A study of Fairness in Machine Learning in the presence of Missing Values

Aeysha Aziz Bhatti

Thesis presented in the partial fulfilment
of the requirement for the degree of
MCom (Statistics)
at the University of Stellenbosch

Supervisor: Dr. T. Sandrock

Degree of confidentiality: A

March 2023

PLAGIARISM DECLARATION

1. Plagiarism is the use of ideas, material and other intellectual property of another's work and to present it as my own.
2. I agree that plagiarism is a punishable offence because it constitutes theft.
3. Accordingly, all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.
4. I also understand that direct translations are plagiarism.
5. I declare that the work contained in this thesis, except otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this thesis or another thesis.

Student number	Signature
A. A Bhatti	9 November 2022
Initials and surname	Date

ACKNOWLEDGEMENTS

I would like to thank Stellenbosch University and the Department of Statistics and Actuarial Science for giving me this opportunity to complete an MCom in Statistics. Next, I would like to thank my supervisor Dr. Sandrock for providing guidance along the way at different stages of the thesis.

I would like to thank my dear husband for his unconditional love and support of my endeavours. I would also like to thank my mother for her support during the difficult times.

Last but not least, I want to thank Cathy O'Neil for giving me the inspiration to pursue the topic of fairness in machine learning.

ABSTRACT

Fairness of Machine Learning algorithms is a topic that is receiving increasing attention, as more and more algorithms permeate the day to day aspects of our lives. One way in which bias can manifest in a data source is through missing values. If data are missing, these data are often assumed to be missing completely randomly, but usually this is not the case. In reality, the propensity of data being missing is often tied to socio-economic status or demographic characteristics of individuals. There is very limited research into how missing values and missing value handling methods can impact the fairness of an algorithm. In this research, we conduct a systematic study starting from the foundational questions of how the data are missing, how the missing data are dealt with and how this impacts fairness, based on the outcome of a few different types of machine learning algorithms. Most researchers, when dealing with missing data, either apply listwise deletion or tend to use the simpler methods of imputation versus the more complex ones. We study the impact of these simpler methods on the fairness of algorithms. Our results show that the missing data mechanism and missing data handling procedure can impact the fairness of an algorithm, and that under certain conditions the simpler imputation methods can sometimes be beneficial in decreasing discrimination.

Keywords: fairness in machine learning, missing values, fairness metrics, imputation, algorithmic bias

OPSOMMING

Die regverdigheid van masjienleeralgoritmes is ’n onderwerp wat toenemend aandag geniet, soos al hoe meer algoritmes elke aspek van ons alledaagse lewens deurdring. Een manier waarop sydigheid in ’n databron kan manifesteer is deur ontbrekende waardes. Indien daar ontbrekende data is, word daar dikwels aanvaar dat die data op ’n algeheel ewekansige manier ontbrekend is, maar dit is gewoonlik nie die geval nie. In werklikheid is die geneigdheid vir die afwesigheid van data dikwels verwant aan sosio-ekonomiese status of demografiese eienskappe van individue. Daar is baie beperkte navorsing oor hoe ontbrekende waardes en die hantering daarvan die regverdigheid van algoritmes kan beïnvloed. In hierdie navorsing voer ons ’n sistematiese studie uit, met die basiese vrae as beginpunt, soos op watter manier die data ontbrekend is, hoe die ontbrekende waardes hanteer word en hoe dit regverdigheid beïnvloed, gebaseer op die uitkoms van ’n paar verskillende masjienleeralgoritmes. Meeste navorsers gebruik skrappingsmetodes of eenvoudige imputasiemetodes eerder as meer komplekse metodes wanneer hulle met ontbrekende waardes gekonfronteer word. Ons ondersoek die impak van hierdie eenvoudiger metodes op die regverdigheid van algoritmes. Ons resultate toon dat die onderliggende ontbrekende waarde meganisme en die prosedure vir die hantering van ontbrekende waardes die regverdigheid van ’n algoritme kan beïnvloed, en dat onder sekere kondisies die eenvoudiger imputasiemetodes soms kan help om diskriminasie te verminder.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS AND/OR ACRONYMS

COMPAS	Correctional Offender Management Profiling for Alternative Sanctions
EOD	Equalized Odds difference
FPR	False Positive rate
k-NN	K-nearest neighbours
LD	Listwise Deletion
LR	Logistic Regression
MAR	Missing at Random
MCAR	Missing Completely at Random
MCC	Matthew's Correlation Coefficient
MDM	Missing Data Mechanism
ML	Machine Learning
MNAR	Missing not at Random
NPV	Negative Predictive value
PPV	Positive Predictive value
RF	Random Forest
SPD	Statistical Parity difference
SVM	Support Vector Machine
TPR	True Positive rate

CHAPTER 1

INTRODUCTION

'Big Data processes codify the past. They do not invent the future. Doing that requires moral imagination, and that's something only humans can provide. We have to explicitly embed better values into our algorithms, creating Big Data models that follow our ethical lead. Sometimes that will mean putting fairness ahead of profit.' - Cathy O'Neil, Weapons of Math Destruction (O'Neil, 2017).

The pervasiveness of machine learning algorithms in our everyday lives is undeniable. The breadth of examples that can be given in this regard is substantial: these algorithms are used to award or deny bank loans, manage admission into educational institutions, considerably speed up the process of hiring and even aid law enforcement to decide the severity of sentencing of criminals (Buolamwini and Gebru, 2018; Žliobaitė, 2017; Dunkelau and Leuschel, 2019). Some of the more seemingly innocuous examples include algorithms making movie recommendations, dating websites recommending users a suitable partner choice and retail websites with shopping recommendations (Mehrabi *et al.*, 2021; Barocas *et al.*, 2017; Ntoutsi *et al.*, 2020).

There are many worthwhile reasons for deferring some decisions from humans to computers: they are faster, they can deal with much larger amounts of information efficiently and accurately, they do not get lazy or bored and overall they reduce human error (Dunkelau and Leuschel, 2019; Ntoutsi *et al.*, 2020; Mehrabi *et al.*, 2021). Despite these advantages, there are many examples of where these algorithms have acted far from justly, with respect to certain sensitive or protected attributes such as race, gender or nationality. As examples: the Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) system, which is used by courts in the United States to assess the risk of re-offence, gave higher risk values for black offenders (lower risk values for white offenders) than their actual risk, facial recognition software which has the lowest accuracy on females who are darker skinned, and the online advertisement platform, Google Ads which showed considerably fewer advertisements for high paid jobs to women than to men (Buolamwini and Gebru, 2018; Ntoutsi *et al.*, 2020; Mehrabi *et al.*, 2021).

One of the main reasons for the biased behaviour of algorithms is the quality of data that they are trained on. Historic human biases for or against certain groups are inherent in many data sources. Machine learning algorithms, which are designed to detect and learn patterns in data, use and even amplify these patterns in making decisions.

An important way in which bias manifests in a data source is through the absence of certain fields or inputs. Missing data refers to those instances of data which contain fields which have not been captured or have been lost or deleted (Enders, 2022). If data are missing, these data are often assumed to be missing completely randomly, but usually this is not the case, and propensity of data being missing is often tied to socio-economic status or demographic characteristics of individuals. For convenience sake but to their detriment, the way that data are missing is usually not taken into account before proceeding with the treatment of missing data (Fernando *et al.*, 2021).

In practice, missing data is often dealt with in one of two ways before any statistical analysis: these are listwise deletion (LD), which refers to complete removal of that instance of data which contains the missing fields, or imputation, where at least one value is presented as a placeholder for the field where there is a missing value. Surprisingly, there is very limited research on fairness of algorithms in the context of the above missing data considerations, when it is clear that missing data and bias of an algorithm are related (Fernando *et al.*, 2021). For example, if listwise deletion is implemented and there are many more missing fields for one group over another, the algorithm might give more accurate predictions for the group with more observations and perform comparatively worse for the other group. When applied in a real world context, these biased decisions would have a tangible impact on an individual.

