Pernille Johnsen

Investigating the Cost of Fairness in Automated Decision-Making Systems

Master's thesis in Computer Science

Supervisor: Pinar Øzturk

June 2020

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Kunnskap for en bedre verden

Pernille Johnsen

Investigating the Cost of Fairness in Automated Decision-Making Systems

Master's thesis in Computer Science
Supervisor: Pinar Øzturk
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU

Kunnskap for en bedre verden

Preface

This thesis was written during spring 2020 to fulfill the graduation requirements for the Computer Science master's degree program at the Department of Computer and Information Science, Faculty of Information Technology and Electrical Engineering at the Norwegian University of Science and Technology (NTNU).

I would like to thank my supervisor Pinar Øzturk for her guidance and encouragement through the work with this thesis. Additionally, I would like to thank fellow student Gunnar Strand Jacobsen for helpful cooperation and discussion throughout this project. Finally, I would like to thank my family for their support through this entire master's degree.

Pernille Johnsen

Trondheim, June 4, 2020

Abstract

This thesis investigates the use of multi-objective optimization and Pareto frontiers to explore the cost of fairness in automated decision-making systems.

Artificial intelligence (AI) systems are now being introduced into several facets of society, both in the public and private sector. These systems often make critical decisions about people's lives, in areas like health care, law enforcement, court cases, hiring, credit scoring, lending, and more. Such automated decision-making systems have been shown to reproduce or amplify human biases, sometimes even introducing new ones. These discoveries led to the emergence of research in the field of fair AI systems. In this field there is a general consensus that there exists a tradeoff between the accuracy and fairness. This concern has not received nearly enough research.

Multi-objective optimization (MOO) is a method for optimizing a solution for multiple objectives, and the use of Pareto frontiers is a popular method for combining the multiple objectives. The frontiers present the users with options where they can select the solutions they want from the front based on their desired tradeoff between objectives. In this thesis we build upon the work by Haas [2019], who created a framework for using MOO to generate Pareto frontiers that can be used to examine the tradeoff between accuracy and fairness. The framework includes the selection of data set(s), fairness and accuracy metrics, and classifiers and bias mitigation methods. These are used to build AI models that can be evaluated by studying the tradeoffs they produce.

The main contribution of this thesis is twofold. We further verify the framework by Haas, by applying it on a different data set, and using different bias mitigation methods. Our results generalize the use of MOO and Pareto frontiers as a method for investigating the cost of fairness. In addition, we present a novel method and architecture adapted from Haas' method, that allows for the same study of fairness tradeoffs. However, as opposed to Haas' method, this novel method can be applied on existing AI systems that have already been trained for optimal accuracy with no regard for fairness. The method and architecture builds upon state-of-the-art research in the field of fairness vs. accuracy tradeoffs.

Sammendrag

I denne masteroppgaven undersøker vi bruken av multi-objektiv optimalisering (multi-objective optimization) og Pareto fronter for å utforske konsekvensen rettferdighet kan ha på riktigheten i automatiske beslutningssystemer.

Kunstig intelligens (KI/AI) systemer blir mer og mer brukt i flere deler av samfunnet verden rundt, både i privat og offentlig sektor. Disse systemene tar ofte avgjørende beslutninger for menneskers liv, på områder som helse, rettsprosesser, ansettelsesprosesser, kredittscore kalkulering, låneinnvilgelse, med mer. Ofte har slike automatiske beslutningssystemer vist seg å reprodusere eller øke menneskelige fordommer, til og med introdusere nye i noen tilfeller. Slike funn har ført til en fremtreden av forsking på områder som omhandler rettferdighet og diskriminering i AI systemer. På dette området er det stort sett konsensus om at å forsøke å gjøre slike systemer mer rettferdige fører til at riktigheten til systemet synker.

Multi-objektiv optimalisering (MOO) er en metode for å optimalisere løsninger for flere formål. Bruken av Pareto fronter er en populær metode for å kombinere slike formål. Frontene presenterer brukere ett sett med de beste mulige løsningene, hvor en løsning kan velges basert på ønsket utbytte mellom formålene. Denne masteroppgaven bygger på arbeid av Haas [2019], som utviklet et rammeverk som bruker MOO og Pareto fronter for å utforske kostnaden rettferdighet har på riktigheten til AI systemer. Rammeverket krever valg av datasett, rettferdighet- og riktighetsmetrikker, og klassifiseringsalgoritmer og diskrimineringsforebyggende metoder. Disse blir brukt til å bygge AI modeller som kan evalueres ved å studere Pareto frontene de produserer.

Hovedbidraget til denne masteroppgaven er todelt. Vi bidrar til å verifisere rammeverket til Haas, ved å anvende et annet datasett og bruke flere diskrimineringsforebyggende metoder. Resultatene våre generaliserer bruken av MOO og Pareto fronter som en metode for å undersøke kostnaden av rettferdighet. I tillegg presenterer vi en ny metode og arkitektur videreutviklet fra Haas sin metode. Denne nye metoden tilbyr samme muligheter for undersøking av kostnaden av rettferdighet, men kan bli anvendt på eksiterende AI systemer som allerede har blir trent for optimal riktighet uten å ta rettferdighet i betraktning. Denne metoden og dens arkitektur bygger på aktuell forskning.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Artificial Intelligence (AI) and Machine Learning (ML) has seen a new wave of popularity in recent years. Due to current optimism, achievements and better hardware the field has taken a big step forward. AI systems are now being introduced into several facets of society, both in the public and private sector. These systems often make critical decisions about people's lives, in areas like health care, law enforcement, court cases, hiring, credit scoring, lending, and more. Such automated systems held the promise of removing human biases from the decision-making process. However, in practice they have been shown to reproduce or amplify human biases, sometimes even introducing new ones. One such famous discovery was made by journalists from ProPublica, who in 2016 published an article detailing how a system used to predict recidivism in the US was biased against black people [Angwin et al., 2016]. Such discoveries led to the emergence of research in the field of fair AI and ML systems.

Most of the work done in this field have focused on methods for mitigating bias and help ensure fairness, as well as formulating how best to measure fairness. However, there is a general consensus that there is a tradeoff the accuracy and fairness of such systems. This concern has not received nearly as much research. Reducing such a tradeoff would help further energize efforts to ensure fairness. This thesis will build on what was learnt through the Specialization Project in the subject TDT4501, where a review was done of the current state-of-the-art in the field of fair AI and ML. Based on this research the focus was narrowed for this thesis. In this thesis we will investigate what the cost of fairness may be for automated decision-making systems.

Multi-objective optimization (MOO) is a method for optimizing a solution for

multiple objectives. There exist several ways to combine the multiple objectives when using MOO, but one of the most popular methods is to produce Pareto frontiers. Such frontiers represent the optimal set of solutions, where one solution might be worse at one objective but must therefore be better at another objective. These frontiers present the users with options where they can select the solutions they want from the front based on their desired tradeoff between objectives. In effect it moves the decision on how to tradeoff the different objectives to the end of the process, where all available options can be judged. MOO is frequently used in complex resource allocation problems, e.g. scheduling problems, Internet bandwidth allocation, etc. We believe using MOO along with Pareto fronts will provide fruitful ground to investigate the tradeoff between accuracy in fairness in automated decision-making systems. A paper by Haas released late 2019 further inspired this idea [Haas, 2019].

1.1 Goals and Research Questions

The overarching goal of this master's thesis is to:

Goal *Investigate the tradeoff between fairness and accuracy in automated decision-making systems.*

This is a substantial and complex goal, and the scope is narrowed in order to fit into the timeframe of a master thesis. More specifically we will explore the following research questions:

Research question 1 *What is the state-of-the-art in research concerning the tradeoff between fairness and accuracy in automated decision-making systems?*

Based on the current state-of-the-art, we will attempt to investigate the possibilities for a method that can be used to answer the following research question.

Research question 2 *Using multi-objective optimization and Pareto fronts to optimize feature selection and classifier hyperparameters, what type of tradeoffs can be observed?*

We will then expand on this method based on interesting results from state-of-the-art papers that suggest that classification thresholds play a key role in the tradeoff between accuracy and fairness.

Research question 3 *Can we use this multi-objective optimization method to optimize classification thresholds, and what effect will this have on the type of tradeoff that can be observed?*

1.2 Thesis Structure

This thesis consists of five main parts. In chapter 2 we will present the background theory needed to understand the contents of this thesis. Chapter 3 will present the related work that cover the current state-of-the-art in research into the tradeoff between accuracy and fairness, as well as how multi-objective optimization and Pareto fronts are being used in this field. Chapter 4 will describe the methods and approaches we use, as well as describe the architectures used in this thesis. In chapter 5 we will describe the experiment plan and show, analyze and discuss the results of the experiments. Finally, in chapter 6 we will conclude the thesis and present our vision for future work.

Chapter 2

Background Theory

This chapter presents the theoretical background of this thesis. First, we present the topic of fairness in Machine Learning, in section 2.1. This section covers the relevant definitions, metrics and mitigation methods. In section 2.2 we cover the concept of multi-objective optimization (MOO). Related to this concept, in section 2.2.1, we cover Pareto optimality and Pareto frontiers, sometimes used in multi-objective optimization methods. In section 2.3 we cover Evolutionary Algorithms, and more specifically the NSGA-II algorithm. In section 2.4 we cover Support Vector Machines (SVMs) which will be used in this thesis. Lastly, we cover the `scikit-learn` Python package that we will be using in our implementation of the architecture.

The first section, section 2.1 on fairness, features some direct or rewritten reiteration of relevant background theory from the Specialization Project report [Johnsen, 2019]. However, as there has been some changes and updates to focus the research questions for this thesis, further relevant background theory has been found that present topics not covered in as much detail in the project report. This new material mainly consists of more detailed descriptions of fairness metrics and mitigation methods that follow in sections 2.1.1 and 2.1.3.

2.1 Fairness in Machine Learning

Fairness is an intricate concept that is hard to define precisely. The notion of fairness and justice has been a topic for philosophical discussion for thousands of years. In general, one might say that fairness is the absence of any prejudice or favoritism towards an individual or a group. This is closely related to the term

discrimination, where one might equate fairness with the absence of discrimination. The term discrimination is often used more specifically when someone is receiving unfair treatment because of some intrinsic or acquired traits, like race, gender, age, sexuality, etc. Such traits are often called *protected* or *sensitive attributes*.

To prevent discrimination and ensure fairness, many countries have enacted laws to prohibit many forms of discrimination. The Norwegian Equality and Anti-Discrimination Act [Norwegian Ministry of Culture, 2017] states that its purpose is to ... *prevent discrimination on the basis of gender, pregnancy, leave in connection with childbirth or adoption, care responsibilities, ethnicity, religion, belief, disability, sexual orientation, gender identity, gender expression, age or other significant characteristics of a person* and that *this Act shall apply in all sectors of society*. Such anti-discrimination laws provide a somewhat of a foothold for what constitutes fairness in legal terms and denote the minimum requirement for companies looking to develop AI systems. One complication is that the legal doctrine is not static, but differ from country to country, and over time. In order to determine whether a system is fair (outside of legal action), in an efficient manner, a clear mathematical definition is needed. There have been several attempts at creating such a definition, but there is no consensus on which is the 'best' definition.

2.1.1 Fairness Definitions and Metrics

There exist many different fairness definitions. The differences between them are not just theoretical, but produce entirely different outcomes [Bellamy et al., 2019]. For example, ProPublica and Northpointe had a public debate on the issue of predictive recidivism (in Northpointe's COMPAS system, [Angwin et al., 2016]) revolving entirely around which fairness definition should be used to make such decisions. Kleinberg et al. [2017] compare three formalized definitions and find that, except in highly constrained special cases, all three fairness conditions cannot be satisfied simultaneously. This shows the complexities around the concept of fairness, and the use of these definitions in order to ensure fairness in AI systems.

Often definitions of fairness are divided into two categories; *Group Fairness* and *Individual Fairness*. Group fairness definitions are focused around treating groups equally, often measured by certain metrics (e.g. equal accuracy, or equal false positive or negative rates). Individual fairness definitions are centered around treating individuals equally, independent of group membership. Not all existing fairness definitions are easily measurable, but in both categories there exists some measurable metrics for fairness.

Group Fairness

The following metrics are popular group fairness metrices.

Statistical Parity was introduced early in fair AI literature, often used interchangeably with the term group fairness at the time [Zemel et al., 2013]. The term *demographic parity* has also been used to refer to this metric. The idea behind it is to ensure that the proportion of members in a protected group ($G = 1$) receiving positive classification ($\hat{Y} = 1$) should be equal to the proportion of the population as a whole. Which means that the probability of receiving a positive classification is independent of group membership.

$$P(\hat{Y} = 1|G = 1) = P(\hat{Y} = 1)$$

A special case of statistical parity is *conditional statistical parity*, where protected groups and the population as a whole should have equal probability of receiving positive classification, given a set of legitimate factors ($L = 1$).

$$P(\hat{Y} = 1|G = 1, L = 1) = P(\hat{Y} = 1|L = 1)$$

The metric used to measure statistical parity is defined as the difference between the percentage of people in the protected group receiving positive classification and the percentage of people in the unprotected group ($G = 0$) receiving positive classification.

$$SP_{Diff} = |P(\hat{Y} = 1|G = 0) - P(\hat{Y} = 1|G = 1)| \tag{2.1}$$

Disparate Impact similarly to statistical parity also considers the probability of receiving the positive classification depending on group membership. However, disparate impact considers the ratio between the probabilities for the two groups, rather than difference between them [Feldman et al., 2015].

$$DisparateImpact = \frac{P(\hat{Y} = 1|G = 0)}{P(\hat{Y} = 1|G = 1)} \tag{2.2}$$

Equalized Odds was introduced as an alternative to statistical parity. As opposed to looking at the percentage of observations with positive classification, equalized odds was suggested, which considers the true positive and false positive rates instead. Hardt et al. [2016] defines equalized odds as; A predictor $\hat{Y}$ satisfies equalized odds with respect to protected attribute G and outcome Y, if $\hat{Y}$ and G are independent conditional on Y.

$$P(\hat{Y} = 1|G = 0, Y = y) = P(\hat{Y} = 1|G = 1, Y = y), y \in \{0, 1\}$$

Another way to put it is that an algorithm is considered fair under equalized odds, if the protected and unprotected groups have equal rates of true positives and false positives.

$$FPR_{G=0} = FPR_{G=1}$$
$$TPR_{G=0} = TPR_{G=1}$$

To measure equalized odds, the difference between the true positive and false positive rates are calculated.

$$EqOdds_{Diff} = 0.5 * (|FPR_{G=0} - FPR_{G=1}| + |TPR_{G=0} - TPR_{G=1}|) \quad (2.3)$$

Equal Opportunity is a special case of equalized odds, where only the difference between true positive rates is considered.

$$EqOpp_{Diff} = |TPR_{G=0} - TPR_{G=1}| \quad (2.4)$$

Individual Fairness

Most fair AI literature considers group fairness metrics. Individual fairness is not always as easily quantifiable. However, one popular individual fairness metric is the Theil Index.

The Theil Index builds on the theory of general entropy indices, and measures if individuals are treated in a similar way. Speicher et al. [2018] proposed using such inequality indices to measure algorithmic fairness, which had previously been used in the field of economics. The generalized entropy indices for a problem with n observations is defined as follows:

$$GEI = \frac{1}{n\alpha(\alpha - 1)} \sum_{i=1}^{n} [(\frac{b_i}{\mu})^{\alpha} - 1] \quad (2.5)$$

where $b_i = \hat{y}_i - y_i + 1$ and $\mu = \frac{\sum_i b_i}{n}$. The Theil index is a special case where $\alpha = 1$:

$$Theil = \frac{1}{n} \sum_{i=1}^{n} \frac{b_i}{\mu} log(\frac{b_i}{\mu}) \tag{2.6}$$

2.1.2 Bias Types

When working to ensure fairness it is important to understand the different sources of unfairness, i.e. the bias that may occur or already exist. In this section we will detail these different types of biases.

Through working on the Specialization Project, subject TDT4501, we found several existing taxonomies for bias. Friedman and Nissenbaum [1996] presented the earliest work we are aware of, attempting to establish a framework to understand and remedy bias in computer systems. They sort bias under three overarching categories relating to the timeline of AI system development; *pre-existing bias, technical bias, and emergent bias*. Later, Dobbe et al. [2018] expanded on this early work in a Machine Learning context, focusing on technical and emergent bias. These papers provide a good starting point from which to organize bias, but are not detailed enough to give good practical guidance when designing AI/ML systems.

Some works limit their scope to study bias in specific fields. Olteanu et al. [2019] limit their scope to social data on the web (from websites like Facebook, Wikipedia, etc.), to create their framework. Torralba and Efros [2011] look at bias in image recognition datasets. Rajkomar et al. [2018] present a framework to mitigate unfair bias in the context of health care. Suresh and Guttag [2019] create a framework that looks at bias in the AI design process, basing their work on previous research in fairness. They create five categories relating to data generation, model building and implementation. These five categories are: *Historical bias, Representation bias, Measurement bias, Aggregation bias, and Evaluation bias*. In a survey by Mehrabi et al. they attempt to summarize the work in [Suresh and Guttag, 2019] [Olteanu et al., 2019] as well as some others. However, the summary is very broad and fairly rudimentary, simply a long list of bias types (23 types, too many to list here), with short explanations. No significant attempt has been made by the authors to organize these biases, simply organizing them loosely around data, algorithm and user interaction.

These approaches and taxonomies listed above are often too general, or too specific, either not giving enough detail for practical purposes or only covering certain research areas or certain parts of the AI development lifecycle. Additionally,

the terminology is often not aligned or is sometimes conflicting. A clear agreed upon list of concepts lacking. Inspired by these drawbacks and in cooperation with another student, Gunnar Strand Jacobsen, as well as our supervisor, Pinar Øzturk, we created a new taxonomy for bias. The goal was to establish a unifying taxonomy, with clearly defined and commonly used terms, in order to create a comprehensive approach to bias for ML practitioners. Below follows a summary of this new taxonomy from the Specialization Project [Johnsen, 2019].

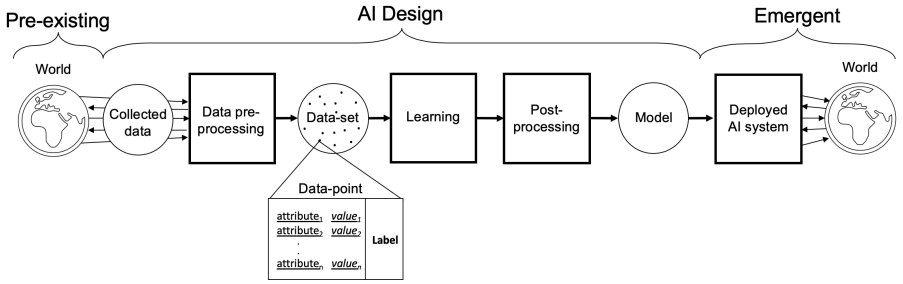


Figure 2.1: *AI development lifecycle*

Figure 2.1 shows a rudimentary overview of the AI development lifecycle, from the current state of the world to the interaction between the world and the finished, deployed AI system. A realistic model of the AI development lifecycle would have more branching and looping paths, but the model has been simplified in order to illustrate the different types of bias that might occur. The figure has been split into three parts; *Pre-existing*, *AI Design* and *Emergent*. These three parts are the overarching categories we used in our taxonomy. They are similar to the taxonomy introduced in [Friedman and Nissenbaum, 1996], separating between bias existing in the world before any development, the decisions made during the design and development of the AI system, and the interaction between the finished system and the world. However, in our case we used *AI Design bias* instead of *Technical bias*, to reflect the whole spectrum of decisions made during the design in the AI system, not just technical. AI Design bias is the most relevant for the contents of this thesis and will receive the most attention in this summary.

Pre-existing bias is the bias that exist in the world, before ML practitioners have even started thinking about building a system. This type of bias can be split into three main sub-types; *Historical*, *Social*, and *Stakeholder* bias. Historical bias is bias related to the passing of time and historical differences. Social bias is similar to historical bias, but is not necessarily related to time. Social bias relates to the bias of people in society in general, not restricted by changing laws

or social norms. Stakeholder bias relates to the biases that might exist in the people making decisions regarding the AI system, from the developers to the company owners. These biases may be subconscious or not.

After the AI system has been finished and released into the world, emergent bias might be introduced. It is therefore important to continue to monitor the system, not only for testing purposes but also to protect against biases that might occur at this time. If emergent biases are introduced after release, the system might have to go back into the design and development phase in order to fix it, or perhaps some solutions could be introduced before the system is released at all. Emergent bias can be split into several sub-types, but we will not cover them here.

AI Design Bias

The term AI design bias is used to cover the process from collection of data through development all the way to the finished system. This is the part where most decisions about the AI system is made, which means it is also the part with most possibilities of bias. Figure 2.2 gives an overview of the different types of biases that might occur during this process.

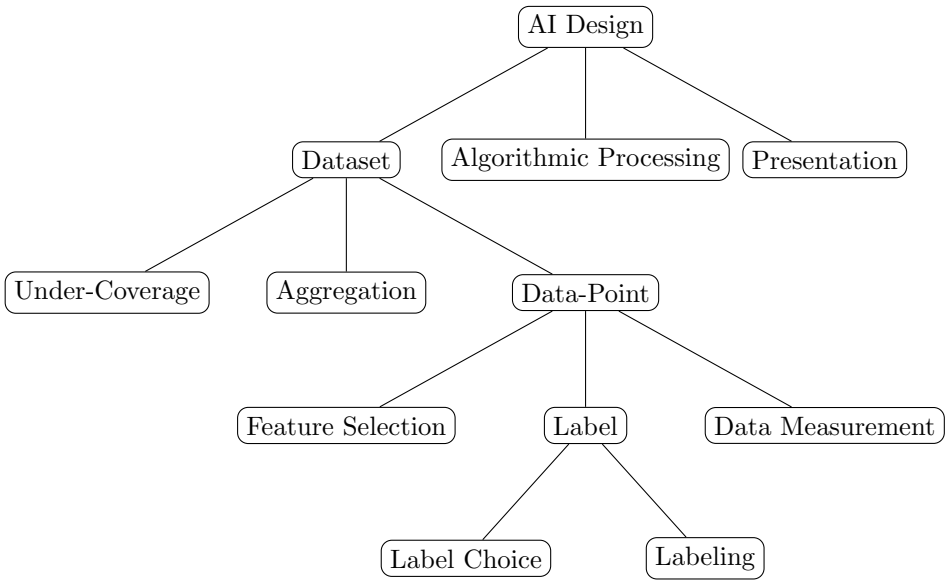


Figure 2.2: *AI design bias overview*

Dataset bias. The dataset is a central part of ML, and it is therefore important to ensure that it is unbiased. First, we will cover types of bias that effect the dataset as a whole; *under-coverage bias* and *aggregation bias*. Afterward we will cover biases that relate to individual parts of the data-points in the dataset.

Under-coverage bias. Under-coverage bias occurs when some underlying group is underrepresented in the dataset. In such cases the ML algorithm will perform worse for the underrepresented groups, because it hasn't had enough data to learn from. Some sources also call this *representation bias, under-representation bias* or *minority bias*.

Aggregation bias. Aggregation bias relates to the underlying groups (e.g. women, white people, etc.) that a system is set to make decisions about. This type of bias can occur when these groups have important differences that should affect the outcome of the decisions. When one algorithm is used to make decisions for such differing groups, the algorithm is often unable to perform well on any of the group, because it is unable to recognize any consistent pattern. If one group has more data-points than any other, the algorithm might become biased towards that group, making bad decisions for people belonging to any other group. Aggregation bias does not only effect how one should handle the dataset, but also stretch into the handling of learning. It might for example be prudent to train different models and/or use different learning algorithms for the different groups.

Data-point bias. Data-point bias relates to individual parts of data-points. Such bias may affect the features, feature values or the label.

Feature selection bias. Feature selection bias occurs during feature selection. Feature selection is an important part of ML, as it aims to assure that the proper features are selected, while not selecting to many, lowering the efficiency of the system. However, feature selection not only affects the accuracy and efficiency of the system, but also how fair the system is. An important discussion here is whether a protected attribute (e.g. gender, race, etc.) should be a feature or if it should be left out entirely.

Label bias. Label bias is by some regarded as the biggest obstacle for fair Machine Learning. Label bias can come from either the choice of label, or the labeling itself. When the goal of an AI system is to make decisions in the real world, the desired label is often unobservable or partially observable. This can lead to label choice bias. In such cases the label needs to be replaced by a proxy. The goal of the proxy is to represent the label as near to 'the ground truth' as possible. As such, bias in the choice of label can be viewed as a measurement error between the proxy label and the ground truth. Labeling bias occurs when the labels are incorrect, not because of the use of a proxy, but because of some

bias or misconception from whomever assigned the label.

