Testing Algorithm Fairness Metrics for Binary Classification Problems by Supervised Machine Learning Algorithms

R.M.V. Humphris

Master's Thesis to obtain the degree in *Business Analytics*
Vrije Universiteit Amsterdam
Faculty of Science

Author:	R.M.V. Humphris
Date:	June 30, 2020
Graduation Supervisor:	Dr. D. Dobler
Second reader:	Dr. M. Hoogendoorn
External supervisor:	P.J. van Kessel

Testing Algorithm Fairness Metrics for Binary Classification Problems by Supervised Machine Learning Algorithms

R.M.V. Humphris

Master project Business Analytics

Vrije Universiteit Amsterdam
Faculty of Science
De Boelelaan 1085
1081HV Amsterdam

PwC
Thomas R. Malthusstraat 5
1066 JR Amsterdam

Statement of Originality

This document is written by Student Raquel Humphris who declares to take full responsibility for the contents of this document.

I declare that the text and the work presented in this document are original and that no sources other than those mentioned in the text and its references have been used in creating it.

The Faculty of Science is responsible solely for the supervision of completion of the work, not for the contents.

Abstract

Currently, predictive Machine Learning (ML) Algorithms become more and more integrated in our daily lives. Much is unknown about how all the various Algorithm Fairness Metrics compare for different factors. Therefore, this study aimed to research the impact of the dataset, supervised ML classification Algorithm, its Surrounding models, Bootstrapping the train / test split and varying Seed choice for the train / test split on the Algorithm Fairness Metrics that were described by Verma and Rubin (2018). Three ML Algorithms were applied on two datasets to predict the binary classifications of individuals who are respectively being a recidivist or having an income of more than 50K. The ML Algorithms are Random Forest (RF), XGBoost and Linear Support Vector Classifier (LinearSVC). The stability of the Algorithm Fairness Metrics was tested by building Surrounding models around the Basis RF, XGBoost and LinearSVC models. Overall, the Algorithm Fairness Metric results are not stable, their value depend on many factors and its relation is often unclear. It is important to take each metric and feature combination characteristic into consideration when using predictive ML Algorithms on real-life cases. The underlying python code of this study distinguishes itself from other python packages as it computes all the Algorithm Fairness Metrics in one go, given any dataset that has five feature groups with $2, 3, 6, 7, 5$ items. I hope this inspires other researchers to use this as a source to gain insight into how the Algorithm Fairness Metrics interact and differ for their specific predictive ML Algorithms.

Keywords

Algorithm Fairness Metrics, Fairness Random Forest, Fairness Linear Support Vector Classifier, Fairness XGBoost.

Contents

Preface

Recently, an increasing amount of decisions have become automated by Machine Learning (ML) systems, emphasizing the importance of fair decision-making in society and being an incentive of this study. Gender, ethnicity, race and other personal characteristics may wrongly impact predictive ML Algorithms, resulting in unfair outcomes. Where most studies stop after mitigating the unfairness of ML Algorithms, it is worthwhile to look one step further and look at the impact of these Algorithms on the numerous Algorithm Fairness Metrics.

I have written this Master's Thesis whilst working as a Thesis student and Specialist at PwC's People and Organisation department. Part of their daily activity is developing and validating ML models in the banking and insurance industry. In these industries, unfair decision-making can have life long personal consequences. These events contributed to my determination and motivation to research such an important field.

The past nine months have been a real experience. In particular meeting Anand Rao, the Global Artificial Intelligence Lead at PwC, was very special. When discussing my thesis subject with him, he raised some very important points and once more emphasized the current importance of the fairness topic.

I would not have been able to perform this Master's Thesis, and especially the enormous amount of python coding, without the supervision from PwC and VU.

First of all, I would like to thank Pieter-Jan van Kessel from PwC who kindly gave me the opportunity to write my thesis at PwC whilst combining it alongside my part-time job as a Specialist. Alongside this, I have gained some very valuable working experience. Both Pieter-Jan and Wouter Karman from PwC are experts within the ML field and have been supervising me along the way for which I am very grateful.

Next, I would like to thank Dennis Dobler from VU for having discussions about interim result interpretation and subsequent steps, during and after office hours.

Also, I would like to thank Frank van Berkum from PwC who was my second reader of my first Master's Thesis. Without him I would have never got in touch with PwC in the first place.

Last but not least, I would like to thank my parents and sister for their support over the years, who would have ever thought I would finish both my Masters during the corona crisis.

Enjoy reading.

Raquel Humphris

Chapter 1

Introduction

As published at nos.nl by Schellevis and de Jong (2019), based on a Government Information Act and a confidential inventory of 54 government agencies, it became clear that the Dutch government uses predictive Machine Learning (ML) Algorithms on a large scale. As a reaction on a new Dutch Algorithm Law, a journalist from NRC.nl (2019) described that companies and governments are currently increasingly relying on predictive ML Algorithms to determine, for example, whether a job applicant should be hired, whether someone is committing fraud or whether you are eligible for a credit or mortgage application. To solve such classification problems, ML Algorithms often use labeled data, and therefore mostly supervised ML classification Algorithms. These ML Algorithms seem to offer a good outcome in the decision-making process, but there are major risks, as stated by Wilson (2019).

In the summer of 2019, the People's Party for Freedom and Democracy, a conservative-liberal Dutch political party, proposed new rules to protect society against the misuse of ML Algorithms and Artificial Intelligence, as described in a Dutch financial newspaper by van Wijnen (2019). In that same newspaper, Baurichter (2019) described that this raised the discussion whether the current General Data Protection Regulation (GDPR) is protecting us enough. The proposal of the Dutch political party followed after the proposal of the Democrats in the United States to create the Algorithmic Accountability Act (AAA) that should protect us from being discriminated by ML Algorithms.

Even though Dutch privacy lawyers Elisabeth Thole and Özer Zivali state that the GDPR is sufficient if it is maintained properly, the AAA was accepted by the United States senate, as mentioned by New (2019). Hence, firms with existing and new high-risk automated decision systems are required to submit an Impact Assessment Report at the Federal Trade Commission. In this, a company must be able to demonstrate that self-acting systems do their work accurately, fairly and without prejudice. This is to prevent ML Algorithms from discriminating against customers based on skin color, place of residence or gender. The system must also prevent third parties from accessing the sensitive private data of customers, as described by Baurichter (2019).

The lawsuit over a now controversial government program, as described by Schellevis (2019), and the outcome of the Correctional Offender Management Profiling for Alternative Sanctions (COMPAS), as mentioned by Larson et al. (2016), illustrate once more the problems with Algorithm unfairness that governmental applied models show.

The Algorithm Fairness topic is increasing in popularity and its discussions are not over yet. By testing the Algorithm Fairness Metrics and analyze the outcomes, this study aims to complement other research in the Algorithm Fairness field. Also, it may act as an aid for researchers to gain insight into how the Algorithm Fairness Metrics interact and differ for their supervised ML classification Algorithms in solving binary classification problems.

1.1 The Problem

Currently, predictive ML Algorithms become more and more integrated into existing practices and products. When their outcomes are unfair, this may have huge consequences for our daily lives. Past research mostly focused on applying ML Algorithms, measuring its performance, applying Algorithm Fairness Metrics and mitigating the unfairness based on this Fairness Metric.

Verma and Rubin (2018) distinguish 32 Algorithm Fairness Metrics. Much is unknown about how these 32 Algorithm Fairness Metrics compare for different dataset, supervised ML classification Algorithms, feature combinations, as well as their stability.

1.2 Research Question

To test the Algorithm Fairness Metrics for supervised ML classification problems, this study aims to answer the following research question: What is the impact of the dataset, supervised ML classification Algorithm choice, its Surrounding models, Bootstrapping the train / test dataset split and varying Seed choice for the train / test dataset split on the Algorithm Fairness Metrics?

1.3 Thesis Outline

The rest of this paper is organised as follows: Section 2 employs an overview of the literature about ML classification Algorithms, Algorithm Fairness Metrics and Algorithm performance metrics; Section 3 describes the two datasets that are used for this study; Section 4 contains the Method; Section 5 shows the results for the two datasets; Section 6 involves the Conclusion; Finally, Section 7 mentions potential Further Research in the Discussion.

Chapter 2

Literature Review

This Chapter contains a Machine Learning (ML) Overview which explains what part of ML is covered for this study, a description of important Supervised ML Classification Algorithms, followed by the Re-sampling Methods and the Model Tuning part. Also, the definition, causes, challenges, mitigation and metrics of Algorithm Fairness are described. The Chapter ends by a description of Algorithm Performance Classification Metrics as well as the approach of obtaining the threshold value from the ROC-curve.

2.1 Machine Learning Overview

As described by Sharma (2019), Artificial refers to something which is made by human or non natural things. Intelligence means the ability to understand or think. Artificial Intelligence (AI) is an intelligence where we want to add all the capabilities to machines that human contain.

Patel (2018) and Sharma (2019) address that ML is the learning in which machines can learn by its own without being explicitly programmed. It is an application of AI that provides systems the ability to automatically learn and improve from experience. In ML problems, the goal is to make predictions of unknown data by using ML Algorithms.

According to Yona (2017), ML Algorithms are, in essence, big computational machines that are trained to recognize and leverage statistical patterns in the data.

As explained by Pedregosa et al. (2019a), ML problems can be divided into four categories: Reinforcement Learning, Unsupervised Learning, Semi-Supervised Learning and Supervised Learning. This study only addresses Supervised Learning.

2.2 Supervised Learning Problems

As described by Patel (2018), in Supervised Learning, the training data is labeled. It establishes a learning process, compares the predicted results with the actual results of the training data, and continuously adjusts the predictive model until the predicted results of the model reach an expected accuracy, such as classification and regression problems. Figure 2.1 shows the Supervised Learning scheme where labeled training data is used for the Learning Algorithm to make a prediction, based on Zhu and Singh (2017).

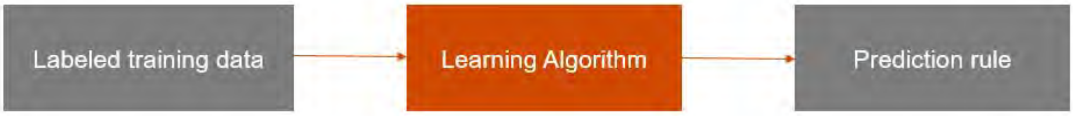

Figure 2.1: The Supervised Learning scheme.

As described by Garbade (2018), in classification problems, samples belong to two or more classes and we want to learn from already labeled data how to predict the class of unlabeled data. This study addresses three supervised ML classification Algorithms for binary classification. They are described in the next Section.

2.3 Supervised Machine Learning Classification Algorithms

This Section describes the characteristics of Decision Trees, Gradient Boosting, Random Forest, XGBoost and the Linear Support Vector Classifier. Also, the advantages and disadvantages are mentioned.

2.3.1 Decision Trees

Chakure (2019) explains that Decision Trees (DTs) are trees that are built through a process known as binary recursive partitioning. This is an iterative process of splitting the data into partitions, and then splitting it up further on each of the branches. To create a tree, the training examples are divided into subsets, this is the start of the tree development. This continues until the DT represents the training set. Hence, a new example is classified by submitting it to a series of tests that determine the class label of the example. These tests are organized in a hierarchical structure called a DT. DTs follow the Divide-and-Conquer Algorithm because it splits the data into subsets before it is split into even smaller subsets. Figure 2.2 shows a DT scheme.

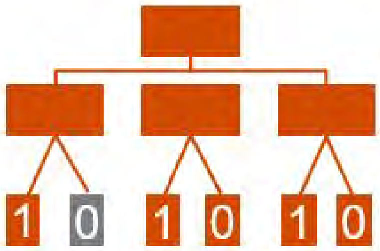

Figure 2.2: Decision Tree scheme.

Advantages: As described by Dhiraj (2019), DTs require, compared to other ML Algorithms, less effort for data preparation during pre-processing, requires no normalization or scaling of the data and it is very intuitive and easy to explain to technical teams, as well as stakeholders.

Disadvantages: As described by Chakure (2019) and Dhiraj (2019), overfitting is common, the decision boundary is restricted to being parallel to attribute axes, and small changes in the training data can result in large changes in the decision logic. For larger datasets, result interpretation becomes harder and the training time increases.

2.3.2 Gradient Boosting

As described by Elsinghorst (2018), Gradient Boosting (GB) is another technique for performing supervised ML tasks where weak learners are converted into strong learners. DTs are used as the weak learner in GB. It is common to constrain the weak learners in specific ways, such as a maximum number of layers, nodes, splits or leaf nodes. This is to ensure that the learners remain weak, but can still be constructed in a greedy manner, as described by Brownlee (2019b). Sing (2018) explains each new tree is a fit on a modified version of the original data set, which is where the AdaBoost Algorithm enters the scene.

Brownlee (2019b) explains the three elements of GB: A loss function to be optimized, a weak learner to make predictions and an additive model to add weak learners to minimize the loss function.

As explained by Grover (2017), the intuition behind the GB Algorithm is to repetitively leverage the patterns in residuals and strengthen a model with weak predictions and make it better. Once we reach a stage that residuals do not have any pattern that could be modeled, we can stop modeling residuals (otherwise it might lead to overfitting). Algorithmically, we are minimizing our loss function, such that test loss reach its minimal.

Brownlee (2019b) describes that the first realization of boosting that saw great success in application was Adaptive Boosting, or AdaBoost for short. The AdaBoost Algorithm begins by training a DT in which each observation is assigned an equal weight. After evaluating the first tree, we increase the weights of those observations that are difficult to classify and lower the weights for those that are easy to classify. The second tree is therefore grown on this weighted data. The idea is to improve upon the predictions of the first tree, as stated by Sing (2018).

Compared to XGBoost, see Section 2.3.4, GB Machines (GBMs) use the metric error to evaluated measure the model, instead of a bunch of metrics such as error and log loss, as described by Sing (2018). Figure 2.3.2 shows the GB Scheme where the error reduces each tree, it is inspired by Serengil (2019).

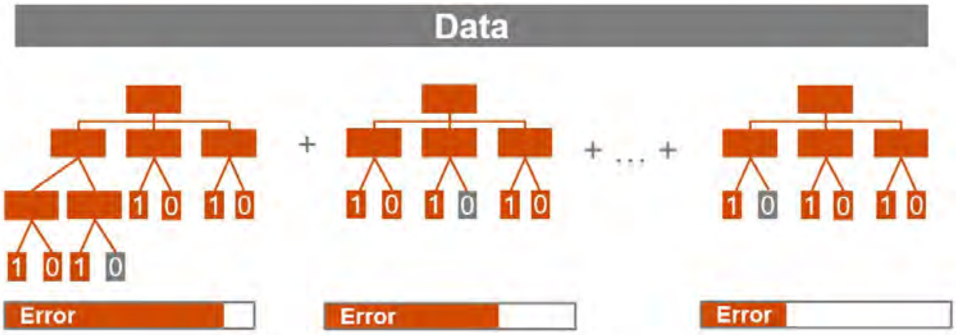

Figure 2.3: Gradient Boosting scheme.

Advantages: Brownlee (2019b) states that a new boosting Algorithm does not have to be derived for each loss function that may want to be used, instead, it is a generic enough framework that any differentiable loss function can be used. Also, Github (2019a) describes that its predictive accuracy cannot be beat, no data pre-processing is required and it handles missing data well.
Disadvantages: Brid (2018) explains that GBMs will continue improving to minimize all errors. According to Kumar (2020), this can overemphasize outliers and cause overfitting. You have to use cross-validation to neutralize this. GBMs can also be computationally expensive since it often require many trees (> 1000). Other disadvantages are the high flexibility results in many hyperparameters that interact and influence heavily the behavior of the approach which requires a large Grid Search during hyperparameter tuning and its result are less interpretable, as described by Github (2019a).

2.3.3 Random Forest

Glen (2019) describes Random Forests (RFs) as a large number of DTs, combined at the end of the process. Morde and Setty (2019) states that DTs are a graphical representation of possible solutions to a decision based on certain conditions. In the evolution of tree-based Algorithms, after DTs and bagging, RF evolves.

According to Morde and Setty (2019), bagging is an ensemble meta-Algorithm combining predictions from multiple DTs through a majority voting mechanism. Hastie et al. (2009) explains that RFs combine the simplicity of DTs with flexibility resulting in a vast improvement in accuracy. Figure 2.4 shows the Random Forest Scheme.

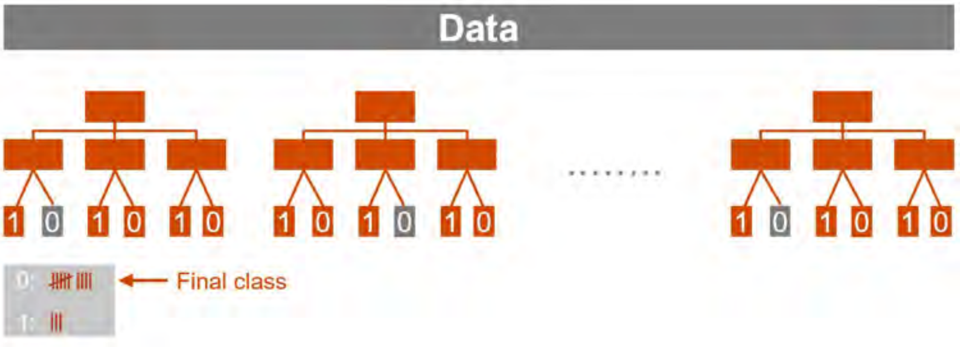

Figure 2.4: Random Forest Scheme.

Advantages: Kho (2018) describes the benefits of RFs as being Impressive in Versatility, being Parallelizable, being great with high dimensional data, being faster to train than DT because we are working only on a subset of features in this model. He also explains that each DT has a high variance, but low bias. But because we average all the trees in RF, we are averaging the variance as well so that we have a low bias and moderate variance model.
Disadvantages: Some drawbacks for RFs are that they are not all that interpretable; they are like black boxes, and for large data sets, the size of the trees can take up a lot of memory, as described by Kho (2018) and Jansen (2020).

The Random Forest Algorithm

Hastie et al. (2009) describe the RF Algorithm for classification:

1. For b = 1 to B:
    (a) Draw a Bootstrap sample Z^* of size N from the training data.
    (b) Grow a random-forest tree T_b to the Bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
        i. Select m variables at random from the p variables.
        ii. Pick the best variable/split-point among the m.
        iii. Split the node into two daughter nodes
2. Output the ensemble of trees $\{T_b\}_1^B$.
    - To make a prediction at a new point x: Let $\hat{C}_b(x)$ be the class prediction of the b^{th} random-forest tree. Then $\hat{C}_r^B f(x) = majority\ vote\ \{\hat{C}_b(x)\}_1^B$.

2.3.4 XGBoost

As explained by Morde and Setty (2019) and Brownlee (2016), XGBoost stands for eXtreme Gradient Boosting (GB) and is a DT-based ensemble ML Algorithm that uses a GB framework. As mentioned by Morde and Setty (2019), over the years, DTs have evolved to bagging, RF, boosting, GB and then finally to XGBoost. Two widely used ensemble learners are bagging and boosting which both help to reduce the variance in any learner, as described by Sundaram (2018).

As explained by Mandot (2019), in boosting, the trees are built sequentially such that each subsequent tree aims to reduce the errors of the previous tree. Each tree learns from its predecessors and updates the residual errors. Hence, the tree that grows next in the sequence will learn from an updated version of the residuals. The combining process of the DTs start at the beginning, instead of at the end like RFs, mentions Glen (2019).

GB involves creating and adding DTs to an ensemble model sequentially. New trees are created to correct the residual errors in the predictions from the existing ensemble. The simplest hyperparameters are the maximum depth of the DTs being trained, the loss function being used, and the number of classes in the dataset. The eta Algorithm requires special attention since it gives us a chance to prevent overfitting. The eta can be thought of more intuitively as a learning rate. Rather than simply adding the predictions of new trees to the ensemble with full weight, the eta will be multiplied by the residuals being adding to reduce their weight. This effectively reduces the complexity of the overall model, as all explained by Seif (2019).

Advantages: As explained by Elsinghorst (2018) and Brownlee (2016), advantages of using XGBoost are its execution speed and model performance. Compared to GBMs, XGBoost has a number of nifty tricks that make it exceptionally successful, particularly with structured data. It has advanced regularization which improves model generalization and it computes second partial derivatives. The latter provides more information about the direction of gradients and how to get to the minimum of our loss function.
Disadvantages: Jain (2016) explains that the XGBoost model requires parameter tuning to improve and fully leverage its advantages over other ML Algorithms. Also, the implementation of gradient boosted machines is relatively slow, due to the model training that must follow a sequence. They, therefore, lack scalability due to their slowness, as stated by CFI (2020).

The XGBoost Algorithm

Sundaram (2018) describes the XGBoost Algorithm, since we use the notation from Hastie et al. (2009), the notation of a qualitative response variable is G. He defines model F_0 as the initial model to predict G, model H_1 is fit to the residuals of this initial model. F_0 and H_1 combined gives F_1, the boosted version of F_0. Mathematically, the steps are as followed:

1. Define a function F_0 which minimizes the loss function or MSE:
$$F_0(x) = argmin_\gamma \sum_{i=1}^{n} L(G_i, \gamma) = argmin_\gamma \sum_{i=1}^{n} (G_i - \gamma)^2. \tag{2.1}$$
The corresponding residual is denoted by $G - F_0$.
2. Fit model H_1 to the residual of step 1. This will be a regression tree which will try and reduce the residuals from the previous step. Its output will help in predicting the successive function $F_1(x)$ that brings down the residuals, here $F_1(x) = F_0(x) + H_1(x)$. Hence, $H_1(x)$ learns from the residuals of $F_0(x)$ and suppresses it in $F_1(x)$.
3. Repeat to compute model H_2 and H_3, each makes use of the residuals from the preceding function.

Gradient Boosting is then:

1. $F_0(x)$ is as defined for boosting.
2. The gradient boost of the loss function is:
$$r_{im} = -\alpha[\frac{\delta(L(G_i, F(x_i))}{\delta F(x_i)}]_{F(x)=F_{m-1}(x)}. \tag{2.2}$$
3. Fit each model $H_1, H_2, ...$ on the gradient obtained at each step.
4. Derive the multiplicative factor $\gamma_1, \gamma_2, ...$ and the boosted model $F_1, F_2, ...$ by:
$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x), \quad \text{for m = 1, 2, ...} \tag{2.3}$$

2.3.5 Linear Support Vector Classification

The Linear Support Vector Classification (LinearSVC) is a scalable Linear Support Vector Machine (SVM) for classification, implemented using liblinear, as mentioned by Pedregosa et al. (2020a). SVMs is the technique you use when you want to optimally separate hyperplanes for two classes that are not linearly separable and the classes overlap, as described by Hastie et al. (2009). It produces nonlinear boundaries by constructing a linear boundary in a large, transformed version of the feature space.

According to Navlani (2018), SVMs offers higher accuracy than other classifiers such as DTs and it is known for its kernel trick to handle nonlinear input spaces. The classifier separates data points using a hyperplane with the largest amount of margin to find an optimal hyperplane which helps in classifying new data points, explains Navlani (2018).

Cortes and Vapnik (1995) explain, in the support-vector network, three ideas are combined. First, the solution technique from optimal hyperplanes which allows for an expansion of the solution vector on support vectors. Secondly, the idea of convolution of the dot-product which extends the solution surfaces from linear to non-linear. Thirdly, the idea of soft margins which allow for errors on the training set.

Figure 2.5 shows the high level view of what the SVM does, the yellow line is the hyperplane separating orange and grey points, it is inspired by Medium (2019) and Sanjeevi (2017).

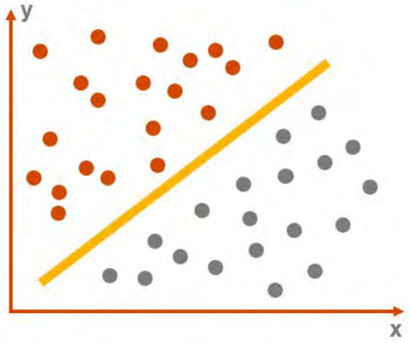

Figure 2.5: Support Vector Machine scheme.

Advantages: The advantages of using SVMs are its effectivity in high dimensional spaces and it is still effective in cases where number of dimensions is greater than the number of samples, as explained by Pedregosa et al. (2019b). Also, it used a subset of training points in the decision function (called support vectors), so it is also memory efficient. Finally, SVMs are versatile, meaning that different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels, as stated by Pedregosa et al. (2019b).
Disadvantages: Pedregosa et al. (2019b) describe the disadvantages of the SVMs, they state that if the number of features is much greater than the number of samples, avoiding over-fitting in choosing Kernel functions and regularization term is crucial. Moreover, they do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

Using Linear kernels or LinearSVC may be an alternative to reduce the computation time significantly, as explained by Pedregosa et al. (2019b). Pedregosa et al. (2020a) described that LinearSVC is similar to SVC with parameter $kernel = linear$, but implemented in terms of liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples.

The Support Vector Machine Algorithm

As described by Hastie et al. (2009), for training data pairs $(x_1, y_1), ...(x_N, y_N)$ with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$, define a hyperplane by:

$$x : f(x) = x^T \beta + \beta_0 = 0, \tag{2.4}$$

where β is a unit vector $||\beta|| = 1$. A classification rule induced by $f(x)$ is

$$G(x) = sign[x^T \beta + \beta_0]. \tag{2.5}$$

The convex optimization problem, also known as the support vector criterion for separated data is:

$$min_{\beta,\beta_0} ||\beta|| \text{ subject to } y_i(x_i^T \beta + \beta_0) \geq 1, i = 1, ..., N, \tag{2.6}$$

where the norm constraint on β is dropped and $M = \frac{1}{||\beta||}$. For the non-separable case, also a convex optimization problem, the support vector classifier is:

$$min||\beta|| \text{ subject to } \begin{cases} y_i(x_i^T \beta + \beta_0) \geq 1 - \xi_i \forall i, \\ \xi_i \geq 0, \sum \xi_i \leq \text{constant}. \end{cases} \tag{2.7}$$

The value ξ_i is is the proportional amount by which the prediction $f(x_i) = x_i^T \beta + \beta_0$ is on the wrong side of its margin. By bounding it, the total proportional amount by which prediction fall on the wrong side of their margin. By bounding its sum, the total number of training misclassifications at a constant value is bounded. For convenient computation, the support vector classifier for the non-separable case is:

$$min_{\beta,\beta_0} \frac{1}{2} ||\beta||^2 + C \sum_{i=1}^{N} \xi_i \text{ subject to } {}_i \geq 0, y_i(x_i^T \beta + \beta_0) \geq 1 - \xi_i \forall i, \tag{2.8}$$

where C is the cost parameter that replaces the previous constraint. Larger values of C focus attention more on (correctly classified) points near the decision boundary, while smaller values involve data further away. Either way, misclassified points are given weight, no matter how far away. Using cross-validation, the optimal choice for C can be made. The quadratic programming solution can be described by Lagrange multipliers, the Lagrange primal function is:

$$L_P = \frac{1}{2} ||\beta||^2 + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i [y_i(x_i^T \beta + \beta_o) - (1 - \xi_i)] - \sum_{i=1}^{N} \mu_i \xi_i, \tag{2.9}$$

minimizing this with respect to β, β_0 and ξ_i and setting the respective derivatives to zero you obtain six equations that characterize the solution to the primal and dual problem. First, you get $\beta = \sum_{i=1}^{N} \alpha_i y_i x_i, 0 = \sum_{i=1}^{N} \alpha_i y_i, \alpha_i = C - \mu_i, \forall i$ with $\alpha_i, \mu_i, \xi_i \geq 0 \forall i$. The Lagrangian dual objective function then becomes:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha i' y_i y_{i'} x_i^T x_{i'}^T, \tag{2.10}$$

which you maximize subject to $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$. The last three equation are the Karush-Kuhn-Tucker conditions which are $\alpha_i [y_i(x_i^T \beta + \beta_0) - (1 - \xi_i)] = 0, \mu_i \xi_i = 0, y_i(x_i^T \beta + \beta_0) - (1 - \xi_i) \geq 0$. Figure 2.6 shows the support vector classifiers for the nonseparable case where ξ_i represents the points that are on the wrong side with a distance of $M * \xi_i$, as explained by Navlani (2018) and Hastie et al. (2009).

