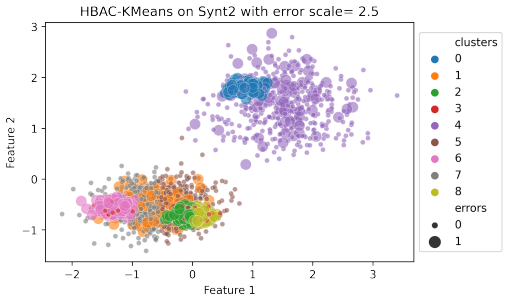


Figure 5.26: error scale 2.5

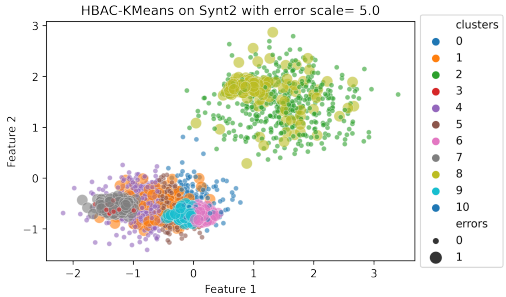


Figure 5.27: error scale 5.0

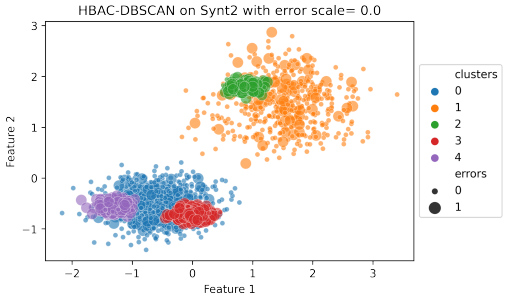


Figure 5.28: error scale 0.0

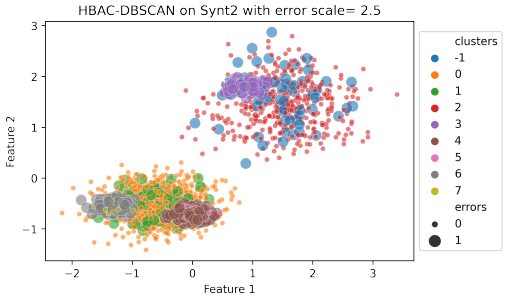


Figure 5.29: error scale 2.5

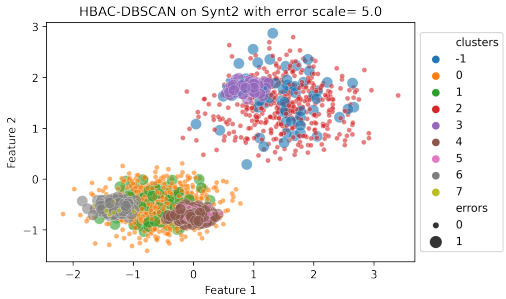


Figure 5.30: error scale 5.0

Figure 5.31 shows the results for the discriminated clusters for each error scaling factor.

We observe that for HBAC-KMeans, the highest discrimination bias achieved when the errors are scaled with 4, while the highest discrimination bias is already achieved at a scaling factor of 1 for HBAC-DBSCAN. The non-deterministic nature of HBAC-KMeans can partially explain the varying results of this algorithm. For HBAC-DBSCAN, we previously observed that this clustering method already performed well at identifying the error clusters in this dataset, which implies that the role of the error feature is less impactful for this clustering method.

Based on the obtained results, we conclude that scaling the errors with a high value leads to finding a higher discriminating bias, although there is a threshold after which HBAC-DBSCAN does not find a cluster with a higher bias. A scaling factor between 0.5 to 1 results in the best balance between including the errors as a clustering "guide", while not overpowering the other features in the clustering space.

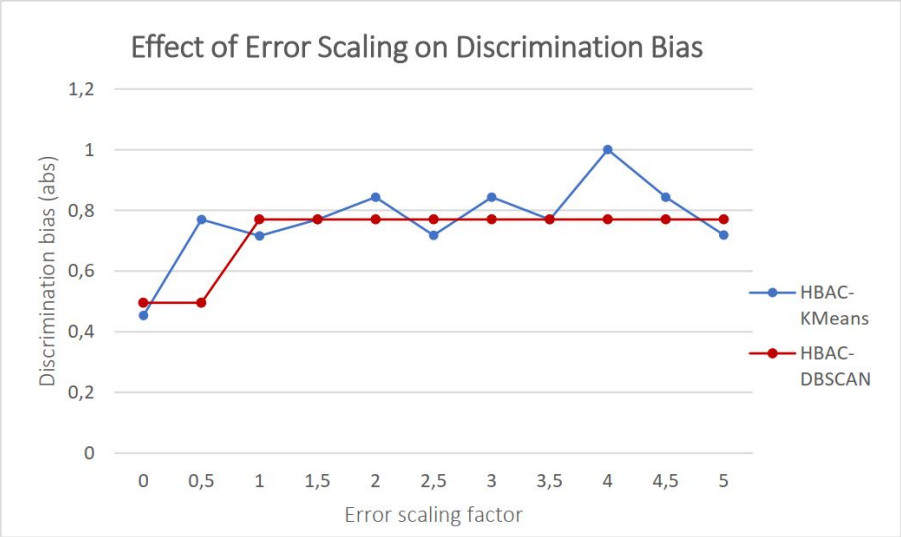


Figure 5.31: Discrimination bias per error scaling factor

5.3 Indirect Evaluation

Finally, for the last analysis we evaluate the HBAC algorithms on the properties mentioned in Table 4.4.

Scalability
HBAC-KMeans performed best on three of the four datasets, which contained varying densities and sizes. Additionally, this algorithm was the fastest when compared with DBSCAN and MeanShift.

Robustness
In terms of robustness, the density-based clusters yielded the same results after multiple runs, while the initial placement of the centroids in KMeans often affected which discrimination bias was found. Moreover, DBSCAN allowed a more refined "query" to find a discriminated cluster when compared with MeanShift and KMeans due to its two parameters.

Interpretability and meaningfulness of found clusters
This property was relatively difficult to assess. On real-world datasets, we observed that

K-Means found clusters with appropriate size, whereas DBSCAN and MeanShift found either too small or too large clusters, as their parameters were more difficult to tune. However, DBSCAN and MeanShift found more useful clusters for the synthetic dataset, which illustrates that they are more applicable for datasets that are normally distributed, as these clustering methods primarily form clusters based on densities.

Parameter tuning complexity
Although KMeans and MeanShift both contained a single parameter, KMeans was evidently simpler to tune. DBSCAN allowed to find more refined error clusters, which was illustrated by the results obtained on Synt2, but the K-Distance plot was often insufficient to find the highest discriminating bias. Moreover, MeanShift and DBSCAN required to iteratively decrease the parameters to find the discriminated clusters with HBAC.

Role of the Domain Expert
The role of the domain expert for HBAC-KMeans is to indicate how large the groups of interest should be. For HBAC-MeanShift, developers and domain experts could increase the error scaling factor to find which of the dense regions contains more errors. Using this information, HBAC-MeanShift could be applied to identify the discriminated clusters. Thus, more technical expertise is required for MeanShift when compared with KMeans. Lastly, for DBSCAN, the same holds as for MeanShift: more technical expertise is required to find the discriminated clusters. The domain experts and developers need to, for example, closely inspect how many samples each cluster should at least have (with *MinPts*) and how the *epsilon* should be iteratively decreased.

Thus, based on the found discriminating bias and the indirect evaluation, HBAC-KMeans performed better than HBAC-DBSCAN and HBAC-MeanShift as a fairness assessment tool.