Data Measurement bias. Data measurement bias effects the feature values in the data-points. This type of bias occurs when these values are inaccurate. For example, in image processing, it is known that images cannot represent the world completely because they are limited by camera technology.

Algorithmic processing bias. Often, when discussing bias in ML, it is said that the learning algorithm is completely neutral and simply reflect bias in the data. However, bias might still be introduced in the learning step, depending on which decisions are made regarding the algorithms used [Danks and London, 2017].

Presentation bias. When using ML systems in general society a user interface (UI) is often needed, especially when the end users are regular people, unfamiliar with ML and programming in general. The design and development of the UI often fall outside the jurisdiction of ML practitioners, depending on the size of the development team and the budget. Regardless of who the UI is made by it is important to note that bias can be introduced depending on how information is presented [Mehrabi et al., 2019].

2.1.3 Mitigation Methods

Mitigation is a huge part of the fair AI discussion. After discovering a problem, the innate response is to find ways to fix it. An attractive approach to mitigating unfairness for computer scientist is to develop technical solutions to the problem, as the problem comes from technical systems. Fairness is a complex concept, and many point out that technical solutions are likely not enough to fix the problem entirely [Whittaker et al., 2018]. However, they are still an essential part of the solution.

Mitigation methods can be split into three categories: pre-, in-, and post-processing. *Pre-processing* methods are methods intended for use on the original data, before the main learning phase. In general, they work by taking the original collected data as input, performing some algorithm, and outputting a 'new' dataset. *In-processing* methods generally run as part of the main learning phase, either integrated into an existing algorithm, or as entirely new algorithms. *Post-processing* methods are used after the learning phase, working on the output from the learning algorithm. These methods can also be used when the systems are otherwise a *black-box* and there is no way to make any changes to the outputted model. Below we will present a brief summary of the three mitigation methods used in this thesis. All of them are pre-processing algorithms.

Reweighing

The Reweighing method was introduced by Kamiran and Calders [2012]. The method generates weights differently for each (group, label) combination in the training data, to unbias the data set. This results in data sets where each data point is given a weight depending on what (group, label) combination it contains. For example, data points where the sensitive attribute (e.g. race) is part of an unprivileged group (e.g. black) and the label is positive will be given higher weights then data points part of the unprivileged group with a negative label. Oppositely, data points that are part of the privileged group (e.g. white) are given lower weights if the label is positive and higher weights if the label is negative. The weight (W) for each data point (X) are assigned using the following formula:

$$W(X) = \frac{P_{exp}(S = X(S) \wedge Class = X(Class))}{P_{obs}(S = X(S) \wedge Class = X(Class))} \tag{2.7}$$

where S denotes the sensitive attribute. Kamiran and Calders state: "The weight of an object will be the expected probability to see an instance with its sensitive attribute value and class given independence, divided by its observed probability". Using the Reweighing method, no data point values are changed, they are given weights that compatible classifiers can use during training.

Disparate Impact Remover

The Disparate Impact Remover was introduced by Feldman et al. [2015]. The method attempts to increase group fairness while preserving rank-ordering within groups, by editing feature values. The fairness measure used in the method is the disparate impact metric we described in section 2.1.1, hence the name Disparate Impact Remover. The feature values are edited so that any attributes in the data set (D) that could be used to predict the sensitive attribute are changed so that the transformed data set ($\bar{D}$) can be certified as having no disparate impact. In the resulting data set $\bar{D}$ only feature values are transformed, not the protected attribute and the label. The goal of the method is also to preserve the relative per-attribute ordering, rank, in the data set so that it is still able to predict the label.

The algorithm creates $\bar{Y}$, such that for all $y \in \bar{Y_S}$, the corresponding $\bar{y} = F_A^{-1}(F_S(y)$. $\bar{Y}$ denotes the set of features excluding the protected attribute. S denotes the protected attribute. F_A^{-1} is the quantile function of the median distribution A: $F_A^{-1} = \text{median}_{\ s \in S} F_S^{-1}(u)$. F_S ranks the values of Y_S (i.e. $F_S^{-1}(1/2)$ is the value of y such that $P(Y \geq y | X = x) = 1/2$).

Since the 'repair' process outlined by the algorithm is likely to degrade the ability to classify accurately, Feldman et al. adds a 'repair level' parameter $\lambda \in [0, 1]$ to the algorithm. $\lambda = 0$ yields the unmodified data set and $\lambda = 1$ yields the fully repaired data set as described above. In our thesis we will use a repair level of $\lambda = 0.8$ in our experiments. It is, however, also possible to add this parameter to the set of parameters to be optimized by our method and architecture as described in chapter 4.

Optimized Pre-Processing

The Optimized Pre-processing method by Calmon et al. [2017] learns a probabilistic transformation that, using group fairness, transforms the original data set so that it satisfies three properties: *discrimination control*, *distortion control*, and *utility*. This transformation edits the features and labels in the data set.

The first objective, discrimination control, is to limit the dependence of the transformed label $\bar{L}$ on the protected attribute S. The second objective, distortion control, is to satisfy distortion constraints. These constraints restrict the transformation to reduce or eliminate certain large changes (e.g. a very low credit score mapping to a very high credit score). In addition to these constraints on individual distortion, the third objective, utility, requires that the distribution of $(\bar{Y}, \bar{L})$ is close to the distribution of (Y, L). Y denotes the features in the data set, excluding the protected attribute. This objective ensures that the model learned from the transformed data set is not too different from one learned from the original data set.

2.1.4 Aif360

`Aif360` [Bellamy et al., 2019] is an open source toolkit that we will be using in this thesis. The toolkit includes a Python module for examining, reporting and mitigating bias and discrimination in machine learning models. The package contains over 70 fairness metrics, and 10 state-of-the-art mitigation algorithms, that can be integrated throughout the AI lifecycle. The package also provides easy access to some common data sets often used when studying fair AI: The Adult Data Set, The Bank Marketing Data Set, The COMPAS Data Set, and the German Credit Data Set.

2.2 Multi-Objective Optimization

When performing any task, the goal is to reach and *optimize* an *objective*. This is also the case when AI systems perform tasks. During training and development, the performance of the system is often measured by calculating how well the

task was performed, i.e. if and how well the desired objective was reached. The function used the calculate the measure is often called a *fitness function*. In automated decision-making systems the main objective is often measured by the *accuracy* of the system, as the ultimate goal for the system is to always make the correct decision. However, some tasks may also have multiple objectives. For examples, when making a software product, a clear goal is for the software to be able to handle the requirements correctly, but there are often other goals as well. It is desirable for the software to have good usability, to be easy to maintain and update, and there might be limits on the timeframe or the expense.

The goals of meeting requirements, good usability, and easy maintenance may be referred to as objectives, while limits on the timeframe and the expense are *constraints*. AI systems may also have multiple objectives and/or constraints. In the context of AI systems, objectives are measurable goals (e.g. accuracy, false/positive rates, etc) used to create fitness functions. Meanwhile, constraints limit the solution space by declaring some solutions *infeasible* [Floreano and Mattiussi, 2008]. For example, while the software requirements create a measurable solution space, containing only software reaching the requirements, a time or expense limit decreases the solution space by declaring all software solutions that break the limit infeasible.

In some cases, the distinction between an objective and a constraint is not entirely clear, as some goal may be declared to be either or. For example, when introducing fairness into an AI system. The desired solution may be formulated as a system with high accuracy that doesn't violate a fairness constraint, or fairness can be viewed as a second objective. In both cases a new measurement is needed, as measuring accuracy says nothing about the fairness of the system. The difference between regarding fairness as an objective versus a constraint, is that having a constraining value is a simple quantitative approach, while viewing fairness as an objective opens up for more qualitative considerations.

There are several methods for handling multiple objectives. One method is to rank objectives by *priority*, in which case the system will focus on the most important objective first, before taking into account objectives with lower priority [Floreano and Mattiussi, 2008]. A second method is the use of objective *targets*, where each objective has a value representing its target. These targets can be used to convert the multiple objectives into a single scalar fitness function, in which case a single-objective algorithm can be used [Floreano and Mattiussi, 2008]. A third method is to use *weights*. Assuming one knows the *tradeoff* between different objectives, it is possible to create an aggregated fitness function where each objective function is weighted based on the known tradeoff between them. This creates a single aggregated fitness function that, again, can use single-objective algorithms for optimization. This approach is often used because

of its simplicity. However, the approach relies on the tradeoffs being known and easy to quantify. The more objectives, the more difficult creating weights will be. Additionally, it might not always be the case that the objectives can be combined linearly. A fourth approach that doesn't require knowledge of tradeoffs beforehand is based on the concept of *Pareto optimality*.

2.2.1 Pareto Optimality and Pareto Frontiers

Italian economist Vilfredo Pareto (1848–1923), in his study of distributional efficiency, developed the concept of *Pareto optimality*.

Pareto optimality. *An allocation is considered Pareto optimal if no alternative allocation could make someone better off without making someone else worse off. [Mock, 2011]*

The concept has long been used to study markets and society, but it is also applicable for multi-objective optimization purposes. In more mathematical terms (from [Floreano and Mattiussi, 2008]):

Pareto dominance. *A solution x_1 is said to **dominate** x_2 if*
(1) it is at least as good with respect to all objective functions, and
(2) strictly better with respect to at least one objective.
*If a solution is not dominated by any other solutions it is **Pareto optimal**.*

Though this condition is fairly strict, in the space of feasible solutions it is possible for there to be a whole set of candidate solutions that are not dominated by any other solutions. This set is called the *Pareto-optimal set* or the *Pareto frontier/front*.

In general, it is often the case that there are several optimal solutions rather than a single optimal individual. Such a set of solutions allows for considerations of the tradeoffs between each objective, allowing for informed consideration before a final solution is picked, or for interesting research opportunities. Figure 2.3 shows an example of a Pareto frontier in a case with two objective functions where the goal is to maximize each objective. The general shape of the front will vary depending on whether objective functions are maximized, minimized, one of each, etc. Such a plot provides an easy illustration of the tradeoffs between each objective. However, having more objectives adds more dimensions to plots of Pareto fronts, diminishing the effect of such a plot. There exist several algorithms for multi-objective optimization (MOO) and a subset of them use the concept of Pareto optimality and find Pareto fronts. These algorithms are often based on the theory of evolutionary algorithms.

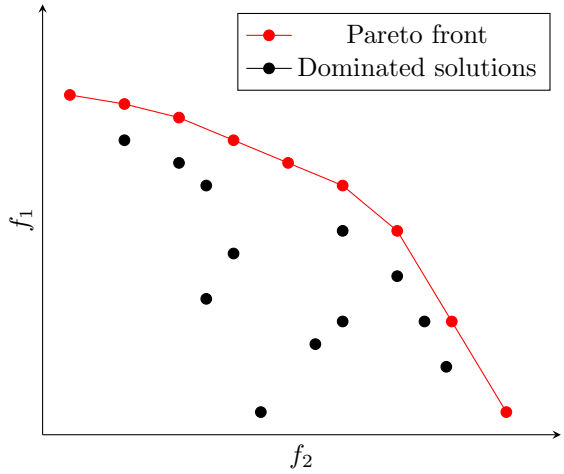


Figure 2.3: *Example of a Pareto frontier in the case of two objective functions f_1 and f_2, where both functions are maximized.*

2.3 Evolutionary Algorithms

Evolutionary algorithms are based on the natural evolutionary process. In the 60s and 70s computer scientist and engineers began developing algorithms inspired by this natural evolution. Most evolutionary algorithms are based on the same four pillars of natural evolution: (1) maintenance of a population; (2) creation of diversity; (3) a selection mechanism; and (4) a process of genetic inheritance [Floreano and Mattiussi, 2008]. This can be summed up in the famous line: *"Survival of the fittest"*. There are several ways to handle each of these four main pillars, pillars (1) and (4) are determined based on the specific problem at hand, while pillars (2) and (3) are generally determined by which evolutionary algorithm is chosen. However, all steps have an effect on the outcome of the algorithm. Some famous evolutionary algorithms are; Genetic Algorithm (GA), Evolutionary Strategies (ES), Genetic Programming (GP), and Evolutionary Programming (EP). In general, all of these algorithms follow the procedure as shown in Algorithm 1.

When using such algorithms important decisions include how to (a) represent the population; (b) generate the initial population; (c) design the fitness function; (d) design the crossover and mutation operators; (e) determine the termination condition. The decision made on how to represent the population has an effect on all the other decisions. In EAs the population consist of individuals represented as chromosomes. The chromosomes represent potential solutions to a problem, and

Algorithm 1 Evolutionary Algorithm

```
1: generate initial population P
2: generation: t = 0
3: while NOT termination condition do
4:    calculate fitness of each individual in P
5:    select parents based on fitness score
6:    perform crossover to create offspring from parents
7:    perform mutation on the offspring
8:    P = selected individuals from set of parents and offspring
9:    t = t + 1
10: end while
```

performing genetic operations on them ensure the process of genetic inheritance. The representation of a chromosome can be split into two parts, *the genotype* and *the phenotype*. The genotype is the encoding of the solution that is used by the evolutionary algorithms, while the phenotype is the encoding that represent the real-world solution. To calculate fitness scores, in general, the genotype has to be decoded into a phenotype that can be used to calculate these scores.

There are two main types of genetic operators; crossover, and mutation. The crossover operator is used to combine two parent chromosomes to generate children chromosomes, and the mutation operator is used to add a slight mutation to a child chromosome. In general parameters defining the probability of crossover and the rate of mutation are used in EAs to define how often crossover and mutation is performed. Often the probability of crossover is fairly large, but not 100%, while the rate of mutation is a lot smaller, closer to 0%. These two operators are performed in consecutive order, where crossover is performed first, and mutation is performed on the children produced from the crossover. Because the crossover probability is not 100%, some parent chromosomes will become part of the next generation, but it is still possible that some mutation will occur.

Introducing multiple objectives into EAs brings with it further considerations as described in section 2.2. There exist several algorithms developed to handle multi-objective optimization (MOEAs), most of them based on existing single-objective algorithms like the ones listed above. Some example algorithms are: Multiple Objective Genetic Algorithm (MOGA), Strength Pareto Evolutionary Algorithm (SPEA), Niched Pareto Genetic Algorithm (NPGA), Pareto-Archived Evolution Strategy (PAES), Multi-Objective Messy Genetic Algorithm, Nondominated Sorting Genetic Algorithm (NSGA) and NSGA-II. All of these algorithms have different advantages and disadvantages. In this thesis we will be using NSGA-II. This algorithm is popular as it is fast, not overly complex, and it

results in a Pareto front that can be used to study tradeoffs.

2.3.1 NSGA-II

Deb et al. [2002] introduced an upgrade to the Nondominated Sorting Genetic Algorithm (NSGA) called NSGA-II, aiming to improve upon the original and address the main criticisms it received. In their paper they find that their improved algorithm also mostly outperforms two other popular MOEAs, PAES and SPEA, in many common test problems used to evaluate MOEAs. NSGA-II improves upon the original NSGA by introducing a *fast nondominated sorting* algorithm, incorporating *elitism*, and making it parameterless. The algorithm follows the same general procedure as single-objective algorithms, as it is based on GA. However, instead of a single fitness function more complexity is needed to determine the "best" individuals. This is done through the fast nondominated sorting algorithm and a *crowding-distance* algorithm is used to preserve diversity in the population.

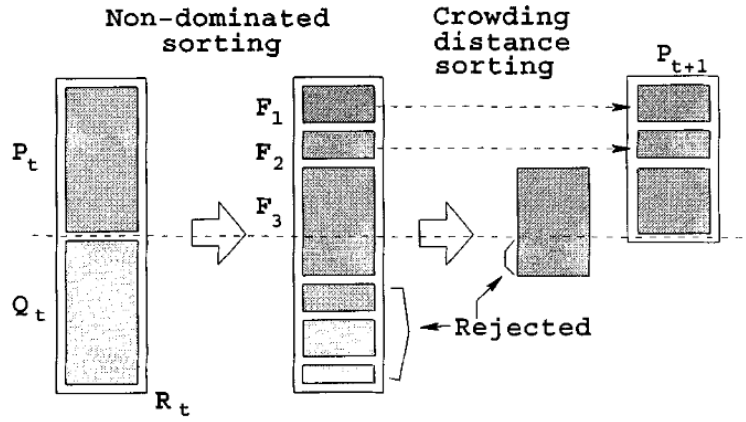


Figure 2.4: *NSGA-II selection procedure (from [Deb et al., 2002])*

Figure 2.4 exhibit the three essential principles involved in NSGA-II; elitism, fast nondominated sorting, and crowding-distance. The procedure starts with a population of chromosomes, R_t consisting of both the parents, P_t, and the children, Q_t. This is what the principle of elitism boils down to; rather than just discarding the parent population after children are generated, the parents are also considered when evaluating the population and are kept if they perform well. This principle helps ensure that if a good solution has been found in the middle of the run this solution is not lost because the children might be evolved

in worse directions. From this combined population fast nondominated sorting is performed sorting the entire population into Pareto fronts, by removing the found fronts from the population before finding the next. Leading to the population being sorted by whether they are in the best F_1, second best, F_2, to worst fronts F_n. Finally, a crowding-distance algorithm is used to preserve diversity by applying Tournament selection to reduce the population, where solutions compete by comparing their crowding-distance (a measure of the density of solutions). After ranking the solutions in this procedure, the algorithm follows the same approach as most EAs, creating a new population using selection, crossover and mutation, before the loop starts over again. Pseudocode for the main loop of the NSGA-II algorithm can be seen in Algorithm 2.

Algorithm 2 NSGA-II Main Loop (from [Deb et al., 2002])

$R_t = P_t \cup O_t$	Combine parent and offspring pop
$F =$ fast-non-dominated-sort(R_t)	$F = (F_1, F_2, ..)$ all nondominated fronts
$P_{t+1} =$ and $i = 1$	
repeat	
crowding-distance-assignment(F_i)	Calculate crowding-distance
$P_{t+1} = P_{t+1}$	Include ith nondominated front
$i = i + 1$	Check next front
until $\lvert P_{t+1}\rvert + \lvert F_i\rvert \leq N$	Until the parent population is filled
Sort$(F_i, \prec_n)$	Sort in descending order using $\prec_n$
$P_{t+1} = P_{t+1} \cup F_i[1 : (N - \lvert P_{t+1}\rvert)]$	Choose the first $(N - \lvert P_{t+1}\rvert)$ elements
$Q_{t+1} =$ make-new-pop(P_{t+1})	Using selection, crossover and mutation
$t = t + 1$	Increment generation counter

NSGA-II is a popular MOEA, often used on common EA problems that need multiple objective measures to ensure the best outcome. A prevalent set of problems are resource allocation problems, which work well with EA. In most real-world resource allocation problems, many objectives are needed to fit the complexity of the real world. Example problems include memory management, Internet bandwidth allocation, and scheduling problems. In these areas NSGA-II is in common usage. Using NSGA-II for fairness measures, however, is not as frequently explored.

2.4 Support Vector Machines

Support Vector Machines (SVMs) are popular and flexible class of classifiers for supervised learning problems. Russell et al. [2010] list three properties that make SVMs attractive:

1. SVMs construct a *maximum margin separator* — a decision boundary with the largest possible distance to example points. This helps them generalize well.
2. SVMs create a linear separating hyperplane, but they have the ability to embed the data into a higher-dimensional space, using the so-called *kernel trick*. Often, data that are not linearly separable in the original input space are easily separable in the higher-dimensional space. The high-dimensional linear separator is actually nonlinear in the original space. This means the hypothesis space is greatly expanded over methods that use strictly linear representations.
3. SVMs are a nonparametric method—they retain training examples and potentially need to store them all. On the other hand, in practice they often end up retaining only a small fraction of the number of examples—sometimes as few as a small constant times the number of dimensions. Thus, SVMs combine the advantages of nonparametric and parametric models: they have the flexibility to represent complex functions, but they are resistant to overfitting.

The different SVMs are defined by the type *kernel function* used. Kernel functions specify the type decision boundary that can be learned by the classifier, and the calculations needed to ensure this. Both linear and non-linear kernels can be used. Examples of non-linear kernels are radial kernels, and polynomial kernels. Though non-linear kernels are more flexible, they are also more complex and require more computational power, especially as the size of the training data set grows. The choice of kernel function is therefore an important consideration when working with SVMs. The choice of kernel also effects what hyperparameters need to be provided to the SVM classifier.

The main hyperparameter for SVM is independent of the chosen kernel. This is the penalty parameter C. As mentioned above SVMs find a separating hyperplane with the maximal margin. The C parameters determines how to tradeoff the maximization of this margin with the correct classification of training samples [Hsu et al., 2016]. This is done by determining the penalty to apply to the error term for each training data point. A small C will encourage a larger margin at the cost of training accuracy, while a larger C will accept small margins at the cost of complicating the decision function in order to fit more training points. In addition to the C parameters, another important hyperparameter is the kernel parameter. The kernel parameter is determined by which kernel is used in the SVM. In our thesis we will be using the radial basis function (RBF) kernel. The

RBF kernel has the following kernel function:

$$K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2), \gamma > 0 \tag{2.8}$$

where x_i and x_j are training points, and γ is the kernel parameter [Hsu et al., 2016]. The γ parameters determines the influence of each training point over other training points, a low γ means the influence reaches 'far', while a high value means the influence 'close'. The optimal value for both C and γ is dependent upon the dataset.

2.5 Scikit-learn

`Scikit-learn` [Pedregosa et al., 2011] is an open source machine learning module for Python. We will be using this module in our thesis. The package integrates a wide range of state-of-the-art machine learning algorithms, including SVM, nearest neighbor, decision trees, naive Bayes, linear models, neural network models, ensemble methods and more. The package also provides methods for model selection, preprocessing, and evaluation, etc. The authors of `scikit-learn` state that the package focuses on bringing machine learning to a wide range of users, with an emphasis on ease of use and performance. This means that though the module might not be ideal for very complex machine learning, it provides an excellent use case for exploring and studying medium scale problems.

Chapter 3

Related Work

This chapter will answer RQ1 from chapter 1 and present the state-of-the art in the field of fairness vs. accuracy tradeoffs. First, in section 3.1 we cover papers that study the tradeoff between accuracy and fairness in a general and theoretical manner. Then we present research using Pareto frontiers and Multi-Objective Optimization techniques to study fairness, in section 3.2.

3.1 The Tradeoff Between Accuracy and Fairness

There is a prevailing theory that fairness and accuracy are at odds with each other. In this section we will cover some relevant papers that cover this tension in the domain of automated decision-making systems.

Zliobaite [2015] study the relation between fairness and accuracy in binary classification. They measure discrimination, i.e. fairness, as the difference between the rates of acceptance. This means that in the extreme cases, where either everyone or no one is accepted, there is no discrimination, but also very low accuracy. Zliobaite introduces a normalized version of the discrimination measure and show that the upper bound for accuracy and discrimination remain linear. In practice the relation means that a classifier would reduce discrimination by either reducing the acceptance rate for the favored group, and/or increase the acceptance rate for the protected group. The acceptance rate is the rate at which data points are 'accepted', i.e. given the positive label. Classification thresholds control the acceptance rate. A low classification threshold means more data points are accepted, and a high threshold means that less data points are accepted. The resulting decrease in accuracy from changing the acceptance rate would be

linearly proportional to the discrimination in the data. The main message of the paper is that because changing the acceptance rate, i.e. the classification threshold, changes the baseline accuracy and discrimination, classifiers are not comparable unless acceptance rates are taken into account.

Zliobate's paper contribute a relevant element to consider when comparing classifiers, but otherwise the papers has a fairly narrow contribution. It only considers one measure of fairness and makes no attempt to generalize the results.

Menon and Williamson [2018] also study the cost of fairness in binary classification. They take a largely theoretical approach, looking at the inherent tradeoffs in learning classifiers with a fairness constraint, not specific to any algorithm. They formalize the *fairness-aware learning problem*, in which the goal is to predict the label well, the performance measure R_{perf}, while not being able to predict the sensitive feature well, the fairness measure R_{fair}. The objective is to minimize the function

$$R_{perf}(f; D) - \lambda \cdot R_{fair}(f; \overline{D})$$

for tradeoff $\lambda \in \mathbb{R}$. The tradeoff parameter λ determines how to balance the competing goals of fairness and accuracy. In their paper they investigate more closely a subset of such problems, that they call *cost-sensitive fairness-aware learning*, where cost-sensitive risks are used as a measure for both performance and fairness. They relate two popular fairness measures *disparate impact* and *mean difference* to cost-sensitive risks. Further they show that for such cost-sensitive fairness measures the Bayes-optimal classifier is an instance-dependent thresholding of the class-probability function. These results are largely theoretical. Lastly, they show that in fairness-aware learning problems, the tradeoff between fairness and accuracy is dependent on how 'dissimilar' the label and sensitive attribute is, the *disalignment*. For example, in a loan application process, if a sensitive attribute, e.g. gender, perfectly coincides with how well a person can repay their loans, then perfect accuracy would entail that only applicants of this gender would receive loans, and attempting to make the process fairer between genders, would result in a large cost to the accuracy. Meanwhile, if gender was perfectly independent from loan repayment capabilities, then it would be possible to have perfect accuracy and fairness simultaneously. This means that the more disalginment, i.e. independence, there are between the label and sensitive attribute the less of an effect introducing a fairness constraint will have on the accuracy.

