`THE PENNSYLVANIA STATE UNIVERSITY

SCHREYER HONORS COLLEGE

DIVISION OF ENGINEERING, BUSINESS AND COMPUTING

Fairness in Artificial Intelligence Algorithms

Alina Rodriguez
Spring 2024

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Information Technology
with honors in Information Sciences and Technology

Reviewed and approved* by the following:

Abdullah Konak
Distinguished Professor of Information Sciences and Technology
Thesis Supervisor

Jeanne Marie Rose
Associate Professor of English
Honors Adviser

* Electronic approvals are on file.

ABSTRACT

As the widespread use of Artificial Intelligence (AI) Algorithms increases, the need to evaluate the fairness, or equitable decision-making, of such algorithms arises. It is crucial that an algorithm's decision-making recommendations do not reflect bias or discrimination as the algorithm's output is used to inform real-world outcomes and therefore impacts people's lives. This study aims to leverage previous research and propose new method to improve the fairness of an Artificial Intelligence model without sacrificing its performance. To that end, the study employs multiple historically used fairness metrics to build a Random Forest Model. The metrics are optimized, and their trends are analyzed to explore the balance needed to build a fair, equitable, and unbiased model that is still accurate and able to inform important decisions. This paper focuses on employing the Multi-Objective Ensemble Learning (MEL) method, as the algorithm considers both the model's performance and its fairness metrics.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

This thesis was made possible by multiple people and organizations who provided valuable assistance. I'd like to say thank you to several of them here:

To Dr. Abdullah Konak, for years of dedication and patience while conducting research with me, and his gift for teaching not only me, but many other students.

To Dr. Jeanne Marie Rose, for hours spent meticulously reviewing this document and giving me feedback. Her effort made not only this thesis, but also my overall writing style, inordinately better.

To Dr. Sandy Feinstein, whose honesty, leadership, and wit truly helped me to understand what honors work is and why it's important.

To Mom, whose listening ears, delicious meals, and endless supply of coffee kept me focused and joyful while I worked on this project.

To Dad, whose unceasing energy, direction, and passion inspired me to keep going when I thought I couldn't.

To Hannah, who always brought me back to reality when I got overwhelmed and always managed to keep me laughing when things got difficult.

To Abby, who reminded me that much can be fixed with a hug.

To Tyrus, whose determination, steadiness, and faith have unspeakably encouraged me.

To the Cohen and Hammel Families whose generous Cohen-Hammel Fellowship has made this project possible.

Chapter 1

Introduction

In recent years, Artificial Intelligence (AI) and Machine Learning (ML) algorithms or models have been increasingly used for automation of tasks in multiple industries, including banking, healthcare, legal services, and education (Beck et al., 1996). Although AI has been in use for decades, the rise of ML, a subfield of AI, has accelerated during the last decade (Allen, 2020). Machine Learning involves training an algorithm itself instead of following pre-programmed rules. This self-training, however, requires a large amount of data. As data was difficult to collect, ML models were historically costly.

With the rise of large datasets, ML models' practicality has increased, causing a significant rise in their popularity. Today, Machine Learning Artificial Intelligence is employed for various tasks, such as candidate screening (Liem et al., 2018), surveillance (Feldstein, 2019), and medical image analysis (Rajpurkar et al., 2022). According to a report by McKinsey, AI adoption globally is two and a half times what it was in 2017; the number of AI capabilities employed by businesses, including automation and computer vision, has doubled since 2018 (*The state of AI,* 2022). As Machine Learning usage increases in so many industries, concerns over AI's equity and fairness have arisen (John-Matthews et al., 2020).

Previous research has established shared understandings of fairness, bias, equality, and equity in the context of AI algorithmic processing. Fairness in decision making can generally be defined as the lack of favoritism towards individuals or groups for any of their inherited or learned traits (Mehrabi et al., 2021). Bias in the algorithmic decision-making process involves a

societally unacceptable disparity among demographic groups in an algorithm's decision-making (Barocas et al., 2017). There are three distinct kinds of bias. Preexisting bias stems from society, technological bias is the amplification of the preexisting bias by the model's internal processes, and emergent bias comes from the user's interaction with the system (Stoyanovich et al., 2022). Equality in decision making generally refers to the equal treatment, or an equal chance of an outcome, for everyone, no matter their background, while equity refers to giving extra assistance to those whose background gives them a distinct disadvantage (Minow, 2021).

There are several prominent incidents which raised questions about whether AI is fair in its treatment of individuals and groups, and whether it should be used for decision making. One such incident occurred in 2014 at Amazon Inc., where the human resources team began utilizing an AI-based tool to review job application resumes (Kodiyan, 2019). This software was adapted and heavily used by the company, but by 2015 the company noticed that for the more technical job applications, such as software engineers and data architects, the AI algorithm was showing strong bias towards male resumes. The algorithm would rank any resume that included "women's," such as "women's rugby" or "women's chess club," as being less viable for the position than resumes that did not include such keywords. After Amazon's engineers investigated the problem, they found that the dataset had been trained on a greater number of male resumes because of male dominance in those positions, prompting the algorithm to prefer resumes describing males.

Amazon addressed the model's discrimination toward females, rewriting the algorithms to be neutral to the two sexes. However, such bias has raised ethical questions in the machine learning community. If AI is used so heavily for something as important as hiring, how can it be ensured that the model does not discriminate?

Amazon's resume tool was not the only one that unintentionally produced biased decisions. In Broward County, Florida, an algorithm called COMPAS was being used to predict recidivism for defendants (Silberg & Manyika, 2019). It was found that the algorithm was making false predictions about the recidivism status for African American defendants twice as often as it did for their white counterparts. Similarly, US-based law enforcement has used facial recognition systems to determine the characteristics and propensities of people, such as potential to commit a crime or to be a terrorist. It was found that those algorithms have lower accuracy in people who are labeled female, black, or between the ages of 18 and 30 (Buolamwini & Gebru, 2018).

The current study's goal is to leverage previous research and build models which are optimized to improve the fairness of an AI model without sacrificing its performance. The study will use performance and fairness metrics chosen based on previous research. This step in the research is critical, as the testing completed in the study will only be as valid as the metrics which are used to evaluate the models. If the metrics do not give a complete picture of the model's overall performance or fairness, the research will be equally as incomplete and invalid.

In order to find a balance between fairness and performance, this study will create many models trained on a randomly selected subset of the same dataset. The fairness and performance metrics will be calculated for each model. The nondominated models, that is, the models which have better performance and fairness metrics than the other models, were chosen for prediction. The metrics will be reviewed, and with many trials, a relationship between the fairness and performance of a model can be extracted from data analysis. This study seeks to answer the following questions:

1. Is there a relationship between a model's performance and its fairness?

2. Is there a way to improve fairness without sacrificing predictive performance by combining the predictions of many models having different levels of fairness and performance?

Chapter 2

Background

Fairness, bias, and equity in AI have been evaluated historically using several different types of measurements, or metrics. Below is a list of the AI fairness metrics that were considered in optimizing the algorithm constructed for this paper. Although many metrics are defined below, only one is employed for the model's training constraints.

Demographic Parity

This metric can be defined as the likelihood that the positive outcome will be the same for the protected and unprotected group members as given in Eq (1). In the equations below, $\hat{Y}$ is the predictor representing an algorithm's decision, Y is the actual outcome the algorithm is attempting to approximate, and A is a protected attribute. This fairness metric has been used in the past to evaluate the effectiveness of preprocessing, in-processing, or post-processing training methods for algorithms (Ding et al., 2021).

$$P\left(\hat{Y} = 1 \middle| A = 0\right) = P(\hat{Y} = 1 | A = 1) \quad (1)$$

Equal Opportunity

A model is said to have equal opportunity when the "probability of a person in a positive class being assigned to a positive outcome [is] equal for both protected and unprotected" group members as defined in Eq (2) (Mehrabi et al., 2021). This fairness metric should be used only when the proportionate number of false positives among groups does not have an effect on overall fairness, as it uses the true positive rate, which does not consider false positives (Makhlouf et al., 2021). The true positive rate is the rate at which the algorithm predicts positive outcomes that are actually positive in the dataset. The false positives are outcomes which are deemed positive by the algorithm, but which are not actually positive in the dataset.

$$P\left(\hat{Y} = 1 \middle| Y = 1, A = 1\right) = P\left(\hat{Y} = 1 \middle| Y = 1, A = 0\right) \qquad 2$$

Equalized Odds

A model has equalized odds when "the probability of a person in the positive class being correctly assigned a positive outcome and the probability of a person in a negative class being incorrectly assigned a positive outcome [is] the same for the protected and unprotected group members," as shown below in Eq (3) (Mehrabi et al., 2021). The concept of equalized odds involves a stricter definition of fairness, as it requires that not only the True Positive Rate (TPR), but also the False Positive Rate (FPR) are equal between groups (Yogarajan et al., 2022). Equalized odds and equal opportunity can be used to determine whether a model is discriminating between protected and unprotected groups, such as different ethnicities or sexes.

$$P\left(\hat{Y} = 1 \middle| Y = x, A = 1\right) = P\left(\hat{Y} = 1 \middle| Y = x, A = 0\right) \; given \; x \in \{0,1\} \qquad 3$$

Some have utilized a combination of metrics to determine the fairness of a model, such as the demographic disparity, used to balance the performance and fairness metrics (Ghosh et al., 2022). Quantile Demographic Disparity (QDD) looks at the model's scores for the protected and unprotected groups in question within a predefined bin. Essentially the data is segmented into the equivalent of quantiles, then the disparity between groups is calculated. This approach improves upon current fairness metrics because the binning process allows insight on the trends at a more individual level. Such consideration of the individual can reveal further bias and improve the model's performance, which may not be achieved using other fairness metrics.