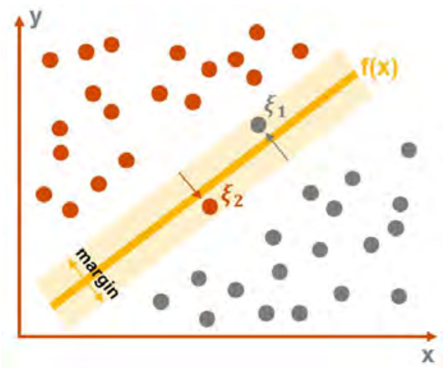

Figure 2.6: Support Vector Machine with margins and ξ_i.

2.4 Re-sampling Methods

As described by Sagar (2019) and Peixeiro (2019), re-sampling is the method of taking samples iteratively from the original data samples which is an indispensable tool in modern statistics. According to Sagar (2019), the method of re-sampling is a non-parametric method of statistical inference which means that the parametric assumptions that ignore the nature of the underlying data distribution are avoided. Peixeiro (2019) explains that it involves repeatedly drawing samples from a training set and refitting a model of interest on each sample in order to obtain additional information about the fitted model. Two commonly used re-sampling methods are Bootstrap and Cross Validation, as mentioned by Sagar (2019).

2.4.1 Bootstrap

As explained by Brownlee (2018a), the Bootstrap method is a re-sampling technique used to estimate statistics on a population by sampling a dataset with replacement. It is used in applied ML to estimate the skill of ML models when making predictions on data not included in the training data. Frost (2020) describes the Bootstrap method:

- The Bootstrap method has an equal probability of randomly drawing each original data point for inclusion in the re-sampled datasets.
- The procedure can select a data point more than once for a re-sampled dataset. This property is the "with replacement" aspect of the process.
- The procedure creates re-sampled datasets that are the same size as the original.

The process ends with your simulated datasets having many different combinations of the values that exist in the original dataset. Each simulated dataset has its own set of sample statistics, such as the mean, median, and standard deviation. Bootstrapping procedures use the distribution of the sample statistics across the simulated samples as the sampling distribution, as stated by Frost (2020).

Hence, as described by Raschka (2016), the Bootstrap method can be used when you are particularly interested in estimating the uncertainty of the performance estimates.

2.4.2 Cross Validation

Brownlee (2018b) describes Cross Validation (CV) as being a statistical method used to estimate the skill of ML models. It is commonly used in applied ML to compare and select a model for a given predictive modeling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods. That K-fold CV is a procedure used to estimate the skill of the model on new data, as explained by Brownlee (2018b).

As stated by Malik (2018), the process of K-Fold CV is straightforward. You divide the data into K folds. Out of the K folds, K-1 sets are used for training while the remaining set is used for testing. The Algorithm is trained and tested K times, each time a new set is used as testing set while remaining sets are used for training.

Figure 2.7 shows the scheme of K-Fold CV, as inspired by Krishni (2018). In the end, the result of the K-Fold CV is the average of the results obtained on each set.

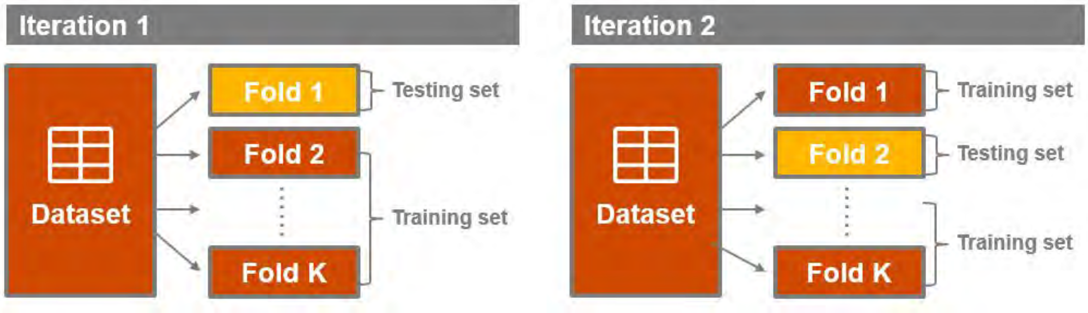

Figure 2.7: K-Fold CV scheme.

2.5 Model Tuning

Ippolito (2019) describes three common approaches to hyperparameter optimization, they are Grid Search, Random Search and Genetic Algorithms. Scikit-learn contains the GridSearchCV and RandomizedSearchCV objects from Pedregosa et al. (2020b) and Pedregosa et al. (2020c), which enables the use of Cross Validation (CV) and parameter tuning, while Jain (2018) describes the Genetic Algorithm for tuning hyperparameters.

2.5.1 Cross Validation

ML models tuning is a type of optimization problem where both CV and hyperparameters are important, as stated by Ippolito (2019).

As described by Mandava (2018), model parameters are internal to the model whose values can be estimated from the data while hyperparameters are external to the model and cannot be directly learned from the regular training process. These parameters express “higher-level” properties of the model such as its complexity or how fast it should learn. Hyperparameters are model-specific properties that are ‘fixed’ before you even train and test your model on data.

Pedregosa et al. (2020b) explains both CV and hyperparameter tuning are commonly done at the same time in data pipelines, and are defined as followed:

- CV is the process of training learners using one set of data and testing it using a different set.
- Hyperparameter tuning is the process to selecting the values for a model’s parameters that maximize the accuracy of the model.

2.5.2 Grid Search

Grid Search CV conducts an exhaustive search for the combination of hyperparameters that maximizes the CV performance, according to some user-defined score function, as stated by Lopez de Prado (2020). Pedregosa et al. (2020b) describes that GridSearch CV implements a “fit” and a “score” method where the hyperparameters of the estimator that are used to apply these methods, are optimized by a cross-validated Grid Search.

The theory on DataCamp (2020) explains that GridSearch CV can be computationally expensive, especially if you are searching over a large hyperparameter space and dealing with multiple hyperparameters. A solution is the use of RandomizedSearchCV.

Mantovani et al. (2015) describes that, the hypothesis that a simple Random Search method is sufficient to adjust the hyper-parameters of SVMs is investigated. A set of experiments compared the performance of five tuning techniques: three meta-heuristics commonly used, Random Search and Grid Search. The experimental results show that the predictive performance of models using Random Search is equivalent to those obtained using meta-heuristics and Grid Search, but with a lower computational cost.

2.5.3 Random Search

An advantage of using Random Search is that not all hyperparameter values are tried out, as stated by the theory on DataCamp (2020). Instead, a fixed number of hyperparameter settings is sampled from specified probability distributions. Koehrsen (2018a) explains that both Grid Search and Random Search are hands-off, but require long run times because they waste time evaluating unpromising areas of the search space.

Pedregosa et al. (2020b) mentions the two main benefits of the Random Search. First of all, a budget can be chosen independent of the number of hyperparameters and possible value. Secondly, adding hyperparameters that do not influence the performance does not decrease efficiency.

Stalfort (2019) went through the conceptual explanation of Grid Search and Random Search to understand which one is better for the Random Forest Algorithm. The conclusion was that Random Search is preferred.

2.5.4 Genetic Algorithm

Wirsansky (2020) describes that the usage of the Genetic Algorithm approach allowed them to improve the results of the Grid Search. Instead of trying out every possible combination of hyperparameters, using evolutionary Algorithms instead of Grid Search in Scikit-learn, will cause to evolve only the combinations that give the best results while also reducing the time required to find the best hyperparameters for your estimator, as stated in the Github (2019b) that describes Sklearn-deap.

Especially for the XGBoost, where many hyperparameters were tuned in this study, Genetic Algorithm may reduce the computation time a lot. The theory behind this is explained by Jain (2018) and is as followed:

1. Initialization: The initial population, which will have certain traits and will be tested in a certain environment to observe how well the individuals (parents) in this population perform, based on a predefined fitness criteria.
2. Selection: Based on the fitness value, the top performing parents are selected and labelled as the survived population.
3. Crossover: The parents in the survived population will mate to produce offspring, using a combination of two steps. Crossover is the first step where the genes (parameters) from the mating parents will be recombined, to produce offspring, with each children inheriting some genes (parameters) from each parent.
4. Mutation: The second step is mutation where some of the values of the genes (parameters) will be altered to maintain a genetic diversity. This allows the nature/Genetic Algorithm to typically arrive at a better solution.

Then, the second generation of the population contain both survived parents as well as children. The survived parents are kept to retain best fitness hyperparameters in case the offspring's fitness value turns out to be worse than the parents. The corresponding Python code forms the inspiration for the code of this study.

2.6 Algorithm Unfairness

This section will discuss the definition, causes, metrics and mitigation of Algorithm Unfairness. As described by Jacobson (2019), despite the seeming objectivity of the process of training ML Algorithms to maximize prediction accuracy on training data, it sometimes results in ML Algorithms that, while computationally correct, produce outputs that are biased and unjust from a human perspective.

2.6.1 Causes And Challenges Of Algorithm Unfairness

Jacobson (2019) states that a large piece of the challenge is that ML Algorithms can only be as fair as the data itself. If the underlying data is biased in any way, there is a risk that its structural inequalities will not only be replicated but possibly even amplified in the ML Algorithm. ML engineers must be aware of their own blind spots and implicit assumptions; all the small decisions they make about finding, organizing, and labeling training data for these models can be as impactful as their choice of ML techniques, as described by Jacobson (2019).

According to Hao (2019), often, the training data is blamed for being biased and therefore are the reason for Algorithm Unfairness. The Algorithm Unfairness causes can be distinguished by the following three stages: the framing problem, data collection problem and data preparation problem. Hao (2019) describes them as followed:

- Framing problem: The variable that you want to predict can be defined in various ways, this is decided based on on the context of the goal. Also, as explained by Baron and Hershey (1988), these decisions are made for various business reasons other than Fairness or discrimination.
- Data collection problem: Training data may be unrepresentative of reality, or it reflects prejudices. Prejudices may be reflected if the data contains historical decisions that show that men are hired more often than women. The ML Algorithm will hence do the same thing. By selecting which features to use for the ML Algorithm to make a prediction, you consequently introduce bias.
- Data preparation problem: Choosing which attributes to consider or ignore can significantly influence your model's prediction accuracy. But while its impact on accuracy is easy to measure, its impact on the model's bias is not.

2.6.2 Definitions Of Algorithm Unfairness

Whatever the causes of Algorithm Unfairness or its corresponding challenges, its definition is not straightforward. There are many definitions of Unfairness, depending on the context, the corresponding dataset but also the researcher in question. However, the majority agrees this is not a trivial task and that Unfair ML Algorithms are skewed toward a particular group, as can be read from for example Jacobson (2019), Mehrabi et al. (2019) and Yona (2017).

Verma and Rubin (2018) describe 32 definitions of Fairness for the Algorithmic classification problem, see Table 2.1 for an overview. After explaining the rationale behind the definitions, they apply them on a case-study.

2.6.3 Notation

This study uses the notations that are inspired by the paper of Verma and Rubin (2018). For each dataset, N characteristics define individual k with $k = 1, ..., K$. Characteristic n_i with $i = 1, ..., N$, having M_{n_i}. The collection of features is binary, characteristics of individual k are denoted by:

$$X_k([n1; n_N]) = [m(n_1, 1), ..., m(n_1, M_{n_1}), ..., m(n_N, 1), ..., m(n_N, M_{n_N})]. \tag{2.11}$$

Statistical Fairness Measures	1-12. Statistical Metrics	
	Definitions based on predicted outcome	13. Group Fairness
		14. Conditional statistical parity
	Definitions based on predicted and actual outcomes	15. Predictive parity
		16. Predictive quality
		17. Equal opportunity
		18. Equalized odds
		19. Conditional use accuracy equality
		20. Overall accuracy equality
		21. Treatment equality
	Definitions based on predicted probabilities and actual outcome	22. Test-fairness
		23. Well-calibration
		24. Balance for positive class
		25. Balance for negative class
Similarity-Based Measures	26. Causal discrimination	
	27. Fairness through unawareness	
	28. Fairness through awareness	
Causal Reasoning	29. Counterfactual Fairness	
	30. No unresolved discrimination	
	31. No proxy discrimination	
	32. Fair inference	

Table 2.1: Fairness definitions, as distinguished by Verma and Rubin (2018).

For a specific characteristic n_i, individual k can be described by:

$$X_k(n_i) = [m(n_i, 1), ..., m(n_i, M_{n_i})]. \tag{2.12}$$

To compare outcomes for different characteristics, take $j(i) = 1, ..., M_{n_1}$ and $l(i) = 1, ..., M_{n_1}$ where $j(i) \neq l(i)$. The classification of individual k is $Y_k = c$ with $C \in 0, 1$. The predicted probability of individual k being part of classification c is:

$$S_{k,c} = P(Y_k = c | X_k([n_1; n_N])). \tag{2.13}$$

The predicted classification of individual k is $d_k = c$. In general, if $s_k > g$, $d_k = 1$ where g is a certain threshold.

2.6.4 Fairness Metrics

The following subsections describe the 32 Algorithm Fairness Metrics of Verma and Rubin (2018), by applying the notation as described above. Six Algorithm Fairness Metric types can be distinguished. The Algorithm Fairness Metric model numbers are indicated in the brackets.

Algorithm Fairness Metrics Type 1: Statistical Metrics $(1-12)$

Verma and Rubin (2018) explain that Statistical Metrics, the first 12 Algorithm Fairness Metrics, form the basis of all the Algorithm Fairness Metrics.

As explained by Sunasra (2017), the Confusion matrix contains four values, the True Positive (TP), False Negative (FN), False Positive (FP) and the True Negative (TN) values. They form the first four Algorithm Fairness Metrics and are described as:

1. TP values occur when you predict a “Yes” which is the same as the actual value.
2. FN values occur when you predict “No” while the actual value is “Yes”.
3. FP values occur when you predict a “Yes” while the actual value is “No”.
4. TN values occur when you predict a “No” which is the same as the actual value.

The remainder eight Algorithm Fairness Metrics can be found in Table 2.2.

Statistical metric	Description
Positive predictive value (PPV)	$\frac{TP}{TP+FP} = P(Y = 1\|d = 1)$
False Discovery Rate (FDR)	$\frac{FP}{TP+FP} = P(Y = 0\|d = 1)$
False Omission Rate (FOR)	$\frac{FN}{TN+FN} = P(Y = 1; d = 0)$
Negative Predictive Value (NPV)	$\frac{TN}{TN+FN} = P(Y = 0\|d = 0)$
True Positive Rate (TPR)	$\frac{TP}{TP+FN} = P(d = 1\|Y = 1)$
False Positive Rate (FPR)	$\frac{FP}{FP+TN} = P(d = 1\|Y = 0)$
False Negative Rate (FNR)	$\frac{FN}{TP+FN} = P(d = 0\|Y = 1)$
True Negative Rate (TNR)	$\frac{TN}{FP+TN} = P(d = 0\|Y = 0)$

Table 2.2: Algorithm Fairness Metric 4-12.

Algorithm Fairness Metrics Type 2: Def. based on predicted outcome $(13, 14)$

The second Algorithm Fairness Metric type consists of the following two Algorithm Fairness Metrics:

1. Group Fairness: $P(d = 1|m(n_i, b(i)) = P(d = 1|m(n_i, l(i))$
2. Conditional Statistical Parity: Using a set of attributes $L = m(n_i, j(i)), ...$, this is calculated by $P(d = 1|L, m(n_i, n(i)) = P(d = 1|L, m(n_i, l(i))$

where $b(i) = 1, ..., M_{n_i}$ and $l(i) = 1, ..., M_{n_i}$ where $b(i) \neq l(i)$.

Algorithm Fairness Metrics Type 3: Def. based on predicted and actual outcomes $(15 - 21)$

The third Algorithm Fairness Metric type includes the following seven Algorithm Fairness Metrics:

1. Predictive parity: $P(Y = 1|d = 1, m(n_i, b(i))) = P(Y = 1|d = 1, m(n_i, l(i)))$
2. Predictive quality: $P(d = 1|Y = 0, m(n_i, b(i))) = P(d = 1|Y = 0, m(n_i, b(i)))$
3. Equal opportunity: $P(d = 0|Y = 1, m(n_i, b(i)) = P(d = 0|Y = 1, m(n_i, l(i)))$
4. Equalized odds:
   $P(d = 1|Y = 1, m(n_i, b(i))) = P(d = 1|Y = 1, m(n_i, l(i))) \qquad \wedge$
   $P(d = 1|Y = 0, m(n_i, b(i))) = P(d = 1|Y = 0, m(n_i, l(i)))$
5. Conditional use accuracy equality:
   $P(Y = 1|d = 1, m(n_i, b(i))) = P(Y + 0|d = 0, m(n_i, l(i)))$
6. Overall accuracy equality: $P(d = Y, m(n_i, b(i))) = P(d = Y, m(n_i, l(i)))$
7. Treatment equality: $\frac{FN}{FP} m(n_i, b(i)) = \frac{FN}{FP} m(n_i, l(i))$

where $b(i) = 1, ..., M_{n_i}$ and $l(i) = 1, ..., M_{n_i}$ where $b(i) \neq l(i)$.

Algorithm Fairness Metrics Type 4: Def. based on predicted probabilities and actual outcome $(22 - 25)$

The following four Algorithm Fairness Metrics belong to the fourth Algorithm Fairness Metric type:

1. Test Fairness: $P(Y = 1|S = s, m(n_i, b(i))) = P(Y = 1|S = s, m(n_i, l(i)))$
2. Well calibration: $P(Y = 1|S = s, m(n_i, b(i))) - P(Y = 1|S = s, m(n_i, l(i))) = s$
3. Balance for positive class: $E(S|Y = 1, m(n_i, b(i))) = E(S|Y = 1, m(n_i, l(i)))$
4. Balance for negative class: $E(S|Y = 0, m(n_i, b(i))) = E(S|Y = 1, m(n_i, l(i)))$

for $s = [0, 0.1, ..., 1.0]$. and where $b(i) = 1, ..., M_{n_i}$ and $l(i) = 1, ..., M_{n_i}$ and $b(i) \neq l(i)$ holds.

Algorithm Fairness Metrics Type 5: Similarity based measures (26 – 28)

The fifth Algorithm Fairness Metric type addresses the following three Algorithm Fairness Metrics:

1. Causal discrimination: For identical characteristics, except for one, create $X_{m(n_i,n(i))} = X_{m(n_1,l(i))}$ and also $d(m(n_i,n(i))) = d(m(n_1,l(i)))$ where $b(i) = 1,...,M_{n_i}$ and $l(i) = 1,...,M_{n_i}$ where $b(i) \neq l(i)$.
2. Fairness through unawareness: After removing sensitive attribute, create $X_{m(n_i,b(i))} = X_{m(n_1,l(i))}$ and also $d(m(n_i,n(i))) = d(m(n_1,l(i)))$ where $b(i) = 1,...,M_{n_i}$ and $l(i) = 1,...,M_{n_i}$ where $b(i) \neq l(i)$.
3. Fairness through awareness: The distance between the outcome probabilities for two applicants is: $D(i,J) = S(i) - S(j)$. The k value is defined as being the difference is average of each feature option. For each Feature option, the average D is calculated and the percentage violating cases is defined as the percentage for which $D(M(x), M(y(x)) \leq k(x,y)$.

Algorithm Fairness Metrics Type 6: Causal Reasoning (29 – 32)

The following definitions are used for the creation of the Causal graph:

- Using the chi-squared test to determine the dependency. When there is a dependency, a path is drawn between the two features.
- When the value of one feature can be used to derive a value of another, it is called a proxy attribute.
- When a feature is independent of another, meaning one cannot draw a path between them, but indirectly you can arrive at it, this is called a resolving attribute.
- When the indirect path is via a two proxy attributes, it is called a discriminatory path.
- When the indirect path is via a resolving attribute, it is called a non-discriminatory path. There is a distinction between protected and unprotected features.

The following four Algorithm Fairness Metrics belong to this sixth Algorithm Fairness Metric type:

1. Counterfactual fairness: Fairness is achieved when the predicted outcome is independent of the protected features. In other words, there is no direct descendant from a protected feature to the predicted outcome.
2. No unresolved discrimination: Fairness is achieved when there is only an indirect link, hence via a resolving feature, from the protected feature to the predicted outcome.
3. No proxy discrimination: Fairness is achieved when there exist no path from the protected feature to the predicted outcome that is blocked by a proxy variable.
4. Fair inference: Paths are classified as either legitimate or illegitimate. When a feature is a proxy for the protected attribute, this path is illegitimate. Hence, Fairness is achieved when there are no illegitimate paths from the protected feature to the predicted outcome

2.6.5 Unfairness Mitigation Problems And Solutions

Many research has been done to mitigate the Unfairness to achieve Algorithmic Fairness. According to Jacobson (2019), achieving this seems as difficult as achieving Fairness in human-led decision-making systems. Human systems are biased in all of the ways that Algorithmic systems are biased — since both are human creations — and human decision-makers are additionally biased in ways that machines are not.

Hao (2019) describes four reasons of why Algorithm Unfairness is hard to fix. First, the introduction of bias is not always obvious during a model's construction because you may not realize the downstream impacts of your data and choices until much later. Removing explicit gendered words like "women" and "men" may not be enough since there may also be implicitly gendered words, being words that are apparently highly correlated with the explicit gendered words. This is defined as the unknown unknowns. Secondly, many standard practices are not designed with bias detection in mind. Since data is split into the training set and testing set, the data you use to test your ML Algorithm will have the same bias as the part that was used to train the ML Algorithm. Therefore, skewed or prejudiced results can be unwitnessed. The third reasons is about the lack of social context. An ML Algorithms can be used for different tasks in different contexts. According to Selbst et al. (2019), this causes the portability trap since ML Algorithms ignore a lot of social context. He states: "You can't have a system that you apply for 'fair' criminal justice results then applied to employment. How we think about Fairness in those contexts is just totally different." Finally, the definition of Unfairness has a long history of debate in philosophy, social science and law. In the ML field, its definition should be defined in mathematical terms which is a task on itself. The difference within the ML is that the concept of Fairness has to be defined in mathematical terms. As stated by Cage (2016), there is a mathematical limit to how fair any ML Algorithm or human decision-maker can ever be.

According to Yona (2017), hiding information from the ML Algorithm may be the best way to treat different groups the same. However, removing the protected attribute alone is not sufficient since it is often redundantly encoded in the rest of the observed features. Two issues were described by Yona (2017), the aware versus the unaware issues as a Fairness of the process and Fairness of the outcome, their definition is as followed:

- The unaware approach ensures Fairness of the process since it forces the fact that during the learning phase, the ML Algorithm does not in any way treat individuals differently based on their protected attribute. However, the final outcome can actually be less fair towards the protected and non protected sub-groups.
- The aware approach uses a process that is not fair since it explicitly uses gender information, and learns different classification rules for people of different parts of the population. However, aware approach can actually reach an outcome that is more fair towards the minorities.

Another approach is applying Treatment equality or Conditional procedure accuracy equality, describes Berk et al. (2017). Treatment equality is achieved by a classifier that yields a ratio of false negatives and false positives that is the same for both protected group categories. Conditional procedure accuracy equality is achieved when conditioning on the known outcome, the classifier is equally accurate across protected group categories. This is equivalent to the FN rate and FP rate being the same for men and women. However, since there is a trade-off, it is not easy to determine which of the two methods is the best. Ultimately it is up to the stakeholders to determine the trade-offs.

According to Hellman (2019), another alternative is focusing on whether the scores produced by the ML Algorithm are equally predictive for each group or we could focus on whether the error rates produced by the ML Algorithm are equal. The controversy appears to focus on those two measures. To summarize, ML Algorithms are used to predict some endpoint of interest – sickness, recidivism, or a multitude of other possible traits. These ML Algorithms generally avoid the use of classifications that are protected by antidiscrimination law, like race or sex. The ML Algorithm can exhibit equal predictive value such that scores will be equally predictive of the target trait for members of one group as for members of the other. Or, the ML Algorithm can exhibit error rate balance such that people of each group who have or lack the target variable are equally likely to be accurately scored by the test.

2.7 Algorithm Performance Classification Metrics

To estimate the generalization error of a model, it is required to test the model with a dataset which it has not seen yet, this is the testing dataset, as stated by DeZyre (2019). One has to determine how to divide the dataset into the training dataset and testing dataset. With less training data, your parameter estimates have greater variance. With less testing data, your performance statistic will have greater variance. When the dataset contains "enough" data, there is not a big difference between using 90% or 80% of the dataset for training. In both cases, the testing dataset is large enough to yield meaningful results and is representative of the dataset as a whole, as mentioned by Google Developers Machine Learning (2019).

To measure the performance of a classification model, methods such as Precision-Recall, Classification Accuracy, F1 score, Log-loss, AUC, Confusion matrix, Classification report can be used. Classification accuracy is the number of correct predictions made as a ratio of all predictions made. This is the most common evaluation metric for classification problems, as mentioned by Brownlee (2019a).

According to Brownlee (2019a), it is also the most misused evaluation metric. They state that it is really only suitable when there are an equal number of observations in each class (which is rarely the case) and that all predictions and prediction errors are equally important, which is often not the case.

2.7.1 Confusion matrix

The Confusion matrix is one of the most intuitive and easiest metrics used for finding the correctness and accuracy of the model. It is used for classification problem where the output can be of two or more types of classes, as described by Sunasra (2017).

Yona (2017) states the Confusion matrix has one undeniable mathematical fact. There are relationships between the cell counts since they must sum to the total number of observations. Therefore, the different kinds of Fairness are also related and there are many trade-offs.

2.8 ROC-curve And The Threshold

The Confusion matrix is useful for measuring, for example the accuracy and the ROC curve. AUC stands for the Area under the ROC curve and indicates how well the probabilities from the positive classes are separated from the negative classes, as explained by Agarwal (2019). AUC is both a scale and-invariant as a classification-threshold-invariant. As stated by Agarwal (2019), it measures the quality of the model's predictions irrespective of what classification threshold is chosen, unlike F1 score or accuracy which depend on the choice of threshold.

Brownlee (2020) describes that the default threshold for the ML classification Algorithms is 0.5. He explains that the default threshold can result in poor performance. As such, a simple and straightforward approach to improving the performance of a classifier that predicts probabilities on an imbalanced classification problem is to tune the threshold used to map probabilities to class labels. In some cases, such as when using ROC-curves and Precision-Recall curves, the best or optimal threshold for the classifier can be calculated directly.

In other cases, it is possible to use a Grid Search to tune the threshold and locate the optimal value, as explained by Brownlee (2018c) and Larrabee (2020).