Menon and Williamson's paper provide interesting theoretical results, but lack somewhat in real practical solutions. The tradeoff parameter λ present a clear picture of the concept, but unless the desired tradeoff is already determined,

tuning of this parameter is needed. Additionally, not every fairness problem can be reduced to what they call a *fairness-aware learning problem*. In some domains it is illegal to use sensitive attributes in the decisions making process, which the authors approach relies on.

Corbett-Davies et al. [2017] study the cost of fairness in algorithmic decision making, specifically in pretrial release decisions. Such algorithms predict defendants risk scores, which are supposed to indicate how likely they are to commit a violent crime. These algorithms don't use race explicitly as an input, but have still been shown to be substantially more likely to incorrectly classify black defendants than white. One such famous algorithms is COMPAS [Angwin et al., 2016]. Corbett-Davies et al. reformulate algorithmic fairness as constrained optimization, where the objective is to maximize public safety while satisfying formal fairness constraints designed to reduce racial disparities. They show that for many past definitions of fairness, the optimal algorithms require applying multiple, race-specific threshold for individuals' risk scores. However, the optimal *unconstrained*, i.e. safety maximizing, algorithm requires applying a single, uniform threshold for all defendants. Such a threshold satisfies another important understanding of equality, where all individuals are held to the same standard, irrespective of race. The difference between the constrained and unconstrained algorithms highlight the tension between fairness, i.e. reducing racial disparity, and accuracy, i.e. maximizing safety.

Corbett-Davies et al.'s paper has a slightly narrower scope then the above papers, which leads to more practical and specialized results. Though, the results might not necessarily be applicable in other domains, outside predictive risk scoring for pretrial release decisions. However, as the authors write themselves, the principles they discuss could be applied to other domains, and perhaps even to human decisions makers as well. The results are similar to the results from Menon and Williamson's paper [Menon and Williamson, 2018], in that they suggest that instance dependent thresholds (e.g. based on race, gender, etc.) are the optimal algorithms for optimization constrained by fairness goals. This means there are also similar problems to consider. One is the legality of using protected attributes as inputs for the algorithm. Another similar problem is the requirement of a predetermined, or alternatively having to tune, a fairness threshold parameter.

While the above papers consider the fairness/accuracy tradeoff in regard to classifiers and mitigation methods, Wick et al. [2019] take a slightly different approach by considering where the bias leading to unfairness might come from. In particular they focus on label and selection bias. They point out that when a mitigation method works to ensure fairness in the model using the training data, often no changes are being made to the evaluation data. It is therefore only natural that accuracy is decreased, when measured on the biased evaluation data. However,

the true accuracy of the classifiers, if it could be measured against unbiased data, might not be as drastically reduced as one might think. In some cases, they posit, increasing fairness might in fact increase the accuracy as well. If protected attributes in fact have no effect on a person's intelligence, potential, qualifications, etc., then it follows that enforcing fairness should normally increase the accuracy. Although, as they specify, such assumptions do not always hold. There might be historical, systematical, and/or biological reasons that can lead to different groups having different limitations or advantages. In which case results similar to those found in the papers mentioned above would likely hold instead. One concern when attempting to consider the origin of the bias is the dataset while evaluating the fairness/accuracy tradeoff, is that having a 'perfect', *ground truth* dataset is in most cases impossible. Wick et al. use data simulation to attempt to circumvent this problem and create a ground truth dataset. However, they acknowledge the lack of reliability, as there is no way of measuring how good the simulated data is. Though, if one assumes that this method works somewhat well, their results largely supporters their hypothesis.

A common through line in all the papers mentioned above is the use of group fairness measures. Speicher et al. [2018] consider individual fairness instead. The core idea of their paper is to use existing inequality indices from economics to measure fairness. In addition, they perform some comparative analysis, first comparing the tradeoff between accuracy and their notion of individual fairness, and secondly between individual and group fairness. The former analysis is the most relevant for this thesis. They make a proposition suggesting, similarly to Wick et al. [2019], that their notion of fairness should be in perfect harmony with accuracy, i.e. that eliminating error should mean the elimination of fairness as well. Like Wick et. al. they acknowledge this is often only true in special cases and not in general. Speicher et al.'s empirical analysis of the fairness vs. accuracy tradeoff use a generalized entropy index to measure individual fairness. See eq. (2.5) for the definition of generalized entropy indices. In their analysis they use the index with $\alpha = 2$. They experiment with two real-world datasets: The Adult Income Dataset [Dua and Graff, 2017], and the ProPublica COMPAS Dataset [Larson et al., 2016]. Three standard classifier models are used; logistic regression, SVM with RBF kernel, and random forest classifier, all of which are optimized for accuracy. In addition, they compare these classifiers with an 'oracle' that can perfectly predict the label for each instance. Then they use a classification threshold $0 \leq \tau \leq 1$ to predict a label for each instance based on the likelihoods output from the classifiers. Increasing the threshold results in more instances being rejected, and vice verca. In their experiment they vary this threshold to study the effect it has on accuracy and fairness. They find that for the oracle increasing τ produces results that support their original proposition, up until a point, after which the trend reverses (close to 0.75 in the Adult data

and 0.45 in the COMPAS data, corresponding to the fraction of instances in the negative class in the respective datasets). For the other (non-oracle) classifiers, the more general case is true. The optimal threshold for (imperfect) accuracy is not the same as the optimal threshold for fairness.

One mayor takeaway from this related research is that most of the papers use group fairness measures. This is not surprising, as such measures are more easily calculated, an essential part of being able to evaluate tradeoffs. The papers by Zliobaite [2015], Menon and Williamson [2018], and Corbett-Davies et al. [2017] all mostly examine classifiers and how to make them fair. They reach results that show that the optimal classifiers constrained by fairness requires multiple, group-specific thresholds for when to accept an individual. They all find that using such classifiers, however, reduces the accuracy. It also requires either a predetermined threshold parameter, or alternatively some tuning of the threshold. Menon and Williamson also highlight that the correlation between protected attributes and the label can have a large effect on how good or bad the tradeoff will be, in some cases (with no correlation) they suggest that there will be no tradeoff at all, and that both accuracy and fairness can be maximized. Wick et al. [2019] take this one step further, hypothesizing that if one assumes that an individuals protected attributes have no effect on their capabilities, then increasing fairness can in fact *increase* accuracy. They consider where the bias comes from, and claim that because most methods only change the model made from the training data, and continue to evaluate the accuracy on evaluation data that is likely also biased, it is no wonder that the new model does not fit the evaluation data as well as before. Their results support their hypothesis and show that the tradeoff increases or decreases based on how well this bias is mitigated in the evaluation data. Speicher et al. [2018] is the only paper that specifically considers an individual fairness metric. Similarly to Zliobaite [2015], Menon and Williamson [2018], and Corbett-Davies et al. [2017] they consider classification thresholds in their analysis. However, they take a more empirical rather than theoretical approach. Additionally, they only use one threshold for the whole population, rather than group specific thresholds. One might hypothesize that using group specific thresholds only holds for group fairness measures.

In this thesis we wish to investigate several of the findings discovered in this research. In addition, we believe using MOO and Pareto frontiers presents a different and more practical approach to the question of fairness vs. accuracy tradeoffs. This belief was further inspired by a paper by Haas [2019]. In the next section we will cover this paper and explore how MOO and Pareto frontiers have been used to study fairness previously.

3.2 Multi-Objective Optimization for Studying Fairness

Most research using multi-objective optimization (MOO) to study fairness is centered around resource allocation problems, like Internet bandwidth allocation, memory management, scheduling tasks, etc. Fairness concerns in this domain is centered around a fair allocation of resources, which entails using slightly different fairness measures then the ones we consider in this thesis. Bertsimas et al. [2019] attempt to quantify the 'price' of fairness relative to efficiency loss, in research allocation problems. They examine other types of fairness specific to the resource allocation domain and explore these is several common resource allocation problems. Additionally, some papers cover fairness in offspring diversity in MOO problems [Friedrich et al., 2011]. Both of these examinations of fairness are slightly out of the scope of this paper, and nothing more will be covered about this research. However, there is fortunately one paper which focuses on automated decision-making systems and use MOO to examine the fairness vs. accuracy tradeoff in such systems. This paper was a big inspiration, and we built on its results and research in this thesis.

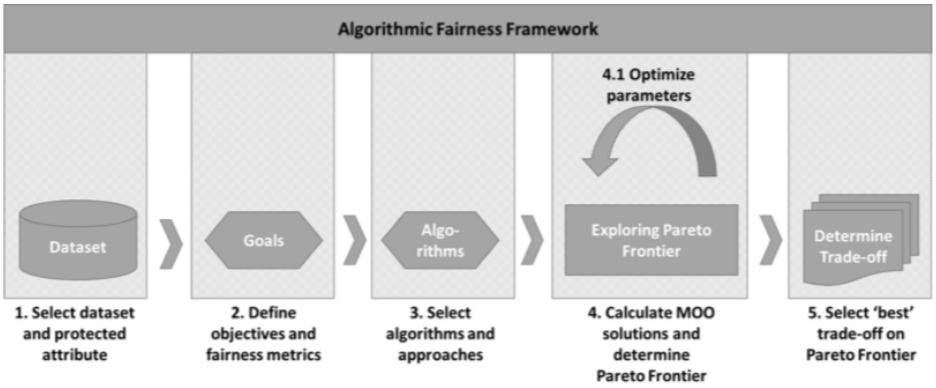


Figure 3.1: *General framework to explore algorithmic fairness tradeoffs (from [Haas, 2019])*

Haas [2019] rightly states that the tradeoff between fairness and accuracy (or other performance measures) hasn't been researched nearly enough. They present a generic framework for exploring these tradeoffs in algorithmic fairness. Additionally, they perform a case study to compare several fairness metrics and mitigation methods. The framework consists of five separate steps that can be seen in fig. 3.1. Step 1-3 consists of selecting the dataset and protected attributes, met-

rics and other objectives, and classifier algorithms and mitigation approaches. In step 4 the Pareto frontier is calculated using MOO. Based on the results from step 4, the 'best' tradeoff is selected in step 5 through analysis of the Pareto front.

In their case study they use the German Credit Card data set [Dua and Graff, 2017], with Age as the protected attribute, where <25 is considered young and part of the unprivileged group, and $\geq$ 25 not young and therefore part of the privileged group. They run two scenarios, one where the group fairness metric statistical parity difference [eq. (2.1)] is used, and one where the individual fairness metric Theil Index [eq. (2.6)] is used. In both scenarios AUC is used as the performance metric. SVM is used as the baseline classifier, while the pre-processing mitigation method Reweighing is used before SVM. The post-processing method Reject Option classifier (ROC) is used after SVM. Additionally, as an in-processing method, the Meta-Fair classifier is used as an alternative classifier. To determine the Pareto frontier, NSGA-II is used to select hyperparameters and features for the classifiers. This results in a Pareto front consisting of the different non-dominated classifiers based on the selected objectives (the fairness and performance metrics). 5-fold cross validation is used to determine the performance for the objectives. Table 3.1 summarizes the parameters for the case study.

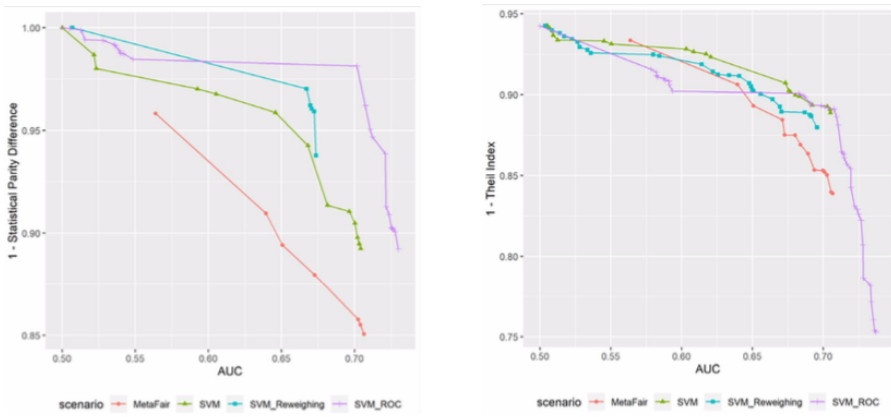
Figure 3.2 shows the resulting Pareto fronts from both scenarios. In the graphs the y axis shows the fairness metric, while the x axis shows the performance metric. For both metrics the 'ideal' value is 1, which indicates that the approach is perfectly 'fair' or perfectly 'accurate'. In addition to these Pareto fronts, they also investigated the distribution of values for other potentially relevant metrics.

Their results show that the framework can be used to systematically analyze the differences between fairness and performance metrics, by evaluating a set of strategies and metrics for a given data set. For example, the results show that for their case study, optimization for a group fairness metric resulted in observable differences between the different approaches, whereas optimization for an individual fairness metric resulted in much more similar behavior between approaches.

Though the framework shows great potential, there is yet more that can be done with it to explore more complex situations and tradeoffs. Haas only cover a small amount of possibilities, and as they state themselves, exploring more data sets, more metrics and more mitigation methods would be beneficial. We wish to leverage and build upon this framework in our thesis. Unfortunately, they don't provide their code from the case study, but our hope is that we will be able to somewhat recreate and further develop our own version based on

Parameter	Values	Description
Dataset	German Credit Data Set	[Dua and Graff, 2017]
Protected Attribute	Age	< 25: young; ≥ 25: not young.
Algorithms	SVM, $\text{SVM}_{Reweighing}$, SVM_{ROC}, $Meta - Fair$	The different mitigation approaches. See section 2.1.3 for further descriptions.
Metrics	Performance metric: AUC Fairness metrics: SP_Diff or $Theil$	The metrics used in the two case study scenarios. Scenario 1 uses AUC and Statistical Parity Difference, Scenario 2 AUC and the Theil Index. See section 2.1.1 for further descriptions.
NSGA-II Parameters	Generations: 100 Mutation rate: 0.05 Cross-over probability: 0.7 Population Size: 50	The parameters used for NSGA-II

Table 3.1: *Overview of parameters from Haas' case study. (adapted from [Haas, 2019])*



(a) Scenario 1 (b) Scenario 2

Figure 3.2: *Pareto fronts from Haas [2019] case study.*

the limited description given in the paper. Additionally, we believe there are some other avenues that can be explored using the framework Haas created.

Haas optimizes SVM hyperparameters and feature selection in order to generate different classifiers along the Pareto frontiers. Based on state-of-the-art research stating the importance of classification thresholds (see section 3.1), we want to investigate whether it is possible to optimize such thresholds instead of SVM parameters. Further we find that although Haas analyze their resulting frontiers in light of which final solution to choose, their analysis of the tradeoff itself is lacking. Therefore, we want to realize our own results using their framework and analyze them in light of the current state-of-the-art research in the fairness vs. accuracy tradeoff. In chapter 4 we will describe how our method and architecture differs from Haas' method. In chapter 5 we will describe the experiments we will perform in order to investigate these additional avenues of interest, as well as analyze and discuss our results.

Chapter 4

Method and Architecture

This chapter will describe our method and the architectures we are using to answer RQs 2 and 3 from section 1.1. RQ2 was somewhat answered by section 3.2 already where we discussed the paper by Haas [2019] that established a framework based on multi-objective optimization (MOO) to investigate the cost of fairness. In this thesis we wish to leverage the framework introduced by Haas. In our first experiment we will be implementing our version of this framework to provide results we can analyze and build upon in our following experiments. We will investigate a different data set from Haas and some different mitigation methods, to further answer RQ2 and further confirm the validity of the framework. As we discussed in section 3.1 much research in the field of fairness vs. accuracy tradeoffs highlight the importance of classification thresholds. This led us to formulate RQ3 related to optimizing classification thresholds. In order to answer this third research question our other experiments will therefore be adapting the method and architecture to study whether optimizing classification thresholds will give similar tradeoffs to optimizing classification parameters. Because of the slight difference between the goals of each experiment, a slight difference between the architectures are also required. The biggest difference is that experiments 2 and 3 require predefined and trained classifiers, that will be selected from experiment 1 results.

This chapter will first present the method that forms the basis for our approach in this thesis. Secondly, we will describe our method and how it differs from Haas' method. Thirdly, we will present the architectures needed to implement our method. Finally, we will cover relevant implementation details that will further explain how our architecture works.

4.1 Hyperparameter and Feature Selection for SVMs using Genetic Algorithms

When working with machine learning, two important steps of the process are feature selection and hyperparameter selection. Feature selection is used to identify the optimal number of features to use from the original dataset. More features mean more patterns to recognize, more time needed for computation, and more examples needed for learning. Therefore, the goal of feature selection is for the model to be able to generalize well with as few provided features as possible. Once the appropriate features have been selected, most machine learning classifiers require careful selection of parameters in order to tune the classifier to ensure it performs at its optimal capacity given the data provided. Such parameters, often called hyperparameters are specific to the type of classifiers used, for example the number of neighbors (k) in a k-nearest neighbors classifier or the maximum depth for a decision three classifier. Genetic algorithms (GAs) have been shown to be very efficient in automating this parametrization process [Huang and Wang, 2006]. Huang and Wang show that their proposed GA based method for SVM significantly improves results, when compared to another common parametrization option; Grid Search. As opposed to Grid Search, using GAs also allows for automated feature selection in the same process. Such GA based methods like the one suggested by Huang and Wang [2006], allows the feature selection and hyperparameter selection processes to be combined. In Huang and Wang's paper they use a single objective GA where they combine the criteria of classification accuracy, number of selected features and the feature cost into a single objective fitness function using weights. Another option would have been to use a multi-objective GA, with each criterion as an objective function. The latter is what we will be doing in this thesis but using an accuracy metric as one of the objective functions, and a fairness metric as another objective function.

In this thesis we will be using an SVM classifier. Similarly to Haas [2019] and Huang and Wang [2006] we will be using the radial basis function (RBF) kernel. The hyperparameters that should be optimized for SVM with an RBF kernel is the soft margin constant (penalty parameter) C, and the kernel parameter gamma (γ) used by RBF. For more on C and γ see section 2.4. In addition to optimizing these SVM hyperparameters we will optimize which features to include as well.

In order to implement this approach, the following decisions have to be made; (a) how to represent the hyperparameters and features as a chromosome, (b) how to design the crossover and mutation operators, and (c) how to generate the initial population. Our implementation of these decisions, as well as how this approach integrates into our method will described in section 4.2.

4.2 Method Description

Our method and architecture used in the thesis is based on the framework described by Haas [2019], which itself appears to be based on the method developed by Huang and Wang [2006]. Figure 3.1 shows Haas' framework. In this thesis we will be leveraging this framework, which includes five steps; (1) Select dataset and protected attribute; (2) Define objectives and fairness metrics; (3) Select algorithms and approaches (i.e. classifiers and mitigation methods); (4) Calculate MOO solutions and determine Pareto frontier; (5) Select 'best' tradeoff on Pareto frontier. As the goal for this thesis is to study tradeoffs, not to select a solution, we will be disregarding step 5. Step 1-3 entails selecting parameters for the experiments. Section 5.1 describes the dataset and protected attribute we have selected for our experiments (step 1). In section 5.2.1 we will describe which objectives, metrics, algorithms, and approaches we will be using (step 2 and 3). Step 4 is the main interest for this chapter, chapter 4. This step is where all the calculations are done to determine the Pareto frontier.

Step 4 states; Calculate MOO solutions and determine Pareto Frontier. In order to perform this step Haas [2019] uses the method created by Huang and Wang [2006] to optimize SVM hyperparameters and feature selection in order to generate classifier solutions. However, instead of optimizing a single objective like Huang and Wang, Haas optimizes two objectives, one fairness objective and one accuracy objective. In order to perform this optimization a MOO algorithm is therefore needed. Haas uses the NSGA-II algorithm. See section 2.3.1 for more on NSGA-II.

In experiment 1, we will be optimizing parameters and features for SVM, similarly to Haas [2019] and Huang and Wang [2006]. Meanwhile in experiment 2 and 3, we will be optimizing classification thresholds for trained classifiers. These trained classifiers will be selected based on results from experiment 1. Therefore, our architecture will differ slightly from experiment 1 to experiment 2 and 3. Further descriptions of the experiments can be found in section 5.2.

In the first experiment our method and architecture will be fairly similar to Haas's method. However, their architecture description is incomplete and limited. It is therefore not possible to be sure that the architecture is perfectly recreated. Additionally, because we will be reusing classifiers found in experiment 1 for experiments 2 and 3, some additional changes have been made. These changes mainly consist of using a defined train/test split rather than the k-fold method used by Haas. Furthermore, we will be using a different dataset, different mitigation methods and a different accuracy metric. For more on the choice of dataset, and the train/test split see section 5.1. In section 5.2.1 we will we discussing which mitigation methods and metrics we use and explain why the changes were made.

In sections 4.3 and 4.4 we will further describe our architectures. First, we will cover the main decisions that influenced the architecture.

In section 2.3 we outlined the four major pillars that define evolutionary algorithms: (1) maintenance of a population; (2) creation of diversity; (3) a selection mechanism (for selecting the best parent chromosomes to evolve into a new generation); and (4) a process of genetic inheritance. Pillar (1) includes making decisions about how to design the chromosome and represent the hyperparameters and features in experiment 1, and represent the classification thresholds in experiment 2 and 3. Each chromosome will represent a potential solution, i.e a set of hyperparameters and features or a set of classification thresholds that represent unique SVM classifiers. In pillar (1) we also include the question of how to generate the initial population of chromosomes. Pillar (2) is determined by which EA is used. Similarly to Haas [2019] we will be using the NSGA-II algorithm. The NSGA-II algorithm uses *crowding distance sorting* to create diversity in the population. For more on crowding distance sorting and NSGA-II see section 2.3.1. Pillar (3) involves two steps: (a) calculating fitness scores for the population; and (b) selecting the 'best' chromosomes based on the fitness scores. Step (b) is again determined by NSGA-II, where chromosomes are selected using *elitism*, *fast nondominated sorting* and crowding distance sorting. For further descriptions on the selection procedure used in NSGA-II see fig. 2.4 and section 2.3.1. Step (a) is where the SVM classifiers and mitigation methods are introduced. In this step the fairness and accuracy objectives need to be calculated in order to provide fitness scores NSGA-II can use to select the best chromosome. Pillar (4) ensures the process of genetic inheritance by determining what type of crossover and mutation operators to use. For more on crossover and mutation operators see section 2.3.

Our decisions for pillars (1), (3a) and (4) forms the basis for our architectures. The choice to use NSGA-II determined the other pillars; (2) and (3b). The descriptions of our architecture will be split into two parts, covering the architecture for experiment 1 and the architecture for experiment 2 and 3 respectively. Both descriptions consist of four main sections based on the three decisions. First, we will cover our choice for pillar (1) by describing the chromosome design. Then we will cover pillar (4) by describing our choice for genetic operators. After the preliminary description of the chromosome and genetic operators we will describe the main loop of our architecture and how everything connects. Finally, we will present the evaluation function that calculates fitness scores for pillar (3a).

4.3 Architecture for Experiment 1

In experiment 1 we will be optimizing hyperparameters and feature selection for SVM. In section 4.2 we outlined the main choices that defines our architecture. First, we will cover our chromosome design for experiment 1. Secondly, we will describe our chosen genetic operators. Thirdly, we will describe the main loop of the architecture and how everything connects. Finally, we will present the evaluation function used to calculate the fairness and accuracy objectives we wish to optimize for in our experiment.

4.3.1 Chromosome Design

In section 2.3 we covered the importance of the chromosome design as each chromosome represents a potential solution to the problem at hand. First, we will describe the chromosome representation we use to represent the solutions. Secondly, we will describe how the initial population of chromosomes is generated.

Chromosome Representation

As stated in section 2.3 the representation of a chromosome can be split into two parts, the *genotype* and the *phenotype*. The genotype is the encoding of the solution that is used by the evolutionary algorithms, while the phenotype is the encoding that represent the real-world solution. In our architecture, the phenotype for experiment 1 is the real valued parameters for SVM and the feature mask representing the features to keep. There exist several options for how to represent this phenotype as a genotype. We have chosen a very simple option, the binary encoding system. This allows for easy implementation of the genotype, as a list of bits is all that is required. In addition, genetic operations are easy to perform on this type of encoding.