The aim of this research is to study the effects of missing values on the fairness of an algorithm. Fairness (or bias) of algorithms in machine learning is quantified by statistical definitions of so called 'fairness metrics', which are based on sub-setting the outcomes of an algorithm in various ways to allow the difference in treatment of one group over another to be measured. Existing literature combining the research of fairness of algorithms and missing data is limited to a handful of studies (Fernando *et al.*, 2021; Zhang and Long, 2021*b*; Wang and Singh, 2021; Zhang and Long, 2021*a*).

A systematic study starting from the foundational questions of how the data are missing, how the missing data are dealt with and how this impacts the fairness metrics, calculated on the outcome of a few different types of machine learning algorithms, does not exist. Most researchers when dealing with missing data either listwise delete or tend to use the simpler methods of imputation versus the more complex ones (Eekhout *et al.*, 2012). The impact of these methods on the fairness of an algorithm has not been studied. In this research, we will start from this point.

To this end, the remaining chapters are organised as follows: In Chapter 2 we cover the relevant background material for fairness in machine learning. In Chapter 3, we go over the relevant background on the subject of missing data. In Chapter 4, we present a literature review of the limited work which combines the two fields of fairness and missing values, and where relevant comparing the work which has been done to the work we will be undertaking. In Chapter 5, we cover the details of our experimental process. In Chapter 6 we present the results of our experiments. We end the thesis with Chapter 7 which covers our conclusions, recommendations and directions for future work.

CHAPTER 2

BACKGROUND: FAIRNESS IN MACHINE LEARNING

2.1 INTRODUCTION

The topic of fairness in machine learning is receiving increasing attention, as machine learning (ML) algorithms continue to permeate all spheres of life, including banking, education, hiring and health. With respect to certain 'sensitive' or 'protected' attributes, such as gender, race, religion and nationality, these algorithms often display discriminatory behaviour. Some of the more high profile examples include: commercial, image-based gender classifiers by Microsoft and IBM all increasingly misclassify when the input individual is dark-skinned or female, with an error rate for dark-skinned females significantly higher than that for light-skinned males; the US COMPAS system for predicting re-offence risk giving higher risk values for black offenders (lower risk values for white offenders) than their actual risk, and Google's Ads tool for targeted advertising showing significantly fewer ads for high paid jobs to women than to men (Buolamwini and Gebru, 2018; Dunkelau and Leuschel, 2019; Ntoutsi *et al.*, 2020). Hence the decisions made by ML algorithms have a tangible impact on many people's lives, and so it is of utmost importance to audit these algorithms at different stages of their development.

This is neatly captured in Barocas *et al.* (2017) (Figure 2.1) depicting the 'Machine learning loop'. It shows the different stages of development and deployment of an ML algorithm where unfairness could come in. For example, in the measurement stage, data for some individuals may be captured incorrectly or lost, leading to a biased data set. Data that is non-representative of the underlying population will then lead to biased models, which output inaccurate decisions for certain individuals. This event in an individual's life then goes on to become part of the state of the world, which again becomes data that is collected. In this simplified but accurate depiction, it is easy to see how disadvantaged groups could continue to remain disadvantaged.

Despite the potential drawbacks, the merits of decision making by algorithms are manyfold, including faster decision making and reduction in human error. For instance, a company could speed up its hiring process by algorithmically filtering through hundreds of applications, leaving a more manageable amount for human review (Dunkelau and Leuschel, 2019). Hence to completely dis-

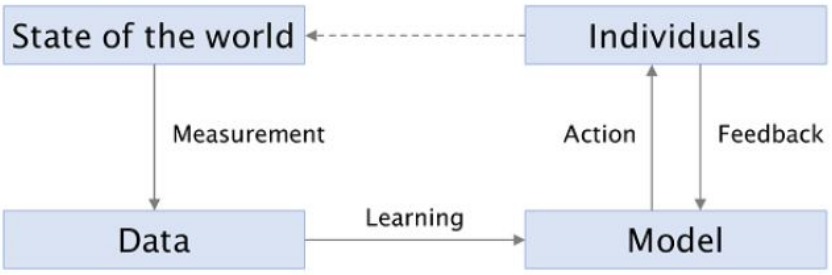

Figure 2.1: The machine learning loop (Barocas *et al.*, 2017, p. 15)

card these algorithms would be short sighted, and so there is a growing body of research into bias detection metrics and bias mitigation procedures for ML algorithms.

2.2 NOTATION

Without loss of generality we assume a binary classification situation. Let Y be the binary classification label taking values in $\{0, 1\}$ where 1 is the positive outcome and 0 is the negative outcome. Considering the case of paying back a loan, an outcome of 1 would mean the loan was paid back in full.

S is the binary sensitive or protected attribute where 1 is the privileged group and 0 is the unprivileged group. If S is the variable race, 1 could be assumed to correspond to white, and 0 to non-white, for example. Straightforward extensions exist for more than 2 classes or groups.

X contains the predictor variables which can be numerical or categorical. X may or may not contain S. Along with Y they form the training data for a classifier. $\hat{Y}$ are the predicted outcomes from a classifier, which can be class predictions in $\{0, 1\}$ or probability scores in $[0, 1]$.

2.3 FAIRNESS THROUGH UNAWARENESS

One might expect that removing the sensitive variable from the model i.e., not using S for training the classifier, might result in a fair classifier. As we expand next, removing the protected attribute from the decision process is insufficient, as other predictors can serve as proxies for the protected

attribute, and the classifier will use these proxy variables to indirectly discriminate. Barocas *et al.* (2017) explain that several predictors that are slightly predictive of the sensitive variable can be used to build high accuracy classifiers for that variable. This is demonstrated in Figure 2.2. The left graph displays a predictor's distribution separated by groups. There is only a slight difference in the two distributions. The right plot shows that if we have many such predictors, we can predict group membership with high accuracy.

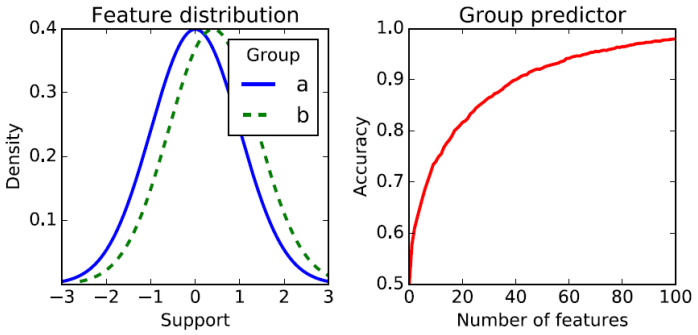

Figure 2.2: Sensitive attribute proxies effect (Barocas *et al.*, 2017, p. 44)

Another enlightening example is given in Calders and Verwer (2010), demonstrated on the Adult income data set (this data set is described in detail in Chapter 5). Calders and Verwer (2010) use a Naive Bayes classifier to predict the outcome, which is to decide whether the income of an individual is high or low. They consider the sensitive variable `sex`, and Figure 2.3 shows the contingency table of co-occurance counts in the data:

	Male	Female
High income	3256	590
Low income	7604	4831

Figure 2.3: Distribution of outcome by gender for the Adult data set (Calders and Verwer, 2010)

As we can see, 30% of all males and only about 11% of all females have a high income, so historically the data has been biased towards males. Then they apply a Naive Bayes classifier to this data, this pattern is recognised by the algorithm and also amplified, as can be seen from the distribution of the predictions in Figure 2.4.

	Male	Female
High income	4559	422
Low income	6301	4999

Figure 2.4: Distribution of predictions when including the sensitive variable in the model (Calders and Verwer, 2010)

The classifier predicts about 42% of all males having a high income, and only 8% of all females. Then they remove the sensitive attribute sex from the predictors in the model, the distribution is as shown in Figure 2.5.

	Male	Female
High income	4134	567
Low income	6726	4854

Figure 2.5: Distribution of predictions excluding the sensitive variable from the model (Calders and Verwer, 2010)

Still, the males are the group favoured by the classifier, more so than even in the training data itself: 38% males against 10% females. In essence, the model has used the predictors that correlate with the sex attribute to indirectly discriminate.

2.4 FAIRNESS METRICS

Broadly, there are two main categories of fairness metrics (Friedler *et al.*, 2019; Žliobaitė, 2017; Verma and Rubin, 2018; Mehrabi *et al.*, 2021; Dunkelau and Leuschel, 2019):

- Group fairness metrics: They aim to ensure non-discrimination across protected groups.
- Individual fairness metrics: They aim to give similar classification to similar individuals.