Chapter 6

Discussion

In this chapter, we reflect on the experiments conducted in this study and their limitations. Finally, we provide suggestions for future work on the Hierarchical Bias-Aware Clustering algorithm.

6.1 Interpretation of Results

In this study, we developed three versions of the Hierarchical Bias-Aware Clustering algorithm using a different underlying clustering approaches HBAC-KMeans, HBAC-DBSCAN and HBAC-MeanShift. From these versions, HBAC-KMeans had the best performance: not only did it find the highest discriminating bias in three of the four datasets, it also scored highest in terms of scalability and understandability of the clusters. Particularly, its performance on the real-world datasets showed that this clustering method was the most suitable method when compared with HBAC-DBSCAN and HBAC-MeanShift.

Still, the usefulness of HBAC-DBSCAN and HBAC-MeanShift should not be overlooked, as they performed better on the more synthetic datasets where they were able to find the error clusters. These error clusters had a different distribution than the remaining datapoints in the other clusters, which was helpful for DBSCAN and MeanShift, as they mainly form clusters based on the differences in densities. Thus, the synthetic datasets provided us with new insights about what kind of datasets are suitable for which kind of clustering technique.

These findings, although preliminary, suggest that HBAC can support developers, civil servants and other stakeholders with further investigating how fair classification algorithms are in terms of the more hidden biases. Not only does HBAC help with finding bias, it also allows to design intuitive visualisations such as the density distributions that could be used to document about how fairness is examined in algorithms. This helps the Public Sector and private sector with becoming more open and transparent towards citizens about the presence of undesired bias in AI technologies. For example, the visualisations, tables and statistical tests could be used by governmental institutions to report publicly on how fairness is evaluated in systems such as Amsterdam's Algorithm Register.

The patterns within discriminated clusters could also help raise awareness for developers and civil servants about what kind of persons and groups might be affected by the classifier's errors. It is particularly this awareness that can help with mitigating undesired

bias, as the stakeholders now have more insights about the impact of their algorithm on the groups of persons that they previously might not have come up with.

Lastly, with HBAC, we offered a new approach of perceiving undesired bias. Though the detection of bias requires a multidisciplinary approach, which can include increasing the diversity in the AI development team or determining which attributes in the dataset contain sensitive information, we presented a bottom-up approach of discovering bias from the results. This approach gives insight into how complex problems in AI can be tackled with accessible information and techniques that have not been considered yet for the use-case of detecting undesired bias.

6.2 Limitations

During our research endeavours, we encountered several challenges and limitations.

First, as with all clustering techniques, HBAC created clusters based on distance metrics. These metrics require numerical attributes, while many datasets within our scope of interest contain sensitive *categorical* attributes. As described in the Methodology, we solved the problem of categorical attributes by transforming them to one-hot encodings, but one-hot encodings were more difficult to analyse and visualise after obtaining the results.

Another limitation of HBAC is its sensitivity to feature normalization and scaling. Although this sensitivity is often inherent to clustering models, it implies that developers working with this bias detection tool should be attentive to how to preprocess their dataset, as this is essential for HBAC's ability to find undesired bias in algorithms and form meaningful clusters. Besides the feature scaling, HBAC was also sensitive to the clustering-specific parameters such as ϵ for HBAC-DBSCAN and the bandwidth for HBAC-MeanShift. These algorithms performed well on the synthetic datasets, but their performance was much worse on the real-world datasets, where they were also more difficult to tune. A solution for HBAC-DBSCAN would be to use HDBSCAN instead: this density-based algorithm considers varying densities rather a single density (Campello, Moulavi, & Sander, 2013). Moreover, HDBDSCAN only requires one parameter (*minimum cluster size*) rather than two for DBSCAN, which also adds to the usability of the HBAC with this clustering method.

Additionally, a limitation that could be addressed in further work are the statistical tests that we used to evaluate whether the difference in means between the discriminated and remaining clusters was significant. In this our experiments, we used Welch's t-test for Unequal Variance. This statistical test not only holds certain assumptions, but its validity also depends on the cluster size, which was varying for each dataset. For instance, for the real-world datasets, the discriminated clusters were fairly small. Small samples increase the likelihood that significant results are found, as small samples are more likely to be unusual. Other peculiar results were found for COMPAS, where Welch's test showed many significant results due to the one-hot encoded features in the dataset. Though we benefited from Welch's t-test in that it gave pointers about which features required closer inspection with a density distribution visualisation, the potential of statistical tests in HBAC remained largely unexplored in this study.

Finally, while we wanted to minimize the role of the domain expert in addressing fairness, this person is still needed to specify the desired properties of a discriminated cluster, such

as the size of the cluster for KMeans. Although the size of the clusters is not specifically required for DBSCAN and MeanShift, their parameters were required to create a "window" that defined the cluster size. Therefore, some tasks within HBAC turned out to be more labour-intensive than anticipated.

6.3 Future Work

The work presented in this study shows many research strands that could be explored in future research. These strands can be categorised into experimenting with results obtained from regression algorithms, using new evaluation metrics, generating different error and data distributions, removing the clustering hierarchy, and enhancing the interpretability of the HBAC results.

A first exploration would be to apply HBAC on the output of regression algorithms. This approach requires using an error metrics suitable for regression algorithms, such as using Mean Squared Errors (MSE) rather than Accuracy. By applying HBAC on regression algorithms, we can use HBAC to identify bias in a wider range of algorithms, which increases the usability and generalisability of this fairness assessment tool.

Another interesting research would be to experiment with multiple error metrics rather than using only accuracy. We used accuracy due to its simplicity, but it would be interesting to form clusters using the F1-score, Precision, or Recall, since this provides domain experts and developers with more options to discover bias from their classification algorithms. Additionally, we could also conduct a classifier with a feature interpretability method (e.g SHAP or LIME) on a dataset to both generate a more realistic error distribution and to retrieve which features played a role into the incorrect classifications

Further work could also be undertaken on the training set of the algorithms: for example, we could conduct HBAC on the training set rather than the test set, after which we could compare the train set results with those of the test set. The advantage of this method is that the training set has much more instances than the test set, and the results of HBAC could then be used to tune the parameters on the validation set or to more precisely investigate the errors. Additionally, the scaling of the errors requires substantially more research, as we observed from 5.2 that the effect of scaling this feature had a large impact on the performance of HBAC. Since we only tested this on one synthetic dataset, the error scaling experiment could also be conducted on real-world datasets.

Synthetic datasets could still be further explored in future works: we could use these to further analyse how the Bias-Aware Clustering algorithms form clusters on non-spheric or non-linear data. The synthetic datasets in this study only comprised normal distributions, where the errors were also normally distributed. This largely contributed to the good performance of the density-based HBAC algorithms, but real world datasets are often more "messy".

In addition, while we did not investigate the favoured clusters, i.e., the clusters with persons for whom the classifier produced substantially less errors than average, comparing the discriminated and favoured clusters may also provide interesting insights about how the classifier performs differently for different groups. For example, this would have especially interesting for the real-world datasets German Credit or COMPAS datasets.

Lastly, another research strand would be to investigate the hierarchical clustering order in

HBAC. We would primarily be interested in whether the cluster hierarchy and splitting mechanism helps the end-user with interpreting the cluster results. In other words, we are curious to find out how the hierarchical structure in the bias-aware clustering algorithm adds to the explainability and interpretability of the model.