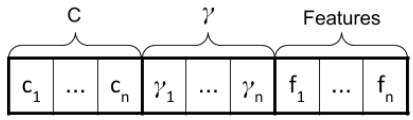


Figure 4.1: *The chromosome for experiment 1 consists of three parts; C, gamma (γ) and the feature mask.*

Using binary encoding one section each of the chromosome will represent the C,

gamma(γ) and the feature mask, as seen in figure 4.1. In fig. 4.1, $c_1...c_n$ represent the value of parameter C, $\gamma_1...\gamma_n$ represent the value of γ, and $f_1...f_n$ represent the feature mask. n_c is the number of bits chosen to represent the value of C, n_γ is the number of bits for γ and n_f is the number of bits representing the features. The values of n_c and n_γ are chosen according to the calculation precision wanted for the value C and γ, and n_f must equal the number of features in the data set.

To decode the feature mask genotype, we use the same method as Huang and Wang [2006] that influenced Haas [2019], where the bit value of '1' represents that the feature is kept, and a bit value of '0' represents that the feature is discarded. In [Huang and Wang, 2006] they use a simple scaling method to decode the genotype of the bit representations of C, γ, which requires many bits to allow a decent amount of precision, as the desired value range increases. Instead we have chosen to decode the bit representations of C and γ using the concept of floating-point numbers to allow for more precision using fewer bits. Symbolically the value of a floating-point number is:

$$\frac{s}{b^{p-1}} * b^e$$

where s is the significand, p is the precision (i.e. the number of digits in the significand), e is the exponent, and b is the base (in our case this will be the number 2.

In the IEEE standardized binary representation of floating-point numbers, the bits are split into three parts. One bit is used to represent whether the value is positive or negative, a set of bits is used to represent the value of the exponent, and the rest is used to represent the significand. Figure 4.2 shows a binary representation where five bits are used to represent the exponent and ten bits are used to represent the significand, in addition to the one bit used for the sign.

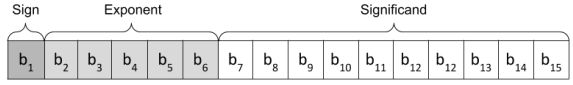


Figure 4.2: *IEEE standardized binary representation of floating-point numbers using 16 bits.*

In our case we can disregard the sign bit, as all our values have to be positive values. The number of bits used to represent the exponent defines the range the number can represent, and the number of bits used to represent the significand

defines the precision. To calculate the decimal value of our binary representation the following equation is used:

$$(\sum_{n=0}^{p-1} *2^{-n}) * 2^{e}$$

In cases where the exponent range is smaller or larger than the desired parameter range, the exponent value is scaled to fit into the desired range.

Generating the Initial Population

Generating the initial population is an important step in EAs, as the initial population sets the basis for which further generations are evolved. It is therefore important to have a diverse initial population. In our case we randomly generate the initial population, placing a random number of 1 and 0 bits into each chromosome at random positions.

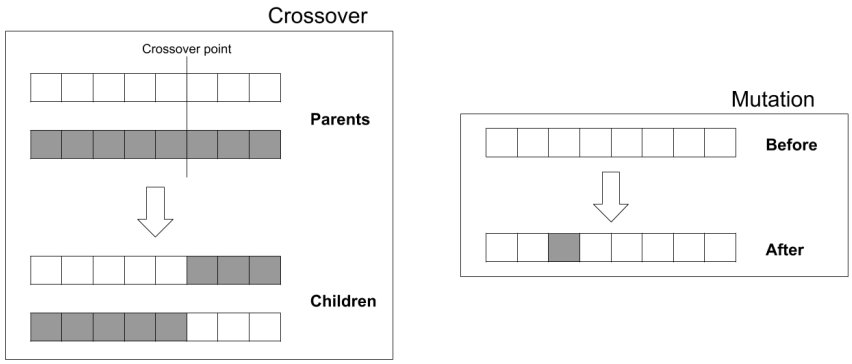


Figure 4.3: *The genetic crossover and mutation operations used in this thesis.*

4.3.2 Genetic Operators

In order to evolve from the initial population genetic operators are needed. In section 2.3 we described the importance of genetic operators and covered the main procedure of the two genetic operators; crossover and mutation. Because we have chosen to use binary encoded chromosomes, crossover and mutation can be done fairly simply. In our case we decided that the crossover operation will

choose a random crossover point at which to split the parents into two parts, where the opposite parts from each parent is used to generate two children. Our mutation operator chooses a random bit in the chromosome and flips the value of the bit. Figure 4.3 provides a visual representation of how these operators work.

4.3.3 The Main Loop

The architecture for experiment 1 is largely inspired by the architectures from Haas [2019] and Huang and Wang [2006]. However, as previously discussed, some changes have been made to ensure that the results from experiment 1 will work with experiments 2 and 3.

Algorithm 3 Experiment 1 - Main Loop

1: $train\_set, test\_set = get\_dataset\_split()$
2: $P_t = create\_population(n)$
3: $t = 0$
4: **repeat**
5: $\quad P_{t+1} = P_t + generate\_children(P_t)$
6: $\quad fitness\_scores = evaluate\_population(P_{t+1}, train\_set, test\_set)$
7: $\quad P_{t+1} = select\_population(P_{t+1}, fitness\_scores)$
8: $\quad t = t + 1$
9: **until** $t == num\_generations$
10: $return\ get\_pareto\_frontier(P_{t+1})$

The pseudocode for the program can be seen in algorithm 3. The main flow consists of two initial steps, performed at lines 1 and 2 of algorithm 3 respectively; (a) Gathering and splitting the dataset, as described in 5.1; and (b) Initializing the population of chromosomes with population size n, as described in 4.3.1. After these two initial steps, the program runs in a loop until some termination condition is met (in our case; a maximum number of generations). In this loop three main steps are performed, in addition to a termination check requiring the iteration of a generation counter ($t = t + 1$) in line 8 in algorithm 3. The three main steps are:

1. *Generate children.* In this step the chromosome 'children' are generated from the selected 'parents'. This is line 5 in algorithm 3. The parents are selected at the end of the loop, in step 3. Therefore, in the first iteration, the initial population is the parents. The children are generated using the genetic operators described in section 4.3.1. These children are then added to the parent population. In this step the population size is increased with the number of children. The population size at this point is therefore usually

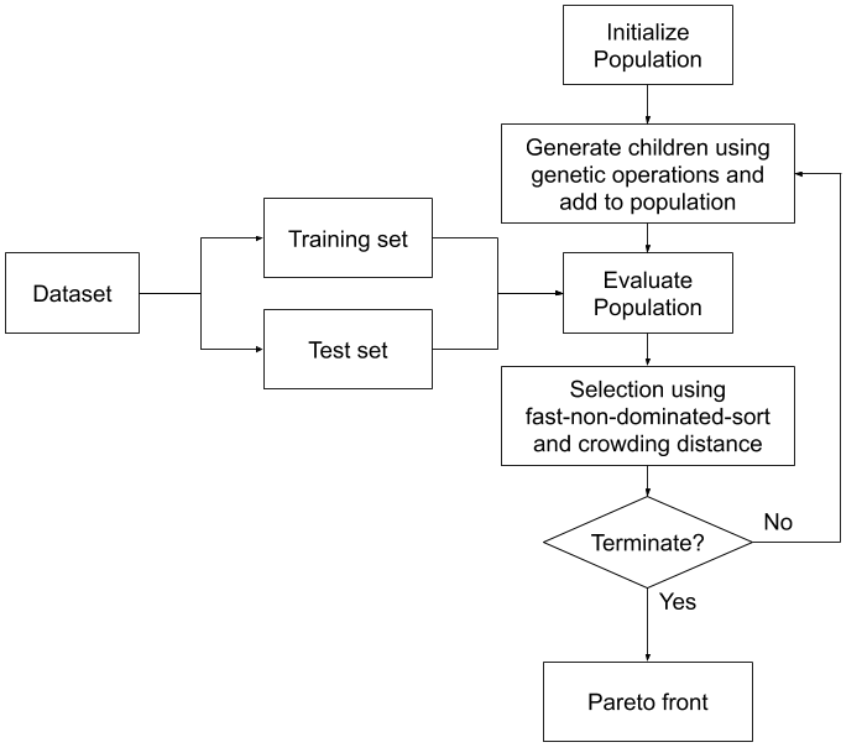


Figure 4.4: *The main flow of the architecture for experiment 1.*

$2n$, as n children are generated. However, because it is possible for some children to be identical to some parents (either because the crossover and mutation probabilities did not hit, or because they coincidentally became equal through the random crossover and mutation operators), the amount of children might be slightly smaller than n because we discard non-unique children to avoid duplicate calculations.

2. *Evaluate population.* This is line 6 in algorithm 3. In this step the population of $2n$ chromosomes are evaluated and fitness scores are calculated. The fitness score for each chromosome is a tuple consisting of a fairness score and an accuracy score. The split data set is introduced in this second step and used to evaluate the population. A more detailed description of this process and how the fairness and accuracy scores are calculated will follow

in our description of the evaluation function.

3. *Selection.* In this step the best chromosomes from the population are selected using principles from NSGA-II; elitism, fast nondominated sort and crowding distance. For further description of this procedure see section 2.3.1. This selection is used to reduce our population of $2n$ to the desired number of n by choosing the 'best' chromosomes from the set of parents and children. This is performed at line 7 in algorithm 3 and returns the n chromosomes that will be used as parents in the next generation.

4. *Terminate?* Finally, if the termination condition is met, i.e. the maximum number of generations, the Pareto front from the final population is returned. Otherwise the program returns to step (1), line 5 in algorithm 3. The frontier is returned in line 10 in algorithm 3.

The main flow of the architecture can be seen in fig. 4.4.

4.3.4 The Evaluation Function

The second step of the main loop (line 6 in algorithm 3), the evaluation of the chromosome population, is where our SVM classifier and mitigation approaches are used. In order to evaluate the population, the classifier and the desired mitigation method needs to be ran for each chromosome in order to calculate accuracy and fairness scores. Figure 4.5 shows the architecture of the evaluation function that runs for each chromosome in the population. Pseudocode for the evaluation function can be seen in algorithm 4. The function takes as input; a chromosome; and the training and test data sets. The main steps of the evaluation architecture are:

1. *Optional: Perform mitigation algorithm.* The step is optional, as we will also be running one approach without mitigation algorithms as a baseline. If this step is performed a mitigation algorithm is used on the training set. In our experiments we will only be using pre-processing mitigation methods which means that this step needs to be performed before training the classifier. If other types of mitigation algorithms were to be used the architecture would need to be expanded to allow for mitigation algorithms both during and after training as well. This step is performed at line 5 in algorithm 4.

2. *Scaling.* In the second step both the training and test sets are scaled using the following function:

$$x_{scaled} = \frac{x - \bar{X}}{s} \tag{4.1}$$

Algorithm 4 Experiment 1 - Evaluation function

1: **Input:** $chromosome, train\_set, test\_set, accuracy\_metric, fairness\_metric$
2: **Output:** $fitness\_scores$
3: **Procedure:**
4:
5: $train\_set = perform\_mitigation\_method(train\_set)$ (Optional)
6: $train\_set, test\_set = perform\_scaling(train\_set, test\_set)$
7:
8: $keep\_features, C, gamma = get\_phenotype(chromosome)$
9: $train\_set, test\_set = feature\_reduction(train\_set, test\_set, keep\_features)$
10: $classifier = train\_SVM(C, gamma, train\_set)$
11:
12: $classifications = test\_classifier(test\_set)$
13: $accuracy\_score = accuracy\_metric(classifications)$
14: $fairness\_score = fairness\_metric(classifications)$
15:
16: $return\ [accuracy\_score, fairness\_score]$

where $\bar{X}$ is the mean of the training samples, and s is the standard deviation of the training samples. This step is performed at line 6 in algorithm 4.

3. *Convert genotype to phenotype.* In this step the chromosome genotype is converted to phenotypes that can be used by the evaluation function. The genotype is converted as described in section 4.3.1. This step is performed at line 8 in algorithm 4.

4. *Keep feature subset.* The phenotype representing the feature mask is in this step used to keep the chosen feature subset and discard the other features from the training and test sets. This step is performed at line 9 in algorithm 4.

5. *Train SVM classifier.* In this step the phenotypes of the values C and γ are used as parameters for the SVM classifier that is trained using the training set. This step is performed at line 10 in algorithm 4.

6. *Test SVM classifier.* In this step the trained classifier is used to predict the probabilities of each label for the test data points. These probabilities are then assigned labels using a classification threshold of 0.5, where probabilities over 0.5 is assigned the positive label and probabilities under 0.5 is assigned the negative label. This step is performed at line 12 in algorithm 4.

7. *Calculate metrics.* Finally, we calculate the desired accuracy and fairness

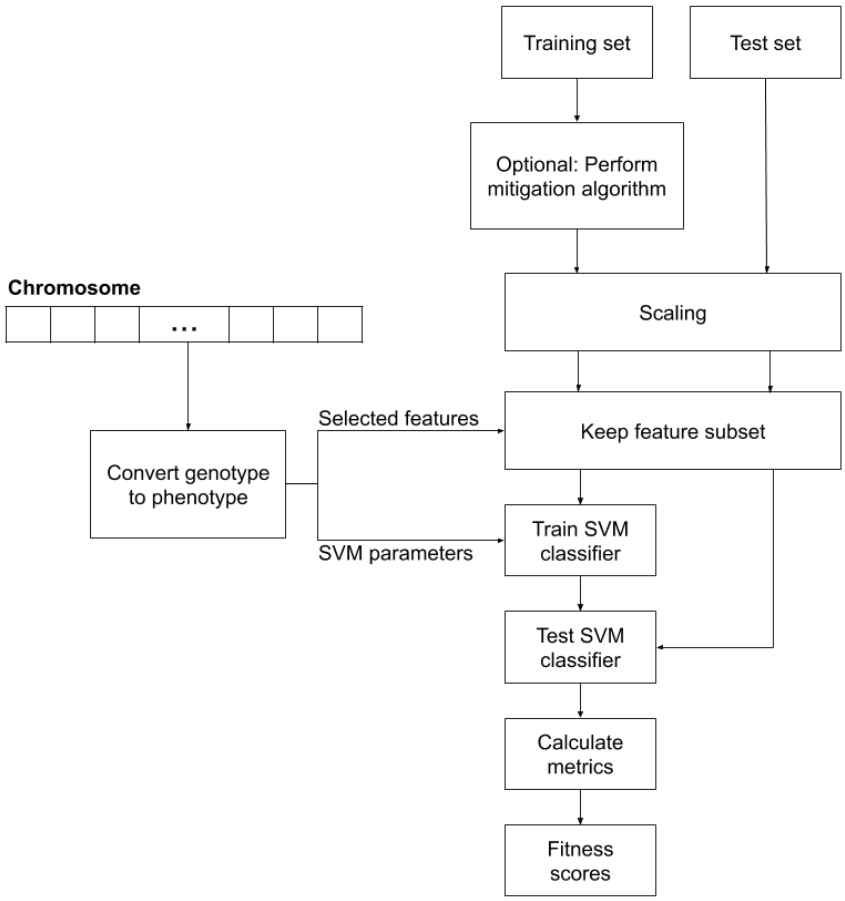


Figure 4.5: *The evaluation function used in experiment 1 to calculate fitness scores for each chromosome in the population.*

metrics based on the predictions from the former step. These steps are performed at line 13 and 14 in algorithm 4. These metrics are then returned, at line 16 in algorithm 4, as the fitness scores for the chromosome.

In experiment 1 a classification threshold of 0.5 is used to label the data points in the test set. The classifier outputs the predicted probability of a data point belonging to the positive label, and if this probability is over 50% the data point

is given the positive label. Else, it is given the negative label. This is a common default value for classification threshold as it follows the logic that the data point is given the label it is most likely to belong to. In experiment 2 and 3 we will be optimizing the classification threshold(s), rather than maintaining the standard 0.5 used in experiment 1.

4.4 Architecture for Experiments 2 and 3

In experiments 2 and 3 we will be using a select few of the classifiers generated in experiment 1. These classifiers will be used to output predictions for the data points in the test set. These predictions will be labeled using classification thresholds that will be optimized using the MOO method. In experiment 2 a single classification threshold will be optimized. Meanwhile, experiment 3 will optimize group specific thresholds that will used to classify data points depending on what group they belong to (i.e. the privileged or unprivileged group). In section 4.2 we outlined the main choices that defines our architecture. This section will follow the same structure as section 4.3 that described the architecture for experiment 1. Some of the architecture will be similar to experiment 1, but there are some changes which we will cover in this section. First, we will cover our chromosome design and genetic operators for experiments 2 and 3. Secondly, we will describe the main loop of the architecture and how everything connects. Finally, we will present the evaluation function used to calculate the fairness and accuracy metrics we wish to optimize for in our experiments.

4.4.1 Chromosome Design and Genetic Operators

Compared to experiment 1 the chromosome design and genetic operators mostly remain the same. Similarly to experiment 1 the initial population is generated randomly. Additionally, the genetic operators are identical. Further descriptions of these procedures can be found in section 4.3.1 and section 4.3.2 respectively. However, because we will be optimizing classification thresholds and the SVM hyperparameters and selected features are already selected based on results from experiment 1, the chromosome representation will differ. In experiments 2 and 3 the solutions are represented by the classification threshold rather than features and SVM hyperparameters.

Chromosome Representation

For experiments 2 and 3 the phenotype of the chromosome is the classification thresholds. The chromosomes will be encoded using the binary encoding system. Using this system one section each of the chromosome will represent each

classification threshold (τ_i) for experiments 2 and 3, as seen in fig. 4.6.

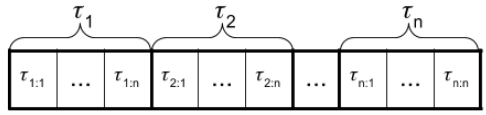


Figure 4.6: *The chromosome for experiment 2 and 3 consists of one or more classification thresholds (τ).*

In fig. 4.6 $\tau_{i:1}...\tau_{i:n}$ and represent the value each classification threshold, τ_i. n_{τ_i} is the number of bits chosen to represent the value of τ_i. The value of n_{τ_i} is chosen according to the calculation precision required. To decode the the bit representations of τ we use the same procedure as described in fig. 4.1 to generate values in the range $[0, 1]$.

4.4.2 The Main Loop

Algorithm 5 Experiment 2 and 3 - Main Loop

$train\_set, test\_set = get\_dataset\_split()$
$train\_set = perform\_mitigation\_method(train\_set)$ (Optional)
$train\_set, test\_set = perform\_scaling(train\_set, test\_set)$
$train\_set, test\_set = feature\_reduction(train\_set, test\_set)$
$classifier = train\_classifier(train\_set)$
$P = create\_population()$
$t = 0$
repeat
    $P_{t+1} = P_t + generate\_children(P_t)$
    $fitness\_scores = evaluate\_population(P_{t+1}, classifier, test\_set)$
    $P_{t+1} = select\_population(P_{t+1}, fitness\_scores)$
    $t = t + 1$
until $t == num\_generations$
$return\ get\_pareto\_frontier(P_{t+1})$

In experiment 2 and 3 the goal is to optimize classification thresholds. Based on the results from experiment 1 classifiers will be chosen that will be used in experiment 2 and 3. This difference changes the architecture slightly from experiment 1 to experiments 2 and 3. The main difference being that the classifiers are trained before entering the main loop, using the selected hyperparameters and features

from experiment 1. Figure 4.7 shows the architecture for these experiments and the pseudocode for the main loop can be seen in algorithm 5.

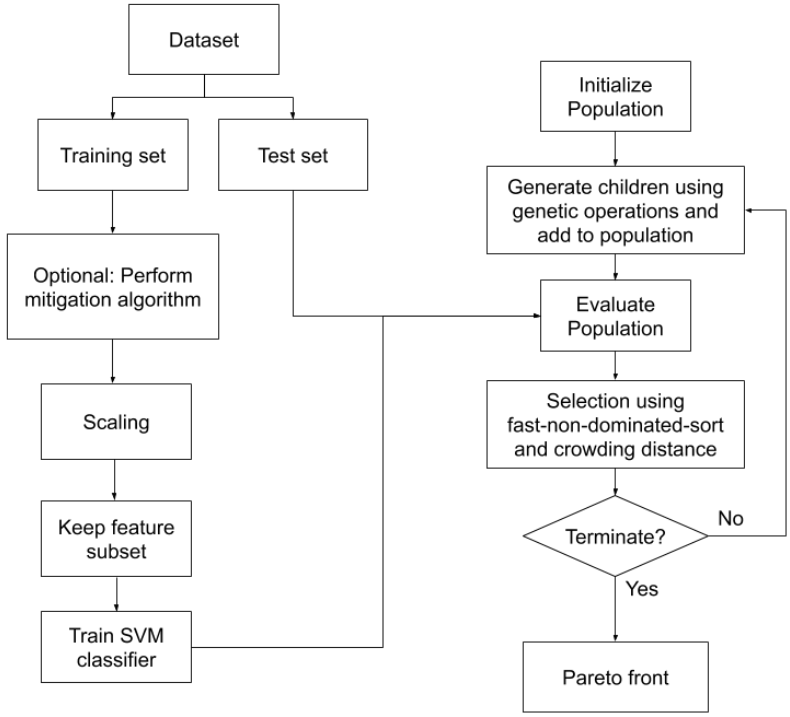


Figure 4.7: *The main flow of the architecture for experiments 2 and 3.*

The main flow of the architecture consists of two initial steps. Step (a) is similar to the evaluation function from experiment 1, up until the testing of the classifiers:

1. *Split data.* First, the data set is split into training and test sets. Line 1 in algorithm 5.
2. *Optional: Perform mitigation algorithm.* Secondly, the optional mitigation algorithm is performed on the training set. Line 2 in algorithm 5.
3. *Scaling.* The training set is scaled using eq. (4.1). Line 3 in algorithm 5.
4. *Keep feature subset.* In the final step of pre-processing the chosen feature subset is kept, and the rest discarded. This time a predefined subset based on results from experiment 1 is used. Line 4 in algorithm 5.

5. *Train SVM classifier.* Finally, the classifier is trained. This time using predefined parameters based on results from experiment 1. Line 5 in algorithm 5.

The classifier resulting from this process as well as the test set is then provided to the evaluation function in the main loop. Step (b) is to generate the initial population (line 6 in algorithm 5). The main loop is identical to experiment 1, the only difference being the evaluation function used in the 'evaluate population' step.

4.4.3 The Evaluation Function

Algorithm 6 Experiment 2 and 3 - Evaluation function

1: **Input:** $classifier, chromosome, test\_set, accuracy\_metric$
2: $fairness\_metric$
3: **Output:** $fitness\_scores$
4: **Procedure:**
5:
6: $test\_set = perform\_scaling(test\_set)$
7: $test\_set = feature\_reduction(test\_set)$
8: $classification\_thresholds = get\_phenotype(chromosome)$
9:
10: $classifications = test\_classifier(test\_set, classification\_thresholds)$
11: $accuracy\_score = accuracy\_metric(classifications)$
12: $fairness\_score = fairness\_metric(classifications)$
13:
14: $return\ [accuracy\_score, fairness\_score]$

The architecture of the evaluation function for experiments 2 and 3 can be seen in fig. 4.8. Pseudocode for the evaluation function can be seen in algorithm 6. The main steps of the evaluation function are:

- *Scaling.* First, the test set is scaled using the same scaler used on the training set. Line 6 in algorithm 6.
- *Keep feature subset.* Secondly, the chosen feature subset is kept identically to the training set. Line 7 in algorithm 6.
- *Convert genotype to phenotype.* In this step the chromosome genotype is converted to phenotypes representing the one or more classification thresholds. Line 8 in algorithm 6. The genotype is converted as described in section 4.4.1.

- *Test SVM classifier.* In this step the phenotype values of the classification threshold(s) are used alongside the trained classifier to predict probabilities for the samples in the test set. If one classification threshold is used the samples are assigned labels based on whether the probability of belonging to the positive label are above or below the threshold. If group specific thresholds are used, the samples belonging to a specific group are assigned labels based on whether the probability of belonging to the positive label is above or below the threshold specific to this group. This step is performed at line 10 in algorithm 6.
- *Calculate metrics.* Finally, we calculate the desired accuracy and fairness metrics based on the predictions from the former step. These calculations are performed at line 11 and 12 in algorithm 6. These metrics are then returned, at line 14 in algorithm 6, as the fitness scores for the chromosome.