In this study we only consider the group metrics and how they are influenced by missing values. Details of individual fairness can be found in Dwork *et al.* (2012).

2.4.1 Group metrics

The measures can be divided into two broad categories: those based on predicted outcome (section 2.4.1.1) and those based on both predicted and actual outcome (section 2.4.1.2) (Friedler *et al.*, 2019; Žliobaitė, 2017; Verma and Rubin, 2018; Mehrabi *et al.*, 2021; Dunkelau and Leuschel, 2019)

2.4.1.1 Measures based on predicted outcome

- **Demographic parity** (Barocas *et al.*, 2017; Dunkelau and Leuschel, 2019): Also known as statistical parity, this measure requires the probability for an individual to be assigned the favourable outcome to be equal across the privileged and unprivileged groups

$$P(\hat{Y} = 1 \mid S = 1) = P(\hat{Y} = 1 \mid S = 0)$$

  Hence this criterion requires the prediction to be statistically independent of the sensitive characteristic. Note that when base rates $P(Y \mid S)$ differ across the two groups, statistical parity rules out the perfect predictor.

- **Disparate impact** (Feldman *et al.*, 2015): Considering the ratio between the groups

$$\frac{P(\hat{Y} = 1 \mid S = 0)}{P(\hat{Y} = 1 \mid S = 1)}$$

  we can require that the rates between the two groups not be different by more than a certain percentage (Friedler *et al.*, 2019).

- **Conditional statistical parity** (Dunkelau and Leuschel, 2019): Instead of considering all differences in treatment between the two groups to be discriminatory, this measure considers some differences to be acceptable. It tries to quantify how much of the difference between the groups can be explained by other attributes of individuals recorded in X, and only the remaining differences are said to be discriminatory. For example, part of the difference in acceptance rates for locals and immigrants may be explained by differences in education levels. Only the remaining unexplained difference should be considered as discrimination

(Žliobaitė, 2017). To this end, let $L \subset X$ be a set of legitimate factors. Then we require

$$P(\hat{Y} = 1 \mid S = 1, L = l) = P(\hat{Y} = 1 \mid S = 0, L = l)$$

There are limitations of demographic parity. It can easily be satisfied by a randomised classifier on the disadvantaged group. Given a trained model, one can measure the acceptance rate of the advantaged group, then simply assign the favourable outcome to disadvantaged individuals at random (or carelessly), with respect to the same rate. Even though the acceptance rates in both groups are identical, it is likely that more unqualified applicants will be selected from the disadvantaged group. As a result, it will appear that members of the unprivileged group performed worse than members of the privileged group, establishing a negative track record for them. This situation might arise without malicious intent, for example a company might have historically hired employees primarily from the privileged group, giving them a better understanding of this group (Dunkelau and Leuschel, 2019; Barocas *et al.*, 2017).

2.4.1.2 Measures based on predicted and actual outcomes

In general the measures in this category can be thought of as asking whether the error rates for each group are similar (Friedler *et al.*, 2019). The criteria which describe the quality of a classifier, which can be formalised in a confusion matrix, come into play in these measures.

- **Equalised odds** (Hardt *et al.*, 2016): we require that both groups experience the same true positive rates (TPR) and false positive rates (FPR)

  $$P(\hat{Y} = 1 \mid Y = i, S = 1) = P(\hat{Y} = 1 \mid Y = i, S = 0),\ i \in \{0, 1\}$$

  (Note that TPR = 1 - FNR, where FNR is the false negative rate, and FPR = 1 - TNR, where TNR is the true negative rate)

Equalised Odds can be relaxed into the following two measures:

- **Equality of opportunity** (Hardt *et al.*, 2016): Requires equality of TPR for both groups

  $$P(\hat{Y} = 1 \mid Y = 1, S = 1) = P(\hat{Y} = 1 \mid Y = 1, S = 0)$$

Note that equality of TPR is equivalent to equality of FNR.

- **Predictive equality** (Dunkelau and Leuschel, 2019): Requires equality of FPR for both groups

$$P(\hat{Y} = 1 \mid Y = 0, S = 1) = P(\hat{Y} = 1 \mid Y = 0, S = 0)$$

Note that equality of FPR is equivalent to equality of TNR.

Swapping the order of Y and $\hat{Y}$ around in the above definitions gives us:

- **Conditional use accuracy equality** (Dunkelau and Leuschel, 2019): This measure requires equality of precision or positive predictive value (PPV) and equality of negative predictive value (NPV):

$$P(Y = \hat{Y} \mid \hat{Y} = i, S = 1) = P(Y = \hat{Y} \mid \hat{Y} = i, S = 0),\ i \in \{0, 1\}$$

This measure can be relaxed into the following measure:

- **Predictive parity or PPV equality** (Verma and Rubin, 2018): A classifier satisfies this definition if both groups have equal PPV:

$$P(Y = 1 \mid \hat{Y} = 1, S = 1) = P(Y = 1 \mid \hat{Y} = 1, S = 0)$$

The fairness criteria which condition on the output variable implicitly assume that the true labels are fair, in other words that the historical data does not contain discrimination. The validity of this assumption should be regarded with caution. In a context such as credit scoring, for example, the output captures whether the loan was paid back or not, so the assumption is realistic. On the other hand, in the context of hiring, for example, if the output variable captures human decisions for who was hired or not, these decisions may not necessarily have been objective (Žliobaitė, 2017).

Finally, there are limitations of group fairness metrics. A fair decision for each individual cannot be guaranteed with these measures, rather 'on average' the decision is fair across the groups defined by the protected or sensitive attribute. Details of the limitations can be found in Dwork *et al.* (2012).

2.4.2 Impossibility theorems

Garg *et al.* (2020) discuss in detail the trade-offs among demographic parity, equalised odds and predictive parity. They detail how these measures relate to one another, as well as whether they are compatible or mutually exclusive.

We assume that the 'base rate' of the two groups is different. The base rate of a group is the ratio of people in the group who belong to the positive class ($Y = 1$) to the total number of people in that group. So, having unequal base rates across the two groups means that $P(Y = 1 \mid S = 0) \neq P(Y = 1 \mid S = 1)$. Firstly, we define some shorthand notation for use in this section.

The group specific TPR:

$$TPR_s = P(\hat{Y} = 1 \mid Y = 1, S = s), s \in \{0, 1\}$$

The group specific FPR:

$$FPR_s = P(\hat{Y} = 1 \mid Y = 0, S = s), s \in \{0, 1\}$$

The group specific PPV:

$$PPV_s = P(Y = 1 \mid \hat{Y} = 1, S = s), s \in \{0, 1\}$$

Next we give proofs which examine the compatibility of demographic parity, equalised odds and predictive parity. Firstly, using the probability relation

$$P(A, B) = P(A \mid B)P(B) = P(B \mid A)P(A)$$

the probability distributions linked with the three metrics can be expressed as follows:

$$\begin{aligned} P(Y, \hat{Y} \mid S) &= P(Y \mid \hat{Y}, S) \times P(\hat{Y} \mid S) \\ &= Predictive\ Parity \times Demographic\ Parity \\ &= P(\hat{Y} \mid Y, S) \times P(Y \mid S) \\ &= Equalised\ Odds \times Base\ Rate \end{aligned} \tag{2.1}$$

- **Demographic parity, equalised odds and predictive parity**: Assuming unequal base rates, we start with the assumption of a predictor that satisfies both demographic parity and equalised odds, then we check if it also satisfies predictive parity. From equation (2.1) we can get

$$P(Y = 1 \mid \hat{Y} = 1, S) = \frac{P(\hat{Y} = 1 \mid Y = 1, S \times P(Y = 1 \mid S)}{P(\hat{Y} = 1 \mid S)} \tag{2.2}$$

  As the predictor satisfies equalised odds, we have that $TPR_0 = TPR_1 = TPR$. Demographic parity also holds, so we have $P(\hat{Y} = 1 \mid S = 0) = P(\hat{Y} = 1 \mid S = 1) = P(\hat{Y} = 1)$. With these conditions we take the difference of the PPV values of the two groups:

$$\begin{aligned} &P(Y = 1 \mid \hat{Y} = 1, S = 0) - P(Y = 1 \mid \hat{Y} = 1, S = 1) \\ &= \frac{TPR[P(Y = 1 \mid S = 0) - P(Y = 1 \mid S = 1)]}{P(\hat{Y} = 1)} \end{aligned} \tag{2.3}$$

  For predictive parity to be satisfied, we need the PPV to be equal for both groups, and this means that the difference on the left side of the above equation should equal zero, which can only happen if the base rates of the two groups are equal (we do not consider the case when TPR is 0 as the usefulness of such a classifier is limited). Thus, in the case that the two groups have unequal base rates, we cannot satisfy all 3 of demographic parity, predictive parity and equalised odds. This is true even in the case of a perfect predictor because when the base rates are unequal a perfect predictor cannot satisfy demographic parity.