Previous Methods

To date, there have been attempts at assessing fairness and analytics and at balancing the fairness of a model with the model's performance, including its accuracy, precision, and other scores. These attempts have been sorted into categories in previous literature, and several of those categorizations are outlined below.

Pre-processing Methods

Much of the literature classifies fairness measures by whether the method is applied to the model before, during, or after the training process. The techniques that "transform a feature space into a representation that as a whole is independent of the sensitive attribute" are considered pre-processing adjustments (Barocas et al., 2017). This definition indicates that there is no correlation between the feature space as a whole and the sensitive attribute. Often this type of technique is applied on the dataset itself before a model is trained utilizing that dataset (Silberg & Manyika, 2019).

An example of a pre-processing technique is data augmentation, where the model is trained on slightly altered copies of the original dataset (Loureiro et al., 2022). Another example of a pre-processing technique is the removal of irrelevant and redundant features, or variables, through feature selection (Felix & Lee, 2019). This method allows the model to be trained only on features which have been deemed relevant and are independent of other training features. Interdependency or correlation between features can lead to multicollinearity in the model, which is the tendency of collinear relationships between features that are treated as independent. The presence of interdependency, and therefore multicollinearity, in a model reduces the accuracy of the model's predictions, and the removal of such interdependency is a pre-processing technique to improve fairness.

There are a few benefits to utilizing a pre-processing method on a dataset. First, these methods are more general because they are not specific to one model. Any number of different models can be trained on a dataset which has utilized pre-processing to decrease bias. Second, training a model on such a dataset can naturally reduce its discrimination (Zhang et al., 2022).

In-processing Methods

Unlike pre-processing techniques, which are applied to the dataset before the model is built, in-processing techniques are applied during the model's training and can directly incorporate fairness into the model's design (Wan et al., 2023). The concepts of fairness and mitigation of bias exist in the form of mathematical constraints, which are integrated into the model's creation (Silberg & Manyika, 2019). The use of in-processing techniques can therefore assist in solving the problem of bias amplification, or the increased prominence of bias already found in the model's data. Bansal (2022) discusses an in-processing technique that groups all of the protected attribute's information, in this case the individual's gender, into a two-part vector. This grouping allows one part of the vector to be used to access the gender information and another part to be used when omitting the gender information for training purposes. Both the grouping and other techniques directly take fairness into account, eliminating any potential bias amplification that may occur.

The benefits of using in-processing methods include the production of intrinsically fair models by directly taking fairness into account when training the model itself. In-processing methods can also be used on pre-trained models for fine-tuning (Wan et al., 2023). This capability allows for bias mitigation without the efforts which it would normally take to retrain the model.

Post-processing Methods

Post-processing refers to the family of techniques that promote fairness in an algorithm by making adjustments to the trained model. These adjustments typically occur after the model's training, making it more suitable for runtime environments as the training processes cannot be accessed (Lohia et al., 2019). The adjustments depend on the protected attribute and the added randomness to achieve independence. An example of a post-processing method is Equalized Odds Post-Processing (EOP), which aims to obtain an equal amount of false positive and false negative rates for the protected and unprotected groups (Lohia et al., 2019). Essentially, the final model's parameters are adjusted in order to make the model more fair. The benefit of utilizing a post-processing method is that it can be used on already deployed ML modeling instead of retraining the model with in-processing methods, which has environmental consequences (Petersen et al., 2021).

Software Toolkits

There are many software toolkits, or assortments of functions, which can be used to help detect or mitigate biases, which are being used today in order to create fair algorithms (Richardson & Gilbert, 2021). These toolkits are generally open access and accessible, and they are offered by several different companies, each of which emphasizes their own defining feature of fairness. For example, "Fairness Indicators" and the "What-If Toolkit" are widget-based software toolkits authored by Google (Richardson & Gilbert, 2021). They are both embedded in the Tensorflow package in Python, but users have the capacity to create custom functions outside

of Tensorflow. The “Fairness Indicators” toolkit focuses on multiclass and binary problems. It allows users to choose what fairness and performance metrics they would like to see on a provided visualization for up to two models. The “What-If Toolkit” focuses on binary and regression problems. It depicts the outcome of fairness transformations on a performance chart and shows the distribution of each feature with the corresponding summary statistics.

Another toolkit is “AI Fairness 360,” which can be used in Python and R. It calculates performance metrics and group and individual fairness metrics, and it offers bias mitigation algorithms during the pre-processing, processing, and post-processing stages of creating an AI algorithm (Bellamy et al., 2018). There are nine total bias mitigation algorithms over the three types of processing. AI Fairness 360 does not offer visualization capabilities, however.

Chapter 3

Methodology

Dataset Description

Several different papers in the field of machine learning have utilized the German Credit dataset. The German Credit dataset was used to build a Bayesian Network, a statistical model which aims to find relationships among variables, tailored to the German Credit dataset (Zonneveldt et al., 2010). That dataset was also used to explore accuracy and fairness in credit score modeling by building a credit score application (Kasmi, 2021).

The German Credit dataset used in this fairness and performance study describes a set of individuals who are deemed as good or bad credit risks, which are classified as Creditable and Non-creditable individuals. The widely available German Credit dataset utilized in this study was retrieved from the Eberly College of Science in the csv format. It was then downloaded and imported into an Integrated Developer Environment (IDE) called Google Colabs. The dataset also includes other attributes which describe those individuals, such as age, sex, marital status, account balance, the credit amount, the length of their current employment, and more. These attributes distinguish whether the individual is a good or bad credit risk. The dataset contains 20 such attributes, 7 numeric and 13 qualitative. It also includes 1000 cases, or 1000 individuals whose creditability and attributes are recorded. There are 300 cases of individuals classified as not creditable, or being a bad credit risk, and 700 cases of individuals who are creditable, or a good credit risk. For the study at hand, 560 of the good credit risk and 240 of the bad credit risk instances were randomly selected from the German Credit dataset for model training. The

remaining 140 good credit risk and 60 bad credit risk instances were used for testing the trained decision tree. Eighty percent of the total German Credit dataset was therefore used for training the models and for preliminary testing to determine dominated models. The remaining twenty percent is used for testing each model's performance and fairness.

By using the dataset in this way, the study at hand will have an equal split of the creditable and non-creditable instances for training and testing. In terms of the other variables in this train and test split, the feature selected to be protected by the fairness metrics was also important to determine. The fairness metrics in question seek to protect an attribute or feature where bias may occur. In the German Credit Dataset there are several potentially sensitive features that are analyzed to determine whether the attribute needs to be assessed for bias.

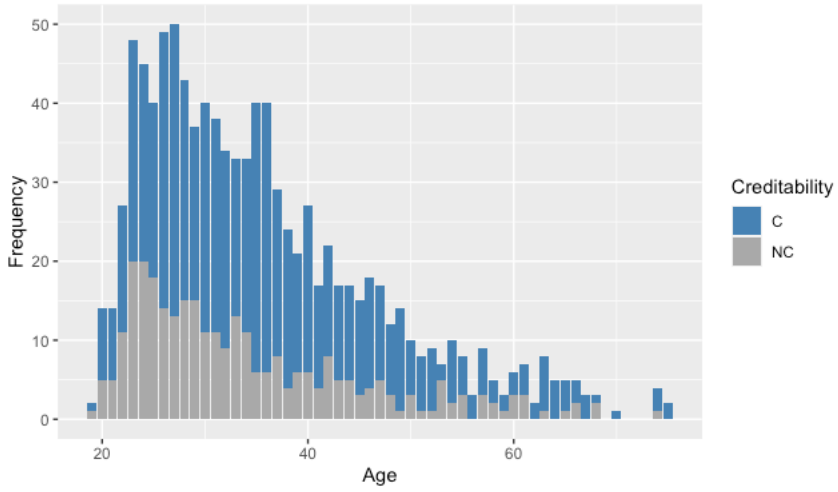

Figure 1. Creditability by Age

As seen Figure 1 above, the dataset is biased based on the age of the individual being evaluated for their credit risk. Those who are close to the age of 25 years old seem to have a higher chance of being classified as non-creditable, denoted in the graph as NC. The ratio of

creditable to non-creditable individuals appears to be around 50 percent at this age, even though there are 700 instances of creditable and 300 of non-creditable. Those individuals who are around the age of 35 years old appear to have a higher chance of being classified as creditable, or C in Figure 1, as the number of creditable individuals for that age appears far greater than the ones classified as non-creditable. This analysis indicates that the dataset, and therefore any model that may be trained on the dataset, may be biased based on an individual's age. As a result, this study must apply fairness metrics to prevent such bias from occurring.