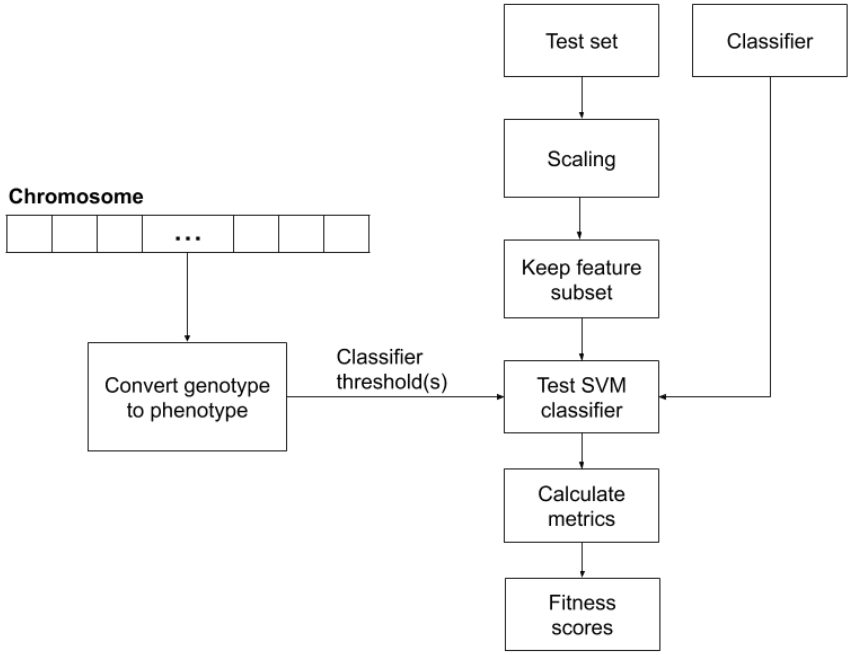


Figure 4.8: *The evaluation function used in experiments 2 and 3 to calculate fitness scores for each chromosome in the population.*

4.5 Implementation Details

The code implementing the architectures described in sections 4.3 and 4.4 was version controlled using `git` and is saved in a Github repository[1]. The code has been written in Python3 and run using version 3.7.3. The necessary Python modules and the version of them we used to run our code is listed in table 4.1. In addition, modules `datetime`, `random`, `json`, `path`, and `os` are used from the Python Standard Library.

Module	Version	Notes
`numpy`	1.16.4	
`scikit-learn` (`sklearn`)	0.21.2	
`aif360`	0.2.3	
`pandas`	0.24.2	At the time of writing our code appears incompatible with the newest versions of `pandas`. We believe this is because of the `pandas` version required by `aif360`.
`matplotlib`	3.1.0	

Table 4.1: *List of Python modules required to run our code.*

We chose to use the popular machine learning tool `scikit-learn` in this thesis. Other popular machine learning tools, like `Tensorflow`, `Keras` and `PyThorch`, were considered but dismissed as options, either for being unnecessarily complex for our use case or simply because we had more experience using `scikit-learn`. For more on `scikit-learn` see section 2.5. The bias mitigation tool used for this thesis is the AI Fairness 360 Open Source Toolkit (or `aif360`). There is not yet an extensive amount of options when it comes to choosing bias mitigation tools, as the area of research is still fairly new. `Aif360` is currently the only viable option, with an extensive set of mitigation algorithms and fairness metrics. For more on `aif360` see section 2.1.4. `Numpy` was used to represent and perform calculations on the chromosomes in our NSGA-II implementation. It is also a requirement for several of the other modules. `Pandas` was a requirement for use of the `aif360` module. At the time of writing our code appears incompatible with the newest versions of `pandas`. We believe this is because of the `pandas` version required by `aif360` not being compatible with newer versions. `Matplotlib` was used to plot the results from our experiments.

Instructions on how to run the code can be found in the `README.md` file in the Github repository[1].

[1] `https://github.com/pernillej/Cost-of-Fairness`

SVM Settings

In addition to providing the C and γ parameters to our SVM classifiers, we also set a maximum number of iterations and a seed for the random number generator. The latter is provided to ensure that the code produces the exact same classifier every time given the same feature subset, SVM parameters and training data. This allows us to be sure that when reusing classifiers from experiment 1 in experiment 2 and 3, these classifiers are the same. In addition, it allows us to speed up the evaluation process in experiment 1. Because this process requires training an SVM classifier and optionally using a mitigation algorithm, the process can be computationally expensive. If we know that the same parameters, i.e. the same chromosome will produce the same classifier, we can check whether an identical chromosome has been evaluated before and simply reuse those scores instead of having to rerun the evaluation function unnecessarily. We know this example will occur because of the elitism of NSGA-II where children and parents are evaluated together.

We set the maximum number of iterations to 10 000, after some experimentation. Because we know that research suggest that the COMPAS data set is neither linearly nor non-linearly separable (as discussed in section 5.1), we know that there is a possibility that the SVM classifier will try to fit the training set indefinitely (or at least for a very long time), especially for some values of C and γ. We wish to ensure that our experiments finish and don't run for an unnecessarily long time. Therefore, the maximum number of iterations is set. Setting this number to 10 000, seems to provide sufficient results, as our initial results seem to align with those previously found for the data set.

NSGA-II implementation

Due to previous experience implementing NSGA-II and the simplicity of the chromosome representation, a decision was made to implement the NSGA-II elements of our code ourselves (with some help from online sources [Allen, 2019]). This ensured we were able to get our experiments up and running fairly quickly, and therefore more time could be afforded to integrating `sklearn`'s SVM classifier with mitigation methods from `aif360` which proved more difficult than expected. Given more time a natural next step would be to implement the NSGA-II elements using modules such as `DEAP`. This would make the code more scalable long term.

Chapter 5

Experiments and Results

In chapter 4 we described the architecture used for the three experiments we run in order to answer RQs 2 and 3 described in section 1.1. In this chapter we will be answering these RQs by presenting our experiment plan, covering the experimental setup, and analyzing and discussing the results from the experiments. First, we will describe the dataset we will be using in our experiments, The COMPAS data set. Secondly, we will present the experiment plan, including the setup and parameters used for the experiments, as well as a detailed description of each of our three experiments. Finally, we will present, analyze and discuss the results from each experiment consecutively, starting with experiment 1 and ending with experiment 3.

5.1 The COMPAS Data Set

In this thesis we will be using the *ProPublica* COMPAS data set [Larson et al., 2016]. This data set was made famous after ProPublica published an article describing how a US predictive recidivism system (called COMPAS) was biased against black people [Angwin et al., 2016]. To predict recidivism means to predict whether a person will commit crime again after being released from prison or on bail. These predictions, often called risk assessment scores, are increasingly used in courtrooms to inform decisions about who to set free at every stage of the criminal justice system.

The COMPAS data set consist of records of criminal defendants from Boward County, Florida, over a two-year period, where the defendants either recidivated or didn't. The COMPAS system generates several scores including 'Risk of Re-

cidivism' and 'Risk of Violent Recidivism'. In this thesis we will be looking at the 'Risk of Recidivism'. The task will be to predict whether a criminal defendant will recidivate (negative class) or not (positive class), based on features such as current charge degree and prior offenses. The sensitive attribute for this type of task can be either or both; gender (Male or Female) and race (Black, Hispanic, White). In this thesis we will be viewing only the race attribute as the sensitive attribute, where 'White' (i.e. 'Caucasian') is the privileged group, and the other races (i.e. 'Not Caucasian') are combined into the unprivileged group.

5.1.1 Data Set Preprocessing

The `aif360` package provides a pre-processed version of the COMPAS data set. The `aif360` package performs the same pre-processing as the original analysis by ProPublica [Larson et al., 2016]. This includes removing rows with missing data and selecting only the most relevant features. Additionally, `aif360` uses one-hot-encoding to encode the categorical features. Before one-hot-encoding the COMPAS data set provided by `aif360` includes 10 features in addition to the label; `sex`, `age`, `age category`, `race`, `juvenile felony count`, `juvenile misdemeanor count`, `juvenile other count`, `priors count`, `charge degree`, and `charge description`. These are the same features used by Dressel and Farid [2018] for their human assessment. We will come back to Dressel and Farid's paper when we analyze the data set. Because `aif360`'s one-hot-encoding of the `charge description` feature increases the feature set substantially (from 1 feature before encoding to 389 features), we have chosen to drop the `charge description` feature, as we find it provides an unnecessary complexity to the data set. Ideally one would circumvent `aif360`'s automatic one-hot-encoding for this feature. However, we found that removing this feature entirely did not substantially impact our results, if at all. This leaves us with a data set containing 12 features, after the age category and charge degree features have been one-hot-encoded. After removing instances (data points) with missing data the data set contains 6172 data points before being split.

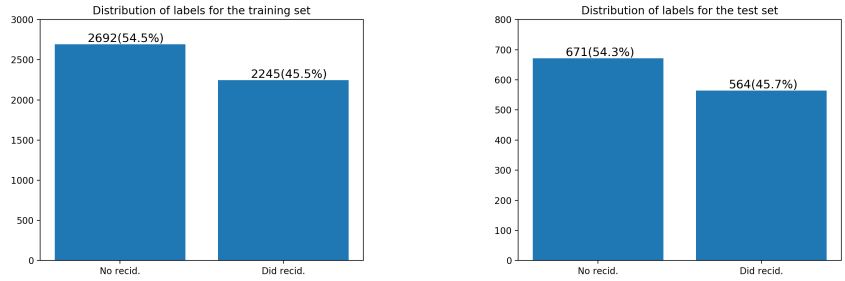
5.1.2 Data set Analysis

We split the data set into an 80/20 split for the training set and test set. After some analyzing of the data set, we found that simply splitting the data set at the data point separating the first 80% and the last 20% maintains the distribution of both the label and the protected attribute, race. Therefore, no random shuffling of the data points was needed. Details about the split can be seen in table 5.1.

As can be seen in fig. 5.1 the label distribution of the data set is maintained in

Data set name	# of features	# of data points	% of total data set
Training Set	12	4937	80%
Test Set	12	1235	20%
Total Data set	**12**	**6172**	**100%**

Table 5.1: *The table shows the two data set splits with their respective amounts of data points and percentage of the total amount of data.*



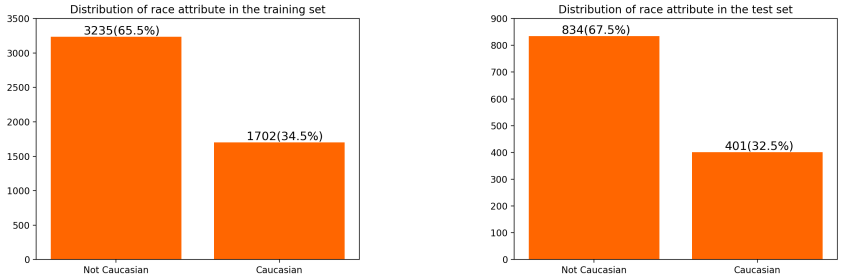
(a) Training set

(b) Test set

Figure 5.1: *Label distributions in the training and test sets. 'No recid.' is the favorable label, while 'Did recid' is the unfavorable label.*

the test set, with both the training and a test set containing circa 54% 'No recid.' labels, and 46% 'Did recid' labels. However, we can also see that there is a slightly uneven distribution between 'No recid.' (positive label) and 'Did recid' (negative label), with the former label having more cases than the latter. Because 'race' is our sensitive attribute, we also analyzed the distribution of this attribute in the data set. Again, we can see that the 'race' distribution is equal in the training in test set. This time however, there is substantially more cases of one of the values. 'Not Caucasian', the unprivileged group, is represented at roughly twice the amount of 'Caucasian', the privileged groups, with 65-68% 'Not Caucasian' to 32-35% 'Caucasian' in both data sets. Figure 5.2 shows the distribution.

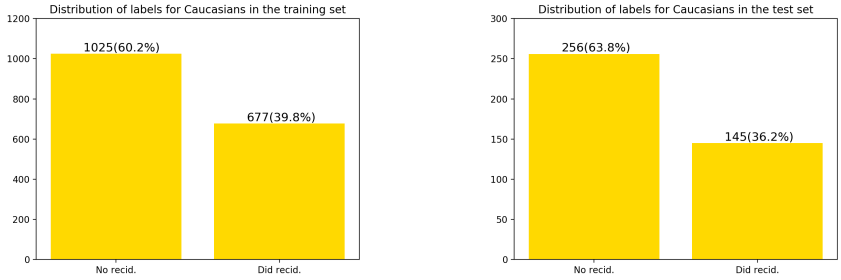
We also investigated the label distribution within each group. Figure 5.3 and 5.4 shows the label distribution for the privileged group and unprivileged group respectively. Once again, we find that the distribution is equal across the training and test sets. We can therefore conclude that the data set split is good. Additionally, we can see that for the unprivileged group the distribution of 'No recid.' and 'Did recid.' is nearly equal at around 50%. However, for the privileged group



(a) Training set (b) Test set

Figure 5.2: *Race distribution in the training and test sets. 'Caucasian' the the privileged group, while 'Not Caucasian' is the unprivileged group.*

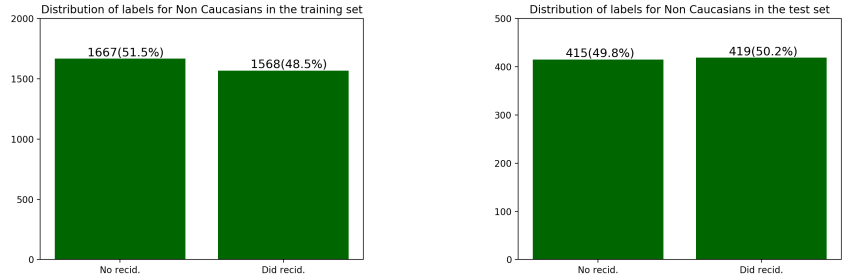
the cases are labeled as 'No recid.' almost twice as much as 'Did recid.' with 60-64% 'No recid.' to 36-40% 'Did recid' in both data sets. Given these distributions it is perhaps not surprising that ProPublica journalists found the system to be biased against black people.



(a) Training set (b) Test set

Figure 5.3: *Label distribution for the privileged group (Caucasians) in the training and test sets. 'No recid.' is the favorable label, while 'Did recid' is the unfavorable label.*

The COMPAS data set has received attention in some other research as well as the ProPublica article [Angwin et al., 2016]. Corbett-Davies et al. [2017] and Speicher et al. [2018] use the COMPAS data set in their analysis of fairness vs. accuracy tradeoffs. For more on these papers see section 3.1. Dressel and



(a) Training set

(b) Test set

Figure 5.4: *Label distribution for the unprivileged group (Not Caucasians) in the training and test sets. 'No recid.' is the favorable label, while 'Did recid' is the unfavorable label.*

Farid [2018] found that the COMPAS system is no more accurate or fair than predictions made by people with little or no criminal justice experience. They find that a simple linear predictor and non-linear predictors using substantially less features then COMPAS yields similar prediction results both in regard to accuracy (roughly 66%) and fairness. This led them to conclude that the data is not separable, neither linearly nor non-linearly. Given the complexities of crime and the real world in general, this finding is perhaps not surprising. We can therefore expect to find similar accuracy scores in our experiments, with the highest accuracies reaching values around 66%.

5.2 Experiments

In this section we will describe the three experiments we will be performing in this thesis. First, we will cover the experimental setup, including which metrics we will use, which mitigation algorithms are added, and other parameters for the experiments. Then we will describe each experiment.

5.2.1 Experimental Setup

Dataset and Protected Attribute

Haas [2019] uses the German Credit Data Set [Dua and Graff, 2017] to test their framework. We found, however, that for our purposes the German Credit Data Set had too few instances (only 1000), and the label distribution was very

uneven. Instead we decided to use the COMPAS Data Set, with 6172 instances and a more even distribution of labels. For a more detailed analysis as well as a description of the pre-processing performed on the dataset, see section 5.1. The protected attribute we will use is the 'race' attribute. The 'race' attribute is split into two groups; the privileged group 'Caucasian', and the unprivileged group 'Not Caucasian'.

Metrics

Similarly to Haas [2019] we decided to run two scenarios in regards to the metrics. One with a group fairness metric and the other with an individual fairness metric. We decided to use the same two metrics as Haas, as these metrics are widely used and popular metrics. The group fairness metric is *Statistical Parity Difference*, eq. (2.1), and the individual fairness metric is the *Theil Index* metric, eq. (2.6). For the accuracy metric we had do deviate from Haas. Because we will be investigating the use of classification thresholds in our final two experiments, we cannot use metrics like AUC, as these use prediction probabilities to calculate the score. Instead we use a simple binary accuracy metric which calculates the percentage of correct classifications. The accuracy is calculated using the following equation;

$$Accuracy = \frac{TP + TN}{P + N} \tag{5.1}$$

where TP and TN are true positives and true negatives respectively, and P and N are the number of positives and negatives in the dataset (which means adding them equal the amount of data points in the dataset). This accuracy metric will be used to compare against both fairness metrics, which means we will have two scenarios to run for each experiment:

Scenario 1: Statistical Parity [eq. (2.1)] vs. Accuracy [eq. (5.1)]

Scenario 2: Theil Index [eq. (2.6)] vs. Accuracy [eq. (5.1)].

Algorithms

To investigate the effect of different mitigation methods our experiments will consider four approaches/algorithms for both scenarios:

1. The baseline: An SVM classifier with no added mitigation methods.
2. SVM + Reweighing: The Reweighing algorithm is performed on the training data before it is used to train an SVM classifier.

3. SVM + Disparate Impact Remover: The Disparate Impact Remover algorithm is performed on the training data before it is used to train an SVM classifier.
4. SVM + Optimized Pre-Processing: Optimized Pre-Processing is used on the dataset before it is used to train and test and SVM classifier.

For each of these four algorithms NSGA-II is used to optimize the feature selection and SVM parameters. Descriptions of each mitigation method can be found in section 2.1.3. All three mitigation methods used are pre-processing methods performed before training the classifier. A decision was made to only use pre-processing methods because we know that they will not distort the effect of classification thresholds. This is because the pre-processing methods are performed before the SVM classifier is trained and the classifier outputs predictions that are labeled using the classification thresholds. Using in-processing and post-processing methods would require that all methods be able to output prediction probabilities rather than direct predictions of the label. This was not possible using the `aif360` module, where all available methods output predicted labels directly without the ability to specify classification thresholds.

Dataset name	# of features	# of data points	% of total dataset
Training Set	10	4222	80%
Test Set	10	1056	20%
Total Dataset	**10**	**5278**	**100%**

Table 5.2: *The table shows the two dataset splits of the Optimized Pre-Processing version of the COMPAS dataset with their respective amounts of data points and percentage of the total amount of data.*

A note about the Optimized Pre-Processing mitigation method: While the two other mitigation algorithms are performed on the COMPAS dataset described in section 5.1, this method requires importing a pre-processed COMPAS dataset from `aif360`, processed according to the Optimized Pre-Processing method. In the paper presenting the Optimized Pre-Processing method [Calmon et al., 2017] one of the datasets they perform their method on is in fact the COMPAS dataset. Details about the resulting version of the COMPAS dataset as well as the training/test split can be seen in table 5.2.

NSGA-II parameters

In order to run NSGA-II four parameters need to be set: Number of generations to run; The size of the chromosome population; The rate of mutation; and the crossover probability. We will be using the same parameters as Haas [2019].

Parameters	Value
Number of generations	100
Population size	50
Mutation rate	0.05
Crossover probability	0.7

Table 5.3: *The NSGA-II parameters for our experiments*

We found that these parameters work well for our exploration purposes, but given a real-world problem increasing both the population size and number of generations would perhaps be beneficial. The length of the chromosome (at least one with bits such as ours) dictates the amount of possible variations, with 2^n variations where n is the length of the chromosome. The longer the chromosome, the larger the population size and number of generations should be. Although with long chromosomes one would never be able to near the amount of variations, the nature of evolutionary algorithms is such that a sufficient solution could still be found because of the targeted evolution that is involved.

Chromosome length: For each experiment the chromosome is represented as described in sections 4.3.1 and 4.4.1 respectively. For the first experiment we will be using a chromosome length of 42, with 15 bits each for the C and γ values, and 12 bits to represent the 12 features of our COMPAS dataset (this will be 10 bits for the SVM + Optimized Pre-Processing algorithm, giving a chromosome length of 40). The C value will be decoded into a range of $[\frac{1}{2^{16}}, 2^{16}]$, while the γ value will be decoded into a range of $[\frac{1}{2^{10}}, 2^3]$.

For the second and third experiments the length of each chromosome section representing a classification threshold will be 10 bits. This means that for experiment 2 with only one threshold the length will be 10, while for experiment 3 with two thresholds the length will be 20. The thresholds will be decoded into a range of [0, 1].

Summary

Table table 5.4 summaries the parameter and experimental setup for our experiments.

Parameter	Values	Description
Dataset	The COMPAS Data Set	See section 5.1
Protected Attribute	Race	Privileged: Caucasian Unprivileged: Not Caucasian
Algorithms	SVM, $\text{SVM}_{Reweighing}$, $\text{SVM}_{DisparateImpactRemover}$, $\text{SVM}_{OptimPreProcessing}$	The different algorithms. See above for further descriptions.
Metrics	Performance metric: *Accuracy* Fairness metrics: SP_Diff or $Theil$	The metrics used in the two case study scenarios. Scenario 1 uses Accuracy [eq. (5.1)] and Statistical Parity Difference [eq. (2.1)], Scenario 2 Accuracy [eq. (5.1)] and the Theil Index [eq. (2.6)].
NSGA-II Parameters	Generations: 100 Mutation rate: 0.05 Cross-over probability: 0.7 Population Size: 50	The parameters used for NSGA-II
Chromosome length	Experiment 1: 42(40) bits Experiment 2: 10 bits Experiment 3: 20 bits	The length of the chromosome for each experiment. See above for further descriptions.

Table 5.4: *Summary of parameters for our experiments.*

5.2.2 Experiment 1 - Optimizing SVM Parameters and Feature Selection

In experiment 1 the feature selection and SVM hyperparameters will be optimized for each of our algorithms described above in section 5.2.1. First, we will run both tradeoff scenarios with the four algorithms, five times each. We will compare the five results for each algorithm with each other to see if the tradeoff varies between each run, and to test whether the NSGA-II parameters are sufficient. The 'best' (chosen randomly if they are fairly equal) Pareto front for each algorithm will then be presented and compared to each other. These fronts will then be analyzed further.

From the resulting Pareto Frontiers, we will select the most 'accurate' and the most 'fair' classifiers for each of the algorithms. These classifier configurations will then be used in experiment 2 and 3.

5.2.3 Experiment 2 - Optimizing a Classification Threshold

In experiment 2 we will be optimizing a single classification threshold. Previous research, discussed in section 3.1, suggest that using a single classification threshold is optimal for accuracy, at the cost of fairness. For more on this see section 3.1. We will be considering this statement by investigating whether optimizing a single classification threshold provides similar or dissimilar tradeoffs to the ones seen in experiment 1. For both tradeoff scenarios we will be running classifiers for each algorithm optimized for accuracy and fairness respectively (i.e. the resulting classifiers from experiment 1), to study whether the same tradeoff can be reached given such differing starting points. Another interesting question is whether an already trained classifier can be 'course corrected' using classification thresholds and provide the same results as a classifier trained to optimize either fairness or accuracy.

5.2.4 Experiment 3 - Optimizing Group Specific Thresholds

In our final experiment we will be optimizing group specific thresholds. Previous research, discussed in section 3.1, suggest that using a group specific classification threshold is optimal for fairness (or at least group fairness), at the cost of accuracy. For more on this see section 3.1. In our case group specific thresholds means two thresholds, one for the privileged group, and one for the unprivileged group. Similarly to experiment 2 we will be running classifiers for each algorithm optimized for accuracy and fairness respectively. We will explore whether using group specific thresholds will give a different tradeoff as opposed to using a single threshold.

5.3 Results and Analysis

In section 5.2 we described our experimental setup and outlined the purpose and details of the three experiments we will run in order to answer RQs 2 and 3 from section 1.1. In this section we will be presenting, analyzing and discussing the results from these experiments. The section will cover each experiment consecutively, starting with experiment 1 and ending with experiment 3.