- **Demographic parity and Predictive parity**: When demographic parity holds we have $P(\hat{Y} = 1 \mid S = 0) = P(\hat{Y} = 1 \mid S = 1) = P(\hat{Y} = 1)$. Next, we take the difference in PPV

across the two groups

$$P(Y = 1 \mid \hat{Y} = 1, S = 0) - P(Y = 1 \mid \hat{Y} = 1, S = 1)$$
$$= \frac{TPR_0[P(Y = 1 \mid S = 0)] - TPR_1[P(Y = 1 \mid S = 1)]}{P(\hat{Y} = 1)} \tag{2.4}$$

For predictive parity to hold, the first line of the equation must be zero, which means that

$$\frac{TPR_0}{TPR_1} = \frac{P(Y = 1 \mid S = 1)}{P(Y = 1 \mid S = 0)} \tag{2.5}$$

Thus, while demographic parity and predictive parity can be simultaneously satisfied even with different base rates, the usefulness of such a classifier is limited when the ratio of the base rates is significantly different from 1, as this implies that the true positive rate for one of the groups would be very low.

- **Equalised odds and Predictive parity**: When equalised odds and predictive parity both hold, we have $TPR_0 = TPR_1 = TPR$, $FPR_0 = FPR_1 = FPR$ and $PPV_0 = PPV_1 = PPV$. We have

$$P(\hat{Y} = 1 \mid S) = \sum_y P(\hat{Y} = 1 \mid Y, S)P(Y \mid S)$$
$$= P(\hat{Y} = 1 \mid Y = 1, S)P(Y = 1 \mid S) + P(\hat{Y} = 1 \mid Y = 0, S)P(Y = 0 \mid S)$$

Hence

$$P(\hat{Y} = 1 \mid S) = TPR[P(Y = 1 \mid S)] + FPR[P(Y = 0 \mid S)] \tag{2.6}$$

Using equation (2.1) we can write

$$P(\hat{Y} = 1 \mid Y = 1, S = 0)P(Y = 1 \mid S = 0)$$
$$= P(Y = 1 \mid \hat{Y} = 1, S = 0)(TPR[P(Y = 1 \mid S = 0)] + FPR[P(Y = 0 \mid S = 0)])$$

Hence

$$TPR[P(Y = 1 \mid S = 0)] = PPV(TPR[P(Y = 1 \mid S = 0)] + FPR[P(Y = 0 \mid S = 0)])$$

$$TPR[P(Y=1 \mid S=0)] = PPV(TPR[P(Y=1 \mid S=0)] + FPR[1 - P(Y=1 \mid S=0)])$$

Hence

$$P(Y=1 \mid S=0) = \frac{PPV \times FPR}{PPV \times FPR + (1-PPV)TPR} \tag{2.7}$$

Similarly we have

$$P(Y=1 \mid S=1) = \frac{PPV \times FPR}{PPV \times FPR + (1-PPV)TPR} \tag{2.8}$$

Hence, in the absence of a perfect predictor (when FPR would be equal to 0 and PPV would be equal to 1), the base rates must be equal if both equalised odds and predictive parity hold. If we have perfect prediction, equations (2.7) and (2.8) take on the indefinite form $\frac{0}{0}$, and therefore we cannot conclude anything about the base rates with surety.

- **Equalised odds and Demographic parity**: If equalised odds is satisfied, then we have $TPR_0 = TPR_1 = TPR$ and $FPR_0 = FPR_1 = FPR$. Then

$$P(\hat{Y}=1 \mid S) = TPR[P(Y=1 \mid S)] + FPR[P(Y=0 \mid S)]$$

Hence

$$P(\hat{Y}=1 \mid S=0) - P(\hat{Y}=1 \mid S=1)$$

$$= TPR[P(Y=1 \mid S=0) - P(Y=1 \mid S=1)] + FPR[P(Y=0 \mid S=0) - P(Y=0 \mid S=1)] \tag{2.9}$$

Hence

$$P(\hat{Y}=1 \mid S=0) - P(\hat{Y}=1 \mid S=1)$$

$$= (TPR - FPR)[P(Y=1 \mid S=0) - P(Y=1 \mid S=1)] \tag{2.10}$$

Assuming demographic parity, the first line of equation (2.10) is 0, this means either TPR = FPR or the base rates are the same. However, we assume that the base rates are different so the only possible option is to have TPR = FPR. Since in general as the goal is to develop a classifier in which the TPR is a lot higher than the FPR, while simultaneously achieving demographic parity and equalised odds is mathematically possible, it is not very useful.

2.5 BIAS MITIGATION ALGORITHMS

Bias mitigation algorithms are classified under three main categories: pre-processing techniques modify the input data so that any algorithm trained on the data will be fair, in-processing techniques which change an existing algorithm which should be fair under any inputs, and post-processing techniques which modify the output of a model to be fair (Friedler *et al.*, 2019). Below we give some details of these three techniques.

2.5.1 Pre-processing methods

This approach aims to remove bias from the training data so that the model does not have to account for discrimination (Dunkelau and Leuschel, 2019). The model is trained on fair data which then gives a fair classifier.

An example is the technique of data massaging. The method takes a number of observations in the training data and changes their outcome values. A ranker R is used which ranks the individuals by their probability to receive the favourable outcome. The higher this probability, the higher the individual will rank.

Let $\epsilon = P(Y = 1 \mid S = 1) = P(Y = 1 \mid S = 0)$ denote the measured discrimination of the training data. The number of individuals M who are modified is calculated as follows

$$M = \epsilon \times \frac{|D_1| \times |D_0|}{|D_1| + |D_0|}$$

where $D_1 = \{X|S = 1\}$ and $D_0 = \{X|S = 0\}$ denote the sets of privileged and unprivileged individuals respectively. The massaging happens on the sets $pr = \{X \in D_0|Y = 0\}$ and $dem = \{X \in D_1|Y = 1\}$ where we sort both sets with respect to their ranks: pr descending and dem ascending. Labels of the top M individuals in both sets get flipped, which are respectively the M individuals closest to the decision boundary.

2.5.2 In-processing methods

These techniques modify particular learning algorithms, by changing the objective function through regularisation or adding additional constraints (Dunkelau and Leuschel, 2019; Friedler *et al.*, 2019).

An example is Discrimination Aware Decision Tree Construction (Kamiran *et al.*, 2010). In this method the splitting heuristic used for learning decision trees is changed. The decision tree iteratively splits the data set D based on the attribute leading to the largest information gain. Assume a split which divides the data into k different data splits $D_1 \dots D_k$. The information gain over the ground truth is defined as

$$IG_Y = H_Y(D) - \sum_{i=1}^{k} \frac{|D_i|}{|D|} H_Y(D_i)$$

where H_Y denotes the entropy with respect to the ground truth. By accounting for the entropy H_S over the protected attribute, the discrimination gain can be measured by

$$IG_S = H_S(D) - \sum_{i=1}^{k} \frac{|D_i|}{|D|} H_S(D_i)$$

For determining the most suitable split during training, for example, $IG_Y - IG_S$ only allows non-discriminatory splits.

2.5.3 Post-processing methods

Post-processing algorithms change the model's predictions to make them fair (Dunkelau and Leuschel, 2019).