Chapter 7

Conclusion

With this study, we proposed an algorithm to locate clusters of bias from the results of classification algorithms. We introduced the bias detection method *Hierarchical Bias-Aware Clustering* and experimented with three clustering techniques of HBAC to establish which of these versions performed best on finding the highest discriminating bias. The HBAC algorithm was applied on the test set, where we also included the classification errors as attribute. Additionally, we investigated the role of the error feature on HBAC's performance. The evaluation of these algorithms comprised a visual inspection of the differences between the discriminated and remaining cluster and a statistical test to find significant results between the means of the discriminated cluster. Furthermore, we conducted an indirect evalution to assess the HBAC's applicability to the intended context of detecting bias.

With our experiments, we aimed to provide answers to the following research questions:

RQ1: How can we use clustering algorithms to automatically detect clusters of error?
With HBAC, we created a clustering model that uses an error metric to calculate the discriminating bias for a given cluster when compared to the other clusters in the dataset. Once a cluster has a higher bias than the remaining clusters, it splits this particular cluster into smaller clusters until it finds the highest discriminating bias.

RQ2: What kind of clustering algorithm is most suitable to automatically detect clusters of error?
Based on results, we conclude that HBAC-KMeans found the highest discriminating bias in clusters when compared to HBAC-DBSCAN and HBAC-MeanShift. Additionally, during indirect evaluation, this clustering algorithm scored highest in terms of scalability and interpretability. However, for datasets with "clear" normal distributions, HBAC-DBSCAN would be a better option, as the parameters of this method provide more options to locate clusters of error.

RQ3: How can the clusters of error be interpreted by public servants and other stakeholders and assessors?
The clusters of error can guide public servants, developers and other stakeholders in finding groups for which a classification algorithm is systematically underperforming. Through visualisations such as parallel coordinate plots and density distributions, stakeholders can gain insight in which group(s) is/are negatively impacted by the classification

algorithm. Depending on the level of found discriminating bias and the context in which the classification algorithm, actions should be undertaken to communicate and mitigate this bias.

RQ4: How can such clustering algorithms for bias discovery complement the existing fairness assessment tools?
Existing fairness assessment tools often address bias caused by sensitive attributes, but in many of these tools, these sensitive attributes need to be pre-specified. With HBAC, we provided a fairness assessment tool that eliminates the need for *a priori* information. As the clustering method is applied on the test results of a classification algorithm, developers can use this tool to further investigate patterns of error. Together with civil servants and other stakeholders, the developers can describe the characteristics of the persons in the discriminated cluster. Thus, HBAC and the analysis of the discriminated cluster complement the existing fairness assessment methods by automatically detecting and describing patterns of error that indicate undesired bias. This information can be used to mitigate this bias, possibly with other fairness assessment tools.

References

Agrawal, A. (2019). Removing Bias in AI isn't enough, it must take intersectionality into account. , 1–5. Retrieved from https://atibhiagrawal.medium.com/removing-bias-in-ai-isnt-enough-it-must-take-intersectionality-into-account-e5e92e76233c

Algemene Rekenkamer. (2021). *Toetsingskader | Algoritmes | Algemene Rekenkamer*. Retrieved 2021-08-02, from https://www.rekenkamer.nl/onderwerpen/algoritmes/algoritmes-toetsingskader/

Algorithmic accountability for the public sector | Ada Lovelace Institute. (n.d.). Retrieved 2021-06-15, from https://www.adalovelaceinstitute.org/project/algorithmic-accountability-public-sector/

Angwin, J., Larson, J., Mattu, S., & Kirchner, L. (2016). *Machine Bias — ProPublica*. Retrieved 2021-07-16, from https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

Arthur, D., & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the annual acm-siam symposium on discrete algorithms* (Vol. 07-09-Janu, pp. 1027–1035).

Arvai, K. (2020). *K-Means Clustering in Python: A Practical Guide – Real Python*. Retrieved 2021-08-04, from https://realpython.com/k-means-clustering-python/{#}overview-of-clustering-techniqueshttps://realpython.com/k-means-clustering-python/{%}0Ahttps://realpython.com/k-means-clustering-python/{#}how-to-build-a-k-means-clustering-pipeline-in-python

Autoriteit Persoonsgegevens. (2018). *Belastingdienst/Toeslagen: De Verwerking van de Nationaliteit van Aanvragers van Kinderopvangtoeslag* (Tech. Rep.). Retrieved from https://autoriteitpersoonsgegevens.nl/sites/default/files/atoms/files/onderzoek{_}belastingdienst{_}kinderopvangtoeslag.pdf

Barocas, S., Hardt, M., & Narayanan, A. (2019). *Fairness and Machine Learning*. Fairmlbook.org. Retrieved from https://fairmlbook.org/

Barocas, S., & Selbst, A. (2016). Big Data's Disparate Impact. *California Law Review*, *104*(3), 671. Retrieved from http://dx.doi.org/10.15779/Z38BG31 doi: 10.15779/Z38BG31

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., ... Herrera, F. (2020, jun). Explainable Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges

toward responsible AI. *Information Fusion*, *58*, 82–115. doi: 10.1016/j.inffus.2019.12.012

Breiman, L. (2001, oct). Random forests. *Machine Learning*, *45*(1), 5–32. Retrieved from https://link-springer-com.vu-nl.idm.oclc.org/article/10.1023/A:1010933404324 doi: 10.1023/A:1010933404324

Buolamwini, J. (2018, jan). *Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification* * (Vol. 81; Tech. Rep.). Retrieved from http://proceedings.mlr.press/v81/buolamwini18a.html

Calders, T., & Verwer, S. (2010). Three naive Bayes approaches for discrimination-free classification. In *Data mining and knowledge discovery* (Vol. 21, pp. 277–292). doi: 10.1007/s10618-010-0190-x

Campello, R. J., Moulavi, D., & Sander, J. (2013). Density-based clustering based on hierarchical density estimates. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 7819 LNAI, pp. 160–172). Springer, Berlin, Heidelberg. Retrieved from https://link.springer.com/chapter/10.1007/978-3-642-37456-2{_}14 doi: 10.1007/978-3-642-37456-2_14

Chouldechova, A. (2017, jun). Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments. *Big Data*, *5*(2), 153–163. Retrieved from https://www.liebertpub.com/doi/abs/10.1089/big.2016.0047 doi: 10.1089/big.2016.0047

Corbett-Davies, S., & Goel, S. (2018, jul). *The measure and mismeasure of fairness: A critical review of fair machine learning.* arXiv. Retrieved from http://arxiv.org/abs/1808.00023

Derpanis, K. G. (2005). Mean Shift Clustering. *Computer*, *1*(x), 1–3. Retrieved from http://www.cse.yorku.ca/{~}kosta/CompVis{_}Notes/mean{_}shift.pdf

de Sousa, W. G., de Melo, E. R. P., Bermejo, P. H. D. S., Farias, R. A. S., & Gomes, A. O. (2019). *How and where is artificial intelligence in the public sector going? A literature review and research agenda* (Vol. 36) (No. 4). Retrieved from https://doi.org/10.1016/j.giq.2019.07.004 doi: 10.1016/j.giq.2019.07.004

Dwork, C., Hardt, M., Pitassi, T., Reingold, O., & Zemel, R. (2012). Fairness through awareness. In *Itcs 2012 - innovations in theoretical computer science conference* (pp. 214–226). doi: 10.1145/2090236.2090255

Edsall, R. M. (2003). The parallel coordinate plot in action: Design and use for geographic visualization. *Computational Statistics and Data Analysis*, *43*(4), 605–619. Retrieved from www.elsevier.com/locate/csda doi: 10.1016/S0167-9473(02)00295-5