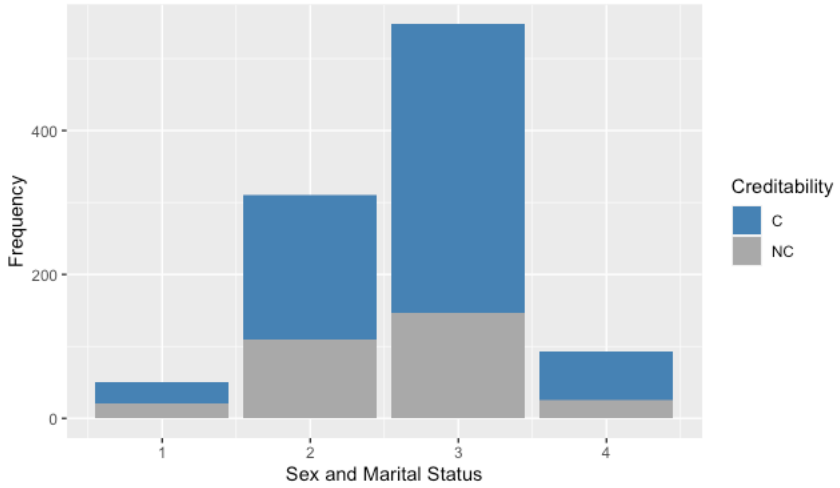

Figure 2. Creditability by Sex and Marital Status

As pictured in Figure 2 above, the Sex and Marital Status Category 1 represents a male who is divorced or living apart from his partner. Category 2 and 3 represent men who are single and married or widowed, respectively. Lastly, Category 4 represents females. There is an obvious disparity between the number of females and the number of males represented in the study. As a result of not having as much information on females applying for credit, the model may have a bias against females. The dataset also has a minimal number of males who are

divorced or living apart from their partners. The included data points on that population of individuals has a far higher ratio of individuals classified as non-creditable to those classified as creditable. This trend indicates that the dataset, and through that the model, may make biased classifications of divorced or separated males. It may classify them as non-creditable even if they are, in fact, good credit risks.

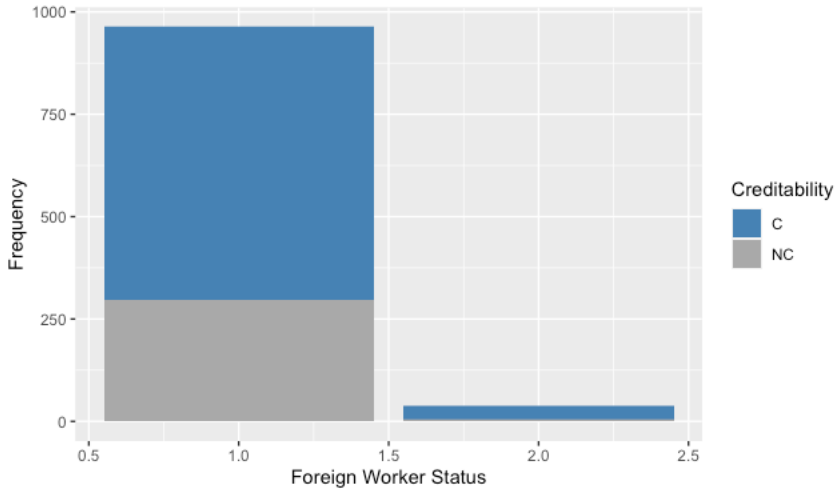

Figure 3. Creditability by Foreign Worker Status

Another potential sensitive attribute that the German Credit Dataset includes is the individual's status as a foreign worker. As seen in Figure 3 above, there are many non-foreign workers, denoted by 0.5 to 1.5 on Figure 3's scale, and there are relatively few foreign workers, denoted by 1.5 to 2.5 on Figure 3's scale. The lack of entries which include foreign worker status indicates that a model built to predict an individual's creditability based on foreign worker status would not be very accurate. As a result of the low number of entries, the foreign worker status variable will not be utilized in the training of this study's models.

Multi-Objective Ensemble Learning (MEL) Method

This study creates the Multi-Objective Ensemble Learning (MEL) method of optimizing a model's fairness and the performance by training many decision trees on the model, choosing the fairest and best performing models, then using those models together to make decisions. First, the decision trees were trained on the training data, which was 90% of the total German Credit Dataset, with equal parts good and bad credit outcomes as the test data. Then the models predicted the remaining 10% of the data, the test data on which the models were not trained. The fairness and performance metrics were assessed for each model based on how well they predicted the test data.

The decision trees that had better fairness or performance than other models, or whose fairness and performance were not worse than other models, were added to a list of "non-dominated" models. These models all "voted" on the test data. All their predictions were averaged, and the majority "vote" from the non-dominated models was recorded. The resulting list of predictions represents the final results, which were assessed for fairness and performance metrics.

In order to determine whether the results of this particular method were productive, the method's performance and fairness metrics described above must be compared to a decision tree trained on the same dataset without the non-dominated and voting methods. Such a decision tree was trained, and its performance and fairness metrics were determined to analyze how much better, if at all, the MEL method performed in comparison to the usual way of training a model.

Model Creation

The first part of the MEL method involves building a set *M* of random models on the German Credit Dataset. The pseudocode describing how set *M* is made is below.

```
M ={}
trials = 100
depths = {3, 4, 5, 6, 7, 8, 9, 10}
for each D ∈ depths:
    for i=1..trials:
        train, test = split of German Credit Data with a test set of 10% and train of 90%
        model =decision tree classifier with maximum depth D
        Split the train data into features (X_train) and target, Creditability (Y_train)
        Split the test data into features (X_test) and target, Creditability (Y_test)
        Fit model to X_train and Y_train values
        Y_pred = model predicts given X_test data
        Performance = f1 score given Y_pred and Y_test
        Fairness = demographic parity given Y_pred, Y_test and X_test's Sex feature
        Add model to set M
```

The pseudocode above creates one hundred models for each of the given max-depths from three to ten. The model's max-depth is the number of splits that it is allowed to make, and it is an attribute of the decision tree. The data is split using a random variable that is consistent throughout all the trials. Ninety percent of the data is used for training the model, and ten percent is used for the data testing. The model is fitted to the test data features and its target, which is creditability. The model predicts the Y_test values from the X_test data. The Y_pred and Y_test values are used to find the F1 score, and are also used in conjunction with the feature "Sex" stored in X_test to find the model's fairness. The model is then added to set *M* of models.

Non-Domination Method

The next part of MEL involves utilizing the method described in the pseudocode below to determine which models are non-dominated or belong to the Pareto front.

Given a set of models in set *M*,

```
dominated ={}
length = length of set M
for each modelA ∈ M:
    for indexB, modelB in length:
        if indexA ≠ indexB:
            Compare fairness of modelA located at indexA and modelB located at indexB
            Compare performance of modelA located at indexA and modelB located at indexB
            If the performance of modelA and fairness of modelA is better than those of model B,
            add model B to the dominated set.
for each model in dominated set:
        remove the model from set M
```

The pseudocode above achieves the goal of removing all dominated models from the dataset by comparing each model to each of the other models. indexA and indexB are both indexes of the set *M*, which contains all the models. If the indexes are not the same, and therefore indexA and indexB refer to different models, then the performance and fairness metrics are compared between the models. A model is classified as dominated if its performance and fairness metrics are worse than the other models'. In this case, the fairness, which is measured by demographic parity, would be a higher value, and the performance, measured by the F1 score, a lower value to be declared worse. The F1 score is a metric that measures both the precision and recall of a model using the harmonic mean. After the models were compared with one another, all the dominant models were added to the dominated set. Then, each model located in the dominated set was removed from the original set of models, leaving only non-dominated models.

The remaining models are the non-dominated Pareto front models. The figures below illustrate a conceptual visualization of the dominated and non-dominated models.