Chapter 3

The Datasets

3.1 Crime Dataset - Predicting Recidivism

3.1.1 General

The dataset is collected by Angwin et al. (2016) from the Broward County Clerk's Office in Florida, United States. It contains data of $11,757$ criminal defendants of the Broward County Jail that entered and left jail between January 2013 and March 2016. Its average population was about $3,500$, as stated by Avg (2016). According to Blomberg et al. (2010), monthly, there were about respectively $4,000$ male and $1,000$ female jail bookings. The initial purpose of the dataset was to obtain the COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) scores which is a popular commercial Algorithm used by judges and parole officers for scoring criminal defendant's likelihood of recidivism. It has been shown that the Algorithm is biased in favor of white defendants, and against black inmates, based on a two year follow up study (i.e who actually committed crimes after two years), as noted by Ofer (2017).

3.1.2 Data Preparation And Data Analysis

This study considers the criminal defendant's characteristics and whether they committed crimes after two years, COMPAS scores are omitted. Hence, the resulting features are: Gender, Age, Age Category, Race, Legal Status, Marital Status, Language, number of crimes while juvenile, overall prior number of crimes and the number of days in jail for their recent crime.

About 10% of the criminal defendants have an unknown jail date. Based on their age and screening data, their jail date is approximated. The date out of jail is approximated by analyzing the crime they commit and the amount of days previous criminal defendants had to be in jail for it. Overall, the number of days in jail varied from one day up to about $2,000$ days for those whose expected date out of jail is after 2016. Table 3.1 shows the features that are used for the Algorithm Fairness Metric calculations.

	Gender	**Age Category**	**Race**	**Marital Status**	**Legal Status**
Features	Male Female	< 25, $25 - 45$, > 45	African-American, Asian, Caucasian, Hispanic, Native American, Other	Divorced, Married, Separated, Significant Other, Single, Widowed, Unknown	Pretrial, Post Sentence, Unknown, Other

Table 3.1: Feature overview.

The criminal defenders of the Crime Dataset have the following characteristics:

- 79.41% ($9,336$) is Male and 20.59% ($2,421$) is Female, see Figure 3.1.
- Ages vary between 18 years and 96 years. More than half of the defendants is between the Age of 25 and 45, see the pie chart in Figure 3.2.

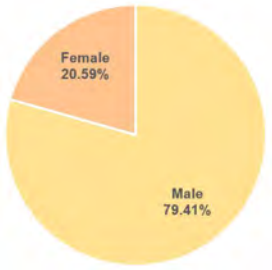

Figure 3.1: Gender distribution.

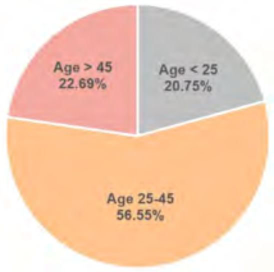

Figure 3.2: Age Category distribution.

- Almost half of the criminal defendants is African-American, followed by more than one third Caucasian, see Table 3.2 for the descending proportion of each Race.
- The majority of the criminal defendants is single (74.69%). See Table 3.3 for the descending proportion of each Marital Status type.
- In 95% of the cases, the criminal defendants are in pretrial during their jail stay. People who cannot afford to post bail — in particular, people from poor communities — remain in jail, often until their cases are resolved, while those who have access to financial resources are able to secure their liberty, as explained by Lokhart (2019). Only a couple of criminal defendants are conditionally released after some days to await their trial. The remainder violated their probation, are in post sentence, deferred sentencing or had an unknown status.

Race	Proportion
African American	49.44%
Caucasian	35.75%
Hispanic	9.36%
Other	5.62%
Asian	0.49%
Native American	0.34%

Table 3.2: Race distribution.

Marital Status	Proportion
Single	74.69%
Married	11.81%
Divorced	4.50%
Significant Other	3.18%
Unknown	2.97%
Separated	2.38%
Widowed	0.47%

Table 3.3: Marital Status distribution.

In total, 68.50% (8, 054) of the criminal defendants turned out to be a recidivist.

- About 85% is male.
- About 60% is in of the age between 25 and 45 years old, followed by about 26% of the age < 25.
- About 60% is of the race African American, followed by about 30% Caucasians, 7% Hispanic and 4% other and less than 1% of Asians and Native Americans.
- About 80% is single, followed by 7% being married, 3% having a significant other or an unknown status, the remainder consists of criminal defendants being either separated or widowed.
- About 95% is in their pretrial, followed by about 2% being in post sentence or being unknown, the remainder is other or probation violator.

3.2 Income Dataset

3.2.1 General

The dataset considers data from the Unites States Census Bureau of 1994, derived from Lichman (2013) where Barry Becker extracted the 1994 Census (2019) Database. Census (2019) considers themselves as the nation's leading provider of quality data about its people and economy. The dataset contains information of 32, 561 United States citizens.

3.2.2 Data Preparation And Data Analysis

In this study, features to predict whether someone makes over 50K a year are: Work Class, Education, Marital Status, Relationship, Race, Gender, Capital Gain / Loss, Working hours per week, Native Country. The features for the Algorithm Fairness Metric calculations have been chosen such that they are comparable to the Crime Dataset.

For the Education features, initially 16 could be distinguished, see Figure 3.3. Finally, four are distinguished, based on the Dutch school system as described by Nuffic (2013). The Dutch “Basisschool”, noted as “Basic School” includes Pre-school, Kindergarten and Elementary school (grade 1 until 6). The Dutch “Middelbare school”, notes as “Middle School”, includes Junior High and Senior High (grade 9 until 12). The university group includes both the Bachelors and Master, but also those who followed “some college”. The remainder belong to the Above University group.

The Weekly working hours were initially a continuous variable which is made categorical by distinguishing those who work less than 40 hours per week, as Part-time, those who work exactly 40 hours, as Full-time, and those who work more than that as “Above full-time”.

Table 3.4 shows the features that are used for the Algorithm Fairness Metric calculations.

	Gender	**Working hours**	**Race**	**Marital Status**	**Education**
Features	Male Female	Part-time, Full-time, Above full-time	African-American, Asian, Caucasian, Hispanic, Native American, Other	Divorced, Married, Separated, Significant other, Single, Widowed, unknown	Basic school, Middle school, University, Above university

Table 3.4: Feature overview.

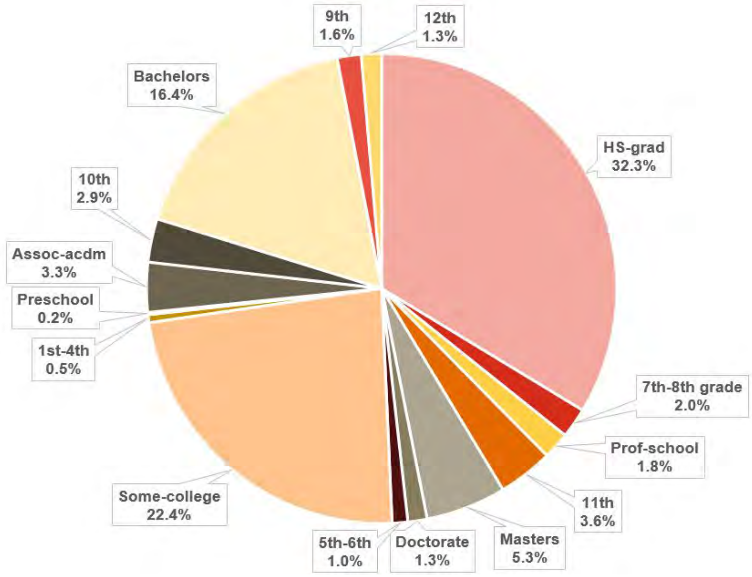

Figure 3.3: Initial Education distribution.

The people of the Income Dataset have the following characteristics:

- Gender: 66.92% (21,790) is Male and 33.08% (10,771) is Female, see Figure for the pie chart 3.4.
- Weekly working hours: The number of hours working for those who work Part-time vary from one hour up to 39 hours. The majority of the individuals work Full-time. The average number of Working hours per week is 40.44.
- Race: The majority is White, followed by Black, Asian-Pac-Islander, Amer-Indian-Eskimo and Other, see Table 3.5 for the details.

- Marital Status: The majority is Married and one third is Single, see Table 3.6 for the proportion per Marital Status.
- Education: The amount of individuals who finished the Dutch Middle school or obtained an University degree is both about 44%. See Figure 3.6 for the pie chart.

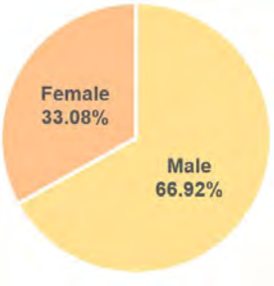

Figure 3.4:
Gender distribution.

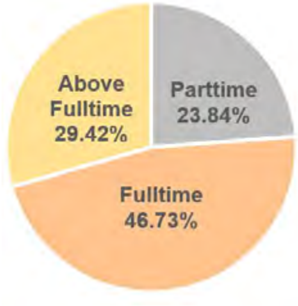

Figure 3.5: Working hours distribution.

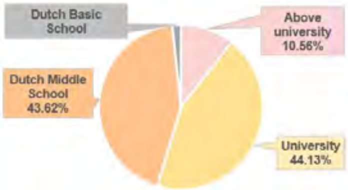

Figure 3.6:
Education distribution.

Race	Proportion
White	85.43%
Black	9.59%
Asian-Pac-Islander	3.19%
Amer-Indian-Eskimo	0.96%
Other	0.83%

Table 3.5: Race distribution.

Marital Status	Proportion
Married	45.99%
Single	32.81%
Divorced	13.65%
Separated	3.15%
Widowed	3.05%
Married, spouse Absent	1.28%
Married, spouse in Armed Forces	0.07%

Table 3.6: Marital Status distribution.

Income Larger Than $50K$

Considering the individuals that have a yearly income of above $50K$, out of the $32,561$ entries, this turned out to be the case for 24.08% ($7,841$).

- About 85% is Male.
- About 49% works above Full-time, 41% works Full-time and 9% works Part-time.
- About 91% are Caucasian, 5% is African-American and 4% is Asian. The remainder consists of the races Native American, Hispanic and Other.
- About 86% of the individuals are Married, 6% is Divorced, 6% is Single and 1% is Widowed. The remainder consist of those who are Separated. There are no individuals whose Marital Status can be classified as Significant Other.
- About 60% obtained an University degree, 24% finished their Dutch Middle School degree, while 17% obtained a degree higher than the University degree. Only 0.28% solely finished their Dutch Basic School.

Chapter 4

Method

This Chapter explains the Method of this study. The first Section describes the Machine Learning (ML) Algorithms, the threshold setup, the hyperparameters of the Basis model, the hyperparameters of the Surrounding models, the Bootstrap approach and the varying Seed choice approach for the Random Forest (RF), XGBoost and Linear Support Vector Classifier (LinearSVC). The second Section explains how the results for each Algorithm Fairness Metric are interpreted, for each dataset. Finally, the programming language and laptop specifications are mentioned.

4.1 Machine Learning Algorithms

Three ML Algorithms are applied on two datasets to predict the binary classifications of individuals who are respectively being a recidivist or having an income above 50K.

The motivation of using two datasets is to conclude whether eventual found patterns are generic or differ for each dataset, this will help to reinforce later conclusions.

The three ML Algorithms are RF, XGBoost and LinearSVC which are three common and often, in accuracy, successful ML algorithms for classification problems, as described in Section 2.3. To analyze the model accuracy and its relation with the Algorithm Fairness Metrics, Basis models and Surrounding models are obtained.

Basis models are generated by a five fold Cross-Validated (CV) hyperparameter Grid Search over hyperparameter settings for RF and LinearSVC, or by Genetic Algorithm for XGBoost. Surrounding models are built around each Basis model of the ML Algorithm.

For the train / test split, similar Seed choice ($random\_state = 0$), a Bootstrap with 10 and 100 iterations or 10 and 100 different Seed choices are used for the Basis models and Surrounding models. Before calculating the Algorithm Fairness Metrics, the models were trained and tested by using the optimal cut-off value from its ROC-curve.

Threshold

Optimal decision threshold g is determined by the optimal cut-off value of the ROC-curve where the True Positive Rate (TPR) is high and the False Positive Rate (FPR) is low. A high TPR is preferred for both datasets since, for the Crime Dataset, it is desirable for our social security to make good classifications as wrong classifications results in liberating a dangerous prisoner. For the Income Dataset, it is in the interest of the bank to make good classifications to prevent them from giving mortgages to those who cannot afford it.

The thresholds are obtained through the $roc\_curve()$ module of Scikit. The optimal cut-off is defined for the case when $TPR - (1 - FPR)$ is as close to zero as possible, as explained by Miler Jerkovic (2019).

Basis Model

For each dataset, the script finds three Basis models (one for each ML Algorithm) for the prediction of the classification after tuning the hyperparameters. This is done through RandomizedSearchCV using a five fold CV hyperparameter Grid Search for RF and LinearSVC and through Genetic Algorithm for XGBoost. The Genetic Algorithm for XGBoost does not include this five fold CV hyperparameter Grid Search since the hyperparameter tuning process is based on the script by Koehrsen (2018b).

The size of the grid is tailored for the computation power of the laptop that was used in this study. Hence, by trial and error, ranges of the hyperparameter grid are defined. The grid and hyperparameter specifics for each ML Algorithm are described in Section 4.1.1, Section 4.1.2 and Section 4.1.3.

Finally, the Basis model is trained and tested by using the optimal cut-off value from its ROC-curve. The optimal hyperparameter combinations of the Basis models form the basis of the Surrounding models.

Surrounding Models

Around each Basis model, to be able to analyze its accuracy and Algorithm Fairness Metric values for different hyperparameters, Surrounding models were defined. These Surrounding models are based on the optimal hyperparameters of the Basis model. The goal is to cover many hyperparameter combinations around this Basis model to be able to analyze and base conclusions on it.

For hyperparameter i its value is a_i and the range is $[a_i - \frac{1}{2}a_i; a_i + \frac{1}{2}a_i]$. There is no scientific reason behind this, it is solely based on the experience with smaller or larger ranges for the hyperparameters.

Smaller hyperparameter ranges resulted in Surrounding models with similar accuracy, AUC and hyperparameters, hence having similar Algorithm Fairness Metric values. This is not desirable since the goal is to analyze and compare the different Algorithm Fairness Metric values.

On the other hand, larger hyperparameter ranges resulted in, especially for the XGBoost models, Surrounding models that are not desirable in two ways. First of all, the were cases where the difference in accuracy was more than 40% or the Surrounding model classified each item with 100% zeros or 100% ones. Secondly, the AUC value was similar to that of a No Skill classifier (AUC value of 0.50). As a result, for both cases, this questioned the value and the meaning of the eventual conclusions for the Algorithm Fairness Metric results. For this reason, the range of $[a_i - \frac{1}{2}a_i; a_i + \frac{1}{2}a_i]$ was chosen.

In total, 200 Surrounding models are created based on the hyperparameter combinations within its range, by randomly selecting 200 random hyperparameter combinations with the $random.seed(0)$ setting. This amount of 200 Surrounding models is chosen in the interest to reduce the computation time while still being able to analyze the course of the accuracy and Algorithm Fairness Metrics and to compare it with the Basis model.

Step sizes were dynamically chosen to obtain 200 Surrounding models. Larger step size result in less models, while smaller step sizes result in more models while still being within the $[a_i - \frac{1}{2}a_i; a_i + \frac{1}{2}a_i]$ range. Very small step size would, especially for the seven hyperparameters for XGBoost, result in tens of thousands different hyperparameter combinations. On the other hand, for RF, since their tuned hyperparameters have an integer value, there may be less than 200 models.

Finally, each Surrounding model is trained and tested by using the optimal cut-off value from its ROC curve.

Bootstrap

To analyze the impact of Bootstrapping for a train / test dataset split of 80% / 20% for the Basis models, using $random\_state = 0$, Bootstraps with 10 iterations as well as a Bootstrap with 100 iterations are performed. The grid and hyperparameter specifics for each ML Algorithm are described in Section 4.1.1, Section 4.1.2 and Section 4.1.3. This resulted in respectively 10 and 100 Basis models for each ML Algorithm.

Note that, to analyze the impact of the Bootstrap process on the Algorithm Fairness Metrics and to reduce the computation time, Algorithm Fairness Metrics are only calculated for the Basis models from the Bootstrap with 10 iterations. Hence, not for the corresponding Surrounding models or for the the Bootstrap with 100 iterations and its Surrounding models. However, the accuracy values and hyperparameter characteristics are known and may contain some useful information.

Seed Choice

For the train / test dataset split of 80% / 20%, the Basis models use a Seed choice of $random\_state = 0$. To analyze the impact of the Seed choice on the Algorithm Fairness Metrics, the already defined Basis models and Surrounding models were fitted for varying train sets by changing the random Seed choice for the train / test dataset split of 80% / 20%.

In total, results are obtained for respectively 10 and 100 varying Seed choices where the $random\_state$ value is chosen by randomly sampling respectively these 10 and 100 values with the $random.seed(0)$ setting.

Note that, to analyze the impact of the Seed choice on the Algorithm Fairness Metrics and to reduce the computation time, Algorithm Fairness Metrics are only calculated for the Basis models and Surrounding models for the 10 varying Seed choices. Hence, not for the 100 varying Seed choices. However, the accuracy and AUC values are known and may contain some useful information.

4.1.1 Random Forest

As described by the Random Forest Classifier of Pedregosa et al. (2011), 19 hyperparameters can be tuned to increase the performance of the model. As described by Meinert (2019), RF is from itself already very good at classification and hence it may not be necessary to do an exhaustive Grid Search. Also, by minimizing the number of hyperparameters for the hyperparameter tuning and using step sizes, the computation time is significantly reduced.

To find the RF Basis model for each dataset, hyperparameter tuning involved the number of trees ($n\_estimators$) and the maximum depth of the tree (max_depth). These two hyperparameters had the biggest impact on the datasets, removing the other hyperparameters from the grid did not seem to decrease the accuracy as much as when $n\_estimators$ or max_depth would be removed from the hyperparameter tuning process. To note, the remainder hyperparameters are set at the default value.

For the RF Basis model, to reduce the computation time, the grids of $n\_estimators$ and max_depth are set from 1 until 100 with step size 10. This process was done based on the scripts from Koehrsen (2018b). Using Scikit-Learn's RandomizedSearchCV() method, Koehrsen (2018b) describes that hyperparameter ranges can be defined to randomly sample from the grid, performing a five fold CV for each value combination.

As described in Section 4.1, the optimal number of $n\_estimators$ is a_1 and the optimal max_depth is a_2 where their range for the Surrounding RF models is $A_i = [a_i - \frac{1}{2}a_i; a_i + \frac{1}{2}a_i]$ for $i = 1, 2$. The 200 hyperparameter combinations for the Surrounding RF models are obtained by randomly choosing, with the $random.seed(0)$ setting, (b_1, b_2) for b_1 in A_1 for b_2 in A_2. Since the tuned hyperparameters have an integer value, there may be less than 200 hyperparameter combinations.

Each hyperparameter combination forms the input of a Surrounding RF model while the remainder hyperparameters are set at the default value. Section 8.1.1 in the Appendix shows the model details of the RF Basis model.

4.1.2 XGBoost

As described by xgboost developers (2020), the XGBoost Classifier involves 23 hyperparameters. The number of hyperparameters for the hyperparameter tuning is minimized to seven but still may results in a significant amount of hyperparameter combinations for different step sizes.

The computation time of RandomizedSearchRV() is not feasible, hence hyperparameter tuning was performed by using Genetic Algorithm. The Genetic Algorithm process was performed based on the scripts from Jain (2018) which does not include the five fold CV hyperparameter Grid Search.

Based on the script, the number of items within the hyperparameter range are based on the number of parents which is set at 8. The *learning_rate*, *min_child_weight*, *gamma*, *subsample* and *colsample_bytree* are calculated by taking the random value of an uniform distribution. The remainder two hyperparameters are calculated by pseudorandom number generators for various distributions. In line with the script of Jain (2018), the number of parents mating and the number of generations are set at 4. The Basis XGBoost model hence contains hyperparameter values $a_1, ..., a_7$. See Section 8.1.2 in the Appendix for the model details.

The ranges of the seven hyperparameters of the Basis XGBoost model are $A_i = [a_i - \frac{1}{2}a_i; a_i + \frac{1}{2}a_i]$ for $i = 1, ..., 7$. The 200 combinations for the Surrounding XGBoost models are obtained by randomly choosing, with the $random.seed(0)$ setting, 200 possible combinations of these seven hyperparameters within their set range. Each hyperparameter combination forms the input of a Surrounding XGBoost model.

4.1.3 LinearSVC

To obtain the Basis LinearSVC model, hyperparameter tuning was applied by using RandomizedSearchCV(). Different values for C value and tolerance value are considered since these hyperparameters had the biggest impact on the LinearSVC performance.

The C value requires restrictions to reduce the computation time, especially for the larger Income dataset. Hence, the maximum C value is 10.

The tolerance value varies from $1e - 5$ up to 100. The ranges of the two hyperparameters of the Basis LinearSVC model are $A_i = [a_i - \frac{1}{2}a_i; a_i + \frac{1}{2}a_i]$ for $i = 1, 2$.

The 200 combinations are obtained by randomly choosing, with the $random.seed(0)$ setting, (b_1, b_2) for b_1 in A_1 for b_2 in A_2. Each hyperparameter combination forms the input of a Surrounding LinearSVC model.

4.2 Interpreting The Results

This Section describes how the Algorithm Fairness Metrics results are interpreted. Their calculations are done based on Verma and Rubin (2018), some Algorithm Fairness Metrics use feature combination differences, while others take the entire dataset into account.

The 32 Algorithm Fairness Metrics can be distinguished in six types and have five different result types. For each dataset, every Algorithm Fairness Metrics result type has its own description, they are explained in the subsections. Overall, the five different Algorithm Fairness Metrics result types are distinguished as followed:

1. The results of the Algorithm Fairness Metric $1 - 12$ can be shown by one or multiple graphs in which the behavior of each metric for the Basis Models and the Surrounding Models for each Algorithm can be distinguished.
2. For Algorithm Fairness Metric $13, 15 - 21, 26$ and 27, the goal is to compare two features in terms of their Unfairness. In total, 46 combinations can be made. To analyze the impact of the amount of data for the Algorithm Fairness Metrics, three groups are created based on the proportion of data of each feature. The first group

considers feature combinations where the amount of data of each item is similar. The second group considers feature combinations of which one item covers almost twice as much, compared to the other. The third group considers feature combination of which I think they may be interesting to analyse such as different genders for the Crime Dataset.

3. Algorithm Fairness Metric 14 is a special Algorithm Fairness Metric since it involves the common feature combinations within the dataset. The unique feature combinations of the dataset are distinguished and, by counting its appearances, they are ranked in decreasing order. Only the top ranked combinations are considered for this Algorithm Fairness Metric.
4. For Algorithm Fairness Metric 28, one table is returned where the changing feature, its k-value, corresponding average D-value and the percentage of violating cases are displayed for each of the ML Algorithms.
5. For Algorithm Fairness Metric 29-32, a Causal graph forms the source. Based on this Causal graphs, the Fairness definitions are either met or not.

4.2.1 Groups Crime Dataset

This subsection describes the result types 2-5 for the Crime Dataset, type 1 is excluded as its result interpretation does not depend on feature combinations for this study.

Result Type 2

Algorithm Fairness Metric $13, 15 - 21, 26$ and 27 compare Gender, Age Category, Race, Marital Status and Legal Status with each other. In total, 46 feature combinations can be made. See Section 8.2.1 in the Appendix for the overview.

To make the results comparable, three groups have been made based on the proportion of the dataset. The following percentages are round to integers. Each group, contains multiple combinations, indicated by Group m.n where m is the group number and n represents the combination number within that group. The first group consists of the following three feature combinations, in descending order of contribution:

- Group 1.1: Race African-American and Caucasian, respectively 49% and 34%;
- Group 1.2: The age group < 25 and > 45, respectively 20% vs. 22%;
- Group 1.3: Divorced vs. Significant other, respectively 5% vs. 4%.

The second group considers the following three combinations, in descending order of contribution they are:

- Group 2.1: group $25 - 45$ vs. < 25, respectively 58% and 20%;
- Group 2.2: Age group $25 - 45$ vs. > 45, respectively 58% and 22%;
- Group 2.3: Married vs. Divorced, respectively 11% vs. 5%.

The third group considers the following four combinations, in descending order of contribution they are:

- Group 3.1: Gender Male vs. Female, respectively 80% vs. 20%;
- Group 3.2: Single vs. Married, respectively 75% vs. 11%;
- Group 3.3: Race Caucasian vs. Hispanic, respectively 34% vs. 10%;
- Group 3.4: Race Caucasian vs. Other, respectively 34% vs. 46%.

For the same data proportion of the two features, such as for Group 1.2 and 1.3, a relationship may be observed. If not, this may be caused by the different feature importances of each model. When the results within a group differ significantly, the proportion of data and the Algorithm Fairness Metrics may hence not have a clear relation. In this case, in the attempt to still find other promising patterns, only the first combination of each group are used for the result interpretation. This is sufficient since the first combinations within each group also represent the largest coverage in terms of data, since the items are grouped in descending order of contribution.

Result Type 3

For Algorithm Fairness Metric 14, the combination of Gender, Age Category, Race, Marital Status and Legal Status matter. Four features are used to make a comparison with the remainder features. The Unfairness value is calculated for all combinations, ranked in descending order of contribution, Section 8.2.2 in the Appendix shows part of the table.

For those who have the same proportion of combination percentage (column 4), only the top cases in terms of proportion are selected (column 4 and 5). It would not be fair to compare groups who have extremely different proportions, for example 18.2% versus 0.1% (first row). Therefore, four combinations are considered:

- Single Males in the Age Category $25 - 45$ in pretrial (cover 33%), comparing Race African-American with Caucasian (respectively 18% and 10% out of the 33%).
- Single African-American Males in pretrial (cover 31%), comparing the Age Category < 25 and $25 - 45$ (respectively 9% and 18% out of the 31%).
- Single African-Americans in the Age Category $25 - 45$ in pretrial (cover 23%), comparing Males and Females (respectively 18% and 5% out of the 23%).
- Single Caucasian Males in pretrial (cover 19%), comparing the Age Category $25 - 45$ and > 45 (respectively 10% and 5% out of the 19%).

Result Type 4

Verma and Rubin (2018) change the age variable to calculate Algorithm Fairness Metric 28. Therefore, for the Crime Dataset, the Age Category is changed. Hence, two additional individuals have been generated for changing Age Category.

Verma and Rubin (2018) generated individuals with increasing age, with steps of five years up to an age addition of 25 years. For the Crime Dataset, this means only addressing the age categories $25 - 45$ and > 45, where there is only a way up. The value k is defined as the normalized difference between the means of each Age Category group, hence before the ages were categorical.

Result Type 5

The causal graphs contain all the attributes that were used for the Basis models of the three ML Algorithms. Each ML Algorithm hence has its own Causal graph. Next to Gender, Age Category, Race, Marital Status and Legal Status, other attributed (and hence nodes) in the graphs are: normalized number of days in jail, normalized year of birth, normalized age in jail, normalized age when leaving jail and whether there are multiple or no previous (juvenile) counts.

Verma and Rubin (2018) used solid and dashed lines to distinguish the relationship being direct or indirect, however this study only uses solid lines since this also enables you to distinguish this.