Elsevier. (2018). *Artificial Intelligence: How knowledge is created, transferred, and used* (Tech. Rep.). Retrieved from https://www.elsevier.com/connect/ai-resource-center

Feldman, R., & Sanger, J. (2006). *The Text Mining Handbook*. doi: 10.1017/cbo9780511546914

Ferreira, L. (2017). *Predicting Credit Risk - Model Pipeline | Kaggle*. Retrieved 2021-07-06, from https://www.kaggle.com/kabure/predicting-credit

-risk-model-pipeline

Fjeld, J., Achten, N., Hilligoss, H., Nagy, A., & Srikumar, M. (2020). Principled Artificial Intelligence: Mapping Consensus in Ethical and Rights-based Approaches to Principles for AI. *Berkman Klein Center for Internet & Society*, *9860*, 60. Retrieved from http://nrs.harvard.edu/urn-3:HUL.InstRepos:42160420{%}0AThis

Friedler, S. A., Choudhary, S., Scheidegger, C., Hamilton, E. P., Venkatasubramanian, S., & Roth, D. (2019, jan). A comparative study of fairness-enhancing interventions in machine learning. In *Fat* 2019 - proceedings of the 2019 conference on fairness, accountability, and transparency* (pp. 329–338). Association for Computing Machinery, Inc. Retrieved from https://doi.org/10.1145/3287560.3287589 doi: 10.1145/3287560.3287589

Gates, S. W., Perry, V. G., & Zorn, P. M. (2002). Automated underwriting in mortgage lending: Good news for the underserved? *Housing Policy Debate*, *13*(2), 369–391. Retrieved from https://www.tandfonline.com/doi/abs/10.1080/10511482.2002.9521447 doi: 10.1080/10511482.2002.9521447

Geiger, G. (2021). *How a discriminatory algorithm wrongly accused thousands of families of fraud*. Retrieved 2021-03-05, from https://www.vice.com/en/article/jgq35d/how-a-discriminatory-algorithm-wrongly-accused-thousands-of-families-of-fraud

Gemeente Amsterdam. (2021). *Amsterdam Algoritmeregister*. Retrieved 2021-08-02, from https://algoritmeregister.amsterdam.nl/en/ai-register/ https://algoritmeregister.amsterdam.nl/

Government of the Netherlands. (2018). *Municipalities' tasks | Municipalities | Government.nl*. Retrieved 2021-06-07, from https://www.government.nl/topics/municipalities/municipalities-tasks

Hao, K. (2019). This is how AI bias really happens—and why it's so hard to fix - MIT Technology Review. *MIT Technology Review*, 1–25. Retrieved from https://www.technologyreview.com/2019/02/04/137602/this-is-how-ai-bias-really-happensand-why-its-so-hard-to-fix/ https://www.technologyreview.com/s/612876/this-is-how-ai-bias-really-happensand-why-its-so-hard-to-fix/

Hardt, M., Price, E., & Srebro, N. (2016). Equality of opportunity in supervised learning. In *Advances in neural information processing systems* (Vol. 29, pp. 3323–3331).

Hofmann, H. (1994). *UCI Machine Learning Repository: Statlog (German Credit Data) Data Set*. Retrieved 2021-07-06, from https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

IBM. (2020). *What Is Machine Learning?* Retrieved 2021-08-04, from https://www.ibm.com/in-en/cloud/learn/machine-learning doi: 10.1007/978-1-4842-3799-1_3

Kamishima, T., Akaho, S., Asoh, H., & Sakuma, J. (2012). Fairness-aware classifier with prejudice remover regularizer. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 7524 LNAI, pp. 35–50). Springer, Berlin, Heidelberg. Retrieved from https://link-springer-com.vu-nl.idm.oclc

.org/chapter/10.1007/978-3-642-33486-3{_}3 doi: 10.1007/978-3-642-33486-3_3

Khosla, S. (2021). *ML | K-means++ Algorithm - GeeksforGeeks.* Retrieved 2021-06-28, from https://www.geeksforgeeks.org/ml-k-means-algorithm/

Kim, M. P., Reingold, O., & Rothblum, G. N. (2018). Fairness through computationally-bounded awareness. In *Advances in neural information processing systems* (Vol. 2018-Decem, pp. 4842–4852).

Köchling, A., & Wehner, M. C. (2020, nov). Discriminated by an algorithm: a systematic review of discrimination and fairness by algorithmic decision-making in the context of HR recruitment and HR development. *Business Research*, *13*(3), 795–848. Retrieved from https://doi.org/10.1007/s40685-020-00134-w doi: 10.1007/s40685-020-00134-w

Kusner, M., Loftus, J., Russell, C., & Silva, R. (2017, mar). Counterfactual fairness. In *Advances in neural information processing systems* (Vol. 2017-Decem, pp. 4067–4077). Neural information processing systems foundation. Retrieved from https://arxiv.org/abs/1703.06856v3

Kuziemski, M., & Misuraca, G. (2020, jul). AI governance in the public sector: Three tales from the frontiers of automated decision-making in democratic settings. *Telecommunications Policy*, *44*(6), 101976. doi: 10.1016/j.telpol.2020.101976

Luft, J., & Ingham, H. (1961). The Johari Window: a graphic model of awareness in interpersonal relations. *Human relations training news*, *5*(9), 6–7.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2019). *A survey on bias and fairness in machine learning.* Retrieved from https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

Misztal-Radecka, J., & Indurkhya, B. (2021, may). Bias-Aware Hierarchical Clustering for detecting the discriminated groups of users in recommendation systems. *Information Processing and Management*, *58*(3), 102519. doi: 10.1016/j.ipm.2021.102519

Morley, J., Elhalal, A., Garcia, F., Kinsey, L., Mökander, J., & Floridi, L. (2018). Ethics as a service: a pragmatic operationalisation of AI Ethics. *Journal of Materials Processing Technology*, *1*(1), 1–8. Retrieved from http://dx.doi.org/10.1016/j.cirp.2016.06.001{%}0Ahttp://dx.doi.org/10.1016/j.powtec.2016.12.055{%}0Ahttps://doi.org/10.1016/j.ijfatigue.2019.02.006{%}0Ahttps://doi.org/10.1016/j.matlet.2019.04.024{%}0Ahttps://doi.org/10.1016/j.matlet.2019.127252{%}0Ahttp://dx.doi.o

Nasiriani, N., Squicciarini, A., Saldanha, Z., Goel, S., & Zannone, N. (2019). Hierarchical clustering for discrimination discovery: A top-down approach. In *Proceedings - ieee 2nd international conference on artificial intelligence and knowledge engineering, aike 2019* (pp. 187–194). Retrieved from https://www.researchgate.net/publication/335078000 doi: 10.1109/AIKE.2019.00041

Ofer, D. (2017). *COMPAS Recidivism Racial Bias | Kaggle.* Retrieved 2021-07-25, from https://www.kaggle.com/danofer/compass

Oxford English Dictionary. (2012). *Officership | Definition of Officership by Oxford Dictionary on Lexico.com also meaning of Officership*. Retrieved 2021-05-16, from https://www.lexico.com/definition/fairnesshttps://www.lexico.com/definition/officership

Rahmah, N., & Sitanggang, I. S. (2016, jan). Determination of Optimal Epsilon (Eps) Value on DBSCAN Algorithm to Clustering Data on Peatland Hotspots in Sumatra. In *Iop conference series: Earth and environmental science* (Vol. 31, p. 012012). IOP Publishing. Retrieved from https://iopscience.iop.org/article/10.1088/1755-1315/31/1/012012https://iopscience.iop.org/article/10.1088/1755-1315/31/1/012012/meta doi: 10.1088/1755-1315/31/1/012012