An example is Reject Option Based Classification (Kamiran *et al.*, 2012). Let $h(X)$ be the predicted probability of an individual to belong to the positive class. This technique changes the prediction of samples for which $h(X)$ is close to the decision boundary by introducing a rejection option. For the classification threshold θ with $0.5 < \theta < 1$ let $[1-\theta, \theta]$ be the critical region. As an example, for $\theta = 0.7$ the critical region would be $[0.3, 0.7]$. If $h(X)$ lies in the critical region, then the prediction is assumed to be dependent on S: disadvantaged individuals receive the positive outcome $\hat{Y} = 1$ and advantaged individuals receive the negative outcome $\hat{Y} = 0$. For those observations

whose prediction score lies outside the critical region, the predictions remain unchanged, i.e., for $h(X) \in [0, 1-\theta)$ the prediction is $\hat{Y} = 0$ and for $h(X) \in (\theta, 1]$ it is $\hat{Y} = 1$.

2.6 SUMMARY

In this chapter, we covered the ideas that are central to the topic of fairness in machine learning. We started by demonstrating that simply removing the sensitive attribute from the model is not sufficient to ensure that a model is fair. In a real world situation, due to legal or other issues, the sensitive attribute may not be accessible in creating a machine learning model, and so in our experiments we will exclude this variable from our models.

We next discussed the group fairness metrics, which are used to quantify the bias of an algorithm. These metrics will be used in the empirical study, detailed in Chapter 5. For theoretical interest, we also investigated the compatibility or conflicts between the different fairness metrics.

Lastly, we discussed the bias mitigation algorithms, which can be used to attempt to remove the bias of an algorithm at different stages of its development.

CHAPTER 3

BACKGROUND: MISSING VALUES

3.1 INTRODUCTION

Missing data presents us with a challenging issue and is pervasive across all fields of research. For example, in the social sciences, it is a given that some respondents will refuse to participate or to answer certain questions. Another example is from the field of medicine: a clinical trial for a new medication in which participants drop out of the study because they are having adverse reactions to the drug (Van Buuren, 2018; Enders, 2022; Kang, 2013). For our use, a practical definition of missing data can be as follows: those observations in a data set that have no value for some predictors (Kuhn and Johnson, 2013; Kang, 2013). Inevitably, with most real world data sets, we are likely to encounter missing values. Most statistical analysis methods can only work on complete data (Soley-Bori, 2013).

One of the most common ways of dealing with incomplete data cases is to discard the entire observation. The default in most statistical software is to remove any such observations entirely (Enders, 2022). There are many problems associated with this approach (Kang, 2013; Beretta and Santaniello, 2016; Van Buuren, 2018): Parameter estimates from such data can be biased, samples are less representative of the population of interest, it has a negative effect on statistical power and it leads to discarding data which is often very costly to collect in the first place. We will elaborate further on this procedure in the listwise deletion section.

3.2 MISSING DATA CAUSES AND PATTERNS

We can broadly categorise the causes of missing data (Fernando *et al.*, 2021) as follows:

- **Partial completion**: This can occur in longitudinal studies where a measurement is repeated after a certain period of time. It results when after collecting a few values of a record, at a certain point in time or place within a questionnaire or a data collection process, the remaining attributes or measurements are missing. This could happen towards the end of a questionnaire, as well as to users more prone to fatigue.

- **Missing by design**: This can arise in two ways, contingency attributes and attribute sampling. In contingency attributes there may be certain questions which are not applicable to all individuals. In attribute sampling, specific design is used to randomly set different subsets of questions to different individuals.
- **Item non-response**: Certain types of questions, for example those related to private information such as income, may generate a non-response. There are three main ways in which a non-response can occur: information is not provided, for example if an answer is not known, the information provided is useless, for example an answer is illegible or impossible, or the information provided is lost, for example through data processing.

A missing data pattern can be defined as the distribution of observed and missing values in a data set. Based on the causes of missing data, we can discern the following missing data patterns (Enders, 2022; Van Buuren, 2018). These are displayed in Figure 3.1. The location of the missing values in the data set is represented by the shaded areas. Assume we have a data set with four variables, Y_1, Y_2, Y_3, Y_4, the different missing data patterns can be described as follows:

- **Univariate pattern**: Missing values are isolated to a single variable.
- **Unit non-response pattern**: This pattern can arise when a researcher administers two measures which are cheap to collect, to the entire sample (e.g., Y_1 and Y_2) and collects two expensive measures (e.g., Y_3 and Y_4) from a subset of cases only.
- **Monotone missing data pattern**: This can occur in longitudinal studies where subjects drop out. This pattern resembles a staircase, in that the cases with missing data on a particular variable are always missing subsequent variable measurements.
- **General missing data pattern**: This is the most common pattern. Missing values are scattered through the data matrix in a random fashion.
- **Planned missing data pattern**: This can be seen as intentional missing data. For example, in panel E we see four questionnaires divided across three forms, such that each form includes Y_1 but is missing Y_2, Y_3 or Y_4.
- **Latent variable pattern**: This pattern is included for completeness as latent variable models are not usually viewed as missing data problems. Here the values of the latent variable are

missing for the entire sample.

Regarding missing data patterns, it is no longer important to distinguish between them because advanced imputation techniques such as multiple imputation are well suited to almost all missing data patterns. To clarify, the missing data pattern describes the location of the missing values, they do not explain why the data are missing.

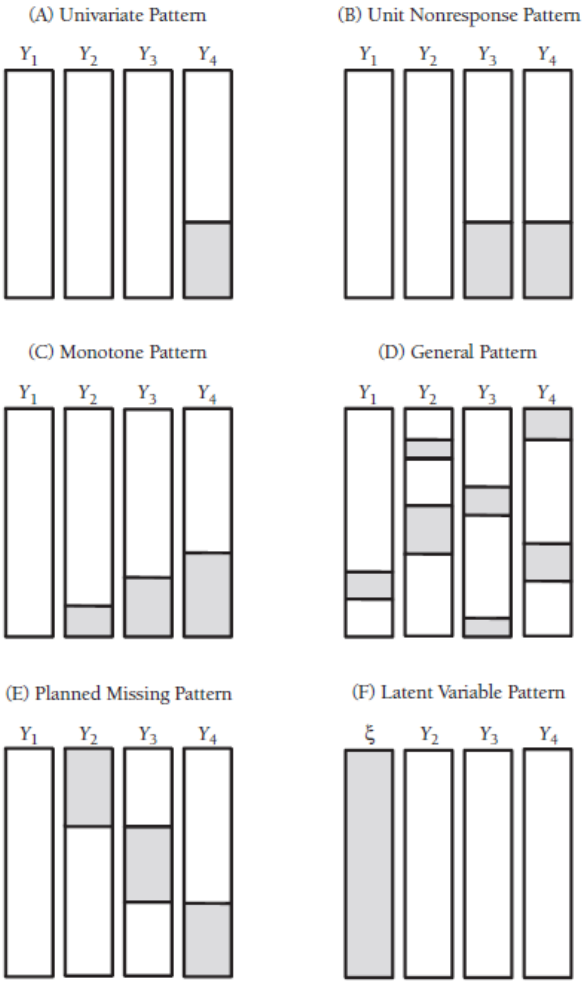

Figure 3.1: Missing data patterns (Enders, 2022, p. 4)

3.3 MISSING DATA MECHANISMS

Missing data mechanisms (MDM's) (Little and Rubin, 2019; Enders, 2022; Van Buuren, 2018; Schouten *et al.*, 2018) describe the relationship between the probability of missingness in a data set and the variables in the data set. There are three MDM's, and as agreed by many researchers the terminology related to these can be confusing.

We define some notation which will be used in the definitions below. As usual let X denote the $n \times p$ matrix which has data values for p predictors on all n observations. Hence X contains the hypothetically complete data. We use R to denote the $n \times p$ matrix containing zeros or ones, such that R contains the locations of the missing values in X. In particular, we denote the elements of X and R as x_{ij} and r_{ij} respectively, such that $i = 1, \ldots, n$ and $j = 1, \ldots, p$. We have that $r_{ij} = 1$ if x_{ij} is observed, and $r_{ij} = 0$ if x_{ij} is missing. We say that X_{obs} is the observed data and X_{mis} stores the missing data. X_{mis} contains those elements x_{ij} such that $r_{ij} = 0$. Hence $X = (X_{obs}, X_{mis})$ would denote the hypothetically complete data. X_{mis} contains real values which are hidden from us, such that R indicates which values are hidden.