Note that no causal graphs have been created for the Surrounding models since, in terms of dependency among the features and the predicted values, they are expected to retrieve the same results.

4.2.2 Groups Income Dataset

Result Type 2

Algorithm Fairness Metric 13, 15−21, 26 and 27 compare Gender, weekly working hours, Race, Marital Status and Education with each other. See Section 8.3.2 in the Appendix for the overview. As in Section 4.2.1, three groups have been made based on the proportion of the dataset. The following percentages are rounded to integers . The first group consists of the following three feature combinations, in descending order of contribution they are:

- Group 1.1: Finishing an University degree vs. Dutch Middle School, respectively 44% and 43%;
- Group 1.2: Working Part-time vs. working above Full-time, respectively 24% and 30%;
- Group 1.3: Being separated vs. Widowed, respectively 3% and 3%.

The second group consists of the following three feature combinations, in descending order of contribution they are:

- Group 2.1: Male vs. Female, respectively 67% and 33%;
- Group 2.2: Working Full-time vs. working Part-time, respectively 47% and 24%;
- Group 2.3: Being Divorced vs. Single, respectively 14% and 32%.

The third group consists of the following four feature combinations, in descending order of contribution they are:

- Group 3.1: Being African-American vs. Caucasian, respectively 9% and 86%;
- Group 3.2: Being Divorced vs. Married, respectively 14% and 48%;
- Group 3.3: Finishing higher than University degree vs. Dutch Middle School, respectively 12% and 43%;
- Group 3.4: Being Married vs. Single, respectively 48% and 32%;

When the results within a group differ significantly, the first combination of each group is used to interpret results, as was explained in Section 4.2.1.

Result Type 3

For Algorithm Fairness Metric 14, the combination of Gender, Working hours, Race, Marital Status and Education matter. As in Section 4.2.1, four characteristics are used to make a comparison with the remaining characteristics. The Algorithm Fairness Metric was calculated for all combinations where the four characteristics covered at least 5% of the data, this led to 99 combinations. See Table 8.6 in the Appendix for the overview.

The most common combination, excluding education, were Married Caucasian Males who are working Full-time which is 17% of the data. Of this, 2% finished a degree above University, 7% has an University degree, 8% has a Dutch Middle School degree and 0.4% only finished Dutch Basic School. Therefore, we only compare those who have an University degree and those who finished Dutch Middle School, since they represent respectively 7% and 8% of the data. Applying this approach to the remainder 99 combinations, this results in the following:

- Married Caucasian Males who are working Full-time (cover 17%), comparing those with an University degree with those who finished Dutch Middle School (7% and 8% out of the 17%).

- Married Caucasian Males with an University degree (cover 16%), comparing those who work are working Full-time with those who are working more than Full-time (7% and 8% out of the 16%).
- Married Caucasian Males who are working more than Full-time (cover 16%), comparing those with an University degree with those who finished Dutch Middle School (8% and 5% out of the 16%).
- Married Caucasian Males who finished Dutch Middle School (cover 15%), comparing those who are working Full-time with those who are working more than Full-time (8% and 5% out of the 15%).

Result Type 4

For the Algorithm Fairness Metric 28, it is impossible to adjust a feature in the way that was done for the Crime Dataset. Even though the Education attribute is ranked, there is no way to define the value of k as being the mean difference between each category group. This requires numerical data, which is unavailable for the Income Dataset.

The Education attribute can be partially converted to numerical data when taking the number of school years as the value. However, those who have a degree which is higher than the University degree, may have very different values.

Result Type 5

The causal graphs will contain all the attributes that were used for the Basis models of the three ML Algorithms. Each ML Algorithm hence has its own Causal graph. Next to Gender, working hours, Race, Marital Status and Education, other attributed (and hence notes) in the graphs are for example their occupation, working type, work branch and the capital information. The graphs are translated in to Tables where a value of one means the null hypothesis is rejected.

4.2.3 Programming Language And Laptop Specifications

Calculations of this study are done through the Python programming language. As stated by Rathi (2019), reasons to choose for this language are its in-built library, it being easy to integrate and to create prototypes, its high productivity and the object-oriented paradigm. The laptop is a Lenovo Thinkpad with an operating system type of 64-bit, x64-based processor and an installed RAM of 16.0GB. The processor has the following specifications: INTEL(R) Core(TM $i7 - 7600$U) CPU @ 2.80GHz 2.90GHz.

Chapter 5

Results

The first Section of this Chapter describes the performance of the Basis models, Surrounding models, Bootstrap approach and the varying Seed choice approach on the train / test dataset split for both datasets and each ML Algorithm. The second Section describes the Algorithm Fairness Metric results for all six types, each subsection covers an Algorithm Fairness Metric where the outcomes for both datasets are mentioned.

The Figures in this Chapter represent the values of each Basis model (dashed horizontal lines) as well as the Ranked Performance of the Surrounding models (solid lines). The values in the Figures are the absolute values, the titles indicate which feature is treated unfair. For example, the title "A vs. B (x)" indicates that the Unfairness value of B is larger than that of A. This results in a negative Algorithm Fairness Metric value which is made absolute for the purpose of the Figures and is presented on the y-axis.

5.1 Machine Learning Algorithms

This Section describes the performance of the Basis models, Surrounding models, Bootstrap approach and the varying Seed choice approach on the train / test dataset split for both datasets and each ML Algorithm. The accuracy, hyperparameters and / or AUC values are compared for the different models and displayed in Figures and Tables.

5.1.1 Crime Dataset

Basis Models

The hyperparameter grid for the Random Forest (RF) Classifier performs calculations for varying number of trees and maximum tree depth. The RandomizedSearchCV() results in a Basis RF model that contains 60 trees and has a maximum tree depth of 14. Its accuracy is 61.57%, based on a threshold value of 0.32. Figure 5.1 shows the ROC-curve and the AUC value of 0.66, compared to a No Skill classifier with an AUC value of 0.50.

Hyperparameter tuning for XGBoost is done through the Genetic Algorithm. The hyperparameters of the Basis XGBoost model are as followed: maximum tree depth of 3, *min_child_weight* is 5, *gamma* value of 4.74, *subsample* value of 0.96, learning rate of 0.82, *colsample_bytree* is 1 and 4 trees. Its accuracy is 61.61%, based on a threshold value of 0.31. Figure 5.2 shows the ROC-curve and the AUC value of 0.66, compared to a No Skill classifier with an AUC value of 0.50.

The hyperparameter grid for the Linear Support Vector Classifier (LinearSVC) considers C values up to 10 and a tolerance value for stopping criteria between $1e-5$ and 100. The RandomizedSearchCV() results in a Basis LinearSVC model that has a C value and a tolerance value of 1. Its accuracy is 61.95%, based on a threshold value of 0.31. Figure 5.3 shows the ROC-curve and the AUC value of 0.67, compared to a No Skill classifier with an AUC value of 0.50.

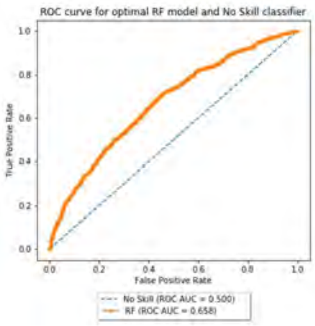

Figure 5.1:
RF ROC-curve
Crime Dataset.

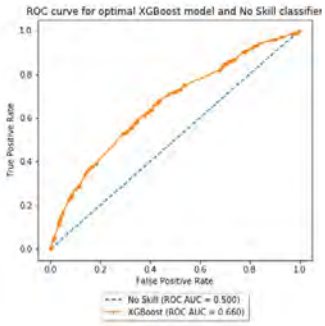

Figure 5.2:
XGBoost ROC-curve
Crime Dataset.

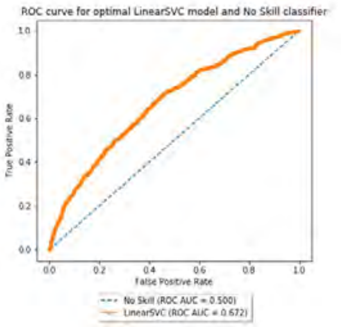

Figure 5.3:
LinearSVC ROC-curve
Crime Dataset.

Surrounding Models

For each ML Algorithm, the Surrounding models have different hyperparameters:

1. Surrounding RF models have between 30 and 90 trees and a maximum tree depth between 7 and 21.
2. Surrounding XGBoost models have the following hyperparameter values: maximum tree depth between 1.5 and 4.5, *min_child_weight* between 2.5 until 7.5, *gamma* value between 2.37 and 7.11, *subsample* value of between 0.48 and 1.44, learning rate between 0.41 and 1.23, *colsample_bytree* between 0.5 and 1.5 and the number of trees between 2 and 6.
3. Surrounding LinearSVC models have C and tolerance values between 0.5 and 1.5.

Table 5.1 shows the accuracy statistics for the Surrounding models for each ML Algorithm, round to three decimals. Figure 5.4 shows the Ranked Performance of the accuracy for all the Surrounding models, it shows the impact of the hyperparameters for each ML Algorithm and emphasizes the wide accuracy range of the Surrounding XGBoost models.

	RF	XGBoost	LinearSVC
min.	0.582	0.430	0.612
max.	0.632	0.699	0.636
avg.	0.612	0.585	0.624
sd.	0.010	0.041	0.004

Table 5.1: Crime Dataset Basis model accuracy.

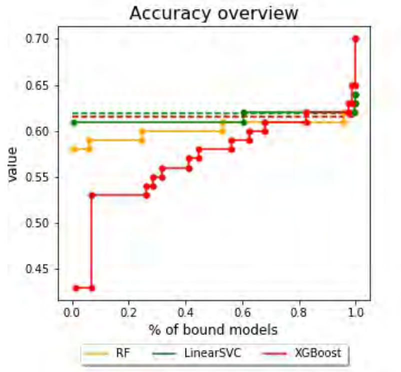

Figure 5.4: Crime Dataset Ranked Performance Surrounding model accuracy.

Bootstrap

Table 5.2-5.4 show the model accuracy and hyperparameters of the Basis models of each ML Algorithm after performing a Bootstrap with respectively 10 and 100 iterations on the train / test dataset split. The values indicate the difference in the model accuracy and hyperparameter values compared to the Basis model for each ML Algorithm that is described in Section 5.1.1.

For the RF, results between the two iterations are rather similar. Both iterations show that the Bootstrap on the train / test dataset split results in varying accuracy, *max_depth* values and *n_estimators* values. This is also illustrated by the values of the standard deviations. After 10 iterations, the Basis RF model has a maximum tree depth of 21 and contains 41 trees. After 100 iterations, the Basis RF model has a maximum tree depth of 21 and contains 60 trees.

For the XGBoost, results between the two iterations vary but still show that the Bootstrap on the train / test dataset split results in varying accuracy and hyperparameters. Using the hyperparameter order in line with the column order of Table 5.3, the Basis XGBoost model has respectively the following hyperparameters after 10 iterations: $8, 6, 5.56, 1, 0.38, 0.94, 29$ and after 100 iterations: $9, 5, 9.6, 0.5, 0.38, 0.73, 48$.

For the LinearSVC, results between the two iterations are similar. Both iterations show that the Bootstrap on the train / test dataset split results in varying accuracy and C values. For both iterations, the Basis LinearSVC model has a C value of 1 and a tolerance value of 0.01.

RF	10 iterations			100 iterations		
	accuracy	max_depth	n_estimators	accuracy	max_depth	n_estimators
min.	(6.24)%	7	(39)	(6.01)%	7	(39)
max.	(2.70)%	67	31	(1.72)%	77	31
avg.	(4.13)%	28	1	(3.76)%	34	7
sd	0.59%	8	(41)	0.60%	11	(40)

Table 5.2: Crime Dataset Bootstrap results for RF.

XGBoost	10 iterations							
	accuracy	max_depth	min_child_weight	gamma	subsample	learning_rate	colsample_bytree	n_estimators
min.	(24.40)%	(1.00)	(4.00)	(3.70)	(0.82)	(0.65)	(0.95)	25.00
max.	12.48%	6.00	4.00	4.35	0.04	0.13	(0.06)	225.00
avg.	(4.83)%	3.60	0.30	0.37	(0.31)	(0.24)	(0.65)	141.40
sd	14.75%	2.69	2.28	2.70	0.25	0.26	0.29	63.50
	100 iterations							
	accuracy	max_depth	min_child_weight	gamma	subsample	learning_rate	colsample_bytree	n_estimators
min.	(25.48)%	(2.00)	(5.00)	(4.66)	(0.93)	(0.80)	(0.97)	0.00
max.	13.13%	6.00	4.00	4.88	0.03	0.18	(0.01)	250.00
avg.	3.14%	2.16	(0.79)	(0.00)	(0.46)	(0.29)	(0.47)	110.96
sd	12.34%	2.81	2.95	2.62	0.27	0.28	0.28	68.46

Table 5.3: Crime Dataset Bootstrap results for XGBoost.

LinearSVC	10 iterations			100 iterations		
	accuracy	C	tol	accuracy	C	tol
min.	(1.89)%	0	(1)	(2.53)%	0	(1)
max.	0.08%	4	0	0.73%	6	0
avg.	(0.77)%	1	(1)	(1.11)%	1	(1)
sd	0.58%	1	0	0.59%	1	0

Table 5.4: Crime Dataset Bootstrap results for LinearSVC.

Seed Choice

For respectively 10 and 100 different Seed choices, the accuracy of the Basis model and Surrounding models are varying. For the case of 10 different Seed choices, their values are $[6311, 6890, 663, 4242, 8376, 7961, 6634, 4969, 7808, 5866]$.

For 100 different Seed choices on the train / test dataset split, the maximum difference in accuracy for respectively the Basis RF, XGBoost and LinearSVC models are 4.7%, 4.9% and 5.3%.

For 10 different Seed choices on the train / test dataset split, the maximum difference in accuracy for respectively the Basis RF, XGBoost and LinearSVC models are 3.3%, 4.0% and 2.3%, see Figure 5.5 for the Basis model accuracy for varying Seed choice.

Figure 5.6 shows the different AUC values of the Basis models for the 10 varying Seed choices on the the train / test dataset split. For different Seed choices, results for the Basis models are varying, there is no clear relation.

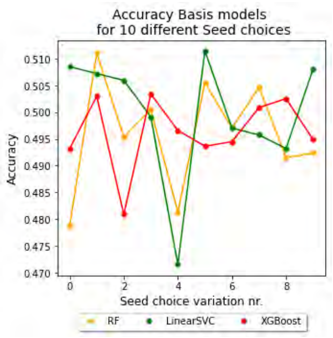

Figure 5.5: Accuracy Basis models for varying Seed choice, Crime Dataset.

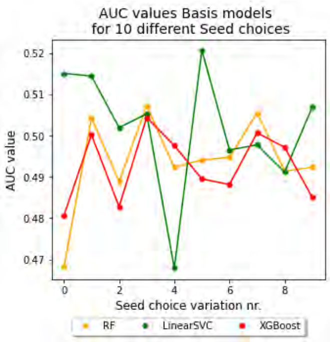

Figure 5.6: AUC values Basis models for varying Seed choice, Crime Dataset.

For the same 10 varying Seed choices on the train / test dataset split, the maximum difference in accuracy for respectively the Surrounding RF, XGBoost and LinearSVC models are 6.3%, 5.4%, 5.1%. For the AUC values, this is respectively 5.9%, 3.7% and 5.4%. For varying Seed choices, results for the Surrounding models of each ML Algorithm are varying, there is no clear relation.

5.1.2 Income Dataset

Basis Models

The hyperparameters for the Basis RF, XGBoost and LinearSVC models are as described in Section 5.1.1. From the RandomizedSearchCV(), the Basis RF model contains 60 trees and has a maximum tree depth of 14. Its accuracy is 78.97%, based on a threshold value of 0.29. Figure 5.7 shows the ROC-curve, the AUC value is 0.88.

Based on the Genetic Algorithm process, the hyperparameters of the Basis XGBoost model are as followed: maximum tree depth of 2, *min_child_weight* is 6, *gamma* value of 9.57, *subsample* value of 0.30, learning rate of 0.04, *colsample_bytree* is 0.83 and 161 trees. The Basis XGBoost model has an accuracy of 78.75%, based on a threshold of 0.28. Figure 5.8 shows the ROC-curve, the AUC value is 0.88.

The hyperparameter grid for the LinearSVC considers C values up to 10 and tolerance values for stopping criteria from $1e{-}5$ up to 100. From the RandomizedSearchCV(), the Basis LinearSVC model has a C value and a tolerance value of 1. Its accuracy is 79.27%, based on a threshold value of 0.88. Figure 5.9 shows the ROC-curve, the AUC value is 0.25.

Surrounding Models

For each ML Algorithm, the Surrounding models have different hyperparameters:

1. Surrounding RF models have between 30 and 90 number of trees and a maximum tree depth between 7 and 21.
2. Surrounding XGBoost models have the following hyperparameter values: maximum tree depth between 1.5 and 3.0, *min_child_weight* between 3 until 9, *gamma* value between 4.79 and 14.36, *subsample* value of between 0.15 and 0.45, learning rate between 0.42 and 1.24, *colsample_bytree* between 0.02 and 0.06 and the number of trees between 80 and 241.
3. Surrounding LinearSVC models have C and tolerance values between 0.5 and 1.5.

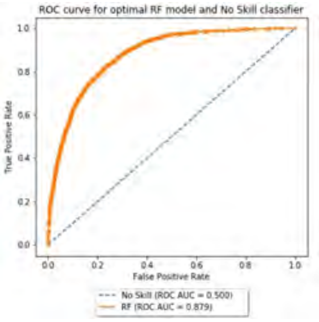

Figure 5.7:
RF ROC-curve
Income Dataset.

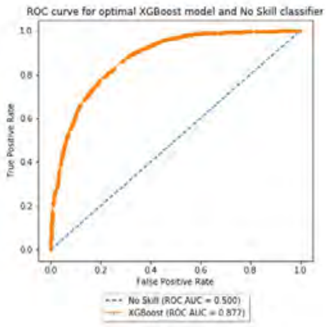

Figure 5.8:
XGBoost ROC-curve
Income Dataset.

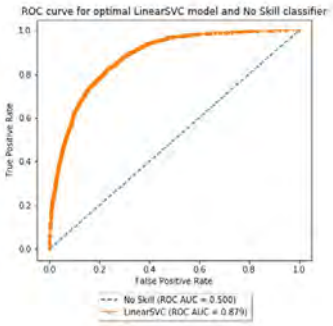

Figure 5.9:
LinearSVC ROC-curve
Income Dataset.

Table 5.10 shows the accuracy statistics for the Surrounding RF, XGBoost and LinearSVC models. Figure 5.5 shows the Ranked Performance of the accuracy for all the Surrounding models. It shows that, in terms of the accuracy, only a few models can be distinguished. There are seven different Surrounding RF models, two different XGBoost Surrounding models and two different Surrounding LinearSVC models.

	RF	XGBoost	LinearSVC
min.	0.777	0.762	0.787
max.	0.794	0.791	0.797
avg.	0.787	0.772	0.792
sd.	0.004	0.009	0.001

Table 5.5: Income Dataset Basis models accuracy.

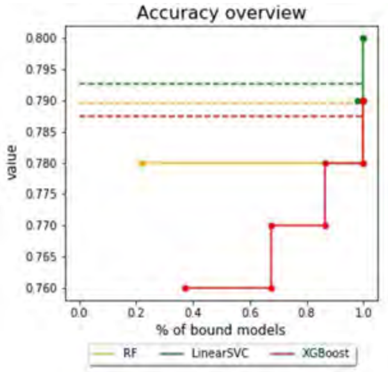

Figure 5.10: Income Dataset Ranked Performance Surrounding model accuracy.

Bootstrap

The Surrounding models of the Bootstrap iterations on the train / test dataset split are similar in accuracy for the Bootstrap with 10 iterations and the Bootstrap with 100 iterations on the train / test dataset split.

Table 5.6-5.8 show the model accuracy and hyperparameters of the Basis models of each ML Algorithm after performing a Bootstrap with 10 and 100 iterations for each Algorithm. The values show that there is a difference in model accuracy and hyperparameter values compared to the Basis model for each ML Algorithm that is described in Section 5.1.2.

For the RF, results between the two iterations are different. Both iterations show that the Bootstrap on the train / test dataset split results in varying accuracy, max_depth values and $n\_estimators$ values. This is also illustrated by the standard deviations. The Basis RF model for 10 iterations has a maximum tree depth of 90 and 91 trees. For 100 iterations, the Basis RF model has a maximum tree depth of 91 and 71 trees.

For the XGBoost, values differ for the two iterations. Using the hyperparameter order in line with the column order of Table 5.3, for 10 iterations, the Basis XGBoost model has hyperparameters: $2, 3, 5.63, 0.25, 0.92, 0.89, 254$, and for 100 iterations: $3, 2, 8.86, 0.23, 1, 0.64, 60$.

For the LinearSVC, results between the two iterations are rather similar. Both iterations show that the Bootstrap on the train / test dataset split results in varying accuracy, *C* values. For the Basis LinearSVC model, after 10 iterations, a C value of 1 and a tolerance value of 0.01 are retrieved. For 100 iterations, the Basis LinearSVC model has a C value of 1 and a tolerance value of 0.001.

RF	10 iterations			100 iterations		
	accuracy	max_depth	n_estimators	accuracy	max_depth	n_estimators
min.	(4.75)%	27	(19)	(5.20)%	27	(39)
max.	(3.60)%	77	31	(2.95)%	77	31
avg.	(4.15)%	51	11	(3.86)%	55	5
sd	0.34%	3	(45)	0.41%	16	18

Table 5.6: Income Dataset Bootstrap results for RF.

XGBoost	10 iterations							
	accuracy	max_depth	min_child_weight	gamma	subsample	learning_rate	colsample_bytree	n_estimators
min.	(6.07)%	0.00	(5.00)	(8.99)	(0.17)	0.01	(0.79)	(131.00)
max.	0.41%	6.00	1.00	0.06	0.59	0.88	0.06	93.00
avg.	(2.16)%	2.80	(1.90)	(5.43)	0.23	0.45	(0.29)	(9.60)
sd	1.98%	1.94	1.81	3.12	0.27	0.29	0.29	71.67
	100 iterations							
	accuracy	max_depth	min_child_weight	gamma	subsample	learning_rate	colsample_bytree	n_estimators
min.	(13.52)%	(1.00)	(6.00)	(9.55)	(0.27)	(0.02)	(0.82)	(157.00)
max.	1.26%	7.00	3.00	0.21	0.70	0.96	0.15	93.00
avg.	(2.52)%	2.77	(1.53)	(4.08)	0.21	0.44	(0.36)	(49.63)
sd	2.84%	2.47	2.81	2.75	0.27	0.29	0.30	72.48

Table 5.7: Income Dataset Bootstrap results for XGBoost.

LinearSVC	10 iterations			100 iterations		
	accuracy	C	tol	accuracy	C	tol
min.	0.29%	0	(1)	(0.04)%	0	(1)
max.	0.86%	2	0	1.51%	4	0
avg.	0.49%	0	(1)	0.57%	1	(1)
sd	0.17%	1	0	0.30%	1	0

Table 5.8: Income Dataset Bootstrap results for LinearSVC.

Seed Choice

For respectively 10 and 100 different Seed choices, the accuracy of the Basis model and Surrounding models are varying. For the case of 10 different Seed choices, their values are $[27670, 12623, 24836, 29171, 13781, 1326, 8484, 31636, 16753, 15922]$.

For 100 different Seed choices on the train / test dataset split, the maximum difference in accuracy for respectively the RF, XGBoost and LinearSVC Basis models are 3.7%, 4.0% and 3.2%.

For 10 different Seed choices on the train / test dataset split, the maximum difference in accuracy for respectively the RF, XGBoost and LinearSVC Basis models are 1.6%, 3.4% and 2.4%, see Figure 5.11. Figure 5.12 shows the different AUC values of the Basis models for the 10 varying Seed choices on the the train / test dataset split. Both Figures show that for different Seed choices, results for the Basis models are varying, there is no clear relation.

For the same 10 varying Seed choices on the train / test dataset split, the maximum difference in accuracy for respectively the RF, XGBoost and LinearSVC Surrounding models are 5.5%, 3.4%, 2.4%. For the AUC values, this is respectively 3.2%, 2.8% and 3.0%. For varying seed choices, results for the Surrounding models of each Algorithm are varying, there is no clear relation.

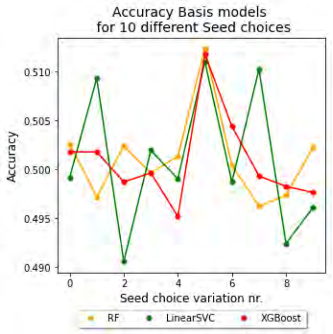

Figure 5.11: Accuracy Basis models for varying Seed choice, Income Dataset.

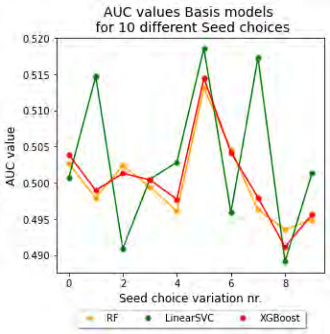

Figure 5.12: AUC values Basis models for varying Seed choice, Income Dataset.

5.2 Results Algorithm Fairness Metric Type 1: Statistical Metrics $(1-12)$

This Section describes the results for the Basis Models, Surrounding Models, the Bootstrap approach and the varying Seed choice approach on the train / test dataset split for the first 12 Algorithm Fairness Metrics. Each subsection starts by describing the results for the Crime Dataset and then shortly mentions the results for the Income Dataset.

5.2.1 Basis Models

The results of the first four Algorithms Fairness Metrics are shown in Figure 5.13. The TP, FP, FN and TN values are counts and show that the Basic LinearSVC model is never the most unfair. Overall, all ML Algorithms have, compared to the TP, FP and FN values, high TN values. Figure 5.14 shows the result of Algorithm Fairness Metric 5-12. Again, the Basis LinearSVC model always performs between the Basis RF model and the Basis XGBoost model.

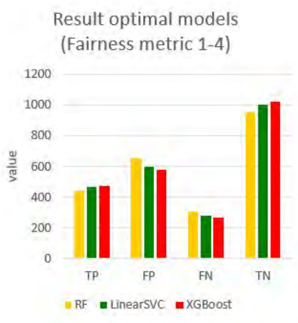

Figure 5.13: Result Basis models Crime Dataset.

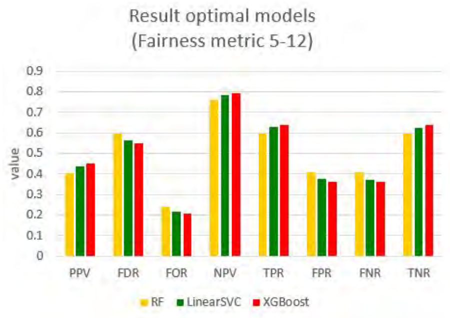

Figure 5.14: Result Basis models Crime Dataset.

The results of the first four Algorithms Fairness Metrics for the Income Dataset are different to that of the Crime Dataset since for each of the 12 Algorithm Fairness Metrics, hardly any difference can be observed. For the Income Dataset, the Basis RF, XGBoost and LinearSVC models have almost the exact same values.

5.2.2 Surrounding Models

Figure 5.15 shows the Ranked Performance of the Surrounding models of the three ML Algorithms for the TP values. The Ranked Performances of the FP, FN and TN are very similar in course, the majority of the models perform similar to the Basis models. This also automatically holds for the TPR values, as well as the FNR values.