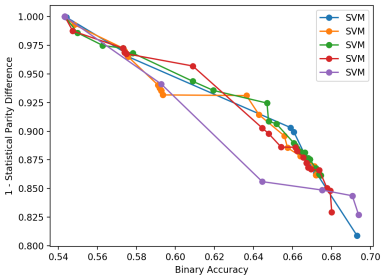
The results are presented using graphs showing the Pareto frontiers, in addition to tables displaying various interesting values. While the optimal score for both the Theil Index and statistical parity difference is zero, all graphs displaying frontiers displays the score as |1 - fairness metric|, where a score of 1 on the y-axis indicated 'perfect' fairness according to the used metric. On the x-axis the accuracy is shown, with a score of 1 indicating 'perfect' accuracy, i.e. no

errors. Because the goal is to maximize both these values the frontiers should look similar to the example frontier shown in fig. 2.3. Each point on the graphs represents one chromosome/solution that generated those exact fairness and accuracy scores. The points therefore represent either the hyperparameter and feature subset combinations used for SVM in experiment 1 or the classification thresholds in experiments 2 and 3. Tables will be used to present the values for these combinations for some relevant solutions. The Pareto frontiers represent the tradeoff found by an algorithm. A good tradeoff means that an increase or decrease in one value doesn't have a large cost on the other objective. If there exist no cost to accuracy when introducing fairness, all frontiers should contain only a single point on the graph with a high fairness score and a high accuracy score. Otherwise, the frontiers should contain several points, were some are more fair then others, but therefore has a lower accuracy. The smaller the cost to each value is at the expense of increasing the other, the better the tradeoff.

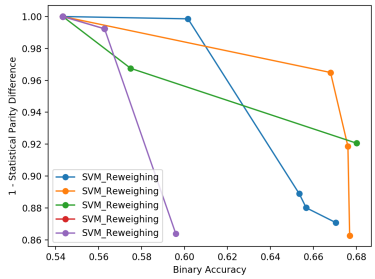
5.3.1 Experiment 1 - Optimizing SVM Parameters and Feature Selection

In experiment 1 the goal is to optimize feature selection and hyperparameters for an SVM classifier with regards to a fairness metric and an accuracy metric. In addition to the baseline SVM classifier, three different pre-processing mitigation methods are used, resulting in four different algorithms; The baseline SVM, and three SVM + mitigation method algorithms. For further descriptions of the four algorithms, see section 5.2.1. A description of the architecture used to perform the experiment can be found in section 4.3. The experiment is run for two different scenarios depending on the fairness metrics we wish to optimize for. In scenario 1 a group fairness metric in addition to the accuracy metric is optimized; Statistical Parity [eq. (2.1)] vs. Accuracy, [eq. (5.1)]. In scenario 2 an individual fairness metric is optimized in addition to the accuracy metric; Theil Index [eq. (2.6)] vs. Accuracy [eq. (5.1)].

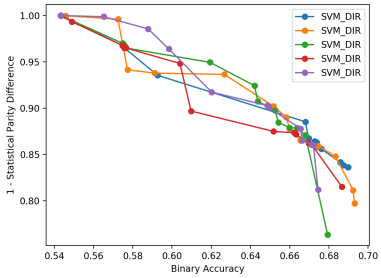
In the first step of experiment 1, all four algorithms were run five times for each scenario. The results from scenario 1 can be seen in fig. 5.5, while the results from scenario 2 can be seen in fig. 5.6. The goal of this first step is to analyze whether the NSGA-II parameters we selected (see table 5.3) sufficiently provides the MOO algorithm with opportunity to search the solution space. Additionally, we will select the 'best' front from each algorithm for further analysis and comparison in step 2 of the experiment. Ideally all the frontiers for a specific algorithm should be fairly identical, and at least display no great deviance from each other. This means that it should be difficult to differentiate the frontiers from each other, and that they should all have reached roughly the same values for both fairness and accuracy. In fig. 5.5, the frontiers in figures (a) and (d), representing the SVM



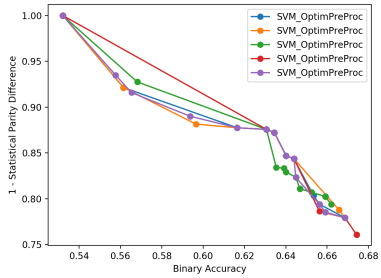
(a) SVM



(b) SVM + Reweighing



(c) SVM + Disparate Impact Remover



(d) SVM + Optimized Pre-Processing

Figure 5.5: *Experiment 1 - Results from five runs of all algorithms for scenario 1: Statistical Parity Difference vs. Accuracy*

and SVM + Optimized Pre-Processing algorithms respectively, all follow roughly along the same general arch. Meanwhile, figure (c), representing the SVM + Disparate Impact Remover algorithm, shows a bit more divergence between the frontiers. However, figure (b), representing the SVM + Reweighing algorithm, displays the largest amounts of divergence between fronts. Additionally, we can see that this algorithm generates much fewer points on the frontiers, i.e. the frontier is very sparse as opposed to the other algorithms with many more points on the frontiers.

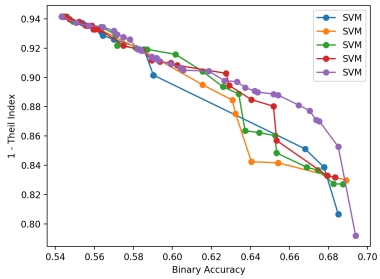
There are two options for reasons as to why SVM + Reweighing produces such varying and sparse frontiers. One reason might be that the NSGA-II parameters need to be adjusted, in particularly increasing the number of generations and the population size. Doing so would broaden the spectrum of the search and might therefore find more possible solutions. Because we know that the Reweighing

method generates instance weights for the training set (see section 2.1.3), this additional complexity during training might need additional search space to optimize. The second reason might be that the Reweighing method and the statistical parity metric simply combines to create such sparse variance, at least when hyperparameters and features selection is optimized. Exactly what about the combination that could give such a result is unclear from our immediate analysis and would require further examination. Likely, the interaction with the weights generated by the Reweighing method combined with the hyperparameters and selected features limits the variance of possible predicted labels. This option is supported by what we can see in fig. 5.6, which is that the same algorithm but run with the Theil Index fairness metric instead of the statistical parity metric produces much more uniform and abundant frontiers. In either case, we can see that at least a few of the frontiers produced by SVM + Reweighing in scenario 1, score within the general values that the other algorithms found.

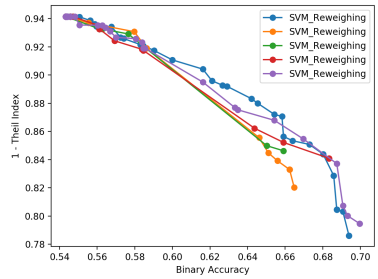
In order to further compare each algorithm with each other we chose the 'best' frontier from each algorithm for comparison. For SVM, SVM + Disparate Impact Remover, and SVM + Optimized Pre-Processing this choice is fairly arbitrary as the frontiers are roughly equal. For SVM the blue frontier was chosen. For SVM + Disparate Impact Remover the orange frontier was chosen. And for SVM + Optimized Pre-Processing the red frontier was chosen. For SVM + Reweighing, the choice is not as arbitrary. The orange frontier was chosen because even though it doesn't strictly dominate neither blue or green, only one point on each blue and green stops orange from strictly dominating them. The chosen frontiers for scenario 1 can be seen in fig. 5.7. Before we analyze this figure, however, we will look at the results from step 1 for scenario 2.

Figure 5.6 shows the results from running each algorithm five times on scenario 2: Theil Index vs. Accuracy. For this scenario we can see that no algorithms stand out like SVM + Reweighing did for scenario 1. All algorithms produced fairly equal frontiers, however, in this case the baseline SVM algorithm is the one with the most divergence. In fact, the purple frontier is nearly strictly dominant on the lower fairness scores. This means that purple frontier appears to be the best candidate to represent the SVM algorithm during further comparison and is therefore the one we selected. For SVM + Reweighing the purple frontier was chosen. For SVM + Disparate Impact Remover the purple frontier was again chosen. And for SVM + Optimized Pre-Processing the red frontier was chosen.

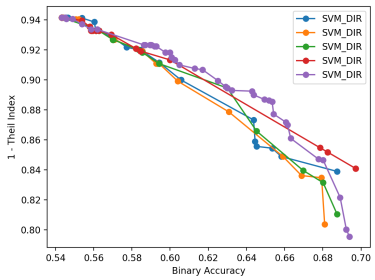
Based on the results from step 1 for both scenarios we can conclude that the chosen NSGA-II parameters (see table 5.3) sufficiently provided results for further analysis. However, the SVM + Reweighing algorithm produced an overly sparse set of frontiers in Scenario 1. Therefore, if our method were to be used for actual decision making, more investigation would be needed to pinpoint the proper



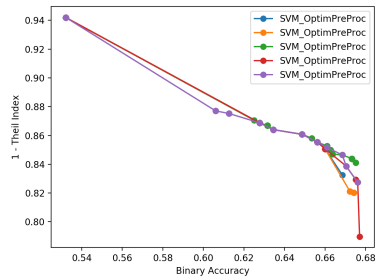
(a) SVM



(b) SVM + Reweighing



(c) SVM + Disparate Impact Remover



(d) SVM + Optimized Pre-Processing

Figure 5.6: *Experiment 1 - Results from five runs of all algorithms for scenario 2: Theil Index vs. Accuracy*

parameters. However, for our purposes they are sufficient. As we shall discuss later, the fairness and accuracy scores produced are within the expected ranges. However, even though EAs in general can rarely be expected to produce identical frontiers each and every run, especially in our case where the genetic operators are random, increasing the number of generations and the population size could be beneficial.

With the chosen frontiers from step 1, we will now compare algorithms with each other. Additionally, we will further analyze each frontier and discuss the type of tradeoffs that can be observed. This will help answer RQ2 from section 1.1. Figure 5.7 shows the four chosen frontiers from scenario 1, and fig. 5.8 shows the four chosen frontiers from scenario 2.

It can be immediately concluded from both figures 5.7 and 5.8 that our architec-

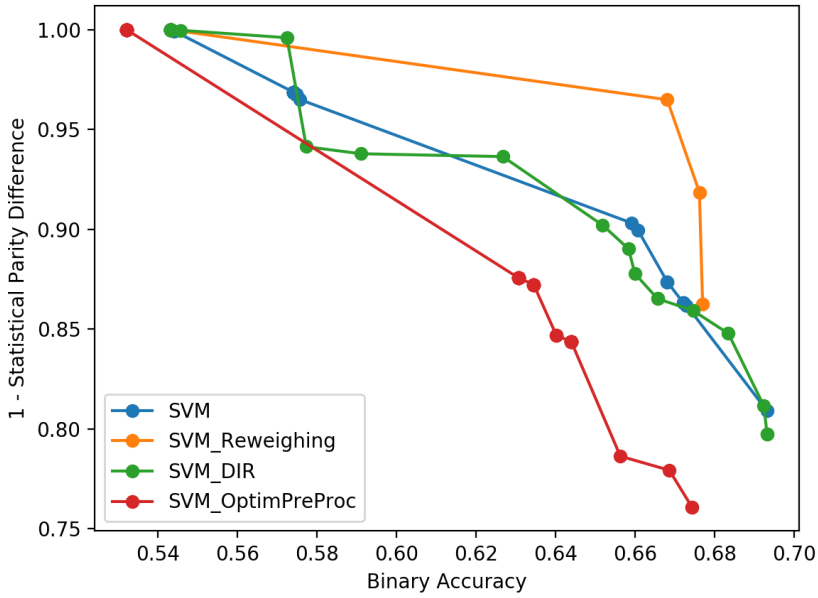


Figure 5.7: *Experiment 1 - The 'best' fronts from all algorithms for scenario 1: Statistical Parity Difference vs. Accuracy*

ture, as described in section 4.3, works. Both figures show the resulting frontiers that can be used to make decisions on what type of algorithm and metrics to use when building fair automated decision-making systems. This result adds validity to the framework originally proposed by Haas [2019] to investigate tradeoffs between fairness and accuracy. In addition, it can immediately be seen that a tradeoff exists. In fig. 5.7 it can be seen that while all the algorithms are able to reach accuracy scores at around 0.68, the fairness scores reach just around 0.80 in that case. Meanwhile, the algorithms are all able to reach a 'perfect' fairness score of 1, but at the cost of the accuracy score which is then lowered to around 0.55. In fig. 5.8 similar results can be seen. All algorithms are able to reach a fairness score of just over 0.94 for the Theil Index, but at the cost of the accuracy score which is lowered from around 0.68 to 0.55 again.

While the results from Haas [2019] (see fig. 3.2) are not directly comparable to ours, as they use a different dataset as well as a different set of mitigation methods, the general overarching conclusion remains the same: A higher fairness score

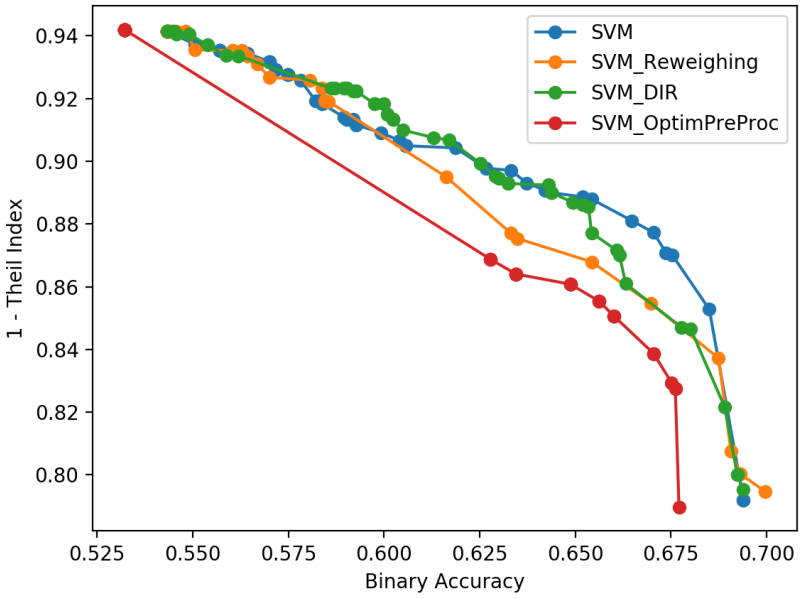


Figure 5.8: *Experiment 1 - The 'best' fronts from all algorithms for scenario 2: Theil Index vs. Accuracy*

comes at the cost of accuracy. These results are supported by other previous work in this field as well. Zliobaite [2015], Menon and Williamson [2018] and Corbett-Davies et al. [2017] all conclude that fairness comes at the cost of accuracy. Wick et al. [2019] and Speicher et al. [2018] also reach similar conclusions, but with a caveat. They both suggest that in theory the notion of fairness (especially individual fairness which Speicher et al. advocate for) should in fact be in perfect harmony with accuracy and not come at the cost of one another. However, they both acknowledge that in most practical cases this proposition doesn't hold. Our results support this fact. Wick et al. take their reasoning of perfect harmony one step further to consider the pre-existing bias in the dataset. They submit that if the accuracy was measured against unbiased data their proposition would hold, but because such unbiased data is a rarity, this is hard to prove. The COMPAS data set used in our experiment is already known to be biased, see section 5.1, therefore our experiment falls in this same pitfall. Attempting to circumventing this pitfall is an interesting direction for further work.

When directly comparing the individual algorithms in figures 5.7 and 5.8, an interesting result is that for both scenarios the SVM + Optimized Pre-Processing algorithm is nearly strictly dominated by all other algorithms, except for a single point from the SVM + Disparate Impact Remover algorithm in scenario 1. Where these results to be used to choose the 'optimal' solution based on the desired tradeoff, one could therefore easily eliminate the SVM + Optimized Pre-Processing algorithm from the competition. The three other algorithms are less easily distinguishable. In fig. 5.7 the SVM + Reweighing algorithm displays less of a tradeoff in the higher fairness scores, but is not able to reach as high accuracy scores as SVM and SVM + Disparate Impact Remover. In fig. 5.8 this is nearly reversed, with SVM + Reweighing reaching the highest accuracy score, but there is less of a divergence from the other two algorithms in this scenario. However, the purpose of our thesis is not to choose an 'optimal' solution and we will therefore conclude our discussion around this decision. Later, however, we will use our results to select the most fair and the most accurate solution for each algorithm in each scenario. These will be used in experiments 2 and 3.

If we look at the actual scores reached in each scenario some interesting results can be seen. The accuracy scores for the most accurate solutions range from 67% to 70% which is low considering the COMPAS system is used to make decisions that affect people's lives. Dressel and Farid [2018] reach accuracies of at most 65% to 67% in their experiments, using both linear and non-linear classifiers. This led them to conclude that the COMPAS data set is not separable, neither linearly nor non-linearly. Our accuracies are slightly higher, but this can likely be written of to the fact our dataset split is not the same as the one used by Dressel and Farid. In fact, they test their classifiers over 100 random 80/20 splits, which is likely what leads to the slightly lower accuracy. Even though we only use a non-linear classifier, SVM with the radial bias function (RBF) kernel, our similar accuracy scores support the conclusion that the COMPAS data set is at least not non-linearly separable.

In addition to the low maximum accuracy scores, the accuracy scores are reduced to around 55% when the fairness score is at its maximum. This supports a statement made by Zliobaite [2015] pointing out that in cases where either no one or everyone is accepted there is no discrimination because everyone receives the same outcome, i.e. 'perfect' fairness. However, it also means that the accuracy is very low. If a dataset contains exactly 50% of each binary label, this would mean that the accuracy would be exactly 50%, either because the negative labels where falsely accepted, or the positive labels were falsely rejected.

Table 5.5 shows the number of data points and percentage of both labels in the test set. The table shows that the percentage of positive labels in the test set is 54%. The accuracy score if all data points were accepted would therefore

Label	# of data points	% of test set
Positive	671	54%
Negative	564	46%
All	**1235**	**100%**

Table 5.5: *The table shows the number of data points for the positive and negative label in the test set as well as the percentage.*

be 54%, which is very close to our average of 55% and therefore supports the statement made by Zliobaite [2015]. If all were rejected the score would be 46%. Because both situations would likely give a fairness score of 1, the case where all are accepted will therefore dominate the case where all are rejected. We can therefore hypothesize that for all our algorithms the most fair solution accepts nearly every data point. An interesting effect of this conclusion is that because every data point is accepted, the classifier must be very bad at generalizing from the training set. We know, from experience in the field of machine learning, that making a bad classifier is much easier than making a good one, as there are only a few hyperparameter values and feature subsets that provide good results. Inversely there is therefore a large set of hyperparameters and feature subsets outside the range of the good ones, that provide bad results. Our results support this fact. Figures 5.7 and 5.8 doesn't show this, but if one investigates our saved result files[1] containing all the chromosomes that make up each frontier, it can be seen that there exist several different chromosomes that reached the same maximum fairness score and low accuracy score. These data points are hidden behind each other in our figures. Conversely, there exists just a few, often just one, chromosome that reached the maximum accuracy score and a low fairness score. Later, as we select our chromosomes, i.e. classifiers that will be used in experiments 2 and 3, we will take a look at some of the hyperparameter values and feature subsets they contain.

In section 3.1 we covered how Menon and Williamson [2018] show that in fairness-aware learning problems the tradeoff between fairness and accuracy depends on the 'similarities' between the label and sensitive feature. They claim that the more disalignment, i.e. independence, there is between the label and the sensitive attribute the less of an effect introducing a fairness constraint will have on the accuracy. In section 5.1 we covered the label distribution with regard to the sensitive attribute, race, in the COMPAS data set. Figures 5.3 and 5.4 displays the label distribution for the privileged group, Caucasians, and the unprivileged group, Not Caucasians, respectively. The figures show that a data point is more

[1] `https://github.com/pernillej/Cost-of-Fairness`

likely to belong to the positive label, no recidivism, if it belongs to the privileged group then if it belongs to the unprivileged group. This shows that the sensitive attribute, race, is not perfectly independent from the label, indicating that if Menon and Williamson's claim holds, this fact aids in explaining the existence of the tradeoff that can be seen in our results. Menon and Williamson also provide a way to theoretically quantify the tradeoff. However, their method requires a predefined degree of fairness in order to calculate. Our thesis is not centered around specifying such a degree of fairness. Nevertheless, making these calculations after choosing a specific solution from our results could be an interesting direction for future work.

In conclusion, the results from experiment 1 support the state-of-the-art research in the field of fairness vs. accuracy tradeoffs. Additionally, our results help verify the Haas [2019] framework by which our architecture was inspired. Finally, the results provide interesting paths for further work investigating further claims made by several of the papers. In our next two experiments we will be investigating another claim made by state-of-the-art research; the importance of classification thresholds in regard to the tradeoff. In addition, we shall see whether we can observe similar tradeoffs in our next experiments given predefined classifiers and optimizing classification thresholds. In order to perform these experiments these predefined classifiers have to be chosen from our results in this first experiment. For each scenario, the most fair, and the most accurate classifiers were chosen for each algorithm. For example, for the baseline SVM algorithm from scenario 1, the most fair classifier is the one with a fairness score of 1, and the most accurate classifier is the one with an accuracy score of close to 0.7. The respective hyperparameters and feature subsets representing all these classifiers can be seen in tables A.1 and A.2 in appendix A. Earlier in our analysis, we discussed how there were many 'fair' classifiers discovered, but just a few that were the most accurate. Because there were many such 'fair' classifiers, one was chosen at random. The values of the hyperparameters, C and γ, as well as the selected feature subset is therefore not that interesting to discuss, as they are simply put, bad. However, the hyperparameter values and feature subsets are more interesting for the most accurate classifiers.

Table 5.6 shows the values of C and γ and the selected features for the most accurate classifiers, representing each algorithm for each scenario. Although optimizing these values and features in regard to a single accuracy objective is the best and simplest way to find optimal values with regard to accuracy, our values and features gives a good indication of what these optimal values and features might be. As can be seen in table 5.6 no set of hyperparameters and features are equal to one another. When comparing algorithms with each other this fact is not surprising, as all the mitigation methods pre-process the training data leading to

Algorithm	C	γ	Selected features
Scenario 1: Statistical Parity Difference vs. Accuracy			
SVM	19.5	0.02001	`age`, `priors count`, `charge degree=misdemeanor`
$SVM_{Reweighing}$	403.75	0.01187	`race`, `juvenile other count`, `priors count`, `age cat.25-45`, `age cat.>45`, `age cat.<25`, `charge degree=misdemeanor`, `charge degree=felony`
SVM_{DIR}	0.99658	0.65440	`age`, `juvenile felony count`, `juvenile misdemeanor count`, `juvenile other count`, `priors count`, `charge degree=misdemeanor`, `charge degree=felony`
$SVM_{OptPreProc}$	0.01566	0.16075	All features
Scenario 2: Theil Index vs. Accuracy			
SVM	$9.98378e^{-7}$	0.00077	`age`, `race`, `juvenile felony count`, `juvenile other count`, `priors count`, `age cat.>45`, `charge degree=misdemeanor`
$SVM_{Reweighing}$	$2.42233e^{-4}$	0.00310	`age`, `juvenile felony count`, `juvenile other count`, `priors count`, `charge degree=felony`
SVM_{DIR}	$2.44141e^{-4}$	0.02075	`age`, `race`, `juvenile other count`, `priors count`, `charge degree=misdemeanor`
$SVM_{OptPreProc}$	$6.66081e^{-6}$	0.15649	`race`, `age cat.>45`, `age cat.<25`, `priors count=0`, `priors count 1-3`, `charge degree=felony`

Table 5.6: *Experiment 1 - Selected SVM hyperparameters and features for the most accurate classifiers from both scenarios.*

each SVM classifier training on different datasets. Therefore, they require slightly different hyperparameters and features to provide the best accuracy. However,

when comparing the hyperparameters and features for each individual algorithm across scenarios, we can see that the C values in Scenario 2 are substantially lower than the ones in Scenario 1. This is surprising, as one would expect the same algorithm would have the same optimal hyperparameter values and features across scenarios, as the only difference is the fairness metric. On reason might just be that the optimal range of hyperparameter values and features is large and that we coincidentally happened to select classifiers exhibiting this result. More investigation would be needed to make any conclusions. It might be interesting to compare our results with results generated from running the method made by Huang and Wang [2006] optimizing for a single accuracy objective using the same algorithms. Additionally, several more runs of our experiment would help investigate whether this is simply a coincidence.