The distribution of R could depend on $X = (X_{obs}, X_{mis})$ and this relation is described by the *missing data model*. We denote the parameters of the missing data model by ϕ.

- **Missing completely at random, MCAR**: The MCAR mechanism is satisfied when the probability of being missing is unrelated to the data. In particular, the data are said to be MCAR if

$$P(R = 0 \mid X_{obs}, X_{mis}, \phi) = P(R = 0 \mid \phi)$$

  The parameter ϕ governs the probability of missingness, but this probability is not related to any variable in the data set. In MCAR missingness, the observed data points can be seen as a simple random sample of the complete data. If missingness exists in a data set then MCAR missingness is the ideal scenario.

- **Missing at random, MAR**: The MAR mechanism requires that the missingness probability

only depends on the observed information, i.e., X_{obs}:

$$P(R = 0 \mid X_{obs}, X_{mis}, \phi) = P(R = 0 \mid X_{obs}, \phi)$$

If MAR is satisfied, i.e., the data are missing as a function of the observed variables, then for example ϕ could contain the coefficients from a logistic regression model.

- **Missing not at random, MNAR**: The data are MNAR if the probability of being missing depends on the missing information X_{mis}, and can also depend on X_{obs}. Hence

$$P(R = 0 \mid X_{obs}, X_{mis}, \phi)$$

does not simplify further.

From the point of view of statistical analysis and inference, removing the rows with missing values before analysis requires the MCAR mechanism to be satisfied, otherwise the parameter estimates produced will be biased.

3.4 MISSING VALUE HANDLING METHODS

3.4.1 Listwise deletion

Also known as complete case analysis, this method simply removes the entire observation that contains missing values for some predictors. It is one of the most frequently used techniques to deal with missing data. Many statisical software packages also use this method as a default. As mentioned in Section 3.3, if the data are MCAR, then listwise deletion produces unbiased parameter estimates. The advantage of listwise deletion is that it is a straightforward convenient technique to use. A disadvantage is that even with MCAR being satisfied, removing entire observations is wasteful and can lead to loss of statistical power. If data are not MCAR then it produces biased parameter estimates, and can reduce the variability of the data (Enders, 2022; Kang, 2013).

3.4.2 Single imputation

There are a variety of single imputation methods but in general, even with MCAR being satisfied, they perform poorly. The meaning of the term 'single' here is that this imputation method generates

a single replacement value per missing value. The advantages of imputation are that you do not have to throw away entire rows of data and you end up with a complete data set. The disadvantage is that even when the data are MCAR, most single imputation techniques produce biased parameter estimates. Also, all single imputation techniques underestimate sampling error (Enders, 2022). Next we consider some popular single imputation techniques:

- **Mean imputation**: In mean imputation, we replace the missing values of a variable with the arithmetic mean of the available cases of that variable. This produces a complete data set. Intuitively, in mean imputation we are imputing with values that are at the centre of a distribution, which reduces the variance. With reduced variability the size of covariances and correlations also reduces. This happens even in the case of MCAR data (Enders, 2022).
- **Mode imputation**: Mode imputation can be described similarly to mean imputation. Here, the mode of a variable is calculated on the available cases of the variable and used to impute its missing values. Mode imputation also leads to an underestimation of the population variance (Nishanth and Ravi, 2016).
- **Regression imputation**: In regression imputation, we replace missing values with scores produced from a regression equation. In the first instance, using the complete variables, we estimate a set of regression equations to predict the incomplete variables. These estimates are generated from the complete cases. Intuitively, this technique makes sense as variables in a data set are likely to be correlated, and we use the information from complete variables to impute incomplete variables. The number of equations depend on the number of missing data patterns in the data set (Enders, 2022).

  With regression imputation, we overestimate correlations but underestimate variances and covariances, but not to the same extent as in mean imputation (Enders, 2022).
- **k-nearest neighbour (k-NN) imputation**: In k-NN imputation, we take the k most similar observations to the missing observation (where similarity is measured by a distance metric), then for a continuous variable you replace the missing value by the mean of the k-nearest neighbours and the mode for a categorical variable (Jonsson and Wohlin, 2004). The choice of distance metric depends on the nature of the variables we want to use in the distance

calculation. A measure which can calculate distance based on a mixture of continuous and categorical variables is the Gower distance (Kowarik and Templ, 2016).

Unlike mean imputation where the replacement value is influenced by all cases, in k-NN imputation these values are only influenced by the most similar cases. k-NN imputation is less influenced by the issue of reduced variance than mean imputation (Jonsson and Wohlin, 2004).

3.4.3 Multiple imputation (MI)

Unlike in single imputation, MI produces multiple replacement values for the missing value. As an overview, see Figure 3.2. MI results in more than one complete data set such that a number of plausible values are imputed for the missing value. The observed data is identical in these data sets but the imputed values differ. Next, the analysis of interest is performed on each imputed data set and the parameters of the model are estimated for each of these. In the last step we pool together the results of the previous analyses in such a way as to take the uncertainty of imputed values into account. For details see Van Buuren (2018).

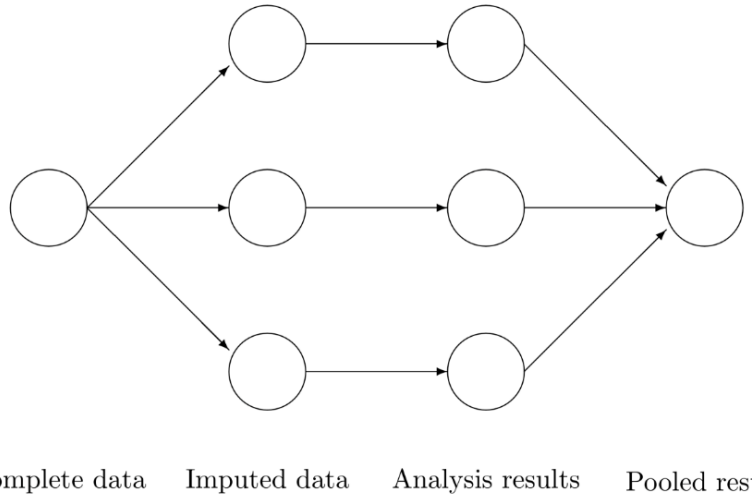

Figure 3.2: Overview of multiple imputation (Van Buuren, 2018)

Despite the fact that approaches like multiple imputation are superior over single imputation methods or listwise deletion (Enders, 2022; Van Buuren, 2018), the latter continue to be used by re-

searchers. Eekhout *et al.* (2012) shows that researchers did not pay attention to the type of missingness before applying a missingness handling method, most studies (81%) performed a complete case analysis, 14% of studies used a single imputation technique and multiple imputation was used in 8% of the studies. Despite the statistical superiority of multiple imputation compared to single imputation, Kowarik and Templ (2016) correctly identify that in a usual statistical analysis, the aim is often to generate one complete data set which is then analysed by researchers, and hence single imputation still has importance in the missing data field.

3.5 SUMMARY

In this chapter, we covered the topics underlying the field of missing data. We started with the causes of missing values and the patterns in missing data.

We then studied the missing data mechanisms, which describe the relationship between the probability of missingness in a data set and the variables in the data set. In our experiments, we will start with a complete data set, and then artificially create missingness in the data sets according to these MDM's.

We then covered missing data handling procedures, including listwise deletion, single imputation and multiple imputation. In our experiments, we will use the methods of listwise deletion and some single imputation methods, in order to remove or impute missing values. We will study the effects of the MDM's and missing value handling procedures on the fairness of classification algorithms.

CHAPTER 4

LITERATURE REVIEW

Existing literature combining the research of fairness of algorithms and missing data is limited to a handful of studies (Fernando *et al.*, 2021; Zhang and Long, 2021*b*; Wang and Singh, 2021; Zhang and Long, 2021*a*).