Figure 5.16 shows that, for the Surrounding XGBoost and RF models, a significant amount performs below the value of the Basis model. The same holds for the NPV values and the TNR values. However, Figure 5.17 shows the opposite, this also holds for the FOR values and the FPR values.

Figure 5.18 shows the correlation coefficients between the Algorithm Fairness Metrics and the accuracy of the Surrounding models. The x-axis contains the values for each of the 12 Algorithm Fairness Metrics where the points indicate the correlation coefficient of, given the color, either the Surrounding RF, LinearSVC or XGBoost models.

For TN, PPV, NPV and TNR, there is a perfect positive correlation between the Algorithm Fairness Metrics and the Surrounding XGBoost model accuracy. Surrounding models with a higher accuracy, have a higher Unfairness value for these specific Algorithm Fairness Metrics. The opposite is the case for the FP, FDR, FOR, TPR and the accuracy of the Surrounding XGBoost models. The correlation coefficients of the Surrounding RF models are less varying, and closer to zero, compared to the Surrounding XGBoost and LinearSVC models.

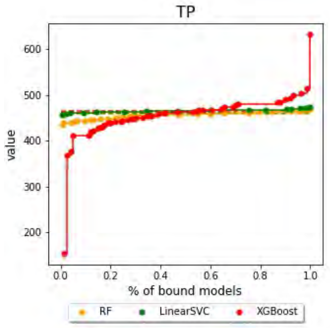

Figure 5.15: True Positive value Ranked Performance Crime Dataset.

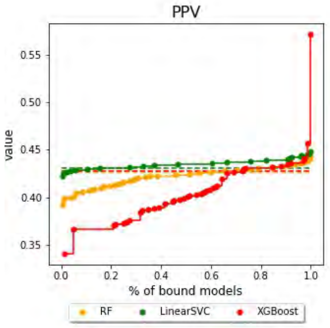

Figure 5.16: Positive Predictive value Ranked Performance Crime Dataset.

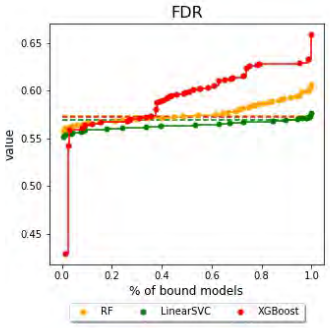

Figure 5.17: False Discovery Rate Ranked Performance Crime Dataset.

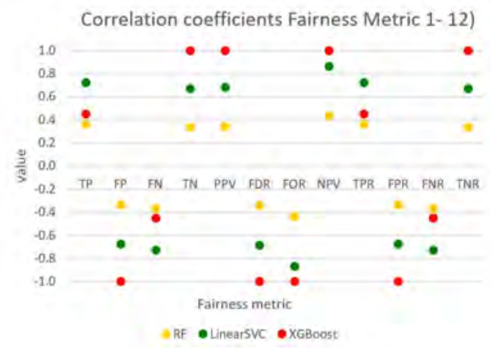

Figure 5.18: Correlation coefficients Algorithm Fairness Metrics 1-12 and Surrounding model accuracy, Crime Dataset.

The Income Dataset shows similar results for the Surrounding LinearSVC models. The Algorithm Fairness Metrics show varying results for the Surrounding RF and XGBoost models. See Figure 5.19-5.21 for the Ranked Performances of each ML Algorithm. The correlation graphs show similar results as for the Crime Dataset.

5.2.3 Bootstrap

For the Crime Dataset, the Bootstrap with 10 iterations on the train / test dataset split, retrieves 10 varying Basis ML values for Algorithm Fairness Metrics $1 - 12$. There is no

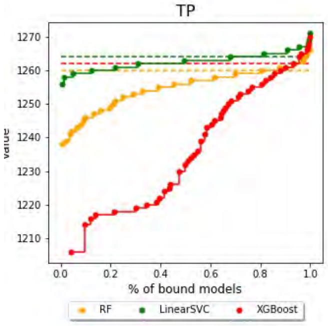

Figure 5.19: True Positive Ranked Performance Income Dataset.

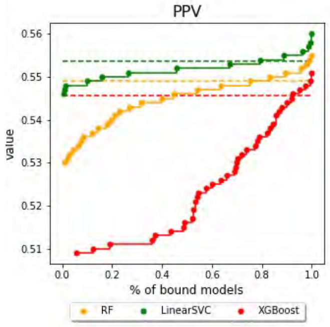

Figure 5.20: Positive Predictive value Ranked Performance Income Dataset.

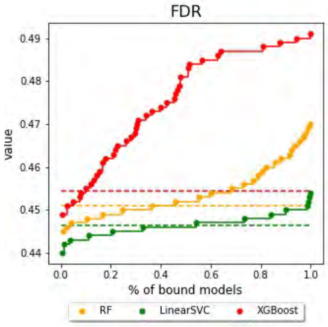

Figure 5.21: False Discovery Rate Ranked Performance Income Dataset.

clear relationship. For the Income Dataset, the same holds. There is no clear relationship as to how the Algorithm Fairness Metric vary for the Basis models from the different Bootstrap iterations.

5.2.4 Seed Choice

For the 10 different Seed choices on the train / test dataset split, varying Algorithm Fairness Metrics $1 - 12$ are retrieved for the Basis models and the Surrounding Models of each ML Algorithm. There is no clear relationship. This holds for both the Crime Dataset and the Income Dataset.

5.3 Results Fairness Metric Type 2: Def. Based On Predicted Outcome $(13, 14)$

This Section describes the results for the Basis Models, Surrounding Models, the Bootstrap approach and the varying Seed choice approach on the train / test dataset split for Algorithm Fairness Metric 13 and 14. Values for the Algorithm Fairness Metrics are defined by the difference in fairness between two features, this hence represents the value of Unfairness. Feature combination groups are as described in Section 4.2.1.

Each subsection describes an Algorithm Fairness Metric in which the results for the Crime Dataset are described, followed by a short description of the results for the Income Dataset.

13. Group Fairness

Overall, values for Algorithm Fairness Metric 13 of the Basis models vary for the three feature combination groups, as well as for other feature combinations. For each first feature combination of a feature combination group, the following Basis model has the highest Unfairness value for the Crime Dataset:

- Group 1.1: XGBoost by more than about 7% compared to RF and LinearSVC;
- Group 2.1: LinearSVC by more than about 6% compared to RF and XGBoost;
- Group 3.1: LinearSVC by more than 12% compared to RF and XGBoost.

For the Income Dataset, for feature combination Group 1.1, the Basis RF, LinearSVC and XGBoost model have a Unfairness values of respectively 31.8%, 30.0% and 31.5%. For each feature combination within the three feature combination groups and for each ML Algorithm, the Unfairness values vary for the Income Dataset.

For the Crime Dataset, Figure 5.22-5.24 show the Ranked Performance of the Algorithm Fairness Metrics for respectively feature combination Group 1.1, Group 2.1 and Group 3.1. The correlation coefficient gives the following results for the feature combination groups:

- Unfairness values of feature combination Group 1.1, 2.1 and 3.1 are positively correlated with the predictions of the Surrounding RF models.
- Unfairness values of feature combination Group 1.1 and 2.1 are negatively correlated with the predictions of the Surrounding LinearSVC models, while this is the opposite for Group 3.1.
- Unfairness values of feature combination Group 1.1 and 3.1 are positively correlated with the predictions of the Surrounding XGBoost models, while this is the opposite for Group 2.1.

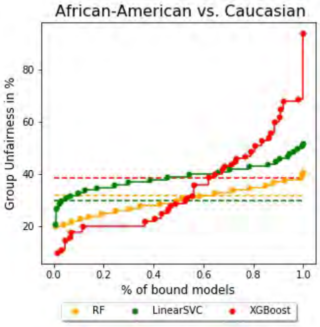

Figure 5.22: Ranked Performance Group 1.1 Crime Dataset.

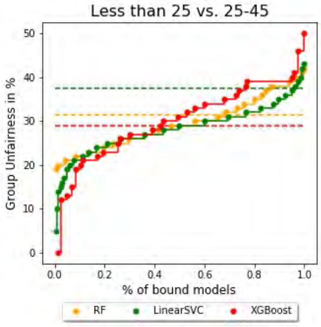

Figure 5.23: Ranked Performance Group 2.1 Crime Dataset.

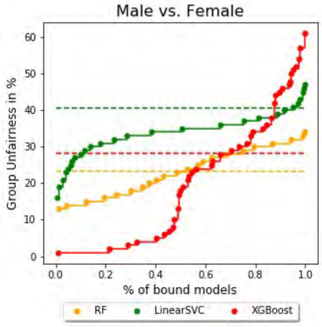

Figure 5.24: Ranked Performance Group 3.1 Crime Dataset.

For the Income Dataset, the Surrounding models vary more than for the Crime Dataset, the majority is close to the Unfairness value of the Basis model for each ML Algorithm. Figure 5.25-5.27 show the Ranked Performance of the Surrounding models for the Unfairness values for each ML Algorithm.

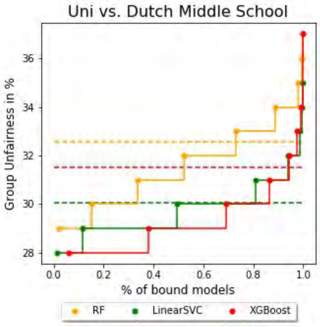

Figure 5.25: Ranked Performance Group 1.1 Income Dataset.

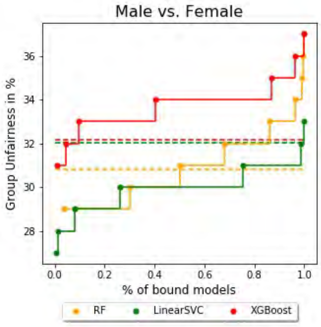

Figure 5.26: Ranked Performance Group 2.1 Income Dataset.

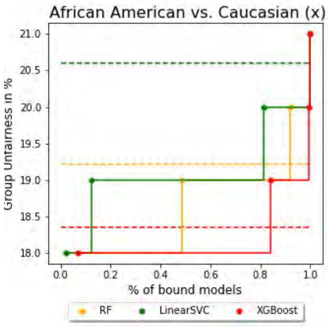

Figure 5.27: Ranked Performance Group 3.1 Income Dataset.

After the Bootstrap with 10 iterations was performed on the train / test dataset split, the following Basis ML Algorithm performs the worst for the Crime Dataset:

- Group 1.1: LinearSVC by more than about 6% and 20% compared to respectively XGBoost and RF.

- Group 2.1: LinearSVC by more than about 4% and 9% compared to respectively XGBoost and RF.

- Group 3.1: LinearSVC by more than about 5% and 20% compared to respectively XGBoost and RF.

For the Income Dataset, results are similar. However, the extent to which the worst performing Basis ML Algorithm (the LinearSVC) performs, compared to the other ML Algorithms, is smaller.

For the Crime Dataset, the Bootstrap with 10 iterations on the train / test dataset split, retrieves 10 varying Basis ML values for Algorithm Fairness Metrics $1 - 12$. There is no clear relationship.

For the Income Dataset, the same holds. There is no clear relationship as to how the Algorithm Fairness Metric vary for the different iterations of the Bootstrap.

For varying Seed Choice on the train / test dataset split, the values for Algorithm Fairness metric 13 are varying for each ML Algorithm and its Basis models and Surrounding models. For both the Crime Dataset and the Income Dataset, there is no clear relation between the varying Seed choice and its varying Algorithm Fairness Metric.

14. Conditional Statistical Parity

For the Crime Dataset, Figure 5.28 shows significant differences for the four combinations for the Basis models of each ML Algorithm. The Basis LinearSVC model retrieves the largest Unfairness value.

Unfairness values for Basis models of each ML Algorithm vary even more for the Income Dataset. There are differences between the ML Algorithms of almost 40%, see Figure 5.28. For Combi 1 and Combi 4 cases, the Basis RF model retrieves remarkable higher Unfairness values compared to the other two ML Algorithms.

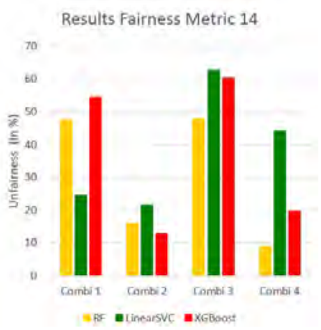

Figure 5.28: Result Basis models Crime Dataset.

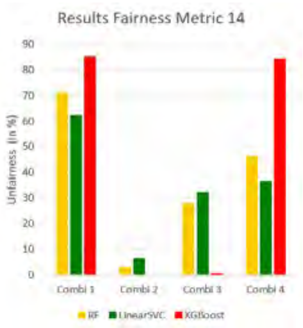

Figure 5.29: Result Basis models Income Dataset.

For both the Crime Dataset and the Income Dataset, the Unfairness values vary among the Surrounding models for each of the four selected combination and each ML Algorithms. There is no clear relationship.

For the Bootstrap with 10 iterations and the 10 varying Seed choice on the train / test dataset split, the varying Algorithm Fairness Metrics do not show a clear relation. This holds for Combi 1, Combi 2, Combi 3 and Combi 4 of the Crime Dataset and the Income Dataset.

5.4 Algorithm Fairness Metrics Results Type 3: Def. Based On Predicted And Actual Outcomes (15 – 21)

This Section describes the results for the Basis Models, Surrounding Models, the Bootstrap approach and the varying Seed choice approach on the train / test dataset split for Algorithm Fairness Metric 15 – 21. Unfairness values are the difference in fairness between two features. Feature combination groups are as described in Section 4.2.1.

For the Crime Dataset, Algorithm Fairness Metric 15 – 21 all show different results, without a clear relation, for the Bootstrap approach and varying Seed choices approach on the train / test dataset split. For the Bootstrap approach, this holds for the 10 Basis models of each ML Algorithm. For the 10 varying Seed choices, this holds for the corresponding Basis models and the Surrounding models of each ML Algorithm. The same is observed for the Income Dataset. However, the extent to which the Algorithm Fairness Metric values differ is smaller compared to the Crime Dataset.

15. Predictive Parity

Figure 5.30 and Figure 5.33 show different results for each group and each of the Basis models. The group differences for the Crime Dataset are larger than those of the Income Dataset. Both dataset indicate positive correlation for the Algorithm Fairness Metric value and the accuracy of the Surrounding ML models, see Figure 5.31 and Figure 5.34.

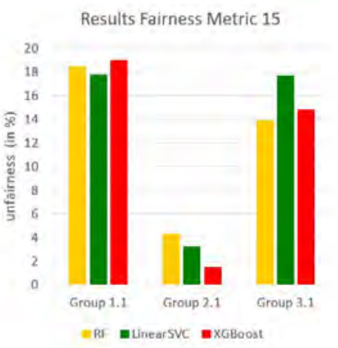

Figure 5.30: Result Basis models Crime Dataset.

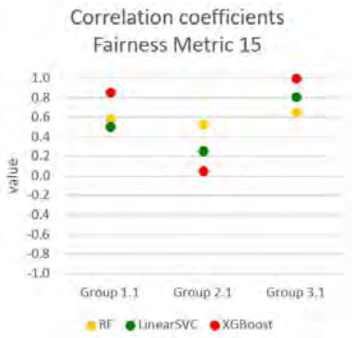

Figure 5.31: Correlation coefficients for Surrounding model accuracy Crime Dataset.

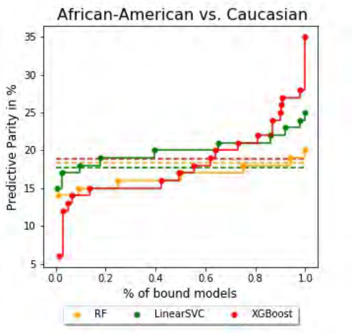

Figure 5.32: Ranked Performance Group 1.1 Crime Dataset.

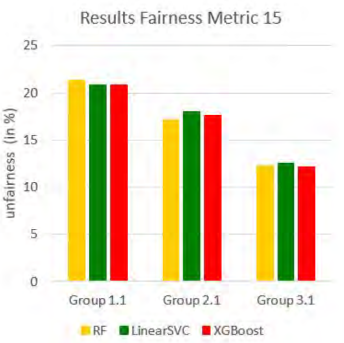

Figure 5.33: Result Basis models Income Dataset.

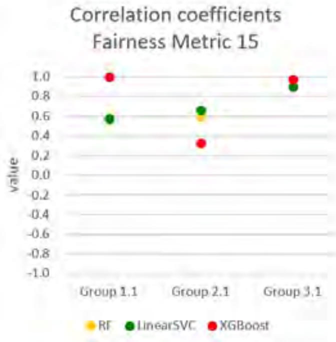

Figure 5.34: Correlation coefficients for Surrounding model accuracy Income Dataset.

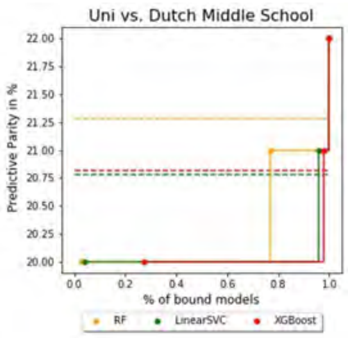

Figure 5.35: Ranked Performance Group 1.1 Income Dataset.

Figure 5.32 shows that for the Crime Dataset, various Surrounding models differ in Unfairness value, when compared to the Basis model. Feature combination Group 2.1 and feature combination Group 3.1 show different results, the Surrounding models have either an overall lower Unfairness value or is very concentrated around the Basis model.

For feature combination Group 1.1 of the Income Dataset, see Figure 5.35, out of the Surrounding ML models, only a handful can be distinguished in terms of its Unfairness value. The Surrounding models for feature combination Group 2.1 and feature combination 3.1 show very similar results.

16. Predictive Equality

The Predictive Equality for the Crime Dataset differs for each group, as well as each Algorithm, see Figure 5.36. The Unfairness value of the three feature combination groups all show a positive correlation with the accuracy of the Surrounding models, as can be seen in Figure 5.37. Figure 5.38 shows the Ranked Performance for feature combination Group 1.1. Surrounding models for feature combination Group 2.1 are different, they are concentrated around the Basis model. The Ranked Performance of feature combination Group 3.1 is similar tot that of feature combination Group 1.1.

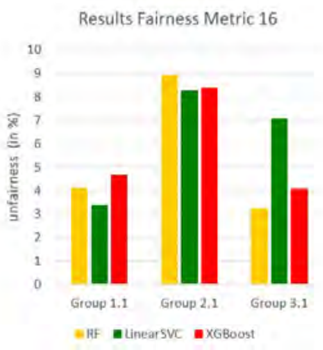

Figure 5.36: Result Basis models Crime Dataset.

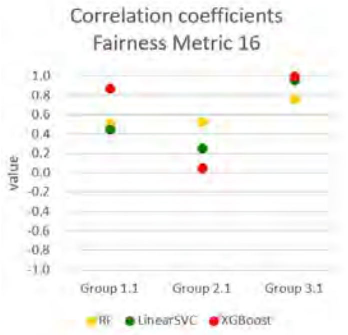

Figure 5.37: Correlation coefficients for Surrounding model accuracy Crime Dataset.

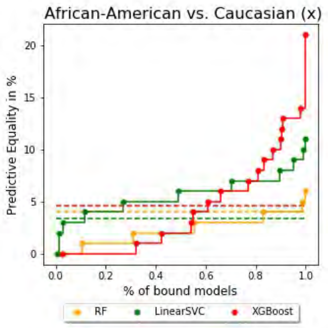

Figure 5.38: Ranked Performance Group 1.1 Crime Dataset.

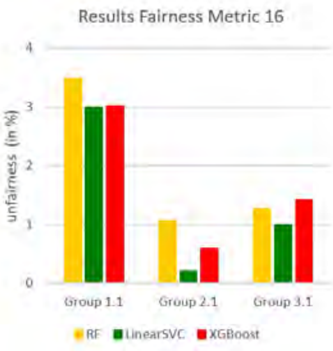

Figure 5.39: Result Basis models Income Dataset.

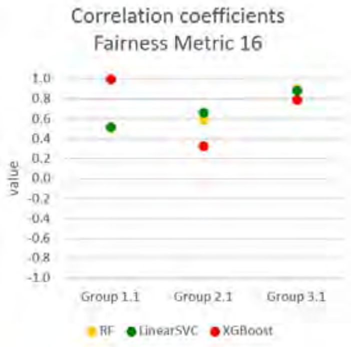

Figure 5.40: Correlation coefficients for Surrounding model accuracy Income Dataset.

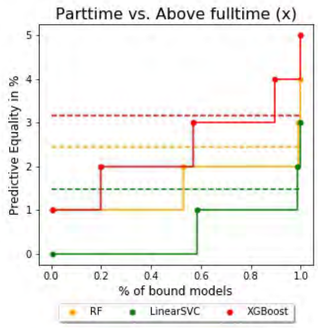

Figure 5.41: Ranked Performance Group 1.1 Income Dataset.

For the Income Dataset, see Figure 5.39 and 5.40, conclusions are similar to that of the Crime Dataset. The Surrounding models show that, for each Algorithm, only a hand full can be distinguished. Figure 5.41 shows the Ranked Performance for feature combination Group 1.1, Group 2.1 and 3.1 show different distributions, the majority of the Surrounding models perform similar to the Basis model.

17. Equal Opportunity

The Predictive Equality for the Crime Dataset differs for each group, as well as each ML Algorithm, see Figure 5.42. The value of Algorithm Fairness Metric 17 of the three feature combination groups all show a positive and a negative correlation with the accuracy of the Surrounding models. The negative correlation holds for Group 2.1 and the Surrounding XGBoost models, as can be seen in Figure 5.43. Figure 5.44 shows the Ranked Performance for feature combination Group 1.1, Group 2.1 has similar results. Surrounding models for Group 3.1 have a broader range in terms of the Algorithm Fairness Metric value.

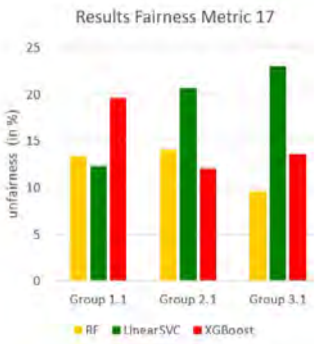

Figure 5.42: Result Basis models Crime Dataset.

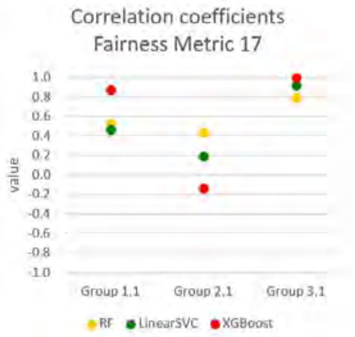

Figure 5.43: Correlation coefficients for Surrounding model accuracy Crime Dataset.

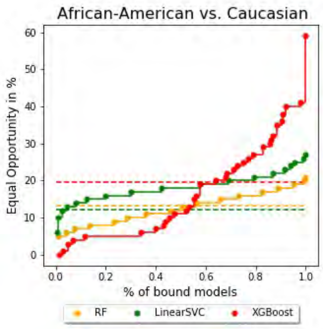

Figure 5.44: Ranked Performance Group 1.1 Crime Dataset.

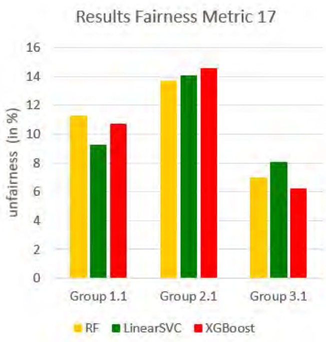

Figure 5.45: Result Basis models Income Dataset.

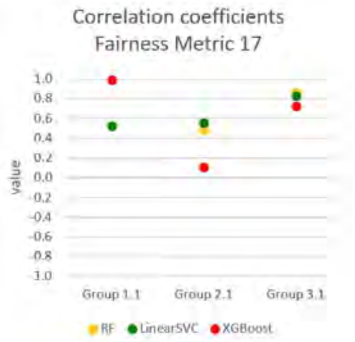

Figure 5.46: Correlation coefficients for Surrounding model accuracy Income Dataset.

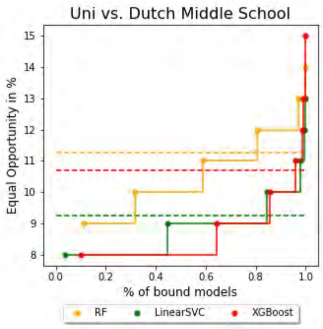

Figure 5.47: Ranked Performance Group 1.1 Income Dataset.

For the Income Dataset, see Figure 5.45 and 5.46, compared to the Crime Dataset, the Basis ML Algorithms are less varying for each feature combination Group. The Surrounding models show that, for each ML Algorithm, only a hand full can be distinguished. Figure 5.47 shows the Ranked Performance for feature combination Group 1.1, feature combination Group 3.1 has similar Ranked Performance.

18. Equalized Odds

Figure 5.48 and Figure 5.51 show different results for each group and each of the Basis models. The group differences for the Crime Dataset are larger than those of the Income Dataset. Both dataset show that the Algorithm Fairness Metric value is positively correlated with the accuracy of the Surrounding models, see Figure 5.49 and Figure 5.52. The Ranked Performance of feature combination Group 2.1 and Group 3.1 are similar to feature combination Group 1.1.

Based on Figure 5.51 and 5.52, compared to the Crime Dataset, the Basis ML Algorithms of the Income Dataset are more varying for each feature combination group. Figure 5.53 shows the Ranked Performance for feature combination Group 1.1, feature combination Group 3.1 has similar Ranked Performance.

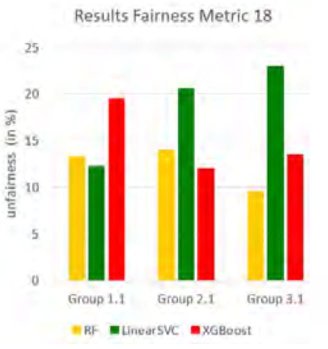

Figure 5.48: Result Basis models Crime Dataset.

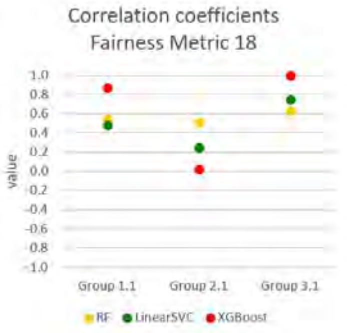

Figure 5.49: Correlation coefficients for Surrounding model accuracy Crime Dataset.

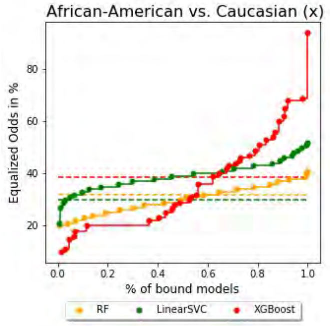

Figure 5.50: Ranked Performance Group 1.1 Crime Dataset.

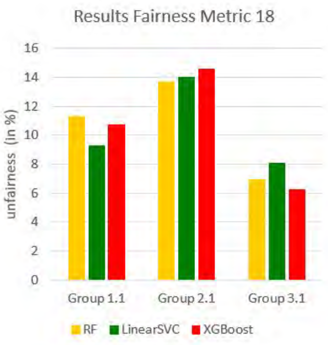

Figure 5.51: Result Basis models Income Dataset.