Another interesting aspect of our selected classifiers is the feature subset selected for each classifier, more specifically in regard to the protected attribute; `race`. An open discussion in fair AI research is whether to include the protected attribute in the model. Initially there was some claims that excluding the protected attribute ensures that there is no possibility of bias. In fact, Saxena et al. [2019] found that ordinary people seem to favor such an approach. However, research has proven that this does not eliminate bias, as several other features in the data sets may be closely connected to the protected attribute, such as names, addresses, etc., and thus circumventing the effect of removing the protected attribute by becoming its proxy. Therefore, many mitigation efforts work to not only lessen the effect of the protected attribute on the predicted label, but also the effect of potential proxy features. However, there is some that claim that including the protected attribute in the model would help the model compensate for the existing bias, and that there may be some cases where the outcome should be correlated Dwork et al. [2012]. For example, when using group specific classification thresholds, as suggested by some literature, the model requires the protected attribute to identify which group specific threshold to apply. This discussion is further complicated by the existence of laws that in effect require the protected attribute to be removed. Many of our selected 'most fair' classifiers, as seen in tables A.1 and A.2 in appendix A, include the `race` attribute, in both scenarios. This suggests that including the protected attribute aids fairness, at least for some approaches. However, because there exist many 'most fair' classifiers in our results, it is possible that some of these don't include the `race` attribute but is able to reach the same fairness score. Further investigation is required to determine the effect of the `race` attribute in each approach.

In our next two experiments we will be using the classifiers from table 5.6 in addition to the selected 'fair' classifiers shown in tables A.1 and A.2 in appendix A. These classifiers will be used to investigate the importance of classification

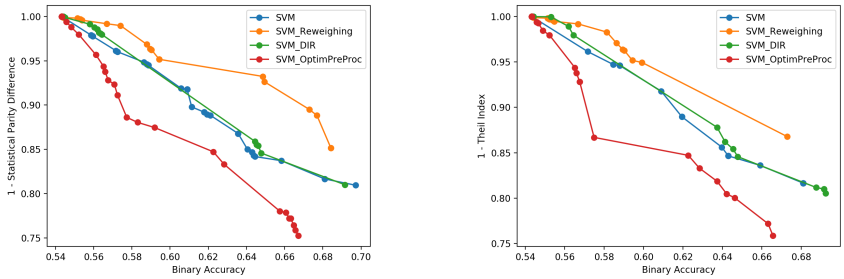
thresholds in regard to the tradeoff. In addition, we shall see whether we can observe similar tradeoffs in our next experiments given predefined classifiers and optimizing classification thresholds.

5.3.2 Experiment 2 - Optimizing a Classification Threshold

In experiment 2 we optimized a single classification threshold using an update version of our method from experiment 1. The goal of the experiment is to investigate whether this approach has a different effect on the fairness vs. accuracy tradeoff as opposed to the effects seen in experiment 1. A description of the differences in architecture used to perform the experiment can be found in section 4.4. The classification threshold will be used to classify the data points of our test set based on the predicted probabilities generated by our algorithms. These algorithms consist of a baseline SVM classifier, as well as three algorithms where a pre-processing mitigation method is used before the SVM classifier. For further descriptions of the four algorithms as well as other experiment parameters, see section 5.2.1. The experiment is run for two different scenarios depending on the fairness metrics we wish to optimize for: Scenario 1; Statistical Parity [eq. (2.1)] vs. Accuracy, [eq. (5.1)], Scenario 2; Theil Index [eq. (2.6)] vs. Accuracy [eq. (5.1)]. In experiment 1 we optimized hyperparameters and feature selection for the SVM classifiers in all four algorithms. From these results the most fair classifier (i.e. the classifier with the highest fairness score) and the most accurate classifier (i.e. the classifier with the highest accuracy score) were chosen for each algorithm in both scenarios. A secondary goal for this experiment is to examine whether it is possible to 'course correct' an already trained classifier using the classification threshold, i.e. whether the most fair classifier is still able to reach high accuracy scores and the most accurate classifier is still able to reach high fairness scores. The selected classifiers represented by their respective hyperparameters and feature subsets can be seen in tables A.1 and A.2 in appendix A.

First, we ran experiment 2 using the most accurate classifiers for both scenarios. Figure 5.9 shows the resulting frontiers. In RQ3 from section 1.1, we asked whether optimizing classification will have a different effect on the tradeoff than we had in experiment 1. At an immediate glance the frontiers from experiment 1 and the ones in fig. 5.9 appear to have the same overall effect. Both experiments produce frontiers exhibiting the existence of a tradeoff that effectively reduces the accuracy from around 68% to 55% in order to increase the fairness to its maximum. However, at a closer look some interesting differences can be seen.

One interesting aspect is the fact that the frontiers in figures 5.9a and 5.9b are much more similar to each other than figures 5.7 and 5.8 from experiment 1. The



(a) Scenario 1

(b) Scenario 2

Figure 5.9: *Experiment 2 - Results from all four algorithms for both scenarios, using the predefined 'most accurate' classifiers selected from experiment 1.*

effect of using different fairness metrics is almost none-distinguishable as opposed to in experiment 1. This is likely because varying a single classification threshold provides a much narrower range of values in which each solution may deviate, as opposed to varying both hyperparameters and feature subsets. Another interesting aspect of these results is that if one compares 5.8 and fig. 5.9b showing the results from scenario 2 in experiment 1 and this experiment respectively, the maximum fairness score, the Theil Index score, has increased from 0.94 to 1. Even more interestingly is that this increase is not at the cost of accuracy. In fig. 5.8 from experiment 1, the accuracy is around 55% when the fairness score is at its highest, 0.94. In fig. 5.9b the accuracy score is again around 55% when the fairness score is at its maximum, but this time the maximum is 1 rather than just 0.94. In fig. 5.9b the accuracy score is around 58% or closer to 60% depending on the algorithm, when the fairness score is at 0.94, the maximum from experiment 1. For Scenario 1, the differences are not that stark, as the frontiers follow roughly the same values as in experiment 1. However, the frontiers are clearly not as sparse (i.e. they contain more points/solutions) as they were in experiment 1, meaning our method from experiment 2 is able to provide an even larger spectrum of choices when deciding on a final solution. Another interesting result is that the SVM + Reweighing algorithm strictly dominates the other algorithms for both scenarios, until the highest accuracy scores are reached. This is a close reflection on the SVM + Reweighing algorithm in Scenario 1 in experiment 1. However, the same effect was not seen in Scenario 2 in experiment 1 where SVM + Reweighing performed roughly equal to the baseline SVM and SVM + Disparate Impact Remover. Meanwhile, in this experiment the SVM + Reweighing frontier strictly dominates in the upper fairness scores in this scenario as well.

This suggests that the Reweighing method is good at improving fairness, but comes at a larger cost to accuracy than the baseline SVM and SVM + Disparate Impact Remover.

There are also some similarities across experiments. Again, the SVM + Optimized Pre-Processing algorithm is nearly strictly dominated by all other algorithms and therefore clearly a worse option. Additionally, as discussed, the maximum accuracy reached is still just at most close to 70%. This further enforces the statement from Dressel and Farid [2018], who found the same low accuracy, that the COMPAS data set isn't non-linearly separable. Furthermore, several other discoveries enforcing by the state-of-the-art literature from experiment 1 still hold in this experiment. The cost to accuracy still exists, as most literature supports. The 'similarities' between the label and sensitive attribute effecting the tradeoff, as proposed by Menon and Williamson [2018] still holds as the same data set is used. An accuracy of around 55% (just as in experiment 1) when the fairness is at its maximum leads us to the same conclusion reached in experiment 1, that as Zliobaite [2015] suggest, accepting (or rejecting) all data points gives a 'perfect' fairness score, but at a large cost to accuracy.

Algorithm	Most Accurate Threshold	Most Fair Threshold
Scenario 1		
SVM	0.3418	0.0361
$\text{SVM}_{Reweighing}$	0.5996	0.2446
SVM_{DIR}	0.4170	0.1333
$\text{SVM}_{OptPreProc}$	0.6133	0.1563
Scenario 2		
SVM	0.4561	0.2495
$\text{SVM}_{Reweighing}$	0.5293	0.0430
SVM_{DIR}	0.4307	0.1528
$\text{SVM}_{OptPreProc}$	0.6152	0.0007

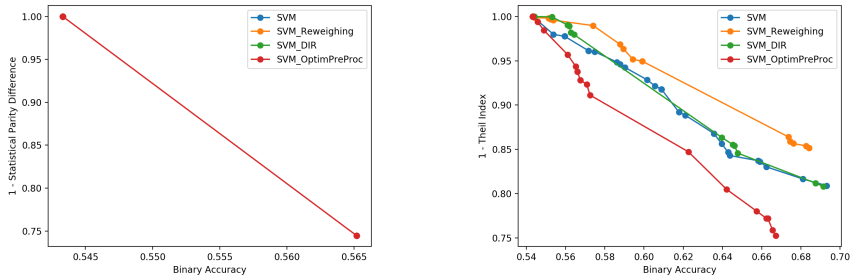
Table 5.7: *Experiment 2 - The most accurate and most fair threshold for both scenarios, generated from the most accurate classifiers.*

Table 5.7 shows the thresholds that resulted in the most fair results (i.e. highest fairness score) and the most accurate results (i.e. highest accuracy score) when we used the most accurate predefined classifiers. In experiment 1, all data points were classified using a threshold of 0.5. In table 5.7 we can see that none of the most accurate or most fair thresholds maintained the same threshold. The most accurate thresholds are in a range of 0.5 ± 0.15. While this is a significant variation from 0.5, the changes are even more significant for the most fair thresholds. The values of the most accurate thresholds still suggest that some data

points are likely to be classified with the negative label. Meanwhile, the most fair thresholds are very low, at most roughly 0.25, but some as low as 0.0007 for SVM + Optimized Pre-Processing in Scenario 2. The lower the threshold, the more data points are accepted, i.e. given the positive label. This supports our hypothesis that nearly all data points are accepted when reaching the highest fairness scores. Because of this fact, our results contain many chromosomes that reached the highest fairness score. The most fair thresholds shown in table 5.7 are simply a threshold selected from this set of chromosomes. However, a quick investigation of the other chromosomes shows similarly low thresholds.

In conclusion, the method from experiment 2 optimizing a single classification threshold on accuracy optimized predefined classifiers does show some different effects to the tradeoff as opposed to optimizing classifier hyperparameters and feature selection. A downside of this method is that a predefined and trained classifier is required. However, this can also be viewed as an upside as it allows AI systems already in use to be tested and adapted using this method. A lot of discussion in the field of fair AI has centered around the statement that enforcing fairness from the start is better than fixing for fairness after the fact. The method from experiment 1, and as suggested by Haas [2019], enforces fairness from the start as the classifiers is 'trained' with fairness in mind. However, our method from this experiment uses an already trained classifier and can be compared to fixing the system after the fact. Although, it is not nearly that black and white. Choosing fairness metrics is still an important discussion that should be taken from the start. Additionally, all our algorithms, except for the baseline SVM algorithm, had mitigation methods introduced from the start. However, our results from this experiment using accuracy optimized classifiers suggest that our method and architecture for experiment 2 is not worse at introducing fairness, and in fact better in some cases. For Scenario 1, with the statistical parity difference fairness metric, no large differences were seen, only that the frontiers were less sparse providing more options. For Scenario 2 with the Theil Index fairness metric, the method provided a different tradeoff than in experiment 1, enabling the classifiers to reach fairness scores of 1 at the same cost to accuracy that experiment 1 required to reach a fairness score of 0.94. This experiment clearly provided better options for final solutions than experiment 1. The validity of this result would still need further experimentation and verification in order to be sure this method and architecture is in fact better.

As discussed, the most accurate classifiers where able to reach similar or potentially better tradeoffs by optimizing a classification threshold than the ones seen in experiment 1. However, can the same be said for the most fair classifiers? Figure fig. 5.10 shows the result of optimizing classification threshold for the most fair classifiers for both scenarios.



(a) Scenario 1

(b) Scenario 2

Figure 5.10: *Experiment 2 - Results from all four algorithms for both scenarios, using the predefined 'most fair' classifiers selected from experiment 1.*

A big caveat for these results is that, as discussed in section 5.2.2, there were many most fair classifiers in our results, and one was chosen at random for each algorithm. Based on claims by Zliobaite [2015] we hypothesized, in our discussion of the results from experiment 1, that these most fair classifiers nearly accepted all data points. Therefore, we know that nearly all predicted probabilities were over 0.5, which was the classification threshold used in experiment 1. To determine how much these values varies above the 0.5 threshold would require further study of all those classifiers. Likely some varied more than others, but because we chose the most fair classifier randomly, we cannot be sure if other classifiers would have provided other results then the ones seen in fig. 5.10. Further investigation of this would be an interesting path for further work. Regardless, we will take a quick look at the results. Figure fig. 5.10b is nearly identical to the results in fig. 5.9 and exhibit the same differences across experiments. Figure fig. 5.10a however, is very different. It is not very clear from the graph, but all algorithms produced only one solution/chromosome in the frontier, or rather several chromosomes that reached the same fairness scores, except the SVM + Optimized Pre-Processing algorithm that found two different sets of fairness scores. They all found a chromosome that reached a fairness score of 1 and an accuracy of roughly 54%. Because they all found the exact same set of scores for the one point, the graph hides these points behind the algorithm added last to the graph, which also happens to have one additional point; SVM + Optimized Pre-Processing. This second point reached a very low fairness score at nearly 0.75 (as low as the lowest fairness scores in our other results), but reached an accuracy of just over 56.5%, which is much lower than the maximum accuracy SVM + Optimized Pre-Processing has reached in all our other results. These incredibly sparse and bad tradeoffs might suggest

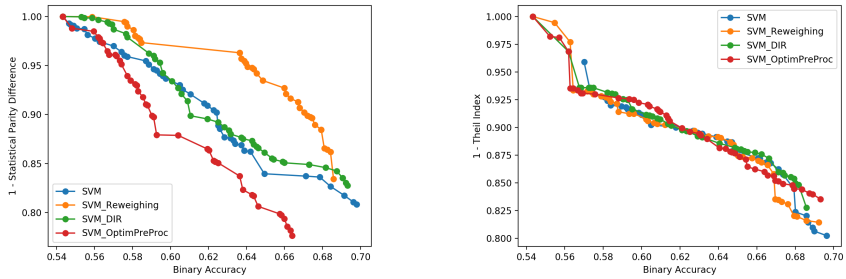
that optimizing for a single threshold using a group fairness metric doesn't work at all when the predefined classifiers are already optimized for fairness. However, as discussed, more investigation would be needed to determine whether this is just a case of a badly selected most fair classifier, or if this holds across more cases. Either way, it can definitively be said that the opposite doesn't hold over all cases.

In conclusion, while optimizing a single classification threshold for the most accurate classifiers similar or better tradeoffs were seen (see fig. 5.9. However, while optimizing a single threshold for the most fair classifiers for scenario 2 exhibited these same tradeoffs, for scenario 1 the method appears to not work at all. Because the most fair classifiers were selected from a large set of options from experiment 1, it is possible that if others were selected the results might be different. In our next experiment we will examine whether optimizing several, group specific thresholds rather than a single threshold will give different results.

5.3.3 Experiment 3 - Optimizing Group Specific Thresholds

In experiment 3 we will optimize group specific classification thresholds. In our case, using the COMPAS data set (section 5.1), there will be two thresholds. One threshold for the privileged group, Caucasians, and one for the unprivileged group, Not Caucasians. This experiment uses the same architecture used for experiment 2, except the chromosome represents the two group specific thresholds rather than one single threshold. The goal of the experiment is to investigate whether optimizing group specific thresholds has a different effect on the fairness vs. accuracy tradeoff as opposed to the effects seen in experiment 2 and experiment 1. A description of the architecture used to perform both experiments can be found in section 4.4. The classification thresholds will be used to classify the data points in each group respectively, where one threshold will be used to classify the privileged group, and the other threshold will be used to classify the unprivileged group. The thresholds will classify the data points by determining whether the predicted probabilities generated by our algorithms are above or below the threshold. There are four algorithms used to generate these probabilities. One is the baseline SVM classifier, while the three others are SVM classifiers with three different pre-processing mitigation methods performed before training the SVM. These are the same algorithms used in both experiment 1 and 2. For further descriptions of the four algorithms as well as other experiment parameters, see section 5.2.1. This experiment consists of two different scenarios depending on the fairness metric that is optimized: Scenario 1; Statistical Parity [eq. (2.1)] vs. Accuracy, [eq. (5.1)], Scenario 2; Theil Index [eq. (2.6)] vs. Accuracy [eq. (5.1)]. In experiment 2 we optimized a single classification threshold for all data points

without considering the race attribute. In order to perform that experiment we needed predefined and trained classifiers to test the thresholds against. These classifiers were selected from experiment 1 where we optimized hyperparameters and feature selection for the SVMs used in all four algorithms. From the results of experiment 1 we selected the most fair classifier (i.e. the classifier with the highest fairness score) and the most accurate classifier (i.e. the classifier with the highest accuracy score) for each algorithm in both scenarios. In this experiment we again need to use predefined classifiers, and we will be using the same classifiers that were used in experiment 2. The selected classifiers represented by their respective hyperparameters and feature subsets can be seen in tables A.1 and A.2 in appendix A.



(a) Scenario 1

(b) Scenario 2

Figure 5.11: *Experiment 3 - Results from all four algorithms for both scenarios, using the predefined 'most accurate' classifiers selected from experiment 1.*

First, we performed experiment 3 using the most accurate classifiers for both scenarios. Figure 5.11 shows the resulting frontiers. In experiment 2 we saw that optimizing a single threshold using the most accurate classifiers produced similar results for Scenario 1, and better results for Scenario 2, compared to the results from experiment 1. State-of-the-art literature, as discussed in section 3.1, suggest that using group specific classification thresholds is better for fairness, but comes at a large cost to accuracy. This literature is largely based on group fairness metrics. Figure 5.11a shows that for scenario 1 the resulting frontiers are nearly identical (i.e. reach the same values) to the results from both experiment 1 and 2. However, the frontiers are once again less sparse, even less sparse then in experiment 2. This can likely be attributed to the fact that using two thresholds rather than one creates more complexity leading to the chromosomes being able to represent even more fairness scores.

For scenario 2, the resulting frontiers are again different from the previous experiment. In this experiment, using the most accurate classifiers for scenario 2, all the differences between each algorithm is gone, and they all follow along the same values. Additionally, a steep tradeoff in fairness can be seen from a score of roughly 0.96 to 0.93. At this tradeoff the accuracy increases minimally. In fact, from this point an onward as the accuracy increases, the algorithms follow more closely the tradeoff seen in experiment 1, then the improved tradeoff from experiment 2. However, they are still able to reach fairness scores of 1 as seen in experiment 2, except the baseline SVM algorithm, reaching just 0.96. Another interesting aspect from scenario 2 is that the SVM + Optimized Pre-processing algorithm performs better than in the previous experiments, reaching an accuracy of nearly 70% with a fairness score of roughly 0.835, a lot better than a fairness score of closer to 0.75 while the accuracy is just over 66%.

These differences in scenario 2 from the previous experiments are interesting, suggesting that perhaps this method of using group specific classification thresholds creates a middle ground between the method from experiment 1 and the method from experiment 2, when using the Theil Index fairness metric. An exception is that the results from the SVM + Optimized Pre-Processing algorithm performed a lot better in the lower fairness scores than seen in both previous experiments, with both higher accuracy scores and higher minimum fairness scores. This result suggests that the differences between our methods are not as clear cut, where for some algorithms this method performs better, but for others it performs worse. However, again all these differences in tradeoffs are only seen in scenario 2. Meanwhile, for scenario 1, with the statistical parity difference fairness metric, the tradeoffs remain the same but the frontiers grow denser for each experiment.

Table 5.8 shows the thresholds that resulted in the most fair results (i.e. highest fairness score) and the most accurate results (i.e. highest accuracy score) when we used the most accurate predefined classifiers. In experiment 1, all data points were classified using a threshold of 0.5, and in experiment 2 the most fair and most accurate thresholds can be seen in table 5.7. In experiment 2, we found that none of the most accurate or most fair solutions maintained a threshold of 0.5. The most accurate thresholds varied in the range 0.5 ± 0.15, while the most fair thresholds very much lower, from 0.25 to $\sim$0. This result supported the hypothesis that nearly all data points are given the positive label when reaching the highest fairness scores. An interesting finding in our results from this experiment, is that just one or a few chromosomes reached the highest fairness score, as opposed to in our previous experiments when many chromosomes reached the highest score. This is likely because of the added complexity the group specific thresholds provide. However, the hypothesis that nearly all data points are

Algorithm	Most Accurate Thresholds		Most Fair Thresholds	
	Privileged Threshold	Unprivileged Threshold	Privileged Threshold	Unprivileged Threshold
Scenario 1				
SVM	0.5586	0.5176	0.0015	0.0459
$SVM_{Reweighing}$	0.5586	0.5547	0.3818	0.2261
SVM_{DIR}	0.2783	0.4590	0.0032	0.0117
$SVM_{OptPreProc}$	0.4434	0.625	0.0005	0.0039
Scenario 2				
SVM	0.4873	0.4971	0.4951	0.125
$SVM_{Reweighing}$	0.4980	0.4834	0.0625	0.0020
SVM_{DIR}	0.4873	0.4766	0.2510	0.0625
$SVM_{OptPreProc}$	0.3291	0.4599	0.1719	0.0625

Table 5.8: *Experiment 3 - The most accurate and most fair thresholds for both scenarios, generated from the most accurate classifiers.*

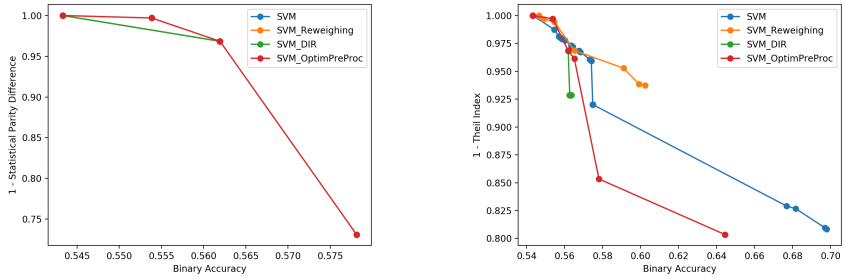
accepted still holds, as the most fair thresholds are again low, in most cases.

Based on the research supporting group specific thresholds (covered in section 3.1), the hypothesis for this experiment is that the resulting thresholds should indicate that the most fair thresholds are lower for the unprivileged group then for the privileged group. This means that more data points from the unprivileged group should be accepted then they would be if a single threshold was used for both groups, thereby mitigating the bias against the unprivileged group. Studying the most fair thresholds from table 5.8, we can see that this hypothesis mostly holds, except in the cases were both thresholds are very small (e.g. the baseline SVM algorithm in scenario 1). In such cases, the deviation between the group specific threshold are insignificant, because we can assume that with such low thresholds all data points are accepted from each group. Another interesting finding is that some of the most fair thresholds are much larger than the ones found in experiment 2, with the baseline SVM algorithm having a privileged threshold of nearly 0.5 in scenario 2. This suggests that most of the data points in the privileged group was already accepted by this algorithm back in experiment 1.

If we study the most accurate thresholds, we can see that the privileged and unprivileged thresholds deviate insignificantly from each other, except in three cases. Additionally, most of these thresholds are close to 0.5 as well. However, in the cases where the thresholds do deviate from each other, the privileged threshold is the lowest (e.g. SVM + Disparate Impact Remover in scenario 1). This suggest that these algorithms are biased against the unprivileged group in

order to reach a high accuracy score.

While scenario 1 retains the same tradeoffs, and the tradeoff for scenario 2 varies for each experiment, some facts still remain constant across experiments. The existence of a tradeoff still remains, as most state-of-the-art literature supports. The same maximum accuracy of nearly 70% is also, consistent, and within the values expected given previously documented accuracies for the COMPAS data set. See section 5.1 for more on the data set. The lowest accuracy also remains consistent at close to 54%-55%. The fairness scores exhibit slightly lower variance for this experiment, varying from a score of 1 to just under 0.8, as opposed to nearly 0.75. This is because the SVM + Optimized Pre-Processing algorithm performed better in this experiment then it did in both previous experiments.



(a) Scenario 1

(b) Scenario 2

Figure 5.12: *Experiment 3 - Results from all four algorithms for both scenarios, using the predefined 'most fair' classifiers selected from experiment 1.*

Finally, we will take a look at the results from the most fair classifiers. Figure 5.12 shows the resulting frontiers. The same caveat stated in experiment 2 still holds for this experiment, as these are the same randomly selected most fair classifiers. In fig. 5.12a we see nearly the same results that was seen for scenario 1 in experiment 2 as well. Some frontiers have a few more solutions, but the conclusion remains the same. Trying to make the fair classifiers more accurate by varying both a single threshold or group specific thresholds is nearly impossible for scenario 1. For scenario 2 we saw no changes in the tradeoff when using the fair classifiers as opposed to the accurate classifiers. In this experiment however, the frontiers look very different. While the baseline SVM algorithm as well as the SVM + Optimized Pre-Processing algorithm reach somewhat similar results as we have seen previously, the frontiers are very sparse. In addition, the SVM + Reweighing and SVM + Disparate Impact Remover algorithms are unable to

'escape' the higher fairness scores.