The first paper we study (Fernando *et al.*, 2021) is the first piece of research which attempts to understand the issue of missing values in the context of machine learning fairness. The authors aim to answer 3 main questions:

- 'Are missing data and fairness related?'
- 'Are those subsamples with missing data more or less unfair?'
- 'Is it the right procedure to delete or replace these values?'

They start by giving a description of the causes of missingness and the causes of unfairness. They then try to tie up the two. With regards to fairness, the missing data might not be evenly distributed between the privileged or disadvantaged groups, which could lead to discrimination in the data and models fitted. Disadvantaged individuals may intentionally omit information if they believe that a complete answer might lead to a discriminatory or unfair action. Fairness strongly depends on both the quality of the data and the quality of the processing of these data, these are the two processes where missing values can appear. Removing data due to a sub-population having missing values more commonly could result in under-representation.

For their empirical work, they use six data sets (two of these, Adult and COMPAS, are used in our study as well, see Appendix A for the data sets descriptions) with two sensitive attributes each, i.e. 12 cases. Using Little's MCAR global test (Little, 1988), the null hypothesis of MCAR is rejected with $p < 0.001$ for all 6 data sets. We will focus on one of these for brevity, the Adult income data set, which is popular in the field of machine learning fairness and contains missing values. The two sensitive attributes in this data set are race and sex. They consider only one fairness metric in their study: Demographic parity (for which they use the term Statistical Parity

Difference (SPD), see Section 2.4.1.1).

The authors start by analysing the training data itself for fairness (this can be seen as applying the metric to the output of a perfect predictor), by looking at 3 subsets of the data separately and applying SPD to them: the data set with all rows, the data set without missing value rows and the data set with only the missing value rows. This is the first step to understand whether the missing data rows are more or less unfair. From the results, in 10 out of the 12 cases, the smallest value of the metric is found on the subsets of instances containing only the missing value rows. The authors interpret that this shows that 'the rows containing missing values are fairer than the rest'. We would like to make a point next about this interpretation, demonstrating it on the Adult income data set with sensitive attribute Race.

The non-MCAR pattern of missingness in the Adult data set for the variables Occupation, Work class and Native country suggests that the rows with missing data likely contain a higher proportion of disadvantaged individuals from both whites and non-whites. This means that the rates of favourable outcome (the favourable outcome being earning an income greater than \$50,000) will likely be lower for both groups in the subset with missing values (than in the full set or the set without the missing rows), irrespective of the racial group. Comparing the metric on the missing data rows to the non-missing data rows is hence akin to comparing the metric on two different data sets. Without standardising with respect to the base rates in each subset, the extent of the differences between the two sets will be falsely inflated.

By looking at the SPD formula on the training data, where Y is the outcome with 1 being the favourable or positive outcome, $S = 1$ indicates whites and 0 indicates non-whites:

$$SPD = P(Y = 1 \mid S = 1) - P(Y = 1 \mid S = 0)$$

For the subset with only missing value rows, both rates in the above statement will be small, which leads to the smaller value of SPD on the missing rows.

Below are the numbers for the Race variable in Adult income data set:

$$SPD_{full} = 0.2539 - 0.1525 = 0.1014$$

and on the subset with missing value rows only the two terms are

$$SPD_{missing} = 0.1399 - 0.1038 = 0.0361$$

We now take a look at the next set of results, the scenario is 100 repetitions of training/test set split (25% of the data for test set) using CART (Classification and Regression Trees) as predictive model. The data subsets again are the data set with all rows, the data set without missing value rows and the data set with only the missing value rows. The authors use the CART model as it can handle missing values, without needing to first remove them or impute them, as is the case with most other machine learning techniques. In 11 out of 12 cases, the rows with missing values give the lowest SPD value.

The next scenario is the same as the previous apart from one difference: the columns with missing values are removed. Compared to the previous case, removing the columns with missing values does not affect the accuracy of the models much. The rows with only missing values still give the lowest SPD. This is an interesting observation, as it seems to suggest that the missing values in these rows themselves are not that important, but rather the non-missing information in the rows with the missing values is more important for accuracy and fairness. Maybe some other variables in the data are correlated with the variables with missing values, acting as their proxy (this is plausible as the MDM is not MCAR).

The last set of experiments aims to understand the effect of listwise deletion or imputation with mean/mode on the fairness of machine learning models, and the trade-off between fairness and accuracy. They use 6 techniques: Logistic regression (LR), Naive Bayes (NB), Neural Network (NN), Random Forest (RF), the RPart decision tree (DT) and a support vector machine (SVM) using a linear kernel. An example of the results is shown in Figure 4.1, on the Adult data set (Fernando *et al.*, 2021) where the y-axis represents SPD and the x-axis Accuracy. For comparison

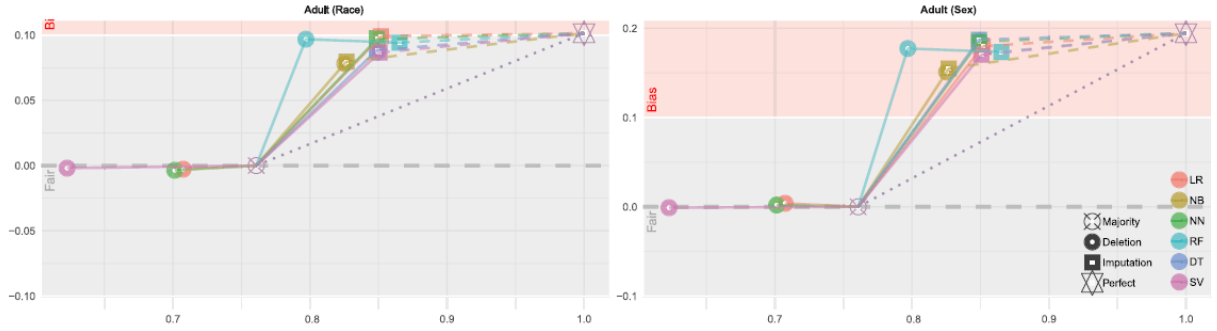

Figure 4.1: SPD vs Accuracy for the six models on adult data set (Fernando *et al.*, 2021, p. 28)

they also plot results from the Majority class model (the model always picks the majority class and has perfect fairness) and the Perfect model (i.e. the model that is 100% accurate but is not unbiased).

Their results show that in terms of accuracy, imputation is generally beneficial for all cases and techniques. In terms of fairness, all methods reduce bias from the perfect model. Regarding the trade-off between fairness and accuracy, if we compare listwise deletion and imputation, we see the expected trend of more accuracy for imputation implying less fairness. The authors recommend that in practice one should try both listwise deletion and imputation to see which models are more relevant for a particular problem, depending on the trade-off between fairness and performance.

The next research paper (Zhang and Long, 2021*b*) investigates the impact of missing data imputation methods on fairness metrics, inspired by the work of Fernando *et al.* (2021).

Based on the widely used fairness metric equalised odds as defined in (Hardt *et al.*, 2016) they define the notion of *equalised odds difference*. Equalised odds difference for classifier h is defined as

$$EOD(h) := |FPR_1(h) - FPR_0(h)| + |FNR_1(h) - FNR_0(h)|$$

Where the subscript 1 refers to the advantaged group and 0 refers to the disadvantaged group. This notion is then used to measure the unfairness in prediction for different imputation methods,

where the (binary) sensitive attributes are gender and race. For gender, male is the advantaged group and for race, white is the advantaged group.

Next they propose a novel notion that measures the fairness of imputation results. For this the following notation is set up. Sensitive attributes are binary, $S = s \in \{0, 1\}$ with 1 being the advantaged group. The complete data matrix (containing all the predictors) without missing values is denoted by $X = (x_{ij}) \in R^{n \times p}$. The missing data indicator is denoted by $R = (r_{ij}) \in R^{n \times p}$ with $r_{ij} = 1\{x_{ij} \text{ is observed}\}$. Next define the group specific entities, let $X^s = (x^s_{ij})$ and $R^s = (r^s_{ij})$ denote the complete data matrix and missing indicator matrix in sensitive group $S = s$, (so that $X = X^0 \cup X^1$ and $R = R^0 \cup R^1$). In group s the data matrix imputed by model g is denoted by $\hat{X}^s(g) = (\hat{x}^s_{ij}(g))$. Assuming that both sensitive groups contain missing data, mean square imputation error of g in group s is defined as

$$MSIE_s(g) = \frac{\sum_{i,j}(\hat{x}^s_{ij}(g) - x^s_{ij})^2(1 - r^s_{ij})}{\sum_{ij} 1 - r^s_{ij}}$$

The novel notion of imputation accuracy parity difference, IAPD for imputation model g is then defined as

$$IAPD(g) = MSIE_1(g) - MSIE_0(g)$$

The usefulness of this measure in regards to measuring fairness is questionable, as higher imputation accuracy does not necessarily mean higher fairness, especially if the data set contains historic biases.