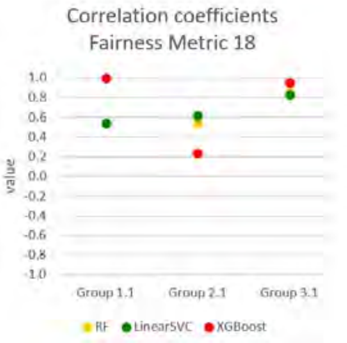

Figure 5.52: Correlation coefficients for Surrounding model accuracy Income Dataset.

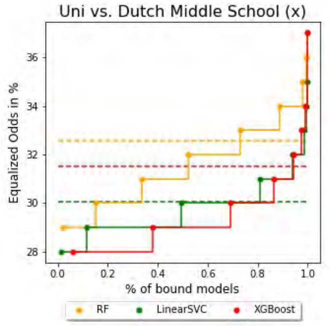

Figure 5.53: Ranked Performance Group 1.1 Income Dataset.

19. Conditional Use Accuracy Equality

For the Crime Dataset, feature combination Group 2.1 and Group 3.1 show higher Unfairness values for the Basis LinearSVC model. For feature combination Group 1.1, the Basis RF model retrieves the highest Unfairness value, see Figure 5.54.

Also, the Algorithm Fairness Metric values are positively correlated with the accuracy of the Surrounding models, as can be seen in Figure 5.55. The same holds for the Crime Dataset, however, the variation of the ML Algorithms within feature combination Group 1.1, Group 2.1 and Group 3.1 are less varying, see Figure 5.57 and Figure 5.59.

For both datasets, the Ranked Performances of the Surrounding models vary among the three Groups for each ML Algorithm, being either very concentrated towards the Basis model, or the opposite.

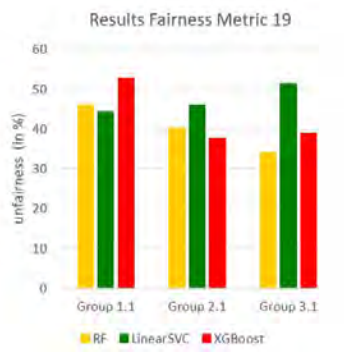

Figure 5.54: Result Basis models Crime Dataset.

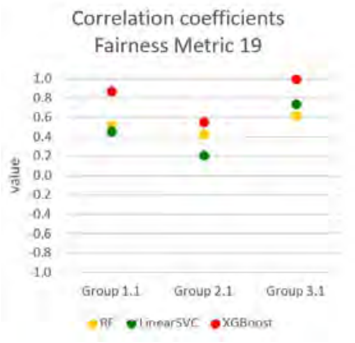

Figure 5.55: Correlation coefficients for Surrounding model accuracy Crime Dataset.

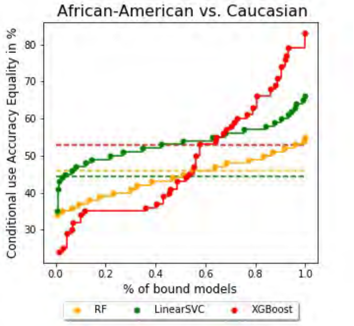

Figure 5.56: Ranked Performance Group 1.1 Crime Dataset.

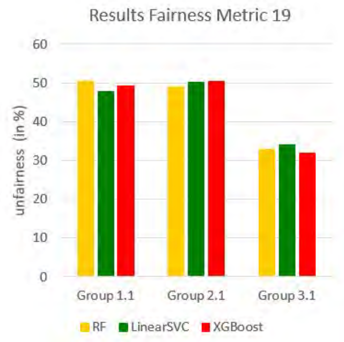

Figure 5.57: Result Basis models Income Dataset.

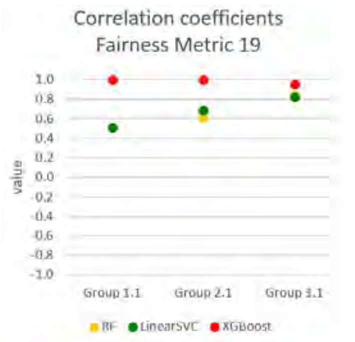

Figure 5.58: Correlation coefficients for Surrounding model accuracy Income Dataset.

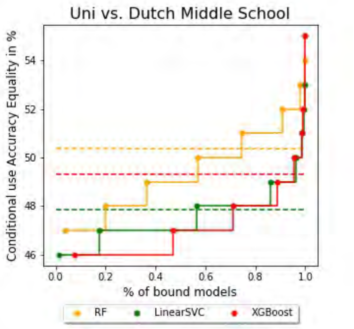

Figure 5.59: Ranked Performance Group 1.1 Income Dataset.

20. Overall Accuracy Equality

Results from Algorithm Fairness Metric 20 are similar to that of Algorithm Fairness Metric 19. Hence, for the Crime Dataset, feature combination Group 2.1 and Group 3.1 show higher Unfairness values for the Basis LinearSVC model. For feature combination Group 1.1, the Basis RF model retrieves the highest Unfairness value.

Also, the Algorithm Fairness Metric values are positively correlated with the accuracy of the Surrounding models. The same holds for the Income Dataset, however, the variation of the ML Algorithms within feature combination Group 1.1, feature combination Group 2.1 and Group 3.1 are less varying.

21. Treatment Equality

The results for Algorithm Fairness Metric 21 are also similar to that of Algorithm Fairness Metric 19. However, instead of the Basis LinearSVC model, feature combination Group 1.1 and Group 2.1 show higher Unfairness for the Basis XGBoost model.

5.5 Algorithm Fairness Metrics Results Type 4: Def. Based On Predicted Probabilities And Actual Outcome (22 – 25)

22-23. Test-Fairness and Well-calibration

Table 5.9 shows the lowerbound for which the Algorithm is calibrated for the Crime Dataset. For both the Basis RF, LinearSVC and the XGBoost model and the various feature combination groups, the classifier satisfies the Test-Fairness definition, Algorithm Fairness Metric 22, for high predicted probabilities but not for low scores.

	RF	LinearSVC	XGBoost
Group 1.1	0.5	0.7	0.5
Group 2.1	0.5	0.7	0.5
Group 3.1	0.5	0.6	0.5

Table 5.9: Lowerbound for the well-calibrated scores s, Crime Dataset.

Table 5.10 shows the lowerbound for which the Algorithm is calibrated for the Income Dataset. The Basis LinearSVC and XGBoost model and the various feature combination groups, satisfy the Test-Fairness definition, Algorithm Fairness Metric 22, for high predicted probabilities and low scores. For the Basis RF model, for feature combination Group 3.1, Group 1.1 and Group 2.1 this is satisfied for high predicted probabilities but not for low scores.

	RF	LinearSVC	XGBoost
Group 1.1	0.5	0.3	0.2
Group 2.1	0.5	0.3	0.3
Group 3.1	0.3	0.3	0.2

Table 5.10: Lowerbound for the well-calibrated scores s, Income Dataset.

24. Balance For Positive Class

Figure 5.60 shows that the Basis models have different results for each feature combination group. For feature combination Group 1.1 and Group 3.1, the Basis XGBoost model has the highest Unfairness value. For Group 2.1, the Basis RF model retrieves the highest Unfairness value. Figure 5.61 shows different results compared to the Crime Dataset. The Basis LinearSVC model has the highest Unfairness value for Group 2.1 and 3.1 while this is the Basis XGBoost model for feature combination Group 1.1.

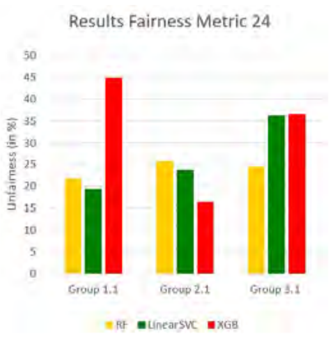

Figure 5.60: Result Basis models, Crime Dataset.

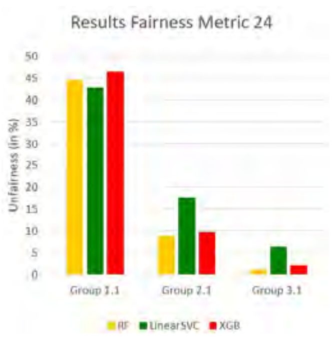

Figure 5.61: Result Basis models, Income Dataset.

Different results are obtained for a varying Seed choices on the train / test dataset split, there is no clear relationship. This holds for both datasets.

25. Balance For Negative Class

Figure 5.62 shows the Balance for Negative Class for the Crime Dataset. The Algorithm Fairness Metric values vary among the ML Algorithms, as well as the groups. The Basis RF model results in the highest Unfairness value for feature combination Group 1.1 and Group 2.1. For feature combination Group 3.1, the Basis LinearSVC model has a significant higher Unfairness value, compared to the Basis RF and XGBoost model.

Compared to the Crime Dataset, the Income Dataset shows different results, see Figure 5.63. For all the three groups, the Basis LinearSVC model has the highest Unfairness value.

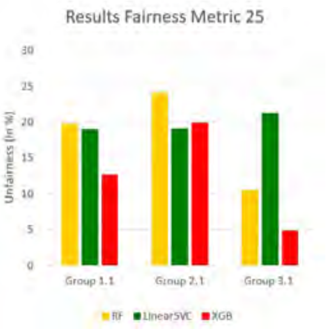

Figure 5.62: Result Basis models, Crime Dataset.

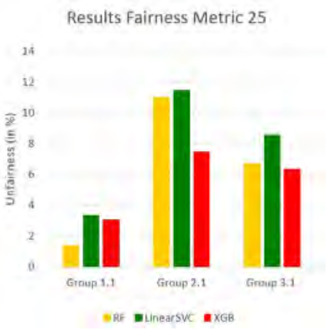

Figure 5.63: Result Basis models, Income Dataset.

Different results are obtained for a varying Seed choices on the train / test dataset split, there is no clear relationship. This holds for both datasets.

5.6 Results type 5: Similarity Based Measures (26 – 28)

26. Causal Discrimination

Figure 5.64 shows the Causal Discrimination for the Basis models for feature combination Group 1.1, Group 2.1, Group 3.1 for the Crime Dataset. For both cases, the XGBoost has the highest Unfairness.

For the Income Dataset, the overall Unfairness values are lower. In specific, the Basis XGBoost models retrieve low (almost zero) Unfairness.

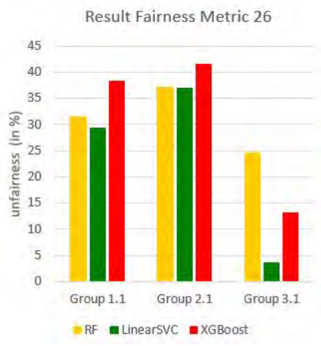

Figure 5.64: Result Basis models Crime Dataset.

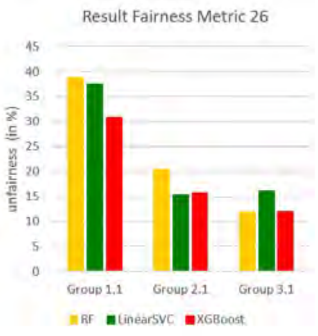

Figure 5.65: Result Basis models Income Dataset.

27. Fairness Through Unawareness

The Basis models for Algorithm Fairness Metric 27 show similar results for feature combination Group 1.1 and Group 2.1. Feature combination Group 3.1 is not calculated since the gender feature is removed from the Crime Dataset for this Metric. The Basis RF model results in the highest Unfairness, compared to the Basis LinearSVC and XGBoost model. The ML Algorithms have similar Unfairness values for feature combination Group 1.1 and 2.1, differing about one percent.

For both datasets, the Surrounding models show different Unfairness metrics for each feature combination Group and ML Algorithm.

28. Fairness Through Awareness

Algorithm Fairness Metric 28 can only be calculated for the Crime Dataset. Table 5.11 shows the results for the Basis RF, LinearSVC and the XGBoost models. Values are round to three decimals and show that the distance metric of the Basis RF and XGBoost model are close to zero. Violating cases decrease for both the Basis RF and LinearSVC models between the age category from 25 years up to 45 years to ages above 45 years.

Age category difference	k	RF avg. D	RF % viol.	LinearSVC avg. D	LinearSVC % viol.	XGBoost avg. D	XGBoost % viol.
25 – 45	0.137	0.068	0.163	0.133	0.431	0	0
> 45	0.277	0.060	0.031	0.188	0.150	0	0

Table 5.11: Fairness Through Awareness, Crime Dataset.

5.7 Algorithm Fairness Metric Results type 6: Causal Reasoning (29 – 32)

Table 5.12 shows the Causal Reasoning Table, they are similar for the Basis RF, LinearSVC en XGBoost models. Values of one indicate a dependency between the two features. Conclusions of Algorithm Fairness Metrics 29-32 are based on this. The predicted Algorithm outcome depends on many features. Either direct or indirect, all the feature are accessible. There are both illegitimate and legitimate paths between the features and the predicted Algorithm outcome.

	norm. nr. days in jail	norm. age jail out	no total counts	multiple juv counts	multiple total counts	no juv counts	norm. year of birth	norm. age jail in	gender	age cat	race	marital status	legal status	language	predicted RF / LinearSVC / XGBoost
norm. nr. days in jail	x	1	0	0	0	0	1	1	0	0	0	0	0	0	0
norm. age jail out	1	x	1	1	1	1	1	1	0	1	1	1	0	0	1
no total counts	0	1	x	1	1	1	1	1	1	1	1	1	0	0	1
multiple juv counts	0	1	1	x	1	1	0	1	0	0	0	0	0	0	0
multiple total counts	0	1	1	1	x	1	1	1	1	1	1	1	0	0	1
no juv counts	0	1	1	1	1	x	0	1	0	0	0	0	0	0	0
norm. year of birth	1	1	1	0	1	0	x	1	0	1	1	1	0	0	1
norm. age jail in	1	1	1	1	1	1	1	x	0	1	1	1	0	0	1
gender	0	0	1	0	1	0	0	0	x	0	0	0	0	0	1
age cat	0	1	1	0	1	0	1	1	0	x	1	1	0	0	1
race	0	1	1	0	1	0	1	1	0	1	x	1	0	1	1
marital status	0	1	1	0	1	0	1	1	0	1	1	x	1	1	1
legal status	0	0	0	0	0	0	0	0	0	0	0	1	x	1	0
language	0	0	0	0	0	0	0	0	0	0	1	1	1	x	0
predicted RF / LinearSVC / XGBoost	0	1	1	0	1	0	1	1	1	1	1	1	0	0	x

Table 5.12: Causal Reasoning Table for the Crime Dataset.

For the Income Dataset, a similar Table is retrieved for the Basis RF, LinearSVC en XGBoost models. See Section 8.3.2 in the Appendix for the Causal Reasoning Table.

For both datasets, the varying Seed choice for the train / test dataset split resulted in similar Causal Reasoning Graphs and Tables.

Chapter 6

Conclusion

This study aimed to research the impact of the dataset, the supervised Machine Learning (ML) classification Algorithm choice, its Surrounding models, Bootstrapping the train / test dataset split and a varying Seed choice for the train / test dataset split on the Algorithm Fairness Metrics that were described by Verma and Rubin (2018).

Except for Algorithm Fairness Metric type 6, the Algorithm Fairness Metrics showed different results for the two datasets in this study. Hence, the results are not generic which reinforces other conclusions of this study.

Compared to the smaller Crime Dataset, for each ML Algorithm, the Basis model and Surrounding models of the Income Dataset were more accurate. Also, their Algorithm Fairness Metric values were more stable than those of the Crime Dataset. Consequently, larger datasets tend to have more accurate ML Algorithms and have therefore more stable Algorithm Fairness Metric values.

The ML Algorithms showed different Algorithm Fairness Metric results. For the datasets in this study, the Basis LinearSVC model was the best performing ML Algorithm. For various Algorithm Fairness Metrics, prediction of the Basis LinearSVC model showed higher unfairness values compared to the less performing Basis Random Forest and XGBoost models.

Surrounding models are models that are built around the Basis model, they differ in accuracy and hyperparameters. This study compared the Algorithm Fairness Metric values of three feature combination. Often, a positive correlation between the accuracy of the Surrounding models and the Algorithm Fairness Metric value was observed. In other words, models with higher accuracy tend to have higher unfairness.

For each of the six Algorithm Fairness Metric types, varying values are obtained when performing a Bootstrap with 10 iterations on the train / test dataset split. Also, there are varying Algorithm Fairness Metric values for the 10 different Seed choices on the train / test dataset split. Hence, the representation of each feature, and its proportion in the training set or test set, tend to be leading for the Algorithm Fairness Metric values.

To summarize, Algorithm Fairness Metric results are not stable and depend on many factors. They are impacted by the dataset, the supervised ML classification Algorithm choice, its Surrounding models, Bootstrapping the train / test dataset split and for varying Seed choice for the train / test dataset split. However, relations are often unclear.

It is important to take each Algorithm Fairness Metric and each feature combination characteristic into consideration when using predictive Algorithms on real-life cases. The underlying python code of this study may be very useful. It distinguishes itself from other python packages as it computes all the Algorithm Fairness Metrics in one go, given any dataset that has five feature groups with $2, 3, 6, 7, 5$ items.

To conclude, this study aims to inspire other researchers to use this as a source to gain insight into how the Algorithm Fairness Metrics interact and differ for their specific predictive ML Algorithms.

Chapter 7

Discussion

First of all, the computation time has been a challenging aspect throughout this study, especially for the larger Income Dataset. This limited the size of the hyperparameter grid as well as certain hyperparameter values such as the C value for LinearSVC which is a good alternative for the computationally heavy Support Vector Machine.

Secondly, the definition of a Surrounding model with their hyperparameter range have been chosen based on trial and error. As a result, especially for the Surrounding XGBoost models, the Algorithm Fairness Metrics of models with very different accuracy were compared. Saving interim results of the RandomizedSearchCV() and the Genetic Algorithm could have resulted in models that one deems to classify as better Surrounding model. With this approach, the results for models with different hyperparameters and almost equal accuracy may be compared.

Thirdly, more iterations for the Bootstrap and broader analyzing its results would have been beneficial to make conclusion about this, as well as the impact of the seed choice for the train/test dataset split before passing the models onto the Algorithm Fairness Metrics calculations.

Fourthly, To make sense of the 32 Algorithm Fairness Metrics, outcomes for the three Algorithms and the two datasets where both had 46 one-on-one feature comparisons. A tool that automatically analyzes each of the Algorithm Fairness Metrics for each dataset and Algorithm, and base conclusion on this, is left for further research. This would help to analyze which Algorithm Fairness Metric is more stable compared to another Algorithm Fairness Metric, and why.

In addition, to make the results comparable, I distinguished feature combination groups based on their data proportion in the dataset. However, this approach leads to Algorithm Fairness Results that vary significantly for each group. Further research should rule out this cause. Stratified sampling may give more insight into this. Also, it may be useful to include some analyses as to how each models classifies a feature in terms of importance.

Next, models with higher accuracy tend to have higher unfairness. The approach of this study has many uncertainties, the Algorithm Fairness Metrics depend on various factors. Further research should find out whether this trade-off also holds for a different approach.

Moreover, the python script that has been created for this study, automatically creates both the excel sheets with fairness values as well as the Ranked Performance graphs. Feature removal is included as well, though, its Algorithm Fairness Metrics calculations and result interpretation is left for further research. Finally, using programs such as PowerBI of Tableau, will contribute to the readability of the Algorithm Fairness Metric results.

Chapter 8

Appendix

8.1 Model Parameter Settings

8.1.1 Random Forest

The parameter grid for the RandomizedSearchCV() of the Base Random Forest model is as followed:
$grid = \{'n\_estimators'$: [x for x in range$(1, 100, 10)]$
$'max\_depth' : [xforxinrange(1, 100, 10)]\}$

The remainder hyperparameter settings are set at default, hence:

$bootstrap = True$
$ccp\_alpha = 0.0;'$
$class\_weight = None;$
$criterion =' gini';$
$max\_features =' auto';$
$max\_leaf\_nodes = None;$
$max\_samples = None; min\_impurity\_decrease = 0.0;$
$min\_impurity\_split = None;$
$min\_samples\_leaf = 1;$
$min\_samples\_split = 2;$
$min\_weight\_fraction\_leaf = 0.0;$
$n\_jobs = None;$
$oob\_score = False;$
$random\_state = None;$
$verbose = 0; warm\_start = False.$

8.1.2 XGBoost

The parameter settings for the Genetic Algorithm of the Base LinearSVC model is as followed:

XGBoost parameter	Uniform distribution parameters
$learning\_rate$	$[0.01; 1]$
min_child_weight	$[0.01; 10]$
$gamma$	$[0.01; 10]$
$subsample$	$[0.01; 1]$
$colsample\_bytree$	$[0.01; 1]$

Table 8.1: XGBoost parameters whose value is from an Uniform distribution.

XGBoost parameter	Random number range
$n\_estimators$	$[10, 1500]$
max_depth	$[1, 10]$

Table 8.2: XGBoost parameters whose value is a random value.

The remainder hyperparameter settings are set at default, hence:

$base\_score = 0.5;$
$booster =' gbtree';$
$colsample\_bylevel = 1;$
$colsample\_bynode = 1;$
$max\_delta\_step = 0;$
$missing = None;$
$n\_jobs = 8;$
$nthread = None;$
$objective =' binary : logistic';$
$random\_state = 0;$
$reg\_alpha = 0;$
$reg\_lambda = 1;$
$scale\_pos_weight = 1;$
$seed = None;$
$silent = None;$
$verbosity = 1.$

8.1.3 LinearSVC

The parameter grid for the RandomizedSearchCV() of the Base LinearSVC model is as followed:
LinearSVC_grid = {’C’: [x for x in range(1,10)], ’tol’: [1e-5, 1e-4, 1e-3, 1e-2, 1, 10,100]}

The remainder hyperparameter settings are set at default, hence:
$class\_weight = None;$
$dual = True; fit\_intercept = True;$
$intercept\_scaling = 1;$
$loss =' squared\_hinge'; max\_iter = 1000;$
$multi\_class =' ovr',$
$penalty =' 12';$
$random_state = None.$

8.2 Crime Dataset

8.2.1 Groups for Algorithm Fairness Metrics $13, 15 - 21, 26, 27$

Combination	**Proportion comparing 1**	**Proportion, comparing 2**
(’Male’, ’Female’)	**79.8%**	**20.2%**
(’Less_than_25’, ’25-45’)	**20.4%**	**58%**
(’Less_than_25’, ’Greater_than_45’)	**20.4%**	**21.6%**
(’25-45’, ’Greater_than_45’)	**58%**	**21.6%**
(’race_African-American’, ’race_Asian’)	49.4%	0.5%
(’race_African-American’, ’race_Caucasian’)	**49.4%**	**34%**
(’race_African-American’, ’race_Hispanic’)	49.4%	10.2%
(’race_African-American’, ’race_Native_American’)	49.4%	0.4%
(’race_African-American’, ’race_Other’)	49.4%	5.6%
(’race_Asian’, ’race_Caucasian’)	0.5%	34%
(’race_Asian’, ’race_Hispanic’)	0.5%	10.2%
(’race_Asian’, ’race_Native_American’)	0.5%	0.4%
(’race_Asian’, ’race_Other’)	0.5%	5.6%
(’race_Caucasian’, ’race_Hispanic’)	**34%**	**10.2%**
(’race_Caucasian’, ’race_Native_American’)	34%	0.4%
(’race_Caucasian’, ’race_Other’)	**34%**	**5.6%**
(’race_Hispanic’, ’race_Native_American’)	10.2%	0.4%
(’race_Hispanic’, ’race_Other’)	10.2%	5.6%
(’race_Native_American’, ’race_Other’)	0.4%	5.6%
(’Divorced’, ’Married’)	**4.5%**	**11.4%**
(’Divorced’, ’Seperated’)	4.5%	2%
(’Divorced’, ’Significant_other’)	**4.5%**	**4%**
(’Divorced’, ’Single’)	4.5%	74.6%
(’Divorced’, ’Widowed’)	4.5%	0.5%
(’Divorced’, ’Unknown_maritalStatus’)	4.5%	3.1%
(’Married’, ’Seperated’)	11.4%	2%
(’Married’, ’Significant_other’)	11.4%	4%
(’Married’, ’Single’)	**11.4%**	**74.6%**
(’Married’, ’Widowed’)	11.4%	0.5%
(’Married’, ’Unknown_maritalStatus’)	11.4%	3.1%
(’Seperated’, ’Significant_other’)	2%	4%
(’Seperated’, ’Single’)	2%	74.6%
(’Seperated’, ’Widowed’)	2%	0.5%
(’Seperated’, ’Unknown_maritalStatus’)	2%	3.1%
(’Significant_other’, ’Single’)	4%	74.6%
(’Significant_other’, ’Widowed’)	4%	0.5%
(’Significant_other’, ’Unknown_maritalStatus’)	4%	3.1%
(’Single’, ’Widowed’)	74.6%	0.5%
(’Single’, ’Unknown_maritalStatus’)	74.6%	3.1%
(’Widowed’, ’Unknown_maritalStatus’)	0.5%	3.1%
(’legal_Pretrial’, ’legal_Post_Sentence’)	95%	1.8%
(’legal_Pretrial’, ’legal_Unknown’)	95%	2.8%
(’legal_Pretrial’, ’legal_other’)	95%	0.4%
(’legal_Post_Sentence’, ’legal_Unknown’)	1.8%	2.8%
(’legal_Post_Sentence’, ’legal_other’)	1.8%	0.4%
(’legal_Unknown’, ’legal_other’)	2.8%	0.4%

Table 8.3: Groups Crime Dataset.