Clearly using group specific thresholds on scenario 2 with the Theil Index, gives very different results from using a single classification threshold. Results from Speicher et al. [2018] suggest that using a single classification threshold works well with individual fairness metrics like the Theil Index up, where both accuracy and fairness increase, until a point (i.e. a threshold value) is reached where the accuracy starts to decrease while the fairness remains increasing. Though Speicher et al. do not investigate the effect of multiple group specific thresholds. Their suggestion that increasing fairness is in harmony with increasing the accuracy clashes with other state-of-the-art literature, where most claim that fairness comes at a cost to accuracy. See section 3.1 for a discussion of the state-of-the-art literature. In this other literature they use group fairness metrics in their reasoning leading them to conclude that group specific thresholds improve fairness at a cost to accuracy, while a single threshold is optimal for accuracy. Combining these claims with Speicher et al.'s proposition we could speculate that a single threshold may be optimal for individual fairness metrics, while group specific thresholds is optimal for group fairness metrics. Our results from experiment 2 and 3 supports this statement in regard to individual fairness. However, the same cannot be said for group fairness, as the results remained the same across experiments.

In conclusion, optimizing group specific classification thresholds provided similar tradeoffs in scenario 1, and showed a different effect on the tradeoffs in scenario 2. For scenario 1 the most accurate classifiers generated slightly more dense frontiers, but they all exhibited the same tradeoff as seen in both previous experiments. The most fair classifiers were again unable to generate substantially higher accuracy scores then was originally found by the classifiers in experiment 1. For the most fair classifiers scenario 2 generated much worse results then seen in experiment 2, but not as bad as for scenario 1. Some of the algorithms were still able to get good accuracy scores at similar costs to fairness as seen previously. The results for the most accurate classifiers in scenario 2 were a combination of results from experiment 1 and 2, where again we were able to reach 'perfect' fairness scores of 1, but for the lower fairness scores the tradeoff remained closer to the ones seen in experiment 1 rather than the slightly better tradeoff from experiment 2. However, the SVM + Optimized Pre-Processing algorithm performed considerably better in this experiment then it had in both previous experiments, suggesting that the choice of method to be used might depend on which mitigation methods will be included. The differences between scenarios also suggest that for some fairness metrics the distinction between methods is inconsequential, but for others, like the Theil Index, the difference can have more of an effect.

Chapter 6

Conclusion and Future Work

This chapter will present our conclusion of the work presented in the previous chapters of this thesis. We will discuss how our work might impact the current state of the fairness vs. accuracy tradeoff research. Finally, we will present our vision for future work.

6.1 Conclusion

In this master's thesis our overarching goal was to:

Goal *Investigate the tradeoff between fairness and accuracy in automated decision-making systems.*

To narrow the scope of our investigation we presented three research questions. Our first research question was:

Research question 1 *What is the state-of-the-art in research concerning the tradeoff between fairness and accuracy in automated decision-making systems?*

The results from our review of the state-of-the-art literature was detailed and discussed in chapter 3. In summary, our findings were that the limited research done in this area mostly concluded that such a tradeoff between fairness and accuracy exists. Additionally, a few papers questioned this assumption, proposing that

in theory these concepts should rather be in harmony. However, they acknowledged that in practice this proposition would rarely hold. Another interesting factor discussed in the state-of-the-art research was the impact of classification thresholds. Research based on group fairness metrics concluded that using a single classification threshold was optimal for accuracy, while using group specific thresholds was optimal for fairness, but at the cost of accuracy. Meanwhile, a paper by Speicher et al. [2018] using an individual fairness metric found that using a single classification threshold lead to fairness and accuracy increasing in harmony, up until a point. These papers highlighted the importance of classification thresholds and indicated that they might have different effects depending on whether group fairness or individual fairness is considered.

In our review of the state-of-the-art research we also found a paper by Haas [2019] that proposed a framework using MOO and Pareto frontiers to investigate the cost of fairness. Before starting the work on our master's thesis, we had taken a course in these concepts, among others. The combination of these factors as well as our findings from our review of the state-of-the-art research lead us to formulate our two last research questions:

Research question 2 *Using multi-objective optimization and Pareto fronts to optimize feature selection and classifier hyperparameters, what type of tradeoffs can be observed?*

Research question 3 *Can we use this multi-objective optimization method to optimize classification thresholds, and what effect will this have on the type of tradeoff that can be observed?*

Research question 2 had already been answered previously by Haas [2019]. In this thesis we used the framework proposed by Haas and adapted the architecture slightly in order to create our own variation of the method. In addition, we applied the framework on a different data set, and different mitigation methods. This provided results that generalize the use of multi-objective optimization and Pareto fronts as a method for investigating fairness vs. accuracy tradeoffs. Our further adaptation of the method was performed so that we could further apply the method in order to answer research question 3. We have found no previous work using a similar adaptation of the method. In chapter 4 we described the methods and architecture we used to answer these two research questions. In chapter 5 we formulated our experiment plan for the experiments that will be used to test our architectures and answer the RQs. In the latter part of the same chapter we presented our results from the experiments and analyzed and discussed the results in the light of the state-of-the-art literature.

In experiment 1 we optimized feature selection and classifier hyperparameters using NSGA-II. The experiment would help us answer RQ 2. We used the COMPAS

dataset (see section 5.1) with 12 features and 6172 data points. This dataset was split into a training set and a test set, containing 80% and 20% of the data points respectively. The mitigation methods we used were the Reweighing method, the Disparate Impact Remover method, and the Optimized Pre-Processing method. Additionally, we had a baseline algorithm that didn't use a mitigation method. All algorithms used an SVM classifier with the RBF kernel. The experiment was performed for two different scenarios depending on the fairness metrics we wished to optimize for. In scenario 1 a group fairness metric in addition to the accuracy metric was optimized; Statistical Parity [eq. (2.1)] vs. Accuracy, [eq. (5.1)]. In scenario 2 an individual fairness metric was optimized in addition to the accuracy metric; Theil Index [eq. (2.6)] vs. Accuracy [eq. (5.1)]. Based on the results from experiment 1 we found that our variation of Haas' method works, and though a different dataset and different mitigation methods were used, the tradeoffs were similar. Our results enforce the existence of a tradeoff, as most state-of-the-art research suggests. Additionally, we found that the particular values we were able to reach with regards to the accuracy and fairness scores, could be supported by previous findings and claims proposed by state-of-the-art research, as well as previous research using the COMPAS dataset. We found that our results support research by Zliobaite [2015] claiming that the most fair classifiers likely accept (or reject) all data points at a large cost to accuracy. Menon and Williamson [2018] show that the tradeoff between fairness and accuracy is dependent on the 'similarities' between the label and protected attribute. Our analysis for the COMPAS data set in section 5.1 showed that there is some correlation between the protected attribute, race, and the label. This correlation aids in explaining the existence of a tradeoff. The most accurate results (i.e. the solutions able to reach the highest accuracy scores) reached accuracies of 66-70%. Similar accuracies were found by previous research of the COMPAS data set [Dressel and Farid, 2018].

In experiments 2 and 3 we optimized a single classification threshold and group specific thresholds respectively. These methods required existing algorithms to generate prediction probabilities. These algorithms were selected from experiment 1. In particular we selected the most fair classifier (i.e. the classifier with the highest fairness score) and the most accurate classifier (i.e. the classifier with the highest accuracy score) for each algorithm in both scenarios. In these experiments the same data set, algorithms, and scenarios from experiment 1 was used. For the most accurate classifiers we found that in scenario 1 both experiments generated nearly equal results. The only difference was that the resulting frontiers became denser with solutions from one experiment to the next. For scenario 2 however, we found some different effects across experiments. In experiment 2 the best overall tradeoff was reached, as the algorithms were able to reach a perfect fairness score of 1 as opposed to just 0.94 at no cost to the accuracy when

comparing the results to experiment 1. In experiment 3 the tradeoff turned out to be a combination of the results from experiment 1 and 2. The frontiers from experiment 3 resembled the frontiers in experiment 2 in the upper fairness scores, but more closely resembled experiment 1 in the lower fairness scores and higher accuracy scores. However, the SVM + Optimized Pre-Processing algorithm performed better than it had in all previous experiments. This led us to conclude that not only does the outcomes of our methods vary depending on which fairness metric is used, they also vary depending on the mitigation method. We can therefore not definitively conclude that using a single classification threshold is better for individual fairness metrics, or at least the Theil Index, but in most cases, this appears to be true.

We also tested the most fair classifiers in experiment 2 and 3, but this came with a caveat. Our results from experiment 1 contained many most fair classifiers, and one was chosen randomly for each algorithm. It is therefore possible that if other classifiers were chosen, our results might have been different. Using our selected classifiers, we found that for scenario 1 the algorithms are never able to 'escape' the low accuracy scores of roughly 55% in both our experiments. This suggests that for group fairness metrics, while classifiers trained for accuracy can be adapted to be more fair using classification thresholds, the classifiers trained for fairness exclusively cannot become more accurate. For scenario 2 the results were slightly different. In experiment 2, using a single classification threshold, the results were identical to the ones reached by the most accurate classifiers. However, in experiment 3 the algorithms had more difficulty 'escaping' the low accuracy scores, but some algorithms were able to reach the same accuracy scores as found by the most accurate classifiers. In either case, the algorithms performed much worse for the most fair classifiers in experiment 3 then they did in experiment 2.

In conclusion, we verified the efficiency of the framework suggested by Haas [2019] in experiment 1. We found that, as most state-of-the-art research suggests, the introduction of fairness comes at the cost of accuracy. In experiment 2 and 3 we found that this result still holds. Additionally, because these methods found similar results to experiment 1 using predefined classifiers rather than using the method to find said classifiers, another conclusion can be reached. We can conclude that the method used in experiments 2 and 3 can be used as a substitute for Haas' method in cases where AI decision making systems have already be trained and in use. In fact, for the Theil Index fairness metric we found that the method from experiments 2 and 3 provides a better tradeoff then the one found in experiment 1.

Based on our conclusions, the research contributions from our work can be summarized as follows:

1. Verification of the Haas [2019] framework for studying the cost of fairness and potentially selection of a final approach and algorithm than can be used to make automated decision-making systems with the desired amount of fairness.
2. A novel method and architecture that allows for the same study and selection, but that can be used on existing AI systems. Or potentially, practitioners can train a model by optimizing accuracy, and afterwards determine what degree of fairness to introduce depending on the tradeoffs found using this novel method and architecture.

6.2 Future Work

There are several directions for future work that can be taken to expand on our current work. Some interesting paths have already been discussed previously in the thesis. Here we will summarize potential avenues for future work.

In their paper Haas [2019] suggest several options for future work that is also relevant for our work in this thesis. This includes testing the method and architecture using other datasets, other mitigation methods, other fairness and accuracy metrics, and other types of classifiers then SVMs. This would help to further verify both the framework suggested by Haas and the novel methods and architecture presented in this thesis.

Based on our work in this thesis we are able to generate some hypothesis as to the difference between using a group fairness metric as opposed to an individual fairness metric. However, we only use one of each type of metric in our experiments. Further investigating whether our hypotheses hold when using more metrics representing both group and individual fairness, would be interesting as it would allow more firm conclusions to be reached.

In our discussion of the state-of-the-art research, as well as in analyzing our results we discussed a paper by Wick et al. [2019]. They claim that in theory fairness should increase along with accuracy and not come at a cost. However, in practice they acknowledge that this would rarely hold. This they claim, is because of the bias that exists in most data sets. For example, in section 5.1 we covered how the COMPAS data set has been repeatedly proven to be biased against non-whites. Wick et al. point out that when a mitigation method works to ensure fairness in the model using the training data, often no changes are being made to the evaluation data. It is therefore only natural that accuracy is decreased when measured on the biased evaluation data, they claim. In their paper they suggest a data simulation method for finding a unbiased data that can be used for evaluation. An interesting path for future work would be to attempt

to use this method or a similar method to investigate whether we would still find a tradeoff between fairness and accuracy in our results.

Finally, we believe that our implementation of our architectures could be improved further. First, given more time, we would have liked to implement our NSGA-II code using recognized Python modules like `DEAP`. This would make the code more scalable long term. In addition, we believe that our code could be expanded and adapted to be more modular and include more dataset options, more fairness metrics, and more mitigation methods. This could potentially lead to a tool that AI practitioners could use without having to implement their own versions of our architecture in order to use the methods suggested in this thesis for analyzing the cost of fairness in their systems.

The source code for our implementation can be found at `https://github.com/pernillej/Cost-of-Fairness`. The code includes a `README.md` file that contains a user guide describing the contents of the code base, as well as how to run and configure the 3 experiments from this thesis. The requirements for running the code are listed in this file as well. See also section 4.5 for a list of requirements as well as some other details about the implementation. A copy of the user guide is found in appendix B.

Bibliography

Allen, M. (2019). 117. genetic algorithms 2 – a multiple objective genetic algorithm (nsga-ii). https://pythonhealthcare.org/2019/01/17/117-genetic-algorithms-2-a-multiple-objective-genetic-algorithm-nsga-ii/. Accessed: 2020-04-11.

Angwin, J., Larson, J., Mattu, S., and Kirchner, L. (2016). Machine Bias. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing. Accessed: 2020-02-03.

Bellamy, R. K. E., Dey, K., Hind, M., Hoffman, S. C., Houde, S., Kannan, K., Lohia, P., Martino, J., Mehta, S., Mojsilovic, A., Nagar, S., Natesan Ramamurthy, K., Richards, J., Saha, D., Sattigeri, P., Singh, M., Varshney, K. R., and Zhang, Y. (2019). AI Fairness 360: An Extensible Toolkit for Detecting and Mitigating Algorithmic Bias. *IBM Journal of Research and Development*, 63(4/5):4:1–4:15.

Bertsimas, D., Farias, V. F., Trichakis, N., Bertsimas, D., Farias, V. F., and Trichakis, N. (2019). The Price of Fairness. *Operations Research*, 59(1):17–31.

Calmon, F., Wei, D., Vinzamuri, B., Natesan Ramamurthy, K., and Varshney, K. R. (2017). Optimized pre-processing for discrimination prevention. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 3992–4001. Curran Associates, Inc.

Corbett-Davies, S., Pierson, E., Feller, A., Goel, S., and Huq, A. (2017). Algorithmic decision making and the cost of fairness. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 797–806.

Danks, D. and London, A. J. (2017). Algorithmic bias in autonomous systems. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 4691–4697.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Dobbe, R., Dean, S., Gilbert, T., and Kohli, N. (2018). A Broader View on Bias in Automated Decision-Making: Reflecting on Epistemology and Dynamics. In *2018 Workshop on Fairness, Accountability, and Transparency in Machine Learning*, pages 1–5.

Dressel, J. and Farid, H. (2018). The accuracy, fairness, and limits of predicting recidivism. *Science Advances*, 4(1):1–5.

Dua, D. and Graff, C. (2017). UCI machine learning repository - statlog (german credit data) data set. `http://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)`. Accessed: 2020-03-11.

Dwork, C., Hardt, M., Pitassi, T., Reingold, O., and Zemel, R. (2012). Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 214–226. Association for Computing Machinery.

Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., and Venkatasubramanian, S. (2015). Certifying and removing disparate impact. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 259–268.

Floreano, D. and Mattiussi, C. (2008). *Bio-Inspired Artificial Intelligence*, chapter 1, pages 1–100. The MIT Press, Cambridge Massachusetts, 1 edition.

Friedman, B. and Nissenbaum, H. (1996). Bias in Computer Systems. *ACM Transactions on Information Systems*, 14(3):330–347.

Friedrich, T., Horoba, C., and Neumann, F. (2011). Illustration of fairness in evolutionary multi-objective optimization. *Theoretical Computer Science*, 412(17):1546–1556.

Haas, C. (2019). The Price of Fairness - A Framework to Explore Trade-Offs in Algorithmic Fairness. In *40th International Conference on Information Systems, ICIS 2019*, pages 1–17.

Hardt, M., Price, E., and Srebro, N. (2016). Equality of opportunity in supervised learning. In *30th Conference on Neural Information Processing Systems*, pages 3323–3331.

Hsu, C.-W., Chang, C.-C., and Lin, C.-J. (2016). A Practical Guide to Support Vector Classification. *Technical Report, Department of Computer Science, National Taiwan University*.

Huang, C. L. and Wang, C. J. (2006). A GA-based feature selection and parameters optimization for support vector machines. *Expert Systems with Applications*, 31(2):231–240.

Johnsen, P. (2019). Fairness and discrimination in machine learning. Project report in TDT4501, Department of Computer Science, NTNU – Norwegian University of Science and Technology.

Kamiran, F. and Calders, T. (2012). Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33:1–33.

Kleinberg, J., Mullainathan, S., and Raghavan, M. (2017). Inherent trade-offs in the fair determination of risk scores. In *Leibniz International Proceedings in Informatics, LIPIcs*, volume 67, pages 1–23.

Larson, J., Mattu, S., Kirchner, L., and Angwin, J. (2016). Data and analysis for 'machine bias'. `https://github.com/propublica/compas-analysis`. Accessed: 2020-04-20.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. (2019). A Survey on Bias and Fairness in Machine Learning. *eprint arXiv:1908.09635*.

Menon, A. K. and Williamson, R. C. (2018). The Cost of Fairness in Binary Classification. In *Proceedings of Machine Learning Research*, pages 1–12.

Mock, W. B. T. (2011). Pareto optimality. In Chatterjee, D. K., editor, *Encyclopedia of Global Justice*, pages 808–809. Springer Netherlands, Dordrecht.

Norwegian Ministry of Culture (2017). Act relating to equality and a prohibition against discrimination (Equality and Anti-Discrimination Act) - Lovdata. `https://lovdata.no/NLE/lov/2017-06-16-51`. LOV-2017-06-16-51.

Olteanu, A., Castillo, C., Diaz, F., and Kıcıman, E. (2019). Social Data: Biases, Methodological Pitfalls, and Ethical Boundaries. *Frontiers in Big Data*, 2:13.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rajkomar, A., Hardt, M., Howell, M. D., Corrado, G., and Chin, M. H. (2018). Ensuring fairness in machine learning to advance health equity. *Annals of Internal Medicine*, 169(12):866–872.

Russell, S., Russell, S., and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach, Third Edition*. Prentice Hall.

Saxena, N. A., Huang, K., DeFilippis, E., Radanovic, G., Parkes, D. C., and Liu, Y. (2019). How do fairness definitions fare? examining public attitudes towards algorithmic definitions of fairness. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '19, page 99–106. Association for Computing Machinery.

Speicher, T., Heidari, H., Grgic-Hlaca, N., Gummadi, K. P., Singla, A., Weller, A., and Zafar, M. B. (2018). A unified approach to quantifying algorithmic unfairness: Measuring individual & group unfairness via inequality indices. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2239–2248.

Suresh, H. and Guttag, J. V. (2019). A Framework for Understanding Unintended Consequences of Machine Learning. *eprint arXiv:1901.10002*, pages 1–6.

Torralba, A. and Efros, A. A. (2011). Unbiased look at dataset bias. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1521–1528. IEEE.

Whittaker, M., Crawford, K., Dobbe, R., Fried, G., Kaziunas, E., Mathur, V., Myers West, S., Richardson, R., Schultz, J., and Schwartz, O. (2018). AI Now Report 2018. *AI Now*, pages 1–62.

Wick, M., Panda, S., and Tristan, J.-B. (2019). Unlocking Fairness: a Trade-off Revisited. In *33rd Conference on Neural Information Processing Systems*, pages 1–10.

Zemel, R., Wu, Y., Swersky, K., Pitassi, T., and Dwork, C. (2013). Learning fair representations. *30th International Conference on Machine Learning, ICML 2013*, 28:1362–1370.

Zliobaite, I. (2015). On the relation between accuracy and fairness in binary classification. In *The 2nd Workshop on Fairness, Accountability, and Transparency in Machine Learning*, pages 1–5.

Appendices

A Selected Classifiers for Experiment 2 And 3

Algorithm	C	γ	Selected features
Highest Accuracy			
SVM	19.5	0.02001	age, priors count, charge degree = misdemeanor
$\text{SVM}_{Reweighing}$	403.75	0.01187	race, juvenile other count, priors count, age cat. 25-45, age cat. >45, age cat. <25, charge degree = misdemeanor, charge degree = felony
SVM_{DIR}	0.99658	0.65440	age, juvenile felony count, juvenile misdemeanor count, juvenile other count, priors count, charge degree = misdemeanor, charge degree = felony
$\text{SVM}_{OptPreProc}$	0.01566	0.16075	All features
Highest Fairness			
SVM	39936.0	1.67129	charge degree = misdemeanor
$\text{SVM}_{Reweighing}$	4412.0	0.03608	race, charge degree = felony
SVM_{DIR}	65376.0	1.00098	charge degree = misdemeanor, charge degree = felony
$\text{SVM}_{OptPreProc}$	0.01602	0.48730	race, charge degree = misdemeanor

Table A.1: *Selected SVM hyperparameters and features from Scenario 1: Statistical Parity Difference vs. Accuracy*

Algorithm	C	γ	Selected features
Highest Accuracy			
SVM	$9.98378e^{-7}$	0.00077	age, race, juvenile felony count, juvenile other count, priors count, age cat. >45, charge degree = misdemeanor
$SVM_{Reweighing}$	$2.42233e^{-4}$	0.00310	age, juvenile felony count, juvenile other count, priors count, charge degree = felony
SVM_{DIR}	$2.44141e^{-4}$	0.02075	age, race, juvenile other count, priors count, charge degree = misdemeanor
$SVM_{OptPreProc}$	$6.66081e^{-6}$	0.15649	race, age cat. >45, age cat. <25, priors count = 0, priors count 1-3, charge degree = felony
Highest Fairness			
SVM	$1.49011e^{-8}$	1.3332	sex, race, juvenile felony count, juvenile other count, age cat. 25-45
$SVM_{Reweighing}$	3.78711	0.85525	race, charge degree = felony
SVM_{DIR}	61504.0	4.18945	age, race, juvenile other count, priors count, charge degree = felony
$SVM_{OptPreProc}$	$6.63102e^{-6}$	0.84112	age cat. 25-45, charge degree = felony, charge degree = misdemeanor

Table A.2: *Selected SVM hyperparameters and features from Scenario 2: Theil Index vs. Accuracy*

B User Guide for the Source Code

The source code for the code used in this thesis can be found at:

`https://github.com/pernillej/Cost-of-Fairness`.

The project consists of code for running 3 different experiments. The code for each experiment is contained in 3 separate folders. In addition, some code is used across experiments and exist in their own folders.

Content Description

- `src/nsga2` - This folder contains the code used to perform NSGA-II operations used to produce Pareto Frontiers.
- `src/util` - This folder contains utility code used to read and write result files, plot results, and convert from binary to decimal values.
- `src/compas_analysis.py` - This file contains the code used to analyse the COMPAS data set that is used in all 3 experiments.
- `src/data.py` - This file contains the code used to gather the COMPAS data set from `aif360`.
- `src/metrics.py` - This file defines the possible fairness and accuracy metrics that can be used in each experiment.
- `src/experiment1`, `src/experiment2`, `src/experiment3` - These folders contain the code used to run each respective experiment. Each folder contains:
  - `/results` - This folder is used to store the results as .txt files containing json dictionaries describing the results and configurations from each run of the experiment.
  - `algorithms.py` - This file defines the 4 algorithms: `svm`, `svm_reweighing`, `svm_dir`, `svm_optimpreproc`. See thesis for the purpose of these algorithms.
  - `config.py` - This file defines the run configurations for the experiment.
  - `main.py` - This file is the file used to define and run the experiment.
  - `plot_results.py` - This file is used to plot the results from the experiment.
  - `baseline.py`, `disparate_impact_remover.py`, `optimpreproc.py`, `reweighing.py` - These files initiate each algorithm into the NSGA-II

optimization approach, by running NSGA-II using the proper parameters, and defining the evaluation function to be used by NSGA-II.

Running an experiment

Each experiment folder contains its own `main.py` file. Running this file will run the entire experiment as it was performed in the thesis.

Run configurations

Each experiment folder also contains its own `config.py` file that can be used to update NSGA-II parameters like number of generations, population size, mutation and crossover rate. The max iterations and seed used for the SVM classifiers can also be changed. Finally, it is also possible to update which fairness and accuracy metrics are used.

Possible accuracy metrics: `auc` and `binary_accuracy`.

Possible fairness metrics: `statistical_parity_difference`, `theil_index`, `equal_opportunity_difference`, `average_odds_difference`, and `disparate_impact`.

Pernille Johnsen

Investigating the Cost of Fairness in Automated Decision-Making Systems

NTNU
Kunnskap for en bedre verden