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (2017, jul). DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems*, *42*(3). Retrieved from https://dl.acm.org/doi/abs/10.1145/3068335 doi: 10.1145/3068335

Scikit-learn. (n.d.). *2.3. Clustering — scikit-learn 0.24.2 documentation*. Retrieved 2021-08-04, from https://scikit-learn.org/stable/modules/clustering.html{#}dbscan

Serra, A., & Tagliaferri, R. (2018). Unsupervised learning: Clustering. In *Encyclopedia of bioinformatics and computational biology: Abc of bioinformatics* (Vol. 1-3, pp. 350–357). doi: 10.1016/B978-0-12-809633-8.20487-1

Silberg, J., & Manyika, J. (2019). Notes from the AI frontier: Tackling bias in artificial intelligence (and in humans). *Mckinsey Global Institute*, 1–8. Retrieved from https://www.mckinsey.com/featured-insights/artificial-intelligence/tackling-bias-in-artificial-intelligence-and-in-humans{#}

The United Nations. (1948). *Universal Declaration of Human Rights*.

Van Aalst, R., Leijten, R., Belhaj, S., van Dam, C., Kuiken, A., Van der Lee, T., ... Van Kooten-Arissen, F. (2021). *Parlementaire Ondervraging Kinderopvangtoeslag* (Tech. Rep.).

Van Der Maaten, L., Postma, E., & Van Den Herik, J. (2009). *Tilburg centre for Creative Computing Dimensionality Reduction: A Comparative Review* (Tech. Rep.). Retrieved from http://www.uvt.nl/ticc

Veale, M., & Binns, R. (2017, dec). Fairer machine learning in the real world: Mitigating discrimination without collecting sensitive data. *Big Data and Society*, *4*(2), 205395171774353. Retrieved from http://journals.sagepub.com/doi/10.1177/2053951717743530 doi: 10.1177/2053951717743530

Verma, S., & Rubin, J. (2018). Fairness definitions explained. In *Proceedings - international conference on software engineering* (Vol. 18, pp. 1–7). ACM. Retrieved from https://doi.org/10.1145/3194770.3194776 doi: 10.1145/3194770.3194776

Virmani, D., Taneja, S., & Malhotra, G. (2015). Normalization based K means Clustering Algorithm. Retrieved from http://arxiv.org/abs/1503.00900

Xu, D., & Tian, Y. (2015, jun). A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science*, *2*(2), 165–193. Retrieved from https://link-springer-com.vu-nl.idm.oclc.org/article/10.1007/

s40745-015-0040-1 doi: 10.1007/s40745-015-0040-1

Yeomans, C., & Xu, J. (2020). *2 Clustering | Introduction to Machine Learning*. Retrieved 2021-07-01, from https://exeter-data-analytics.github.io/MachineLearning/clustering.html

Zafar, M. B., Valera, I., Rodriguez, M. G., & Gummadi, K. P. (2017). Fairness constraints: Mechanisms for fair classification. In *Proceedings of the 20th international conference on artificial intelligence and statistics, aistats 2017*.

Appendix A

Classification Evaluation Metrics

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN}$$

$$Sensitivity = Recall = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{FP + TN}$$

Appendix B

Supplementary Results

B.1 German Credit

B.1.1 HBAC-KMeans

Table B.1: HBAC-KMeans: Cluster differences in German Credit

	Unscaled discriminated	**Unscaled remaining**	**Difference**
Age	31.571429	35.234568	-3.663139
Job	1.571429	1.870370	-0.298942
Credit amount	5338.000000	3098.135802	2239.864198
Duration	23.571429	20.197531	3.373898
Purpose_car	0.000000	0.277778	-0.277778
Purpose_domestic appliances	0.000000	0.006173	-0.006173
Purpose_education	0.857143	0.024691	0.832451
Purpose_furniture/equipment	0.000000	0.240741	-0.240741
Purpose_radio/TV	0.000000	0.290123	-0.290123
Purpose_repairs	0.142857	0.006173	0.136684
Purpose_vacation/others	0.000000	0.012346	-0.012346
Sex_male	0.714286	0.722222	-0.007937
Housing_own	0.857143	0.746914	0.110229
Housing_rent	0.000000	0.179012	-0.179012
Savings_moderate	0.142857	0.092593	0.050265
Savings_no_inf	0.142857	0.191358	-0.048501
Savings_quite rich	0.142857	0.067901	0.074956
Savings_rich	0.000000	0.067901	-0.067901
Check_moderate	0.000000	0.314815	-0.314815
Check_no_inf	0.571429	0.382716	0.188713
Check_rich	0.000000	0.061728	-0.061728
Age_cat_Young	1.000000	0.364198	0.635802
Age_cat_Adult	0.000000	0.382716	-0.382716
Age_cat_Senior	0.000000	0.037037	-0.037037
predicted_class	0.285714	0.172840	0.112875
true_class	0.571429	0.253086	0.318342
errors	0.857143	0.351852	0.505291

B.1.2 HBAC-DBSCAN

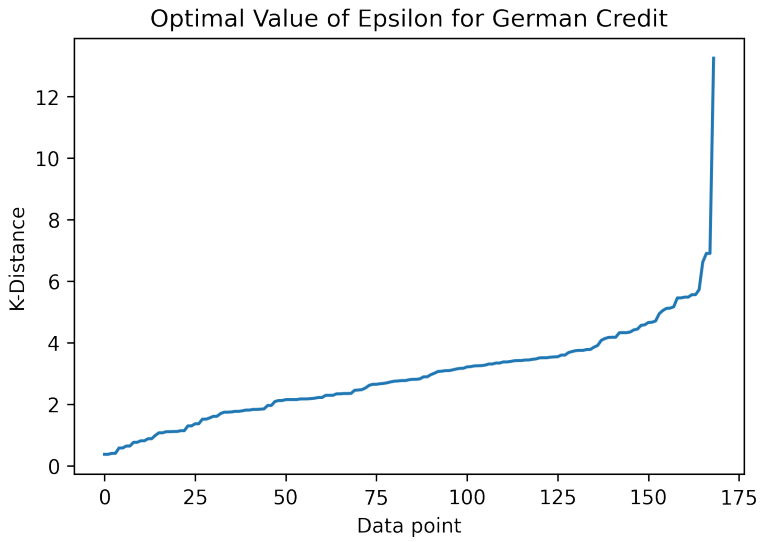


Figure B.1: K-Distance plot for German Credit to determine the epsilon of DBSCAN

Table B.2: HBAC-DBSCAN: Cluster differences in German Credit

	Unscaled discriminated	**Unscaled remaining**	**Difference**
Age	64.6	34.182927	30.417073
Job	2.0	1.853659	0.146341
Credit amount	2538.2	3210.810976	-672.610976
Duration	24.0	20.225610	3.774390
Purpose_car	0.0	0.274390	-0.274390
Purpose_domestic appliances	0.0	0.006098	-0.006098
Purpose_education	0.2	0.054878	0.145122
Purpose_furniture/equipment	0.0	0.237805	-0.237805
Purpose_radio/TV	0.4	0.274390	0.125610
Purpose_repairs	0.0	0.012195	-0.012195
Purpose_vacation/others	0.0	0.012195	-0.012195
Sex_male	0.8	0.719512	0.080488
Housing_own	1.0	0.743902	0.256098
Housing_rent	0.0	0.176829	-0.176829
Savings_moderate	0.0	0.097561	-0.097561
Savings_no_inf	0.4	0.182927	0.217073
Savings_quite rich	0.2	0.067073	0.132927
Savings_rich	0.0	0.067073	-0.067073
Check_moderate	0.2	0.304878	-0.104878
Check_no_inf	0.2	0.396341	-0.196341
Check_rich	0.2	0.054878	0.145122
Age_cat_Young	0.0	0.402439	-0.402439
Age_cat_Adult	0.0	0.378049	-0.378049
Age_cat_Senior	1.0	0.006098	0.993902
predicted_class	0.0	0.182927	-0.182927
true_class	0.6	0.256098	0.343902
errors	0.6	0.365854	0.234146