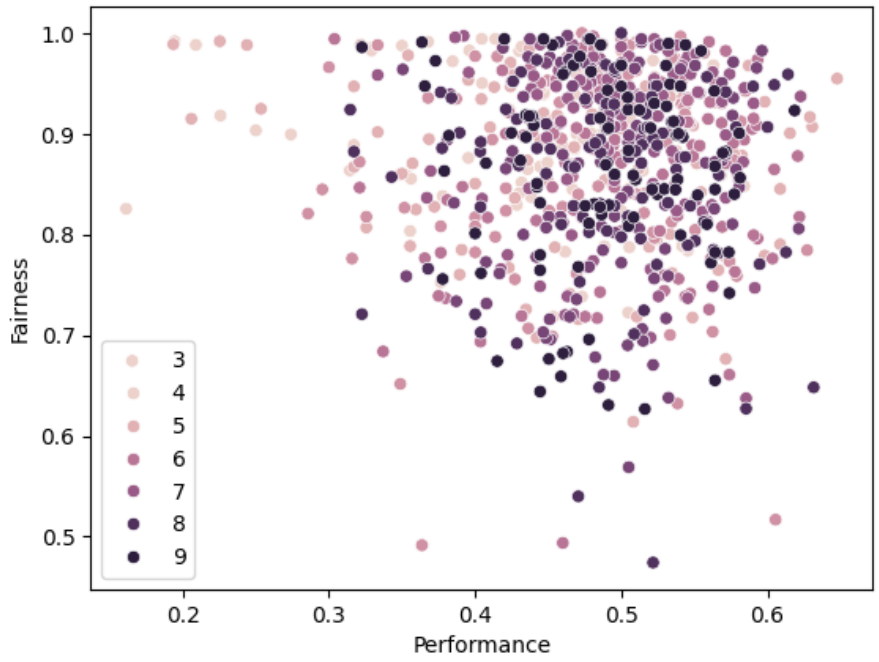

Figure 4. All Models' Fairness, Performance and Max Depth

Figure 4 depicts the full set M of models, plotted based on their fairness and performance measures. The fairness metric, demographic parity, was flipped for the graphing, showing one minus the fairness score. By graphing one minus the demographic parity metric, the values closer to one are more fair, making the visualization more straightforward to understand. The model's max-depth is also depicted through the color of the points plotted. The models were trained with max-depths of 3 to 10.

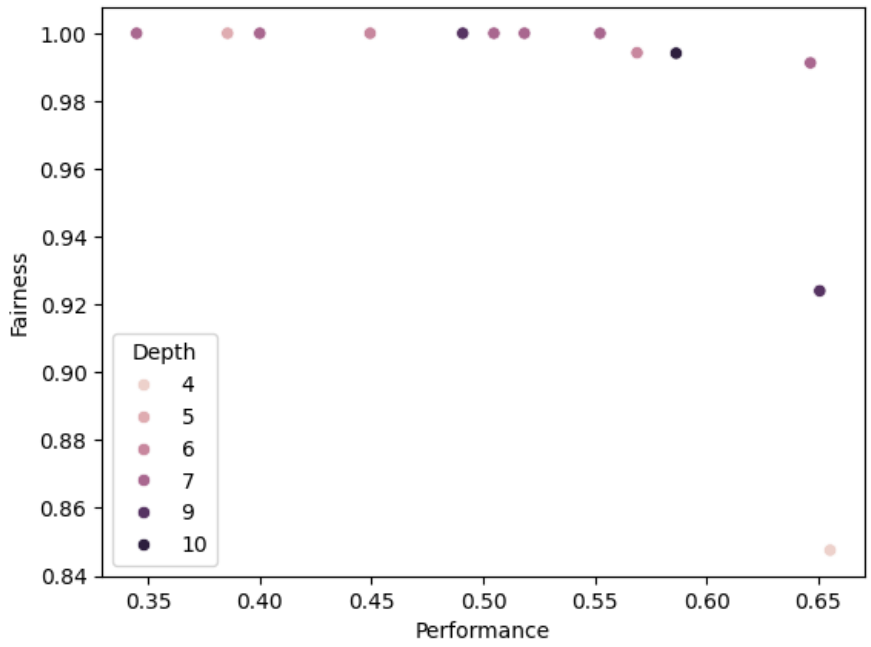

Figure 5. Nondominated Model's Fairness, Performance and Max Depth

Figure 5 shows the non-dominated models, their fairness, performance, and max-depth values. All the models that had lower performance and fairness scores were removed from the set of models used in this study. There is a Pareto Front in Figure 5 which represents a relationship between the fairness and performance of a model. When the performance is lower, the fairness is very high, sitting at almost 1 until the performance reaches .55. Then the performance continues to increase, but the fairness decreases slightly. This trend indicates that there is a tradeoff between the amount of fairness and the amount of performance that a model can attain. If the model is very fair, then it does not perform as well, and if the model performs at its best, it is not fair. This study uses all of these non-dominated models to make a prediction that will consider both fairness and performance by allowing all non-dominated models to vote on predictions.

Ensemble Learning

The study employs the method of ensemble learning, or using multiple models to make a prediction, in order to consider the model's fairness and performance. The non-dominated models chosen from the original set of models vote to determine the final prediction. The way in which the models vote is a simple majority. Each model makes a prediction for every point in the test set, and their collective average decision determines the final outcome. If the average for that point is equal to or above .5, the outcome is 1. If the outcome is below .5, the outcome is 0. As a result, both the models with high fairness and the models with high performance are used to make the final prediction, which may yield a more fair, higher performing model than those used traditionally.

One hundred models were trained, the non-dominated models were selected among those, and the non-dominated models were used to vote on the prediction outcomes for the testing data. This process was repeated one hundred times, and the averages of the fairness and performance metrics from the trials were stored.

Vanilla Model

In order to determine whether the MEL method results are productive, the study needed a standard, unmodified model for comparison purposes. Such a model is known as a vanilla model or algorithm. One hundred vanilla models were trained. The same training data was given to the model to train the decision trees, and the same testing data was used to test its fairness and performance. No modification of the algorithms or ensemble voting was used. The averages of

the hundred models' fairness and performance metrics were used as a comparison baseline for the MEL method results.

Chapter 4

Results

As detailed in the methodology, the German Credit Dataset was used to train both the vanilla models and the MEL models. The model's fairness and performance metrics are plotted below for the results of both the vanilla and MEL methods.

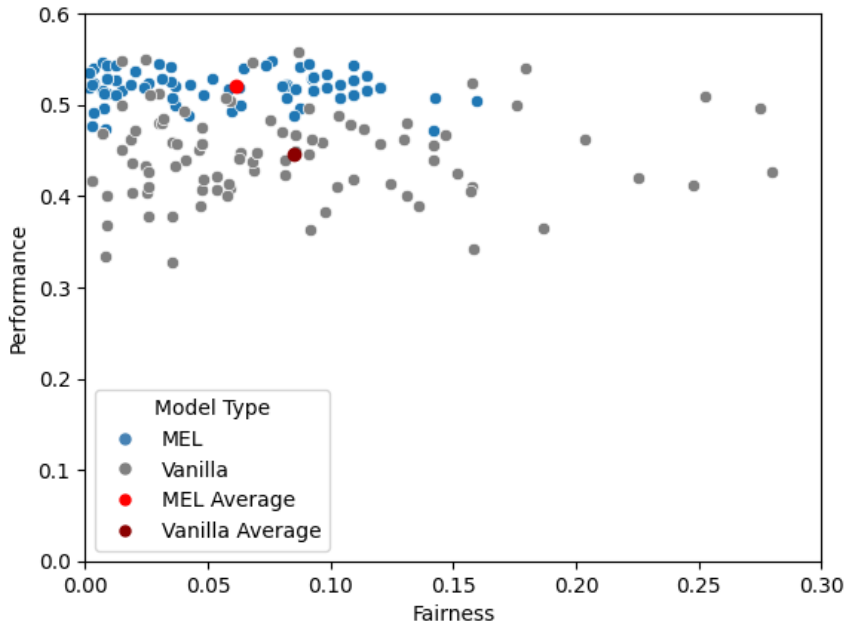

Figure 6. MEL and Vanilla Results

The fairness results for the vanilla model shown in the gray points of Figure 6 are evenly concentrated in their distribution between 0 and around .17 for the demographic parity difference. The more sparse distribution of points is from .17 to .27. The metric's ideal value, the most fair value, is zero. Note that the demographic parity difference is typically negative to represent bias against the protected group and positive to represent bias against the unprotected group. As this study is concerned with overall fairness, the metric's absolute value is utilized.

This means that whether the model is biased towards the protected or unprotected groups, the bias will be shown as positive. The fairness value for the vanilla model is considered good, as the average is .085. The median value is .063, and the mode is .0476.

The performance, or F1 score, ranges from around .35 to around .55. The F1 score has an ideal value of 1 and a worst possible value of 0. If the F1 value is less than .5, the performance is considered not to be good. A score from .5 to .8 is okay, from .8 to .9 is good, and above .9 is excellent. The score falls under the "not good" category, as the average F1 score is .45. The median is also .45, and the mode is .43. Note that the study's concern is not with creating a perfectly fair model but with creating a method to balance fairness and performance. The F1 score of a decision tree trained on the German Credit dataset being .45 means this type of model is not accurate enough to be used reliably for predicting the creditability of individuals. This value will serve as a baseline for comparison with the study's MEL method results, and its "not good" F1 score does not change the validity of that comparison.