8.2.2 Groups for Conditional Statistical Parity

Combination	Proportion of combination	Comparing	Proportion comparing 1	Proportion comparing 2
[’Male’, ’25-45’, ’Single’, ’legal_Pretrial’]	33.3%	(’race_African-American’, ’race_Asian’)	18.2%	0.2%
[’Male’, ’25-45’, ’Single’, ’legal_Pretrial’]	**33.3%**	**(’race_African-American’, ’race_Caucasian’)**	**18.2%**	**10.1%**
[’Male’, ’25-45’, ’Single’, ’legal_Pretrial’]	33.3%	(’race_African-American’, ’race_Hispanic’)	18.2%	3.4%
[’Male’, ’25-45’, ’Single’, ’legal_Pretrial’]	33.3%	(’race_African-American’, ’race_Native_American’)	18.2%	0.1%
[’Male’, ’25-45’, ’Single’, ’legal_Pretrial’]	33.3%	(’race_African-American’, ’race_Other’)	18.2%	1.4%
[’Male’, ’25-45’, ’Single’, ’legal_Pretrial’]	33.3%	(’race_Asian’, ’race_Caucasian’)	0.2%	10.1%
[’Male’, ’race_African-American’, ’Single’, ’legal_Pretrial’]	**31.2%**	**(’Less_than_25’, ’25-45’)**	**9.2%**	**18.2%**
[’Male’, ’race_African-American’, ’Single’, ’legal_Pretrial’]	31.2%	(’Less_than_25’, ’Greater_than_45’)	9.2%	3.8%
[’Male’, ’race_African-American’, ’Single’, ’legal_Pretrial’]	31.2%	(’25-45’, ’Greater_than_45’)	18.2%	3.8%
[’25-45’, ’race_African-American’, ’Single’, ’legal_Pretrial’]	**22.79%**	**(’Male’, ’Female’)**	**18.2%**	**4.6%**
[’Male’, ’25-45’, ’race_African-American’, ’legal_Pretrial’]	22.1%	(’Divorced’, ’Married’)	0.3%	1.7%
[’Male’, ’25-45’, ’race_African-American’, ’legal_Pretrial’]	22.1%	(’Divorced’, ’Seperated’)	0.3%	0.4%
[’Male’, ’25-45’, ’race_African-American’, ’legal_Pretrial’]	22.1%	(’Divorced’, ’Significant_other’)	0.3%	1.4%
[’Male’, ’25-45’, ’race_African-American’, ’legal_Pretrial’]	22.1%	(’Divorced’, ’Single’)	0.3%	18.2%
[’Male’, ’25-45’, ’race_African-American’, ’legal_Pretrial’]	22.1%	(’Divorced’, ’Widowed’)	0.3%	0%
[’Male’, ’25-45’, ’race_African-American’, ’legal_Pretrial’]	22.1%	(’Divorced’, ’Unknown_maritalStatus’)	0.3%	0.1%
[’Male’, ’25-45’, ’race_African-American’, ’legal_Pretrial’]	22.1%	(’Married’, ’Seperated’)	1.7%	0.4%
[’Male’, ’race_Caucasian’, ’Single’, ’legal_Pretrial’]	18.8%	(’Less_than_25’, ’25-45’)	3.8%	10.1%
[’Male’, ’race_Caucasian’, ’Single’, ’legal_Pretrial’]	18.8%	(’Less_than_25’, ’Greater_than_45’)	3.8%	4.8%
[’Male’, ’race_Caucasian’, ’Single’, ’legal_Pretrial’]	**18.8%**	**(’25-45’, ’Greater_than_45’)**	**10.1%**	**4.8%**
[’Male’, ’25-45’, ’race_African-American’, ’Single’]	18.8%	(’legal_Pretrial’, ’legal_Post_Sentence’)	18.2%	0.4%
[’Male’, ’25-45’, ’race_African-American’, ’Single’]	18.8%	(’legal_Pretrial’, ’legal_Unknown’)	18.2%	0%
[’Male’, ’25-45’, ’race_African-American’, ’Single’]	18.8%	(’legal_Pretrial’, ’legal_other’)	18.2%	0.1%
[’Male’, ’25-45’, ’race_African-American’, ’Single’]	18.8%	(’legal_Post_Sentence’, ’legal_Unknown’)	0.4%	0%
[’Male’, ’Less_than_25’, ’Single’, ’legal_Pretrial’]	15.3%	(’race_African-American’, ’race_Asian’)	9.2%	0.1%
[’Male’, ’Less_than_25’, ’Single’, ’legal_Pretrial’]	15.3%	(’race_African-American’, ’race_Caucasian’)	9.2%	3.8%
[’Male’, ’Less_than_25’, ’Single’, ’legal_Pretrial’]	15.3%	(’race_African-American’, ’race_Hispanic’)	9.2%	1.2%
[’Male’, ’Less_than_25’, ’Single’, ’legal_Pretrial’]	15.3%	(’race_African-American’, ’race_Native_American’)	9.2%	0%
[’Male’, ’Less_than_25’, ’Single’, ’legal_Pretrial’]	15.3%	(’race_African-American’, ’race_Other’)	9.2%	1%
[’Male’, ’Less_than_25’, ’Single’, ’legal_Pretrial’]	15.3%	(’race_Asian’, ’race_Caucasian’)	0.1%	3.8%
[’25-45’, ’race_Caucasian’, ’Single’, ’legal_Pretrial’]	13.1%	(’Male’, ’Female’)	10.1%	3%
[’Male’, ’25-45’, ’race_Caucasian’, ’legal_Pretrial’]	13.1%	(’Divorced’, ’Married’)	0.8%	1.4%
[’Male’, ’25-45’, ’race_Caucasian’, ’legal_Pretrial’]	13.1%	(’Divorced’, ’Seperated’)	0.8%	0.3%
[’Male’, ’25-45’, ’race_Caucasian’, ’legal_Pretrial’]	13.1%	(’Divorced’, ’Significant_other’)	0.8%	0.4%
[’Male’, ’25-45’, ’race_Caucasian’, ’legal_Pretrial’]	13.1%	(’Divorced’, ’Single’)	0.8%	10.1%
[’Male’, ’25-45’, ’race_Caucasian’, ’legal_Pretrial’]	13.1%	(’Divorced’, ’Widowed’)	0.8%	0%
[’Male’, ’25-45’, ’race_Caucasian’, ’legal_Pretrial’]	13.1%	(’Divorced’, ’Unknown_maritalStatus’)	0.8%	0%
[’Male’, ’25-45’, ’race_Caucasian’, ’legal_Pretrial’]	13.1%	(’Married’, ’Seperated’)	1.4%	0.3%
[’Less_than_25’, ’race_African-American’, ’Single’, ’legal_Pretrial’]	11.31%	(’Male’, ’Female’)	9.2%	2.1%
[’Male’, ’25-45’, ’race_Caucasian’, ’Single’]	10.5%	(’legal_Pretrial’, ’legal_Post_Sentence’)	10.1%	0.3%
[’Male’, ’25-45’, ’race_Caucasian’, ’Single’]	10.5%	(’legal_Pretrial’, ’legal_Unknown’)	10.1%	0%
[’Male’, ’25-45’, ’race_Caucasian’, ’Single’]	10.5%	(’legal_Pretrial’, ’legal_other’)	10.1%	0.1%
[’Male’, ’25-45’, ’race_Caucasian’, ’Single’]	10.5%	(’legal_Post_Sentence’, ’legal_Unknown’)	0.3%	0%
[’Male’, ’Greater_than_45’, ’Single’, ’legal_Pretrial’]	9.9%	(’race_African-American’, ’race_Asian’)	3.8%	0%
[’Male’, ’Greater_than_45’, ’Single’, ’legal_Pretrial’]	9.9%	(’race_African-American’, ’race_Caucasian’)	3.8%	4.8%
[’Male’, ’Greater_than_45’, ’Single’, ’legal_Pretrial’]	9.9%	(’race_African-American’, ’race_Hispanic’)	3.8%	0.9%
[’Male’, ’Greater_than_45’, ’Single’, ’legal_Pretrial’]	9.9%	(’race_African-American’, ’race_Native_American’)	3.8%	0.1%
[’Male’, ’Greater_than_45’, ’Single’, ’legal_Pretrial’]	9.9%	(’race_African-American’, ’race_Other’)	3.8%	0.3%
[’Male’, ’Greater_than_45’, ’Single’, ’legal_Pretrial’]	9.9%	(’race_Asian’, ’race_Caucasian’)	0%	4.8%
[’Male’, ’Less_than_25’, ’race_African-American’, ’legal_Pretrial’]	9.4%	(’Divorced’, ’Married’)	0%	0.1%
[’Male’, ’Less_than_25’, ’race_African-American’, ’legal_Pretrial’]	9.4%	(’Divorced’, ’Seperated’)	0%	0%
[’Male’, ’Less_than_25’, ’race_African-American’, ’legal_Pretrial’]	9.4%	(’Divorced’, ’Significant_other’)	0%	0%
[’Male’, ’Less_than_25’, ’race_African-American’, ’legal_Pretrial’]	9.4%	(’Divorced’, ’Single’)	0%	9.2%
[’Male’, ’Less_than_25’, ’race_African-American’, ’legal_Pretrial’]	9.4%	(’Divorced’, ’Widowed’)	0%	0%
[’Male’, ’Less_than_25’, ’race_African-American’, ’legal_Pretrial’]	9.4%	(’Divorced’, ’Unknown_maritalStatus’)	0%	0%
[’Male’, ’Less_than_25’, ’race_African-American’, ’legal_Pretrial’]	9.4%	(’Married’, ’Seperated’)	0.1%	0%
[’Male’, ’Less_than_25’, ’race_African-American’, ’Single’]	9.4%	(’legal_Pretrial’, ’legal_Post_Sentence’)	9.2%	0.1%
[’Male’, ’Less_than_25’, ’race_African-American’, ’Single’]	9.4%	(’legal_Pretrial’, ’legal_Unknown’)	9.2%	0%
[’Male’, ’Less_than_25’, ’race_African-American’, ’Single’]	9.4%	(’legal_Pretrial’, ’legal_other’)	9.2%	0.1%
[’Male’, ’Less_than_25’, ’race_African-American’, ’Single’]	9.4%	(’legal_Post_Sentence’, ’legal_Unknown’)	0.1%	0%
[’Female’, ’25-45’, ’Single’, ’legal_Pretrial’]	8.6%	(’race_African-American’, ’race_Asian’)	4.6%	0%
[’Female’, ’25-45’, ’Single’, ’legal_Pretrial’]	8.6%	(’race_African-American’, ’race_Caucasian’)	4.6%	3%
[’Female’, ’25-45’, ’Single’, ’legal_Pretrial’]	8.6%	(’race_African-American’, ’race_Hispanic’)	4.6%	0.7%
[’Female’, ’25-45’, ’Single’, ’legal_Pretrial’]	8.6%	(’race_African-American’, ’race_Native_American’)	4.6%	0%
[’Female’, ’25-45’, ’Single’, ’legal_Pretrial’]	8.6%	(’race_African-American’, ’race_Other’)	4.6%	0.3%
[’Female’, ’25-45’, ’Single’, ’legal_Pretrial’]	8.6%	(’race_Asian’, ’race_Caucasian’)	0%	3%
[’Male’, ’Greater_than_45’, ’race_Caucasian’, ’legal_Pretrial’]	7.8%	(’Divorced’, ’Married’)	1%	1.1%
[’Male’, ’Greater_than_45’, ’race_Caucasian’, ’legal_Pretrial’]	7.8%	(’Divorced’, ’Seperated’)	1%	0.2%
[’Male’, ’Greater_than_45’, ’race_Caucasian’, ’legal_Pretrial’]	7.8%	(’Divorced’, ’Significant_other’)	1%	0.5%
[’Male’, ’Greater_than_45’, ’race_Caucasian’, ’legal_Pretrial’]	7.8%	(’Divorced’, ’Single’)	1%	4.8%
[’Male’, ’Greater_than_45’, ’race_Caucasian’, ’legal_Pretrial’]	7.8%	(’Divorced’, ’Widowed’)	1%	0.1%
[’Male’, ’Greater_than_45’, ’race_Caucasian’, ’legal_Pretrial’]	7.8%	(’Divorced’, ’Unknown_maritalStatus’)	1%	0.1%
[’Male’, ’Greater_than_45’, ’race_Caucasian’, ’legal_Pretrial’]	7.8%	(’Married’, ’Seperated’)	1.1%	0.2%
[’Female’, ’race_African-American’, ’Single’, ’legal_Pretrial’]	7.7%	(’Less_than_25’, ’25-45’)	2.1%	4.6%
[’Female’, ’race_African-American’, ’Single’, ’legal_Pretrial’]	7.7%	(’Less_than_25’, ’Greater_than_45’)	2.1%	1%
[’Female’, ’race_African-American’, ’Single’, ’legal_Pretrial’]	7.7%	(’25-45’, ’Greater_than_45’)	4.6%	1%
[’Male’, ’Greater_than_45’, ’race_African-American’, ’legal_Pretrial’]	6.2%	(’Divorced’, ’Married’)	0.5%	1.3%
[’Male’, ’Greater_than_45’, ’race_African-American’, ’legal_Pretrial’]	6.2%	(’Divorced’, ’Seperated’)	0.5%	0.2%
[’Male’, ’Greater_than_45’, ’race_African-American’, ’legal_Pretrial’]	6.2%	(’Divorced’, ’Significant_other’)	0.5%	0.2%
[’Male’, ’Greater_than_45’, ’race_African-American’, ’legal_Pretrial’]	6.2%	(’Divorced’, ’Single’)	0.5%	3.8%
[’Male’, ’Greater_than_45’, ’race_African-American’, ’legal_Pretrial’]	6.2%	(’Divorced’, ’Widowed’)	0.5%	0.1%
[’Male’, ’Greater_than_45’, ’race_African-American’, ’legal_Pretrial’]	6.2%	(’Divorced’, ’Unknown_maritalStatus’)	0.5%	0.1%
[’Male’, ’Greater_than_45’, ’race_African-American’, ’legal_Pretrial’]	6.2%	(’Married’, ’Seperated’)	1.3%	0.2%
[’Greater_than_45’, ’race_Caucasian’, ’Single’, ’legal_Pretrial’]	6.16%	(’Male’, ’Female’)	4.8%	1.3%
[’Female’, ’25-45’, ’race_African-American’, ’legal_Pretrial’]	5.7%	(’Divorced’, ’Married’)	0.1%	0.5%
[’Female’, ’25-45’, ’race_African-American’, ’legal_Pretrial’]	5.7%	(’Divorced’, ’Seperated’)	0.1%	0.1%
[’Female’, ’25-45’, ’race_African-American’, ’legal_Pretrial’]	5.7%	(’Divorced’, ’Significant_other’)	0.1%	0.3%
[’Female’, ’25-45’, ’race_African-American’, ’legal_Pretrial’]	5.7%	(’Divorced’, ’Single’)	0.1%	4.6%
[’Female’, ’25-45’, ’race_African-American’, ’legal_Pretrial’]	5.7%	(’Divorced’, ’Widowed’)	0.1%	0%
[’Female’, ’25-45’, ’race_African-American’, ’legal_Pretrial’]	5.7%	(’Divorced’, ’Unknown_maritalStatus’)	0.1%	0%
[’Female’, ’25-45’, ’race_African-American’, ’legal_Pretrial’]	5.7%	(’Married’, ’Seperated’)	0.5%	0.1%
[’Male’, ’race_Hispanic’, ’Single’, ’legal_Pretrial’]	5.4%	(’Less_than_25’, ’25-45’)	1.2%	3.4%
[’Male’, ’race_Hispanic’, ’Single’, ’legal_Pretrial’]	5.4%	(’Less_than_25’, ’Greater_than_45’)	1.2%	0.9%
[’Male’, ’race_Hispanic’, ’Single’, ’legal_Pretrial’]	5.4%	(’25-45’, ’Greater_than_45’)	3.4%	0.9%

Table 8.4: Conditional Statistical Parity groups Crime Dataset.

8.3 Income Dataset

8.3.1 Groups for Algorithm Fairness Metrics $13, 15 - 21, 26, 27$.

Combination	**Proportion comparing 1**	**Proportion, comparing 2**
('Male', 'Female')	**66.8%**	**33.2%**
('Fulltime', 'Parttime')	**46.6%**	**24%**
('Fulltime', 'Above_fulltime')	46.6%	29.4%
('Parttime', 'Above_fulltime')	**24%**	**29.4%**
('race_African_American', 'race_Asian')	9.1%	3.3%
('race_African_American', 'race_Caucasian')	**9.1%**	**86%**
('race_African_American', 'race_Hispanic')	9.1%	0%
('race_African_American', 'race_Native_American')	9.1%	0.9%
('race_African_American', 'race_Other')	9.1%	0.7%
('race_Asian', 'race_Caucasian')	3.3%	86%
('race_Asian', 'race_Hispanic')	3.3%	0%
('race_Asian', 'race_Native_American')	3.3%	0.9%
('race_Asian', 'race_Other')	3.3%	0.7%
('race_Caucasian', 'race_Hispanic')	86%	0%
('race_Caucasian', 'race_Native_American')	86%	0.9%
('race_Caucasian', 'race_Other')	86%	0.7%
('race_Hispanic', 'race_Native_American')	0%	0.9%
('race_Hispanic', 'race_Other')	0%	0.7%
('race_Native_American', 'race_Other')	0.9%	0.7%
('Divorced', 'Married')	**14.1%**	**47.9%**
('Divorced', 'Seperated')	14.1%	3.2%
('Divorced', 'Significant_other')	14.1%	0%
('Divorced', 'Single')	**14.1%**	**32.1%**
('Divorced', 'Widowed')	14.1%	2.7%
('Divorced', 'Unknown_maritalStatus')	14.1%	0%
('Married', 'Seperated')	47.9%	3.2%
('Married', 'Significant_other')	47.9%	0%
('Married', 'Single')	**47.9%**	**32.1%**
('Married', 'Widowed')	47.9%	2.7%
('Married', 'Unknown_maritalStatus')	47.9%	0%
('Seperated', 'Significant_other')	3.2%	0%
('Seperated', 'Single')	3.2%	32.1%
('Seperated', 'Widowed')	**3.2%**	**2.7%**
('Seperated', 'Unknown_maritalStatus')	3.2%	0%
('Significant_other', 'Single')	0%	32.1%
('Significant_other', 'Widowed')	0%	2.7%
('Significant_other', 'Unknown_maritalStatus')	0%	0%
('Single', 'Widowed')	32.1%	2.7%
('Single', 'Unknown_maritalStatus')	32.1%	0%
('Widowed', 'Unknown_maritalStatus')	2.7%	0%
('Above_uni', 'Uni')	11.5%	44.2%
('Above_uni', 'Middelbare_school')	**11.5%**	**42.6%**
('Above_uni', 'Basisschool')	11.5%	1.7%
('Uni', 'Middelbare_school')	**44.2%**	**42.6%**
('Uni', 'Basisschool')	44.2%	1.7%
('Middelbare_school', 'Basisschool')	42.6%	1.7%

Table 8.5: Groups Income Dataset.

8.3.2 Groups for Conditional Statistical Parity

Combination	Proportion of combination	Comparing	Proportion comparing 1	Proportion comparing 2
['Male', 'Fulltime', 'race_Caucasian', 'Married']	17.4%	('Above_uni', 'Uni')	2.0%	7.0%
['Male', 'Fulltime', 'race_Caucasian', 'Married']	17.4%	('Above_uni', 'Middelbare_school')	2.0%	7.9%
['Male', 'Fulltime', 'race_Caucasian', 'Married']	17.4%	('Above_uni', 'Basisschool')	2.0%	0.4%
['Male', 'Fulltime', 'race_Caucasian', 'Married']	**17.4%**	**('Uni', 'Middelbare_school')**	**7.0%**	**7.9%**
['Male', 'race_Caucasian', 'Married', 'Uni']	16.4%	('Fulltime', 'Parttime')	7.0%	1.7%
['Male', 'race_Caucasian', 'Married', 'Uni']	**16.4%**	**('Fulltime', 'Above_fulltime')**	**7.0%**	**7.7%**
['Male', 'race_Caucasian', 'Married', 'Uni']	16.4%	('Parttime', 'Above_fulltime')	1.7%	7.7%
['Male', 'Above_fulltime', 'race_Caucasian', 'Married']	15.8%	('Above_uni', 'Uni')	2.7%	7.7%
['Male', 'Above_fulltime', 'race_Caucasian', 'Married']	15.8%	('Above_uni', 'Middelbare_school')	2.7%	5.2%
['Male', 'Above_fulltime', 'race_Caucasian', 'Married']	15.8%	('Above_uni', 'Basisschool')	2.7%	0.2%
['Male', 'Above_fulltime', 'race_Caucasian', 'Married']	**15.8%**	**('Uni', 'Middelbare_school')**	**7.7%**	**5.2%**
['Male', 'race_Caucasian', 'Married', 'Middelbare_school']	15.1%	('Fulltime', 'Parttime')	7.9%	1.9%
['Male', 'race_Caucasian', 'Married', 'Middelbare_school']	**15.1%**	**('Fulltime', 'Above_fulltime')**	**7.9%**	**5.2%**
['Male', 'race_Caucasian', 'Married', 'Middelbare_school']	15.1%	('Parttime', 'Above_fulltime')	1.9%	5.2%
['Male', 'Fulltime', 'race_Caucasian', 'Middelbare_school']	12.9%	('Divorced', 'Married')	1.3%	7.9%
['Male', 'Fulltime', 'race_Caucasian', 'Middelbare_school']	12.9%	('Divorced', 'Seperated')	1.3%	0.3%
['Male', 'Fulltime', 'race_Caucasian', 'Middelbare_school']	12.9%	('Divorced', 'Significant_other')	1.3%	0.0%
['Male', 'Fulltime', 'race_Caucasian', 'Middelbare_school']	12.9%	('Divorced', 'Single')	1.3%	3.4%
['Male', 'Fulltime', 'race_Caucasian', 'Middelbare_school']	12.9%	('Divorced', 'Widowed')	1.3%	0.0%
['Male', 'Fulltime', 'race_Caucasian', 'Middelbare_school']	12.9%	('Divorced', 'Unknown_maritalStatus')	1.3%	0.0%
['Male', 'Fulltime', 'race_Caucasian', 'Middelbare_school']	12.9%	('Married', 'Seperated')	7.9%	0.3%
['Male', 'Fulltime', 'race_Caucasian', 'Uni']	10.7%	('Divorced', 'Married')	0.8%	7.0%
['Male', 'Fulltime', 'race_Caucasian', 'Uni']	10.7%	('Divorced', 'Seperated')	0.8%	0.2%
['Male', 'Fulltime', 'race_Caucasian', 'Uni']	10.7%	('Divorced', 'Significant_other')	0.8%	0.0%
['Male', 'Fulltime', 'race_Caucasian', 'Uni']	10.7%	('Divorced', 'Single')	0.8%	2.6%
['Male', 'Fulltime', 'race_Caucasian', 'Uni']	10.7%	('Divorced', 'Widowed')	0.8%	0.0%
['Male', 'Fulltime', 'race_Caucasian', 'Uni']	10.7%	('Divorced', 'Unknown_maritalStatus')	0.8%	0.0%
['Male', 'Fulltime', 'race_Caucasian', 'Uni']	10.7%	('Married', 'Seperated')	7.0%	0.2%
['Male', 'Above_fulltime', 'race_Caucasian', 'Uni']	10.5%	('Divorced', 'Married')	0.9%	7.7%
['Male', 'Above_fulltime', 'race_Caucasian', 'Uni']	10.5%	('Divorced', 'Seperated')	0.9%	0.1%
['Male', 'Above_fulltime', 'race_Caucasian', 'Uni']	10.5%	('Divorced', 'Significant_other')	0.9%	0.0%
['Male', 'Above_fulltime', 'race_Caucasian', 'Uni']	10.5%	('Divorced', 'Single')	0.9%	1.8%
['Male', 'Above_fulltime', 'race_Caucasian', 'Uni']	10.5%	('Divorced', 'Widowed')	0.9%	0.0%
['Male', 'Above_fulltime', 'race_Caucasian', 'Uni']	10.5%	('Divorced', 'Unknown_maritalStatus')	0.9%	0.0%
['Male', 'Above_fulltime', 'race_Caucasian', 'Uni']	10.5%	('Married', 'Seperated')	7.7%	0.1%
['Male', 'Fulltime', 'Married', 'Middelbare_school']	9.2%	('race_African_American', 'race_Asian')	0.8%	0.2%
['Male', 'Fulltime', 'Married', 'Middelbare_school']	9.2%	('race_African_American', 'race_Caucasian')	0.8%	7.9%
['Male', 'Fulltime', 'Married', 'Middelbare_school']	9.2%	('race_African_American', 'race_Hispanic')	0.8%	0.0%
['Male', 'Fulltime', 'Married', 'Middelbare_school']	9.2%	('race_African_American', 'race_Native_American')	0.8%	0.1%
['Male', 'Fulltime', 'Married', 'Middelbare_school']	9.2%	('race_African_American', 'race_Other')	0.8%	0.1%
['Male', 'Fulltime', 'Married', 'Middelbare_school']	9.2%	('race_Asian', 'race_Caucasian')	0.2%	7.9%
['Fulltime', 'race_Caucasian', 'Married', 'Middelbare_school']	9.0%	('Male', 'Female')	7.9%	1.1%
['Above_fulltime', 'race_Caucasian', 'Married', 'Uni']	8.3%	('Male', 'Female')	7.7%	0.6%
['Male', 'Above_fulltime', 'Married', 'Uni']	8.2%	('race_African_American', 'race_Asian')	0.2%	0.2%
['Male', 'Above_fulltime', 'Married', 'Uni']	8.2%	('race_African_American', 'race_Caucasian')	0.2%	7.7%
['Male', 'Above_fulltime', 'Married', 'Uni']	8.2%	('race_African_American', 'race_Hispanic')	0.2%	0.0%
['Male', 'Above_fulltime', 'Married', 'Uni']	8.2%	('race_African_American', 'race_Native_American')	0.2%	0.0%
['Male', 'Above_fulltime', 'Married', 'Uni']	8.2%	('race_African_American', 'race_Other')	0.2%	0.0%
['Male', 'Above_fulltime', 'Married', 'Uni']	8.2%	('race_Asian', 'race_Caucasian')	0.2%	7.7%
['Fulltime', 'race_Caucasian', 'Married', 'Uni']	8.1%	('Male', 'Female')	7.0%	1.0%
['Male', 'Fulltime', 'Married', 'Uni']	8.0%	('race_African_American', 'race_Asian')	0.6%	0.3%
['Male', 'Fulltime', 'Married', 'Uni']	8.0%	('race_African_American', 'race_Caucasian')	0.6%	7.0%
['Male', 'Fulltime', 'Married', 'Uni']	8.0%	('race_African_American', 'race_Hispanic')	0.6%	0.0%
['Male', 'Fulltime', 'Married', 'Uni']	8.0%	('race_African_American', 'race_Native_American')	0.6%	0.1%
['Male', 'Fulltime', 'Married', 'Uni']	8.0%	('race_African_American', 'race_Other')	0.6%	0.1%
['Male', 'Fulltime', 'Married', 'Uni']	8.0%	('race_Asian', 'race_Caucasian')	0.3%	7.0%
['Male', 'Above_fulltime', 'race_Caucasian', 'Middelbare_school']	7.7%	('Divorced', 'Married')	1.0%	5.2%
['Male', 'Above_fulltime', 'race_Caucasian', 'Middelbare_school']	7.7%	('Divorced', 'Seperated')	1.0%	0.1%
['Male', 'Above_fulltime', 'race_Caucasian', 'Middelbare_school']	7.7%	('Divorced', 'Significant_other')	1.0%	0.0%
['Male', 'Above_fulltime', 'race_Caucasian', 'Middelbare_school']	7.7%	('Divorced', 'Single')	1.0%	1.4%
['Male', 'Above_fulltime', 'race_Caucasian', 'Middelbare_school']	7.7%	('Divorced', 'Widowed')	1.0%	0.1%
['Male', 'Above_fulltime', 'race_Caucasian', 'Middelbare_school']	7.7%	('Divorced', 'Unknown_maritalStatus')	1.0%	0.0%
['Male', 'Above_fulltime', 'race_Caucasian', 'Middelbare_school']	7.7%	('Married', 'Seperated')	5.2%	0.1%
['Male', 'race_Caucasian', 'Single', 'Uni']	6.7%	('Fulltime', 'Parttime')	2.6%	2.3%
['Male', 'race_Caucasian', 'Single', 'Uni']	6.7%	('Fulltime', 'Above_fulltime')	2.6%	1.8%
['Male', 'race_Caucasian', 'Single', 'Uni']	6.7%	('Parttime', 'Above_fulltime')	2.3%	1.8%
['Male', 'race_Caucasian', 'Single', 'Middelbare_school']	6.7%	('Fulltime', 'Parttime')	3.4%	1.9%
['Male', 'race_Caucasian', 'Single', 'Middelbare_school']	6.7%	('Fulltime', 'Above_fulltime')	3.4%	1.4%
['Male', 'race_Caucasian', 'Single', 'Middelbare_school']	6.7%	('Parttime', 'Above_fulltime')	1.9%	1.4%
['Male', 'Fulltime', 'race_Caucasian', 'Single']	6.6%	('Above_uni', 'Uni')	0.5%	2.6%
['Male', 'Fulltime', 'race_Caucasian', 'Single']	6.6%	('Above_uni', 'Middelbare_school')	0.5%	3.4%
['Male', 'Fulltime', 'race_Caucasian', 'Single']	6.6%	('Above_uni', 'Basisschool')	0.5%	0.2%
['Male', 'Fulltime', 'race_Caucasian', 'Single']	6.6%	('Uni', 'Middelbare_school')	2.6%	3.4%
['Female', 'race_Caucasian', 'Single', 'Uni']	6.5%	('Fulltime', 'Parttime')	2.5%	2.8%
['Female', 'race_Caucasian', 'Single', 'Uni']	6.5%	('Fulltime', 'Above_fulltime')	2.5%	1.3%
['Female', 'race_Caucasian', 'Single', 'Uni']	6.5%	('Parttime', 'Above_fulltime')	2.8%	1.3%
['Male', 'Above_fulltime', 'Married', 'Middelbare_school']	5.6%	('race_African_American', 'race_Asian')	0.2%	0.1%
['Male', 'Above_fulltime', 'Married', 'Middelbare_school']	5.6%	('race_African_American', 'race_Caucasian')	0.2%	5.2%
['Male', 'Above_fulltime', 'Married', 'Middelbare_school']	5.6%	('race_African_American', 'race_Hispanic')	0.2%	0.0%
['Male', 'Above_fulltime', 'Married', 'Middelbare_school']	5.6%	('race_African_American', 'race_Native_American')	0.2%	0.0%
['Male', 'Above_fulltime', 'Married', 'Middelbare_school']	5.6%	('race_African_American', 'race_Other')	0.2%	0.0%
['Male', 'Above_fulltime', 'Married', 'Middelbare_school']	5.6%	('race_Asian', 'race_Caucasian')	0.1%	5.2%
['Above_fulltime', 'race_Caucasian', 'Married', 'Middelbare_school']	5.4%	('Male', 'Female')	5.2%	0.2%
['Female', 'Fulltime', 'race_Caucasian', 'Uni']	5.3%	('Divorced', 'Married')	1.4%	1.0%
['Female', 'Fulltime', 'race_Caucasian', 'Uni']	5.3%	('Divorced', 'Seperated')	1.4%	0.1%
['Female', 'Fulltime', 'race_Caucasian', 'Uni']	5.3%	('Divorced', 'Significant_other')	1.4%	0.0%
['Female', 'Fulltime', 'race_Caucasian', 'Uni']	5.3%	('Divorced', 'Single')	1.4%	2.5%
['Female', 'Fulltime', 'race_Caucasian', 'Uni']	5.3%	('Divorced', 'Widowed')	1.4%	0.3%
['Female', 'Fulltime', 'race_Caucasian', 'Uni']	5.3%	('Divorced', 'Unknown_maritalStatus')	1.4%	0.0%
['Female', 'Fulltime', 'race_Caucasian', 'Uni']	5.3%	('Married', 'Seperated')	1.0%	0.1%
['Male', 'race_Caucasian', 'Married', 'Above_uni']	5.2%	('Fulltime', 'Parttime')	2.0%	0.4%
['Male', 'race_Caucasian', 'Married', 'Above_uni']	5.2%	('Fulltime', 'Above_fulltime')	2.0%	2.7%
['Male', 'race_Caucasian', 'Married', 'Above_uni']	5.2%	('Parttime', 'Above_fulltime')	0.4%	2.7%
['Female', 'Parttime', 'race_Caucasian', 'Single']	5.1%	('Above_uni', 'Uni')	0.4%	2.8%
['Female', 'Parttime', 'race_Caucasian', 'Single']	5.1%	('Above_uni', 'Middelbare_school')	0.4%	2.0%
['Female', 'Parttime', 'race_Caucasian', 'Single']	5.1%	('Above_uni', 'Basisschool')	0.4%	0.0%
['Female', 'Parttime', 'race_Caucasian', 'Single']	5.1%	('Uni', 'Middelbare_school')	2.8%	2.0%
['Fulltime', 'race_Caucasian', 'Single', 'Uni']	5.1%	('Male', 'Female')	2.6%	2.5%
['Parttime', 'race_Caucasian', 'Single', 'Uni']	5.1%	('Male', 'Female')	2.3%	2.8%