B.1.3 HBAC-MeanShift

Table B.3: HBAC-MeanShift: Cluster differences in German Credit

	Unscaled discriminated	**Unscaled remaining**	**Difference**
Age	64.000000	34.018405	29.981595
Job	1.833333	1.858896	-0.025562
Credit amount	2576.333333	3213.533742	-637.200409
Duration	23.500000	20.220859	3.279141
Purpose_car	0.000000	0.276074	-0.276074
Purpose_domestic appliances	0.000000	0.006135	-0.006135
Purpose_education	0.166667	0.055215	0.111452
Purpose_furniture/equipment	0.000000	0.239264	-0.239264
Purpose_radio/TV	0.333333	0.276074	0.057260
Purpose_repairs	0.000000	0.012270	-0.012270
Purpose_vacation/others	0.000000	0.012270	-0.012270
Sex_male	0.833333	0.717791	0.115542
Housing_own	0.833333	0.748466	0.084867
Housing_rent	0.166667	0.171779	-0.005112
Savings_moderate	0.166667	0.092025	0.074642
Savings_no_inf	0.333333	0.184049	0.149284
Savings_quite rich	0.166667	0.067485	0.099182
Savings_rich	0.000000	0.067485	-0.067485
Check_moderate	0.333333	0.300613	0.032720
Check_no_inf	0.166667	0.398773	-0.232106
Check_rich	0.166667	0.055215	0.111452
Age_cat_Young	0.000000	0.404908	-0.404908
Age_cat_Adult	0.000000	0.380368	-0.380368
Age_cat_Senior	1.000000	0.000000	1.000000
predicted_class	0.000000	0.184049	-0.184049
true_class	0.666667	0.251534	0.415133
errors	0.666667	0.361963	0.304703

B.2 COMPAS

B.2.1 HBAC-KMeans

Table B.4: HBAC-KMeans: Cluster differences in COMPAS

	Unscaled Discriminated	Unscaled Remaining	Difference
Number_of_Priors	0.953125	3.356823	-2.403698
score_factor	0.812500	0.427852	0.384648
Age_Above_FourtyFive	0.000000	0.205817	-0.205817
Age_Below_TwentyFive	1.000000	0.189038	0.810962
African_American	0.781250	0.506152	0.275098
Asian	0.000000	0.004474	-0.004474
Hispanic	0.015625	0.086689	-0.071064
Native_American	0.000000	0.001678	-0.001678
Other	0.000000	0.052573	-0.052573
Female	1.000000	0.156040	0.843960
Misdemeanor	0.390625	0.359620	0.031005
predicted_class	0.281250	0.418904	-0.137654
true_class	0.437500	0.455817	-0.018317
errors	0.375000	0.325503	0.049497

B.2.2 HBAC-DBSCAN

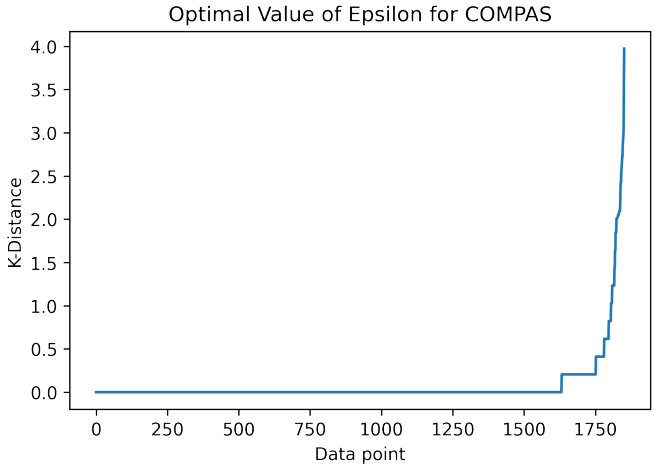


Figure B.2: K-Distance plot for COMPAS to determine the epsilon of DBSCAN

Table B.5: HBAC-DBSCAN: Cluster differences in COMPAS

	Unscaled discriminated	**Unscaled remaining**	**Difference**
Number_of_Priors	1.666667	3.358727	-1.692060
score_factor	0.172043	0.455372	-0.283329
Age_Above_FourtyFive	0.204301	0.198408	0.005893
Age_Below_TwentyFive	0.225806	0.216600	0.009206
African_American	0.000000	0.542922	-0.542922
Asian	0.000000	0.004548	-0.004548
Hispanic	0.000000	0.088687	-0.088687
Native_American	0.000000	0.001706	-0.001706
Other	1.000000	0.000569	0.999431
Female	0.161290	0.186470	-0.025179
Misdemeanor	0.387097	0.359295	0.027802
predicted_class	0.193548	0.425810	-0.232262
true_class	0.376344	0.459352	-0.083008
errors	0.354839	0.325753	0.029085

B.2.3 HBAC-MeanShift

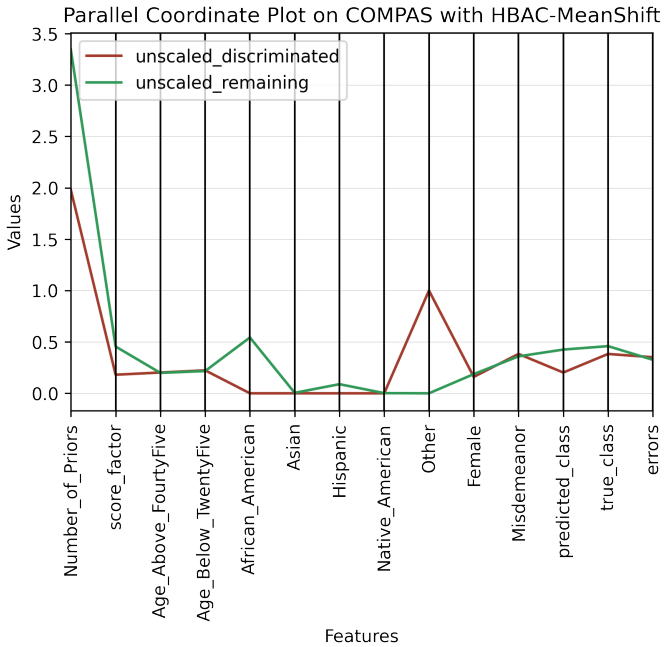


Figure B.3: HBAC-MeanShift: Parallel Coordinate Plot of COMPAS

Table B.6: HBAC-MeanShift: Cluster differences in COMPAS

	Unscaled discriminated	Unscaled remaining	Difference
Number_of_Priors	1.978723	3.343003	-1.364280
score_factor	0.180851	0.455063	-0.274212
Age_Above_FourtyFive	0.202128	0.198521	0.003607
Age_Below_TwentyFive	0.223404	0.216724	0.006681
African_American	0.000000	0.543231	-0.543231
Asian	0.000000	0.004551	-0.004551
Hispanic	0.000000	0.088737	-0.088737
Native_American	0.000000	0.001706	-0.001706
Other	1.000000	0.000000	1.000000
Female	0.159574	0.186576	-0.027001
Misdemeanor	0.382979	0.359499	0.023479
predicted_class	0.202128	0.425484	-0.223356
true_class	0.382979	0.459044	-0.076066
errors	0.351064	0.325939	0.025125