The fairness results for the MEL models shown in the blue points in Figure 6 are concentrated in their distribution between 0 and .13. The average value of the fairness metric, or the demographic parity difference, is .062, which is closer to 0 than the score of the vanilla models, which averaged at .085. The median is .064, and the mode is 0.093. The MEL model's fairness is, therefore, an improvement compared to the vanilla model's fairness. This increase is one of 27.1%, which is a significant improvement in the model's fairness.

The MEL performance results ranged from .47 to .55, which would be ranked as "okay" on the F1 scale. The average performance or F1 score for the models is .52. The median is also .52, and the mode is .52. Compared with the vanilla model's score of .45, the MEL model's performance was a full 15.6% better than that of the vanilla model, which is a significant

increase in performance. Note that the averages of the MEL and vanilla algorithm's fairness and performance are plotted in red and dark red, respectively, in Figure 6. The MEL model's performance score is higher, and its fairness score is lower; therefore, the MEL model dominates the vanilla model, as a high F1 score and low demographic parity score indicate better performance and fairness.

A t-test was performed to determine whether there is, in fact, a significant difference between the means of the fairness and performance for the vanilla and MEL results. The t-test had a confidence interval of ninety-nine percent. The p-value for the performance was 2.2e-16, which is well under the limit of .01 used for the test. This result indicates that the true performance means for the vanilla and MEL methods is not equal to zero. Therefore, there is a statistically significant difference in the means for the performance means for the two methods.

The result for the fairness means t-test was a p-value of .0065, which is less than the .01 tolerance needed to reject the null hypothesis that the difference of the vanilla and MEL means is zero. This result indicates the difference between the vanilla and MEL means is not zero, and there is a statistically significant difference between the means of the two methods.

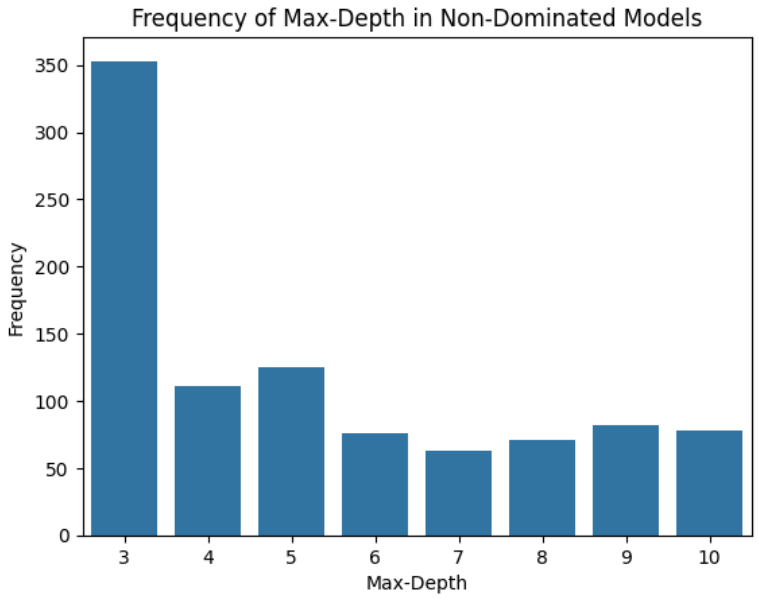

Figure 7. Frequency of Max-Depth in Non-Dominated Models

Figure 7 shows a breakdown of the frequencies at which the max-depth attribute occurs for the non-dominated models in the MEL method. The non-dominated models are chosen since their fairness and performance metrics are better than any of the other models. Assessing the max-depth attribute of only the non-dominated models, therefore, can show what trends are in the max-depth parameter that are related to high performance and fairness. Figure 7 depicts obvious and noticeable trend in the max-depth parameters for the non-dominated models: a lower max-depth, in this case a max-depth of 3, is by far the most frequently observed max-depth. The other max-depth values were accepted in the non-dominated model set, but there are far fewer observations of them.

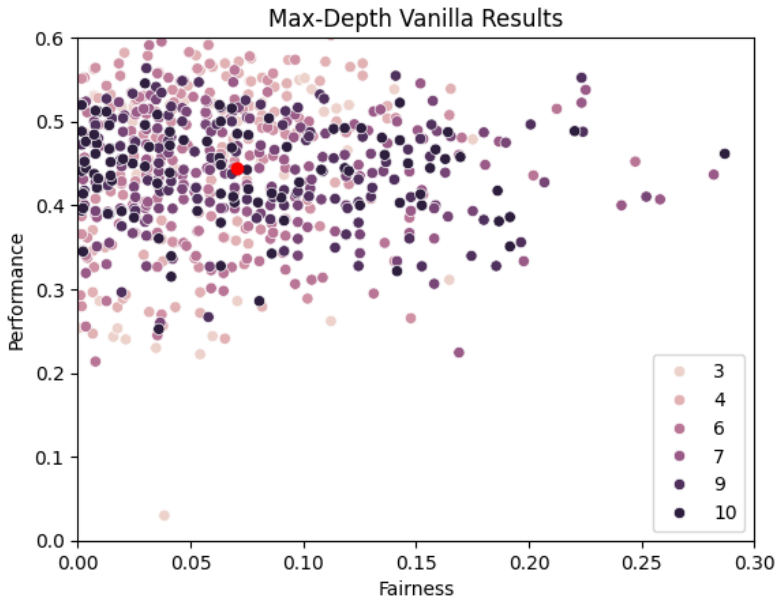

Figure 8. Vanilla Fairness, Performance, and Max-Depth Results

Eight hundred follow-up vanilla models were trained, and the max-depth attribute was altered in order to view any possible correlation between the max-depth attribute and the fairness and performance of a model. In Figure 8, the results of follow-up vanilla models are shown. The max-depth is distinguished in Figure 8 as differently colored points. The darker points have large max-depths, and the lighter-colored points have smaller max-depths. The red point is the average of the fairness and performance. According to the graph, it appears that there is a high concentration of models with a lower, better fairness score with a max depth of 3. This indicates that the observation of a max-depth of 3 from the non-dominated models in the MEL method may have to do with the correlation between the low max-depth and the better fairness score.

While it appears, however, that the points with a max-depth of 3 have a better fairness score, they also have a looser distribution on the performance scale. The models with a max-depth of 3 have some of the highest and some of the lowest observations of performance scores. This trend indicates that while the fairness for the models with a max-depth of 3 tends to be

better, the performance is more varied. Conversely, the larger max-depths tend to have a tighter distribution of performance, but a much wider distribution of fairness. Because the non-dominated models were comprised of models having better fairness and performance scores, the models with max-depth 3 may have been selected over models with other max-depths because of these better fairness scores.

Chapter 5

Conclusion

As discussed, this study introduced and utilized the MEL method to train AI algorithms. The MEL method first used the Pareto front optimization to choose models that are non-dominated in their fairness and performance scores. Those models' predictions were then used to cast a majority vote, which made up the MEL prediction. The MEL method boasted an increase of 27.1% in fairness and 15.6% in performance. These findings indicate that the MEL method has the capability of not only balancing the tradeoffs between having an equitable model and good performance, but also simultaneously increases both the model's performance and fairness at the same time.

The model's max-depth parameter was also investigated in this study in relation to the model's fairness and performance metrics. It was found that the max-depth attribute was correlated with the fairness and performance scores, as the max-depth of 3 seemed to have better fairness, while the higher max-depths seemed to have better performance but a worse fairness score.

The MEL method improved both fairness and performance while balancing the tradeoff between the two. The testing in this study made apparent the merits of employing this method, which can be helpful for researchers as well as those employing AI algorithms in industry, to ensure that their models not only perform well, but are fair and equitable. During the testing of this method the max-depth parameter for decision trees was manually changed for the trials which were used for the non-dominated method. The results of this testing determined a

correlation between the max-depth value and the fairness and performance metrics.

The results have proven that, on a small scale, when building decision trees, the MEL method improves fairness and performance by a significant margin. However, the results open the door for further research to be conducted which investigates the validity of the MEL method with different types of models, such as linear regression, neural networks, random forest models, or support vector machines. Such models may have greater results than building simple decision trees. The type of fairness and performance metrics can be changed in future testing as well. Although the MEL method increased the demographic parity fairness metric and the F1 performance metric, other metrics for fairness and performance may be affected differently when employing the MEL method. The correlation between the max-depth attribute and the fairness and performance of a model may also be cause for future study. The exact relationship between the variables can be further explored through other testing.

The practical application of the study results can be understood from a few previously mentioned examples. For instance, if Amazon had employed a similar method by testing for both performance and fairness in their resume reviewing AI-based tool, they would have seen a large bias against females in the model's results. This information would've allowed the company to amend the tool before utilizing it to filter through resumes. Similarly, if the COMPAS algorithm had been built to optimize both fairness and performance, the inaccuracy in recidivism results may have been addressed. The algorithm may have then made more fair decisions that were not biased against African Americans. The overall effect of using the MEL method is a more fair, well-performing model, which is better for the company employing the model and the people affected by the model's decisions.