Table 8.6: Conditional Statistical Parity groups Income Dataset.

Causal Reasoning

	work Self-emp-not-inc	work Federal-gov	occ Protective-serv	NC South-America	occ Farming-fishing	occ Machine-op-inspct	NC Europe	occ Transport-moving	occ Adm-clerical	occ Priv-house-serv	work Self-emp-inc	occ Armed-Forces	NC Asia	occ Tech-support	occ Handlers-cleaners	capital loss	occ Sales	NC North-America	occ Unknown	fnlwgt norm.	work Private	work Without-pay	work Never-worked	occ Craft-repair	work Unknown	occ Other-service	occ Exec-managerial	work Local-gov	capital gain	work State-gov	NC Unknown	occ Prof-specialty	gender	work time	race	marital status	education	Predicted
work Self-emp-not-inc	x	1	0	0	1	1	0	0	1	0	1	0	0	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1	0	1	0	0	1	1	1	1	0	0
work Federal-gov	1	x	1	0	0	1	0	0	1	0	0	1	0	1	0	0	1	0	1	0	1	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1
occ Protective-serv	0	1	x	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	1	1	1	0	1	0	1	1	1	0	0	0	1
NC South-America	0	0	0	x	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
occ Farming-fishing	1	0	0	0	x	1	0	1	1	0	0	0	0	0	0	0	1	0	1	0	1	1	0	1	1	1	1	0	0	0	0	1	1	1	0	1	1	1
occ Machine-op-inspct	1	1	0	1	1	x	0	1	1	0	0	0	0	1	1	0	1	0	1	0	1	0	0	1	1	1	1	1	0	1	0	1	0	1	1	0	1	1
NC Europe	0	0	0	0	0	0	x	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
occ Transport-moving	0	0	0	0	1	1	0	x	1	0	0	0	0	1	1	0	1	0	1	0	1	0	0	1	1	1	1	0	0	0	0	1	1	1	0	1	1	0
occ Adm-clerical	1	1	1	0	1	1	0	1	x	0	1	0	0	1	1	0	1	0	1	0	0	0	0	1	1	1	1	0	0	1	0	1	1	1	0	1	1	1
occ Priv-house-serv	0	0	0	0	0	0	1	0	0	x	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1
work Self-emp-inc	1	0	0	0	0	0	0	0	1	0	x	0	0	0	0	0	1	0	1	0	1	0	0	0	1	1	1	1	1	1	0	0	1	1	0	1	1	1
occ Armed-Forces	0	1	0	0	0	0	0	0	0	0	0	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NC Asia	0	0	0	0	0	0	0	0	0	0	0	0	x	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
occ Tech-support	0	1	0	0	0	1	0	1	1	0	0	0	1	x	0	0	1	0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	0	0	0	0	1	1
occ Handlers-cleaners	1	0	0	0	0	1	0	1	1	0	0	0	0	0	x	0	1	0	1	0	1	0	0	1	1	1	1	0	1	0	0	1	1	1	0	1	1	1
capital loss	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0	1	0	1
occ Sales	1	1	1	0	1	1	0	1	1	0	1	0	0	1	1	0	x	0	1	0	1	0	0	1	1	1	1	1	0	1	0	1	0	1	0	0	1	1
NC North-America	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	x	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0
occ Unknown	1	1	0	0	1	1	0	1	1	0	1	0	0	1	1	0	1	0	x	0	1	0	1	1	1	1	1	1	0	1	0	1	1	1	0	1	1	1
fnlwgt norm.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
work Private	1	1	1	0	1	1	0	1	0	1	1	0	0	0	1	0	1	0	1	0	x	0	0	1	1	1	0	1	1	1	0	1	0	1	0	1	1	1
work Without-pay	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
work Never-worked	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
occ Craft-repair	1	0	1	0	1	1	0	1	1	0	0	0	0	1	1	0	1	0	1	0	1	0	0	x	1	1	1	1	0	1	0	1	1	1	1	1	1	0
work Unknown	1	1	0	0	1	1	0	1	1	0	1	0	0	1	1	0	1	0	1	0	1	0	0	1	x	1	1	1	0	1	0	1	1	1	0	1	1	1
occ Other-service	0	1	1	0	1	1	0	1	1	0	1	0	0	1	1	1	1	0	1	0	1	0	0	1	1	x	1	0	1	0	0	1	1	1	1	1	1	1
occ Exec-managerial	0	1	1	0	1	1	0	1	1	0	1	0	0	1	1	0	1	0	1	0	0	0	0	1	1	1	x	0	1	0	0	1	0	1	0	1	1	1
work Local-gov	1	1	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	1	0	1	0	0	1	1	0	0	x	0	1	0	1	1	0	0	1	1	0
capital gain	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	1	1	0	x	0	0	1	1	1	0	1	1	1
work State-gov	1	0	1	0	0	1	0	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	1	1	0	0	1	0	x	0	1	0	0	0	0	1	0
NC Unknown	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	x	0	0	0	1	0	0	0
occ Prof-specialty	0	0	1	0	1	1	0	1	1	0	0	0	0	1	1	1	1	1	1	0	1	0	0	1	1	1	1	1	1	1	0	x	0	1	0	0	1	1
gender	1	0	1	0	1	0	0	1	1	1	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	0	1	1	0	0	0	x	1	1	1	0	1
work time	1	1	1	0	1	1	0	1	1	0	1	0	0	0	1	1	1	0	1	0	1	0	0	1	1	1	1	0	1	0	0	1	1	x	1	1	1	1
race	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1	1	x	1	1	1
marital status	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	0	0	1	0	1	0	0	1	1	1	1	1	1	0	0	0	1	1	1	x	1	1
education	0	1	0	0	1	1	0	1	1	1	1	0	1	1	1	0	1	1	1	0	1	0	0	1	1	1	1	1	1	1	0	1	0	1	1	1	x	1
Predicted	0	1	1	0	1	1	0	0	1	1	1	0	1	1	1	1	1	0	1	0	1	0	0	0	1	1	1	0	1	0	0	1	1	1	1	1	1	x

References

Florida county detention facilities average inmate population june 2016. *Prepared by: Florida Department of Corrections Bureau of Research and Data Analysis*, pages 1–7, 2016.

R. Agarwal. Top 5 classification evaluation metrics every data scientist must know, 2019. URL https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226. Last visited on December 3, 2019.

J. Angwin, J. Larson, S. Mattu, and L. Kirchner. How we analyzed the compas recidivism algorithm, 2016. URL https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm. Last visited on May 31, 2020.

J. Baron and J.C. Hershey. Outcome bias in decision evaluation. *Journal of Personality and Social Psychology*, 54(4):569–579, 1988. doi: 10.1037//0022-3514.54.4.569.

R. Baurichter. Amerikaans wetsvoorstel moet big tech dwingen algoritmes uit te leggen, 2019. URL https://fd.nl/economie-politiek/1296854/amerikaans-wetsvoorstel-moet-big-tech-dwingen-algoritmes-uit-te-leggen. Last visited on December 2, 2019.

R. Berk, H. Heidari, S. Jabbari, M. Kearns, and A. Roth. Ffairness in criminal justice risk assessments: The state of the art. 2017. doi: https://arxiv.org/pdf/1703.09207.pdf.

T. Blomberg, W. Bales, K. Mann, R. Meldrum, and J. Nedelec. Broward county jail population: Trends and forecast. *Center for Criminology and Public Policy Research College of Criminology and Criminal Justice Florida State University Tallahassee, Florida*, pages 1–82, 2010.

R.S. Brid. Boosting, 2018. URL https://medium.com/greyatom/boosting-ce84639a805d. Last visited on June 23, 2020.

J. Brownlee. A gentle introduction to xgboost for applied machine learning, 2016. URL https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/. Last visited on November 18, 2019.

J. Brownlee. A gentle introduction to the bootstrap method, 2018a. URL https://machinelearningmastery.com/a-gentle-introduction-to-the-bootstrap-method/. Last visited on June, 29, 2020.

J. Brownlee. A gentle introduction to k-fold cross validation, 2018b. URL https://machinelearningmastery.com/k-fold-cross-validation/. Last visited on November 28, 2019.

J. Brownlee. How to use roc curves and precision-recall curves for classification in python, 2018c. URL https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/,note={LastvisitedonJune24,2020}.

J. Brownlee. Metrics to evaluate machine learning algorithms in python, 2019a. URL https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/. Last visited on December 3, 2019.

J. Brownlee. A gentle introduction to threshold-moving for imbalanced classification, 2020. URL https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/#:~:text=The%20decision%20for%20converting%20a,in%20the%20range%20between%200. Last visited on June 12, 2020.

Jason Brownlee. A gentle introduction to the gradient boosting algorithm for machine learning, 2019b. URL https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/. Last visited on December 3, 2019.

M. Cage. A computer program used for bail and sentencing decisions was labeled biased against blacks. it's actually not that clear., 2016. URL https://www.washingtonpost.com/news/monkey-cage/wp/2016/10/17/can-an-algorithm-be-racist-our-analysis-is-more-cautious-than-propublicas/. Last visited on December 1, 2019.

Census. What we do, our mission, our authority, our goal, 2019. URL https://www.census.gov/about/what.html. Last visited on November 18, 2019.

CFI. What is boosting?, 2020. URL https://corporatefinanceinstitute.com/resources/knowledge/other/boosting/. Last visited on June, 23, 2020.

A. Chakure. Decision tree classification, 2019. URL https://towardsdatascience.com/decision-tree-classification-de64fc4d5aac. Last visited on November 28, 2019.

C. Cortes and V.N. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. doi: https://doi.org/10.1023/A:1022627411411.

DataCamp. Hyperparameter tuning with randomizedsearchcv, 2020. URL https://campus.datacamp.com/courses/supervised-learning-with-scikit-learn/fine-tuning-your-model?ex=11. Last visited on June, 13, 2020.

DeZyre. Metrics for evaluating machine learning algorithms, 2019. URL https://www.dezyre.com/data-science-in-python-tutorial/performance-metrics-for-machine-learning-algorithm. Last visited on November 28, 2019.

K. Dhiraj. Top 5 advantages and disadvantages of decision tree algorithm, 2019. URL https://medium.com/@dhiraj8899/top-5-advantages-and-disadvantages-of-decision-tree-algorithm-428ebd199d9a. Last visited on November 28, 2019.

S. Elsinghorst. Machine learning bias – gradient boosting xgboost, 2018. URL https://shirinsplayground.netlify.com/2018/11/ml_basics_gbm/. Last visited on December 3, 2019.

J. Frost. Introduction to bootstrapping in statistics with an example, 2020. URL https://statisticsbyjim.com/hypothesis-testing/bootstrapping/. Last visited on June, 29, 2020.

M.J. Garbade. Regression versus classification machine learning: What's the difference?, 2018. URL https://medium.com/quick-code/regression-versus-classification-machine-learning-whats-the-difference-345c56dd15f7. Last visited on November 18, 2019.

Github. Gradient boosting machines, 2019a. URL http://uc-r.github.io/gbm_regression#proscons. Last visited on December 3, 2019.

Github. Sklearn deep, 2019b. URL https://github.com/rsteca/sklearn-deap. Last visited on June, 13, 2020.

S. Glen. Decision tree vs random forest vs gradient boosting machines: Explained simply, 2019. URL datasciencecentral.com/profiles/blogs/decision-tree-vs-random-forest-vs-boosted-trees-explained. Last visited on November 18, 2019.

Google Developers Machine Learning. Training and test sets: Splitting data, 2019. URL https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data. Last visited on December 3, 2019.

P. Grover. Gradient boosting from scratch, 2017. URL https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d. Last visited on December 3, 2019.

K. Hao. This is how ai bias really happens – and why it's so hard to fix, 2019. URL https://www.technologyreview.com/s/612876/this-is-how-ai-bias-really-happensand-why-its-so-hard-to-fix/. Last visited on December 1, 2019.

T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning*. Springer, second edition, 2009. ISBN 9780387848587.

D. Hellman. Measuring algorithmic fairness. 2019. URL https://poseidon01.ssrn.com/delivery.php?ID=74409102608209800408201100311508001412104906205206408209500207911907103001110206509904
EXT=pdf. Last visited on November 19, 2019.

P.P. Ippolito. Hyperparameters optimization, 2019. URL https://towardsdatascience.com/hyperparameters-optimization-526348bb8e2d. Last visited on June, 13, 2020.

A. Jacobson. Fairness in the age of algorithms, 2019. URL https://medium.com/berkeleyischool/fairness-in-the-age-of-algorithms-feb11c56a709. Last visited on December 3, 2019.

A. Jain. Complete guide to parameter tuning in xgboost with codes in python, 2016. URL https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/. Last visited on November 18, 2019.

M. Jain. Hyperparameter tuning in xgboost using genetic algorithm, 2018. URL https://towardsdatascience.com/hyperparameter-tuning-in-xgboost-using-genetic-algorithm-17bd2e581b17. Last visited on June, 3, 2020.

S Jansen. Pros and cons of random forests, 2020. URL https://www.oreilly.com/library/view/hands-on-machine-learning/9781789346411/e17de38e-421e-4577-afc3-efdd4e02a468.xhtml. Last visited on June, 20, 2020.

J. Kho. Why random forest is my favorite machine learning model, 2018. URL https://towardsdatascience.com/why-random-forest-is-my-favorite-machine-learning-model-b97651fa3706. Last visited on June 18, 2019.

W. Koehrsen. Automated machine learning hyperparameter tuning in python, 2018a. URL https://towardsdatascience.com/automated-machine-learning-hyperparameter-tuning-in-python-dfda59b72f8a. Last visited on June, 13, 2020.

W. Koehrsen. Hyperparameter tuning the random forest in python, 2018b. URL https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74. Last visited on June, 3, 2020.

Krishni. K-fold cross validation, 2018. URL https://medium.com/datadriveninvestor/k-fold-cross-validation-6b8518070833. Last visited on November 29, 2019.

A. Kumar. Introduction to the gradient boosting algorithm, 2020. URL https://medium.com/analytics-vidhya/introduction-to-the-gradient-boosting-algorithm-c25c653f826b. Last visited on June 23, 2020.

R. Larrabee. *Imbalanced Classification with Python*. Second edition, 2020.

J. Larson, S. Mattu, L Kirchner, and J. Angwin. How we analyzed the compas recidivism algorithm, 2016. URL https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm. Last visited on May 31, 2020.

M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

P.R. Lokhart. Thousands of americans are jailed before trial. a new report shows the lasting impact., 2019. URL ThousandsofAmericansarejailedbeforetrial.Anewreportshowsthelastingimpact. Last visited on May 31, 2020.

M. Lopez de Prado. Chapter 9 hyper-parameter tuning with cross-validation, 2020. URL https://www.oreilly.com/library/view/advances-in-financial/9781119482086/c09.xhtml. Last visited on June, 13, 2020.

U. Malik. Cross validation and grid search for model selection in python, 2018. URL https://medium.com/@jackstalfort/hyperparameter-tuning-using-grid-search-and-random-search-f8750a464b35. Last visited on June, 13, 2020.

S. Mandava. Hyperparameter tuning in xgboost using genetic algorithm, 2018. URL https://medium.com/@mandava807/cross-validation-and-hyperparameter-tuning-in-python-65cfb80ee485#:~:text=Cross%20validation%20is%20a%20technique,without%20any%20overfitting%20and%20underfitting. Last visited on June, 13, 2020.

P. Mandot. How exactly xgboost works?, url = https://medium.com/@pushkarmandot/how-exactly-xgboost-works-a320d9b8aeef, note = Last visited on November 18, 2019, 2019.

R. G. Mantovani, A.L.D. Rossi, J. Vanschoren, B. Bischl, and de Carvalho A.C.P.L.F. Effectiveness of random search in svm hyper-parameter tuning. *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2015. doi: 10.1109/IJCNN.2015.7280664.

Medium. Math behind svm (support vector machine), 2019. URL https://medium.com/@ankitnitjsr13/math-behind-support-vector-machine-svm-5e7376d0ee4d. Last visited on November 23, 2019.

N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. A survey on bias and fairness in machine learning. 2019. doi: https://arxiv.org/pdf/1908.09635.pdf.

R. Meinert. Optimizing hyperparameters in random forest classification, 2019. URL https://towardsdatascience.com/optimizing-hyperparameters-in-random-forest-classification-ec7741f9d3f6. Last visited on June, 25, 2020.

V. Miler Jerkovic. The logistic regression in python — how to prepare a data and find the best model, 2019. URL https://medium.com/@veramiler/the-logistic-regression-in-python-how-to-prepare-a-data-and-find-the-best-model-a85a6 Last visited on June, 25, 2020.

V. Morde and V.A. Setty. Xgboost algorithm: Long may she reign!, 2019. URL https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d. Last visited on November 18, 2019.

A. Navlani. Support vector machines with scikit-learn, 2018. URL https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python. Last visited on November 24, 2019.

J. New. How to fix the algorithmic accountability act, 2019. URL https://www.datainnovation.org/2019/09/how-to-fix-the-algorithmic-accountability-act/. Last visited on December 2, 2019.

NRC.nl. Een nieuwe algoritmewet is overbodig, 2019. URL https://www.nrc.nl/nieuws/2019/07/05/een-nieuwe-algoritmewet-is-overbodig-a3966269. Last visited on December 2, 2019.

Nuffic. Grading systems in the netherlands, the united states and the united kingdom, 2013. URL http://barthokriek.nl/wp-content/uploads/Grades-in-US-UK-and-NL.pdf. Last visited on June 12, 2019.

D. Ofer. Compas recidivism racial bias, 2017. URL https://www.kaggle.com/danofer/compass. Last visited on May 31, 2020.

A. Patel. Machine learning algorithm overview, 2018. URL https://medium.com/ml-research-lab/machine-learning-algorithm-overview-5816a2e6303. Last visited on November 18, 2019.

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

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 1.4 support vector machines, 2019b. URL https://scikit-learn.org/stable/modules/svm.html. Last visited on November 30, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. An introduction to machine learning with scikit-learn, 2019a. URL https://scikit-learn.org/stable/tutorial/basic/tutorial.html#an-introduction-to-machine-learning-with-scikit-learn. Last visited on November 18, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Tuning the hyper-parameters of an estimator, 2020b. URL https://scikit-learn.org/stable/modules/grid_search.html. Last visited on June, 6, 2020.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. sklearn.svm.linearsvc, 2020a. URL https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#:~:text=Linear%20Support%20Vector%20Classification.,to%20large%20numbers%20of%20samples. Last visited on June, 24, 2020.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. sklearn model selection randomizedsearchcv, 2020c. URL https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html?highlight=randomizedsearchcv#sklearn.model_selection.RandomizedSearchCV. Last visited on June, 12, 2020.

M. Peixeiro. How good is your model? — intro to resampling methods, 2019. URL https://towardsdatascience.com/how-good-is-your-model-intro-to-resampling-methods-2831e832042a. Last visited on November 28, 2019.

S. Raschka. Model evaluation, model selection, and algorithm selection in machine learning, 2016. URL https://sebastianraschka.com/blog/2016/model-evaluation-selection-part2.html. Last visited on June, 29, 2020.

S. Rathi. Top 8 python libraries for machine learning artificial intelligence, 2019. URL https://hackernoon.com/top-8-python-libraries-for-machine-learning-and-artificial-intelligence-y08id3031. Last visited on December 1, 2019.

R. Sagar. What is the bootstrap method in statistical machine learning?, 2019. URL https://analyticsindiamag.com/what-is-the-bootstrap-method-in-statistical-machine-learning/. Last visited on June, 29, 2020.

M. Sanjeevi. Chapter 3: Support vector machine with math, 2017. URL https://medium.com/deep-math-machine-learning-ai/chapter-3-support-vector-machine-with-math-47d6193c82be. Last visited on November 28, 2019.

J. Schellevis. Rechtszaak over fraudebestrijding overheid: 'burgers bij voorbaat verdacht', 2019. URL https://nos.nl/artikel/2308110-rechtszaak-over-fraudebestrijding-overheid-burgers-bij-voorbaat-verdacht.html. Last visited on May 31, 2020.

J. Schellevis and W. de Jong. Overheid gebruikt op grote schaal voorspellende algoritmes, 'risico op discriminatie', 2019. URL https://nos.nl/artikel/2286848-overheid-gebruikt-op-grote-schaal-voorspellende-algoritmes-risico-op-discrimi
html. Last visited on December 2, 2019.

G. Seif. A beginner's guide to xgboost, 2019. URL https://towardsdatascience.com/a-beginners-guide-to-xgboost-87f5d4c30ed7. Last visited on December 4, 2019.

A.D. Selbst, D. Boyd, S.A. Friedler, S. Venkatasubramanian, and J. Vertesi. Fairness and abstraction in sociotechnical systems. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 2019. doi: 10.

S.I. Serengil. Gradient boosting decision trees in python, 2019. URL https://www.youtube.com/watch?v=KFsnZKMKNAE. Last visited on December 3, 2019.

A. Sharma. Difference between machine learning and artificial intelligence, 2019. URL https://www.geeksforgeeks.org/difference-between-machine-learning-and-artificial-intelligence/. Last visited on December 2, 2019.

Harshdeep Sing. Understanding gradient boosting machines, 2018. URL https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab. Last visited on December 3, 2019.

J. Stalfort. Hyperparameter tuning using grid search and random search: A conceptual guide, 2019. URL https://medium.com/@jackstalfort/hyperparameter-tuning-using-grid-search-and-random-search-f8750a464b35. Last visited on June, 13, 2020.

M. Sunasra. Perfromance metrics for classification problems in machine learning, 2017. URL https://medium.com/thalus-ai/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d43208
Last visited on December 3, 2019.

R.B. Sundaram. An end-to-end guide to understand the math behind xgboost, 2018. URL https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/. Last visited on November 18, 2019.

J.F. van Wijnen. Vvd wil toezicht op algoritmes, 2019. URL https://fd.nl/ondernemen/1301064/vvd-wil-toezicht-op-algoritmes. Last visited on December 2, 2019.

S. Verma and J. Rubin. Fairness definitions explained. *FairWare '18*, pages 1–7, 2018. doi: 10.1145/3194770.3194776.

A. Wilson. A brief introduction to supervised learning, 2019. URL https://towardsdatascience.com/a-brief-introduction-to-supervised-learning-54a3e3932590. Last visited on December 2, 2019.

E. Wirsansky. *Hands-On Genetic Algorithms with Python: Applying genetic algorithms to solve real-world deep learning and artificial intelligence problems*. Packt Publishing, first edition, 2020. ISBN 1838559183.

xgboost developers. Xgboost python package, 2020. URL https://xgboost.readthedocs.io/en/latest/python/python_intro.html. Last visited on June, 3, 2020.

G. Yona. A gentle introduction to the discussion on algorithmic fairness, 2017. URL https://towardsdatascience.com/a-gentle-introduction-to-the-discussion-on-algorithmic-fairness-740bbb469b6. Last visited on December 3, 2019.

J. Zhu and A. Singh. Semi-supervised learning, 2017. URL http://www.cs.cmu.edu/~10701/slides/17_SSL.pdf. Last visited on December 2, 2019.