B.3 Synt1

B.3.1 HBAC-KMeans

	Unscaled discriminated	**Unscaled remaining**	**Difference**
x1	7.690947	4.936133	2.754815
x2	6.783615	6.050108	0.733507
errors	0.244604	0.207002	0.037603
class	0.690647	1.561644	-0.870996

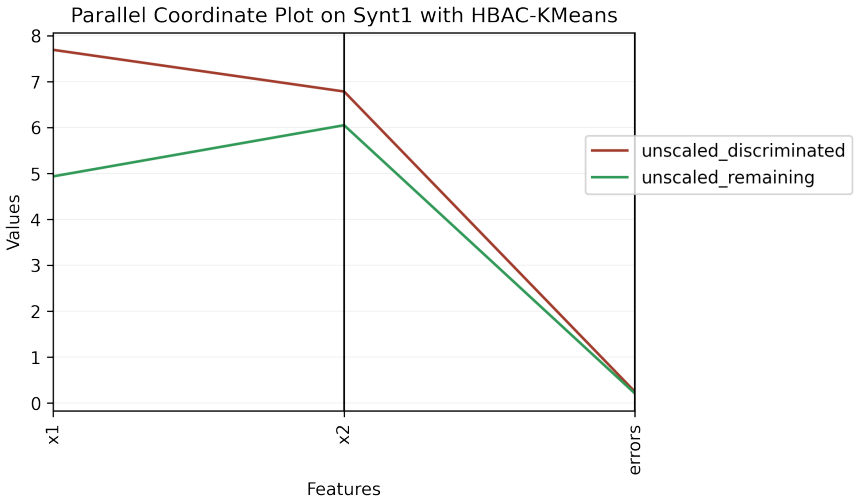


Figure B.4: HBAC-KMeans: Parallel Coordinate Plot of Synt1

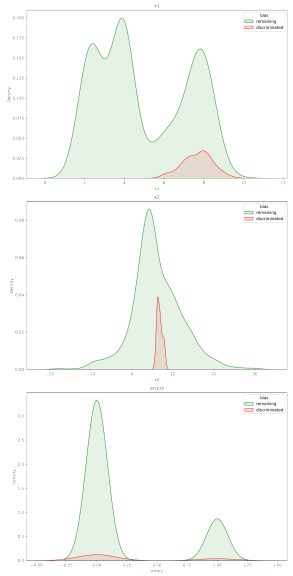


Figure B.5: HBAC-KMeans: Density Distributions in Synt1

B.3.2 HBAC-DBSCAN

Table B.7: HBAC-DBSCAN: Cluster differences in Synt1

	Unscaled discriminated	**Unscaled remaining**	**Difference**
x1	7.957131	4.279177	3.677954
x2	4.169134	6.668098	-2.498964
errors	0.189189	0.215470	-0.026280
class	0.000000	1.948435	-1.948435

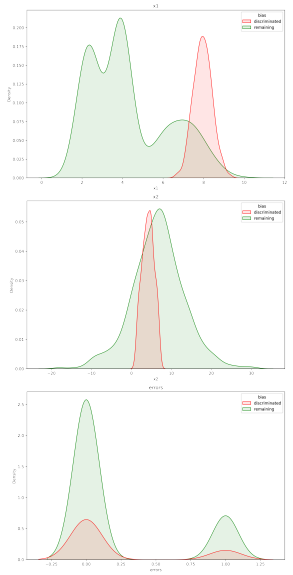


Figure B.6: HBAC-DBSCAN: Density Distributions of Synt1

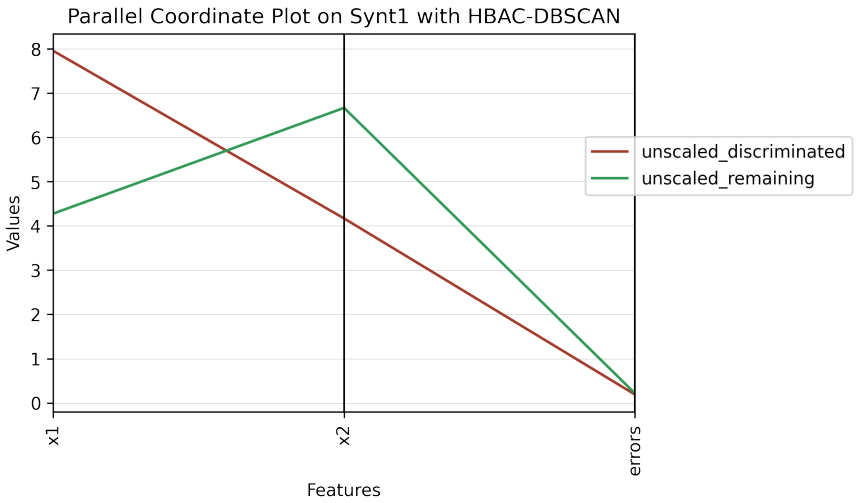


Figure B.7: HBAC-DBSCAN: Parallel Coordinate Plot of Synt1

B.3.3 HBAC-MeansShift

Table B.8: HBAC-MeanShift: Cluster differences in Synt1

	Unscaled discriminated	**Unscaled remaining**	**Difference**
x1	8.495712	5.106367	3.389345
x2	-0.622626	6.120800	-6.743426
errors	0.285714	0.209225	0.076489
class	2.000000	1.502615	0.497385

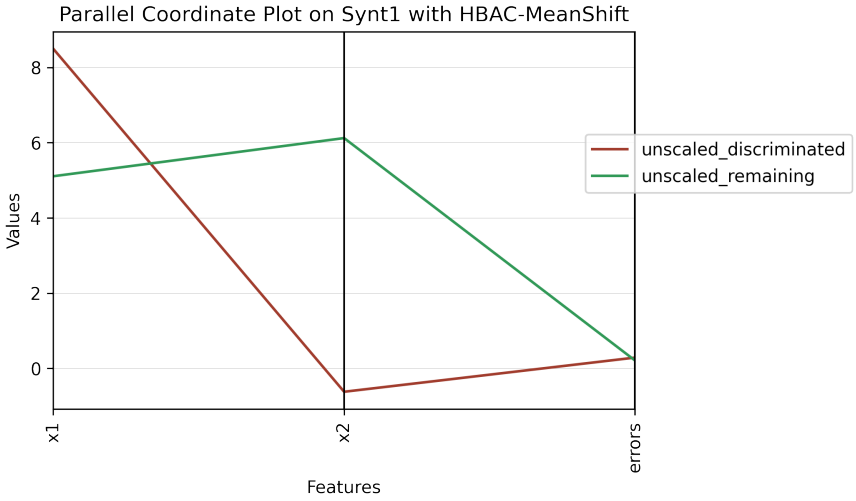


Figure B.8: HBAC-MeanShift: Parallel Coordinate Plot

B.4 Synt2

B.4.1 HBAC-KMeans

Table B.9: HBAC-KMeans: Cluster differences in Synt2

	Unscaled discriminated	Unscaled remaining	Difference
x1	11.730584	12.017920	-0.287335
x2	8.682036	14.050618	-5.368581
errors	0.756637	0.291431	0.465206
class	3.000000	1.163842	1.836158