Bibliography

Allen, G. (2020). Understanding AI technology. Joint Artificial Intelligence Center (JAIC) The Pentagon United States.

Bansal, R. (2022). A survey on bias and fairness in natural language processing. arXiv preprint arXiv:2204.09591.

Barocas, S., Hardt, M., & Narayanan, A. (2017). Fairness in machine learning. Nips tutorial, 1, 2017.

Beck, J., Stern, M., & Haugsjaa, E. (1996). Applications of AI in Education. *XRDS: Crossroads, The ACM Magazine for Students*, *3*(1), 11-15.

Bellamy, R. K., Dey, K., Hind, M., Hoffman, S. C., Houde, S., Kannan, K., ... & Zhang, Y. (2018). AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. arXiv preprint arXiv:1810.01943.

Buolamwini, J., & Gebru, T. (2018, January). Gender shades: Intersectional accuracy disparities in commercial gender classification. In Conference on fairness, accountability, and transparency (pp. 77-91). PMLR.

Ding, F., Hardt, M., Miller, J., & Schmidt, L. (2021). Retiring adult: New datasets for fair machine learning. Advances in neural information processing systems, 34, 6478-6490.

Feldstein, S. (2019). The global expansion of AI surveillance (Vol. 17). Washington, DC: Carnegie Endowment for International Peace.

Felix, E. A., & Lee, S. P. (2019). Systematic literature review of preprocessing techniques for imbalanced data. IET Software, 13(6), 479-496.

Ghosh, A., Shanbhag, A., & Wilson, C. (2022, July). Faircanary: Rapid continuous explainable fairness. In Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society (pp. 307-316).

John-Matthews, J. M., Cardon, D., & Balagué, C. (2022). From reality to world. A critical perspective on AI fairness. Journal of Business Ethics, 178(4), 945-959.

Kasmi, M. L. (2021). Machine learning fairness in finance: An application to credit scoring (Doctoral dissertation, Tilburg University).

Kodiyan, A. A. (2019). An overview of ethical issues in using AI systems in hiring with a case study of Amazon's AI based hiring tool. Researchgate Preprint, 1-19.

Liem, C. C., Langer, M., Demetriou, A., Hiemstra, A. M., Sukma Wicaksana, A., Born, M. P., & König, C. J. (2018). Psychology meets machine learning: Interdisciplinary perspectives on algorithmic job candidate screening. Explainable and interpretable models in computer vision and machine learning, 197-253.

Lohia, P. K., Ramamurthy, K. N., Bhide, M., Saha, D., Varshney, K. R., & Puri, R. (2019, May). Bias mitigation post-processing for individual and group fairness. In Icassp 2019-2019 IEEE international conference on acoustics, speech and signal processing (icassp) (pp. 2847-2851). IEEE.

Loureiro, T. P. P. R. B., Lisboa, F. V. N., Cruz, G. O. R., Peixoto, R. M., de Sousa Guimarães, G. A., dos Santos, L. L., ... & Nascimento, E. G. S. (2022). BIAS AND UNFAIRNESS IN MACHINE LEARNING MODELS: A SYSTEMATIC LITERATURE REVIEW.

Makhlouf, K., Zhioua, S., & Palamidessi, C. (2021). On the applicability of machine learning fairness notions. ACM SIGKDD Explorations Newsletter, 23(1), 14-23.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A survey on bias and fairness in machine learning. ACM computing surveys (CSUR), 54(6), 1-35.

Minow, M. (2021). Equality vs. equity. American Journal of Law and Equality, 1, 167-193.

Petersen, F., Mukherjee, D., Sun, Y., & Yurochkin, M. (2021). Post-processing for individual fairness. Advances in Neural Information Processing Systems, 34, 25944-25955.

Rajpurkar, P., Chen, E., Banerjee, O., & Topol, E. J. (2022). AI in health and medicine. Nature medicine, 28(1), 31-38.

Richardson, B., & Gilbert, J. E. (2021). A framework for fairness: a systematic review of existing fair AI solutions. arXiv preprint arXiv:2112.05700.

Silberg, J., & Manyika, J. (2019). Notes from the AI frontier: Tackling bias in AI (and in humans). McKinsey Global Institute, 1(6).

The state of AI in 2022—and a half decade in review. (2022). McKinsey & Company. https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai-in-2022-and-a-half-decade-in-review

Stoyanovich, J., Abiteboul, S., Howe, B., Jagadish, H. V., & Schelter, S. (2022). Responsible data management. Communications of the ACM, 65(6), 64-74.

Wan, M., Zha, D., Liu, N., & Zou, N. (2023). In-processing modeling techniques for machine learning fairness: A survey. ACM Transactions on Knowledge Discovery from Data, 17(3), 1-27.

Yogarajan, V., Dobbie, G., Leitch, S., Keegan, T. T., Bensemann, J., Witbrock, M., ... & Reith, D. (2022). Data and model bias in artificial intelligence for healthcare applications in New Zealand. Frontiers in Computer Science, 4, 1070493.

Zhang, Z., Wang, S., & Meng, G. (2022, July). A Review on Pre-processing Methods for Fairness in Machine Learning. In The International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (pp. 1185-1191). Cham: Springer International Publishing.

Zonneveldt, S., Korb, K., & Nicholson, A. (2010). Bayesian network classifiers for the German credit data. Bayesian-intelligence.com/publication

Appendix A

Nomenclature

AI	Artificial Intelligence
EOP	Equalized Odds Post-Processing
FPR	False Positive Rate, falsely rejecting the null hypothesis
IDE	Integrated Developer Environment, or a single graphical user interface which combined common developer tools
ML	Machine Learning, a self-trained subset of AI
TPR	True Positive Rate, accurately rejecting the null hypothesis
$\hat{Y}$	Predictor representing an algorithm's decision
Y	Actual outcome the algorithm is attempting to approximate
A	Protected attribute
M	Set of models
MEL	Multi-Objective Ensemble Learning