B.4.2 HBAC-DBSCAN

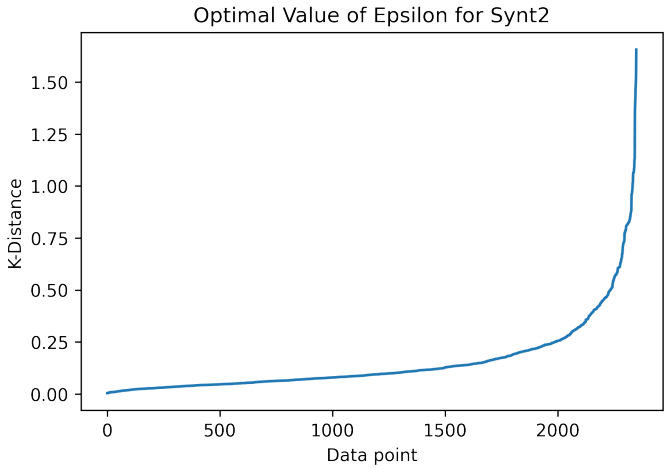


Figure B.9: K-Distance plot

Table B.10: HBAC-HBAC: Cluster differences in Synt2

	Unscaled discriminated	Unscaled remaining	Difference
x1	11.988794	11.990525	-0.001731
x2	8.971583	14.263998	-5.292414
errors	1.000000	0.230010	0.769990
class	3.000000	1.075025	1.924975

B.4.3 HBAC-MeanShift

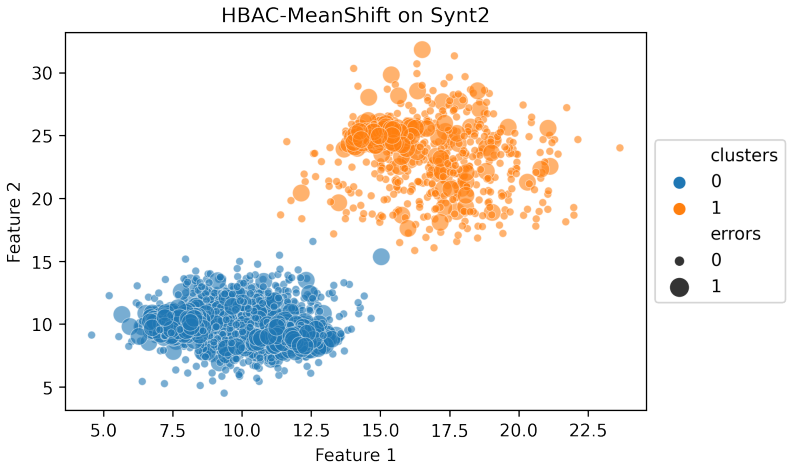


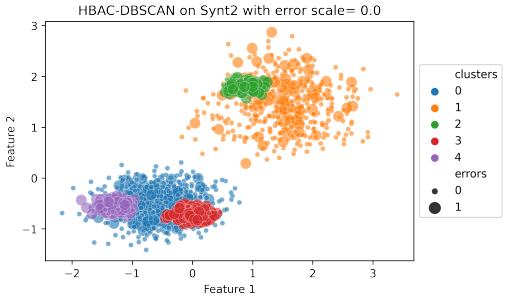
Figure B.10: MeanShift configuration with estimated bandwidth of 5.296

Table B.11: Differences in means for HBAC-MeanShift with bandwidth=3

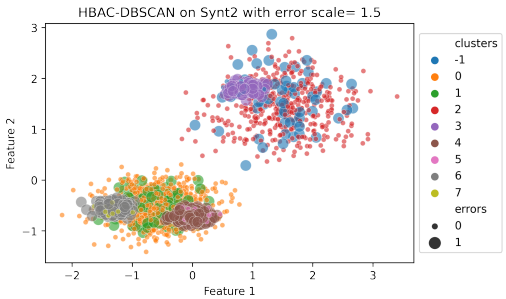
	Unscaled discriminated	**Unscaled remaining**	**Difference**
x1	12.056424	11.973928	0.082496
x2	8.998059	14.656346	-5.658287
errors	0.697425	0.246815	0.450610
class	2.901288	0.954352	1.946935

Appendix C

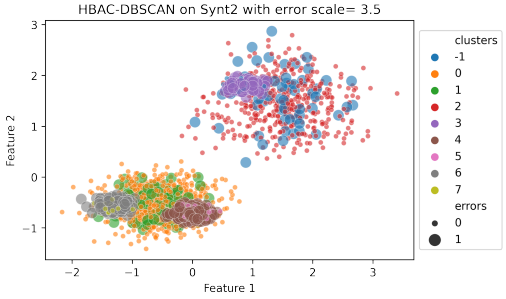
Error Scaling Results



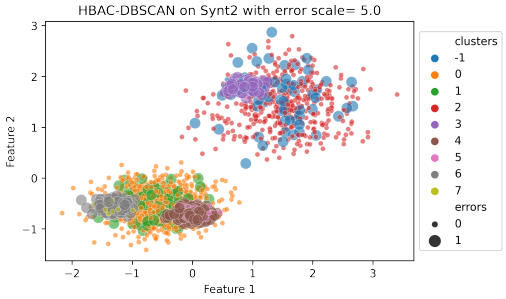
(a) 0.0



(b) 1.5

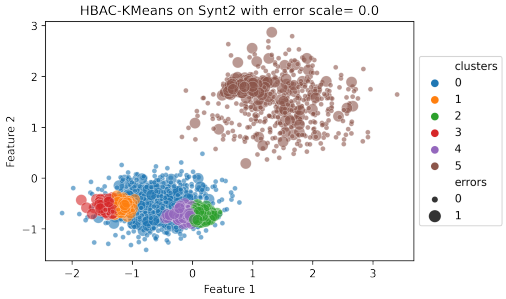


(c) 3.5

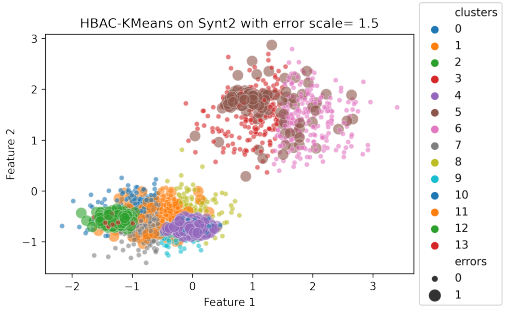


(d) 5.0

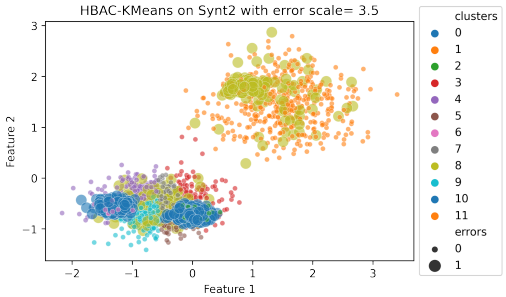
Figure C.1: DBSCAN Error Scaling



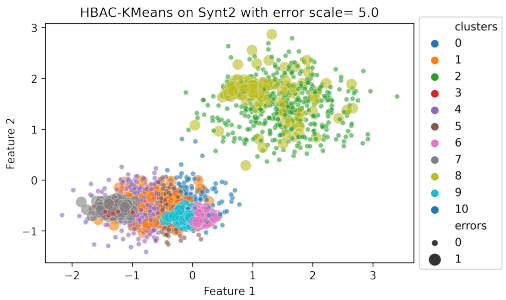
(a) 0.0



(b) 1.5



(c) 3.5



(d) 5.0

Figure C.2: KMeans Error Scaling