Appendix B

Python Code

```
#Import statements
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import tree
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split

# Demographic Parity (Fairness) Function
def calculate_statistical_parity(Y_actual, Y_predicted, X_categorical):
# Create a DataFrame with the actual labels, predicted labels, and categorical feature and
grouping the data
    test = pd.DataFrame({'Category': X_categorical, 'Actual': Y_actual, 'Predicted': Y_predicted})
    grouped_data = test.groupby(['Category', 'Predicted']).size().unstack(fill_value=0)
    statistical_parity_matrix = grouped_data.values
    # Calculate the predicted rates for positive and negative variables
    if grouped_data.shape == (2,2):
        positive_var = grouped_data.iloc[1, 1] / (grouped_data.iloc[1, 1] + grouped_data.iloc[1, 0])
        negative_var = grouped_data.iloc[0, 1] / (grouped_data.iloc[0, 1] + grouped_data.iloc[0, 0])
    else:
        return("Please ensure the data you have entered has two categories")
    # Calculate the difference in predicted rates
    predicted_difference = positive_var - negative_var
    # Returning the absolute value of difference in actual and predicted values
    return(abs(predicted_difference))

# Reading the German Credit Dataset csv file and changing the “Sex & Marital Status” to
represent sex (male and female)
df=pd.read_csv("https://sites.psu.edu/auk3/files/2023/11/german_credit.csv")
df.loc[df['Sex & Marital Status'] < 4, 'Sex & Marital Status'] = 0
df.loc[df['Sex & Marital Status'] == 4, 'Sex & Marital Status'] = 1

# Shuffling the dataframe and manually creating a test and train dataset with equal ratios of
Creditable and Non-creditable cases.
df_shuffle = df.sample(frac=1, random_state=42)
# Separate the data based on Creditability
df_0 = df_shuffle[df_shuffle['Creditability'] == 0]
```

```
df_1 = df_shuffle[df_shuffle['Creditability'] == 1]
# Split each group into training and testing sets, 80% training, 20% testing, equal amt of
creditability 1's and 0's
train_data_0, test_data_0 = np.split(df_0, [int(0.8 * len(df_0))])
train_data_1, test_data_1 = np.split(df_1, [int(0.8 * len(df_1))])
# Concatenate the training and testing sets for both groups
train_data = pd.concat([train_data_0, train_data_1])
test_data = pd.concat([test_data_0, test_data_1])
# Shuffle the combined data
train_data = train_data.sample(frac=1, random_state=42)
test_data = test_data.sample(frac=1, random_state=42)

# Separating the X (features) and Y (target) variables of the test_data dataset for later testing
Y=test_data[['Creditability']]
X=test_data[['Credit Amount','Account Balance','Duration of Credit (month)','Purpose','Value
Savings/Stocks','Length of current employment','Instalment per cent', 'Sex & Marital Status',
'Guarantors', 'Duration in Current address', 'Most valuable available asset', 'Age (years)',
'Concurrent Credits', 'Type of apartment', 'No of Credits at this Bank', 'Occupation', 'No of
dependents', 'Telephone']]


# Creating the plain vanilla algorithm used for comparison
vanilla_accuracy = []
vanilla_fairness = []
for _ in range(100):
  train, test = train_test_split(train_data, test_size=0.2, random_state=np.random.randint(1000))
  vanilla_model = tree.DecisionTreeClassifier()
  Y_train=train[['Creditability']]
  X_train=train[['Credit Amount','Account Balance','Duration of Credit (month)','Purpose','Value
Savings/Stocks','Length of current employment','Instalment per cent', 'Sex & Marital Status',
'Guarantors', 'Duration in Current address', 'Most valuable available asset', 'Age (years)',
'Concurrent Credits', 'Type of apartment', 'No of Credits at this Bank', 'Occupation', 'No of
dependents', 'Telephone']]
  Y_test=test[['Creditability']]
  X_test=test[['Credit Amount','Account Balance','Duration of Credit (month)','Purpose','Value
Savings/Stocks','Length of current employment','Instalment per cent', 'Sex & Marital Status',
'Guarantors', 'Duration in Current address', 'Most valuable available asset', 'Age (years)',
'Concurrent Credits', 'Type of apartment', 'No of Credits at this Bank', 'Occupation', 'No of
dependents', 'Telephone']]
  vanilla_model.fit(X_train, Y_train)
  Y_pred = vanilla_model.predict(X)
  vanilla_accuracy.append(f1_score(Y,Y_pred,pos_label=0))
  vanilla_fairness.append(calculate_statistical_parity(Y['Creditability'], Y_pred, X['Sex &
Marital Status']))
# Printing all results and sum of results
```

```
print(vanilla_accuracy)
print(vanilla_fairness)
print(sum(vanilla_accuracy) / len(vanilla_accuracy) )
print(sum(vanilla_fairness) / len(vanilla_fairness) )


# The vanilla algorithms using the max_depth
vanilla_D_accuracy = []
vanilla_D_fairness = []
vanilla_D_models = []
dp=range(3,11)
for dp_i in dp:
   for _ in range(100):
     train, test = train_test_split(train_data, test_size=0.2, random_state =
np.random.randint(1000))
     vanilla_model = tree.DecisionTreeClassifier(max_depth=dp_i)
     Y_train=train[['Creditability']]
     X_train=train[['Credit Amount','Account Balance','Duration of Credit
(month)','Purpose','Value Savings/Stocks','Length of current employment','Instalment per cent',
'Sex & Marital Status', 'Guarantors', 'Duration in Current address', 'Most valuable available asset',
'Age (years)', 'Concurrent Credits', 'Type of apartment', 'No of Credits at this Bank', 'Occupation',
'No of dependents', 'Telephone']]
     Y_test=test[['Creditability']]
     X_test=test[['Credit Amount','Account Balance','Duration of Credit (month)','Purpose','Value
Savings/Stocks','Length of current employment','Instalment per cent', 'Sex & Marital Status',
'Guarantors', 'Duration in Current address', 'Most valuable available asset', 'Age (years)',
'Concurrent Credits', 'Type of apartment', 'No of Credits at this Bank', 'Occupation', 'No of
dependents', 'Telephone']]
     vanilla_model.fit(X_train, Y_train)
     Y_pred = vanilla_model.predict(X)
     vanilla_D_accuracy.append(f1_score(Y,Y_pred,pos_label=0))
     vanilla_D_fairness.append(calculate_statistical_parity(Y['Creditability'], Y_pred, X['Sex &
Marital Status']))
     vanilla_D_models.append(vanilla_model)
# Printing all performance and fairness results
print(vanilla_D_accuracy)
print(vanilla_D_fairness)
print(sum(vanilla_D_accuracy) / len(vanilla_D_accuracy) )
print(sum(vanilla_D_fairness) / len(vanilla_D_fairness) )

# Creating Figure 8, Vanilla Fairness, Performance, and Max-Depth Results
# Pulling each depth out of its corresponding model
depth = [tree.tree_.max_depth for tree in vanilla_D_models]
sns.scatterplot(x=vanilla_D_fairness, y=vanilla_D_accuracy, hue=depth).set(title='Max-Depth
Vanilla Results', xlabel='Fairness', ylabel='Performance', xlim=(0,.30), ylim=(0,.6))
```

```
plt.scatter(x=sum(vanilla_D_fairness) / len(vanilla_D_fairness), y=sum(vanilla_D_accuracy) /
len(vanilla_D_accuracy), color='r')
plt.show()

# Code for Pareto Front example pictures with max-depth attribute
performances = {'model': [], 'accuracy': [], 'fairness': []}
dp=range(3,11)
for dp_i in dp:
   for _ in range(100):
      train, test = train_test_split(train_data, test_size=0.1,
random_state=np.random.randint(1000))
      model = tree.DecisionTreeClassifier(max_depth=dp_i)
      Y_train=train[['Creditability']]
      X_train=train[['Credit Amount','Account Balance','Duration of Credit
(month)','Purpose','Value Savings/Stocks','Length of current employment','Instalment per cent',
'Sex & Marital Status', 'Guarantors', 'Duration in Current address', 'Most valuable available asset',
'Age (years)', 'Concurrent Credits', 'Type of apartment', 'No of Credits at this Bank', 'Occupation',
'No of dependents', 'Telephone']]
      Y_test=test[['Creditability']]
      X_test=test[['Credit Amount','Account Balance','Duration of Credit
(month)','Purpose','Value Savings/Stocks','Length of current employment','Instalment per cent',
'Sex & Marital Status', 'Guarantors', 'Duration in Current address', 'Most valuable available asset',
'Age (years)', 'Concurrent Credits', 'Type of apartment', 'No of Credits at this Bank', 'Occupation',
'No of dependents', 'Telephone']]
      # Training the model
      model.fit(X_train, Y_train)

      # Make predictions on the test set
      Y_pred = model.predict(X_test)
      # Evaluate performance
      accuracy = f1_score(Y_test,Y_pred,pos_label=0)
      # Evaluate fairness - the closer to 0, the better
      fairness = calculate_statistical_parity(Y_test['Creditability'], Y_pred, X_test['Sex & Marital
Status'])
      performances['model'].append(model)
      performances['accuracy'].append(accuracy)
      performances['fairness'].append(fairness)
#Cleaning out any dominated models
dominated = []
for i, model_i in enumerate(performances['model']):
  for j, model_j in enumerate(performances['model']):
   if i != j:
      # Compare fairness of model j with fairness of model i
      parity_i = performances['fairness'][i]
      parity_j = performances['fairness'][j]
```

```
        # Compare performance of model j with performance of model i
        f1_i = performances['accuracy'][i]
        f1_j = performances['accuracy'][j]
        # Check if model j dominates model i
        if ((f1_i < f1_j) and (parity_i > parity_j)):
          dominated.append(i)
          break
result_models = [value for index, value in enumerate(performances['model']) if index not in
dominated]
result_accuracy = [value for index, value in enumerate(performances['accuracy']) if index not in
dominated]
result_fairness = [value for index, value in enumerate(performances['fairness']) if index not in
dominated]
# Printing the resulting non-dominated performance and fairness scores
print(result_accuracy)
print(result_fairness)

# Flipped fairness for the non-dominated models
flipped_fairness=np.array(result_fairness)
flipped_fairness= 1-flipped_fairness
# Flipping fairness for the entire dataset
flipped_fairness2=np.array(performances['fairness'])
flipped_fairness2= 1-flipped_fairness2

# Depth for the non-dominated models
depth = [tree.tree_.max_depth for tree in result_models]
# Depth for the entire dataset
depth2 = [tree.tree_.max_depth for tree in performances['model']]
# Dataframe for the non-dominated graph
data = pd.DataFrame(
    {'Depth': depth,
     'Fairness': flipped_fairness,
     'Performance': result_accuracy
    })
# Dataframe for the entire dataset graph
data2 = pd.DataFrame(
    {'Depth': depth2,
     'Fairness': flipped_fairness2,
     'Performance': performances['accuracy']
    })
# Creating the non-dominated graph: Figure 5
sns.scatterplot(data, x='Performance', y='Fairness', hue='Depth').set(title='Nondominated Models
Fairness, Performance and Max Depth')
plt.show()
# Creating the graph for the entire dataset graph: Figure 5
```

```
sns.scatterplot(data2, x='Performance', y='Fairness', hue='Depth').set(title='All Models Fairness, Performance and Max Depth')
plt.legend(labels=['3', '4', '5', '6', '7', '8', '9', '10'])
plt.show()

# MAIN TESTING: Creation of the ensemble decision trees
# Building the decision trees, using the max-depth attribute
performance_final = []
fairness_final = []
depth_final = []
for trial in range(100):
   performances = {'model': [], 'accuracy': [], 'fairness': []}
   dp=range(3,11)
   for dp_i in dp:
      for _ in range(100):
         train, test = train_test_split(train_data, test_size=0.2, random_state=np.random.randint(1000))
         model = tree.DecisionTreeClassifier(max_depth=dp_i)
         Y_train=train[['Creditability']]
         X_train=train[['Credit Amount','Account Balance','Duration of Credit (month)','Purpose','Value Savings/Stocks','Length of current employment','Instalment per cent', 'Sex & Marital Status', 'Guarantors', 'Duration in Current address', 'Most valuable available asset', 'Age (years)', 'Concurrent Credits', 'Type of apartment', 'No of Credits at this Bank', 'Occupation', 'No of dependents', 'Telephone']]
         Y_test=test[['Creditability']]
         X_test=test[['Credit Amount','Account Balance','Duration of Credit (month)','Purpose','Value Savings/Stocks','Length of current employment','Instalment per cent', 'Sex & Marital Status', 'Guarantors', 'Duration in Current address', 'Most valuable available asset', 'Age (years)', 'Concurrent Credits', 'Type of apartment', 'No of Credits at this Bank', 'Occupation', 'No of dependents', 'Telephone']]

         # Training the model
         model.fit(X_train, Y_train)

         # Make predictions on the test set
         Y_pred = model.predict(X_test)

         # Evaluate performance
         accuracy = f1_score(Y_test,Y_pred,pos_label=0)

         # Evaluate fairness - the closer to 0, the better

         fairness = calculate_statistical_parity(Y_test['Creditability'], Y_pred, X_test['Sex & Marital Status'])
```

```
            performances['model'].append(model)
            performances['accuracy'].append(accuracy)
            performances['fairness'].append(fairness)

    dominated = []

    for i, model_i in enumerate(performances['model']):
      for j, model_j in enumerate(performances['model']):
        if i != j:
            # Compare fairness of model j with fairness of model i
            parity_i = performances['fairness'][i]
            parity_j = performances['fairness'][j]

            # Compare performance of model j with the performance of model i
            f1_i = performances['accuracy'][i]
            f1_j = performances['accuracy'][j]

            # Check if model j dominates model i
            if ((f1_i < f1_j) and (parity_i > parity_j)):
              dominated.append(i)
              break

    result_models = [value for index, value in enumerate(performances['model']) if index not in
dominated]

    predicted=np.zeros((len(result_models),len(Y)))
    for i in range(0,len(result_models)):
       predicted[i,:]=result_models[i].predict(X)

    votes = np.sum(predicted, axis=0)
    combined_predictions = (votes > (predicted.shape[0] / 2)).astype(int)

    performance_final.append(f1_score(Y,combined_predictions,pos_label=0))
    fairness_final.append(calculate_statistical_parity(Y['Creditability'], combined_predictions,
X['Sex & Marital Status']))

    d = [tree.tree_.max_depth for tree in result_models]
    for i in d:
      depth_final.append(i)

    # Making a prediction using our model, then testing it against the correct creditability scoring
    answers = np.array(Y['Creditability'])
    all_output = pd.DataFrame()

    for i in range(len(result_models)):
```

```
    output = result_models[i].predict(X)
    all_output.insert(i, i, output)

  decision_output = []

  for i in range(len(all_output)):
    threshold_list = []
    for j in range(len(all_output.columns)):
      threshold_list.append(all_output[j][i])
      temp = sum(threshold_list)/len(all_output.columns)
    decision_output.append(temp)

  result_list = [0 if num < 0.5 else 1 for num in decision_output]

  per = f1_score(answers,result_list,pos_label=0)
  fai = calculate_statistical_parity(Y['Creditability'], result_list, X['Sex & Marital Status'])

print(performance_final)
print(fairness_final)

print(sum(performance_final)/len(performance_final))
print(sum(fairness_final)/len(fairness_final))


#Cleaning the data for graphing

data = {'PerfEnsemble': performance_final,
       'FairEnsemble': fairness_final,
       'PerfVanilla': vanilla_accuracy,
       'FairVanilla': vanilla_fairness}
final_df = pd.DataFrame(data)
print(final_df)


#Converting the data to a csv for further analysis using R

csv_data = final_df.to_csv('final_data.csv', index = True)


# Code for Figure 7
from collections import Counter
counter = Counter(depth_final)
# Create the seaborn bar plot
sns.barplot(x=list(counter.keys()), y=list(counter.values()))
plt.xlabel('Max-Depth')
```

```
plt.ylabel('Frequency')
plt.title('Frequency of Max-Depth in Non-Dominated Models')
plt.show()


#Code for Figure 6
sns.scatterplot(final_df, x='FairEnsemble', y='PerfEnsemble').set(xlabel='Fairness',
ylabel='Performance', xlim=(0,.30), ylim=(0,.6))
sns.scatterplot(final_df, x='FairVanilla', y='PerfVanilla', color='grey').set(title='MEL and Vanilla
Results', xlabel='Fairness', ylabel='Performance', xlim=(0,.30), ylim=(0,.6))
legend_labels = ['MEL', 'Vanilla', 'MEL Average', 'Vanilla Average']
legend_colors = ['steelblue', 'grey', 'red', 'darkred']
handles = [plt.Line2D([], [], marker='o', linestyle='None', markersize=5, markerfacecolor=color,
markeredgecolor=color) for color in legend_colors]
ax = plt.gca()
ax.legend(handles=handles, labels=legend_labels, title='Model Type')
plt.scatter(x=sum(final_df['FairEnsemble'])/len(final_df['FairEnsemble']),
y=sum(final_df['PerfEnsemble'])/len(final_df['PerfEnsemble']), color='red')
plt.scatter(x=sum(final_df['FairVanilla'])/len(final_df['FairVanilla']),
y=sum(final_df['PerfVanilla'])/len(final_df['PerfVanilla']), color='darkred')
plt.show()
```

Appendix C

R Code

```
# Importing the German Credit Dataset using a csv file
data = read.csv("german_credit.csv", header=TRUE)
attach(data)

# Loading a r library used to create plots
library(ggplot2)

# Making a dataframe from the csv file for graphing purposes
df = data.frame(data)
attach(df)

# Rewriting the Creditability part of the dataframe so it is C and NC instead of 1 and 0
df$Creditability <- as.factor(ifelse(df$Creditability != '0', 'C', 'NC'))

# Creating Figure 1 Creditability By Age using a Layered GGPlot
ggplot(df, aes(fill=Creditability, x=Age..years., y=frequency(Age..years.))) +
  geom_bar(position='stack', stat='identity') +
  labs(x='Age', y='Frequency', title='Creditability by Age') +
  scale_fill_manual(values = c("C" = "steelblue", "NC" = "darkgray")) +
  theme(plot.title = element_text(hjust=0.5, size=20, face='bold'))

# Creating Figure 2, Creditability by Sex and Marital Status as a layered GGPlot
ggplot(df, aes(fill=Creditability, x=Sex...Marital.Status, y=frequency(Sex...Marital.Status))) +
  geom_bar(position='stack', stat='identity') +
  labs(x='Sex and Marital Status', y='Frequency', title='Creditability by Sex and Marital Status') +
  scale_fill_manual(values = c("C" = "steelblue", "NC" = "darkgray")) +
  theme(plot.title = element_text(hjust=0.5, size=20, face='bold'))

# Creating Figure 3, Creditability by Foreign Worker as a layered GGPlot
ggplot(df, aes(fill=Creditability, x=Foreign.Worker, y=frequency(Foreign.Worker))) +
  geom_bar(position='stack', stat='identity') +
  labs(x='Foreign Worker Status', y='Frequency', title='Creditability by Foreign Worker') +
  scale_fill_manual(values = c("C" = "steelblue", "NC" = "darkgray")) +
  theme(plot.title = element_text(hjust=0.5, size=20, face='bold'))


# Importing python model results (vanilla and ensemble fairness and performance) for analysis
data = read.csv("final_data.csv", header=TRUE)
attach(data)
```

```
# Displaying the Median and Mean values
summary(data)

# Running a T-test on the Ensemble and Vanilla mean performance values
t.test(PerfEnsemble, PerfVanilla,
       mu = 0, paired = FALSE, var.equal = TRUE,
       conf.level = 0.99)

# Running a T-test on the Ensemble and Vanilla mean fairness values
t.test(FairEnsemble, FairVanilla,
       mu = 0, paired = FALSE, var.equal = TRUE,
       conf.level = 0.99)

# Defining a function to find the mode of the values
find_mode <- function(x) {
  u <- unique(x)
  tab <- tabulate(match(x, u))
  u[tab == max(tab)]
}

# Using the function to find the mode of the fairness and performance values
find_mode(PerfEnsemble)
find_mode(PerfVanilla)
find_mode(FairEnsemble)
find_mode(FairVanilla)
```