The authors generate missingness in the first L features in the data set according to the three missing data mechanisms in a univariate manner. Univariate amputation means that missingness is generated one variable at a time. Their notation is as follows: given a sample $\mathbf{z} = (z_1, \ldots, z_p)$ the probability that z_j is missing is given by the following values shown in Figure 4.2 (where the values are truncated inside the unit interval $[0, 1]$) for $\forall j \in \{1, \ldots, L\}$ (Zhang and Long, 2021*b*).

The usefulness of their 11 schemes for generating missingness in categorical variables is not clear, whereas in their case all predictor variables are numeric. Regarding the results in the paper, they are all from only the numerical variables in the COMPAS data set (Store, 2016), whereas we are working with missingness in both continuous and categorical variables. In data sets applicable to

MCAR		MAR		MNAR	
0.1	(1a)	$0.1 + 0.8\mathbf{1}_{\text{male}}$	(2a)	$0.5 - z_j$	(3a)
0.5	(1b)	$0.1 + 0.8\mathbf{1}_{\text{female}}$	(2b)	$0.5 - 0.2z_j$	(3b)
0.9	(1c)	$0.5 - 0.5z_{L+j}$	(2c)	$0.5 + 0.2z_j$	(3c)
		$0.5 + 0.5z_{L+j}$	(2d)	$0.5 + z_j$	(3d)

Figure 4.2: The missing data mechanisms used in (Zhang and Long, 2021*b*)

the real world, often there are many categorical variables with missing values (as these are used to capture personal data) and this situation has not been studied much, in comparison to missingness in numerical or continuous variables (Fernando *et al.*, 2021; Wang and Singh, 2021).

To generate missingness in our research, we have used the `R` function `ampute` in package `mice` (Van Buuren and Groothuis-Oudshoorn, 2011). This function allows one to generate missingness in multiple variables simultaneously, according to the missing data mechanisms. This is beneficial as it can make controlling the overall missingness proportion, i.e., the proportion of cases containing at least one missing value, more straightforward. The merits of using this function over creating your own missingness are detailed in Schouten *et al.* (2018). In particular, amongst other things we are interested in the effects on fairness metrics of varying the overall missingness proportion when creating missingness in multiple variables. This can be difficult to control with a univariate generation process, in that the proportion of missingness desired may not be the proportion that is achieved. In our scenario, using the multivariate amputation approach in `ampute` overcomes such problems of univariate amputation.

Seven imputation methods are investigated: MICE, missForest, K-nearest neighbor (KNN) imputation, two matrix completion methods Soft-Impute and OptSpace, and two deep learning methods Gain and Misgan. Of these, MICE, missForest, Gain and Misgan are multiple imputation methods and the remaining are single imputation methods.

Most of these methods are state of the art and not widely known or used by most researchers. Similarly, most researchers do not investigate the missingness mechanism before applying a miss-

ing data handling technique to it. For example, in the field of epidemiology, the investigation by Eekhout *et al.* (2012) shows that researchers did not pay attention to the type of missingness before applying a missingness handling method, most studies (81%) performed a complete-case analysis, 14% of studies used a single imputation technique and multiple imputation was used in 8% of the studies. Despite the statistical superiority of multiple imputation compared to single imputation, Kowarik and Templ (2016) correctly identify that in a usual statistical analysis, the aim is often to generate one complete data set which is then analysed by researchers, and hence single imputation still has importance in the missing data field. In fairness of machine learning research, most software such as AIF360 (Bellamy *et al.*, 2018), Aequitas and ThemisML simply removes the rows or columns containing the missing values or will result in an error if missing values are encountered (Fernando *et al.*, 2021). Simpler methods dealing with missingness such as complete-case analyses, mean imputation or single-regression imputation are more widely used, and the effects of these on fairness metrics have not been studied. In our research we investigate listwise deletion or complete-case analysis, and the simpler single imputation methods and their effect on fairness.

From their Table 1 results (see Figure 4.3), in their first conclusion, which they term as 'Observation 1', the authors conclude that severe imputation unfairness exists among all the imputation methods, but the threshold of 'severe' has not been defined. In their second conclusion, 'Observation 2', they conclude that despite a similar overall missingness proportion, imputation fairness can be influenced by imbalance of missingness between the sensitive groups. Similarly in 'Observation 3' it is concluded that imputation unfairness tends to grow as missingness proportion increases and imbalance in the size of sensitive classes leads to imputation unfairness. Regarding Observation 3, an argument about 'sample imbalance' is made for the other two datasets, but no comment is made about the imbalance in sensitive class sizes in the COMPAS dataset, for which Table 1 is reporting the results and the IAPD is almost always negative. For comparison, the ratio of whites to non-whites in the COMPAS dataset is 36% : 64%, and the ratio of males to females is 78% : 22%. The ratios in COMPAS in the case of race seem to go against their argument on sample imbalance.

From their Table 2 results in Figure 4.4, in their fourth conclusion, 'Observation 4', the authors

Fairness in Missing Data Imputation

	MSIE								IAPD regarding gender								IAPD regarding race							
Method	I_1	I_2	I_3	I_4	I_5	I_6	I_7	Var	I_1	I_2	I_3	I_4	I_5	I_6	I_7	VarD_g	I_1	I_2	I_3	I_4	I_5	I_6	I_7	VarD_r
MCAR (1a)	0.68	0.68	**0.67**	0.81	0.88	0.73	0.99	0.99	0.45	0.46	0.47	0.50	0.48	**0.36**	0.51	0.55	-0.46	-0.48	-0.48	-0.49	-0.53	**-0.32**	-0.50	-0.53
MCAR (1b)	0.90	0.82	**0.80**	0.91	0.87	0.86	0.95	1.00	0.58	0.51	0.51	0.52	0.50	**0.42**	0.52	0.56	-0.57	-0.56	-0.55	-0.52	-0.54	**-0.42**	-0.49	-0.54
MCAR (1c)	**0.85**	1.09	0.88	0.96	0.88	0.95	0.95	1.00	0.54	0.58	0.53	0.53	0.51	**0.44**	0.47	0.56	-0.55	-0.73	-0.59	-0.53	-0.55	**-0.43**	-0.46	-0.54
MAR (2a)	**0.89**	1.08	0.85	1.03	0.90	1.00	1.04	1.09	0.53	0.79	0.57	0.56	**0.50**	0.60	**0.50**	0.55	-0.63	-0.76	-0.64	-0.61	-0.62	**-0.51**	-0.57	-0.61
MAR (2b)	**0.47**	0.53	0.49	0.62	0.57	0.70	0.69	0.72	0.43	0.35	0.41	0.49	0.49	**0.14**	0.51	0.55	-0.23	-0.27	-0.25	-0.24	-0.28	**-0.15**	-0.22	-0.27
MAR (2c)	0.42	**0.36**	0.54	0.51	0.53	0.80	0.62	0.57	0.15	0.15	0.13	0.18	0.19	**0.11**	0.17	0.21	-0.12	-0.22	**-0.04**	-0.21	-0.23	**-0.04**	-0.06	-0.25
MAR (2d)	**1.27**	1.66	1.35	1.36	1.34	1.37	1.48	1.44	0.86	0.85	0.92	0.83	0.79	**0.73**	0.84	0.83	-0.72	-0.87	-0.81	-0.64	**-0.60**	-0.65	-0.77	-0.62
MNAR (3a)	1.08	1.16	2.44	**0.26**	0.36	0.65	0.64	0.12	0.23	0.21	0.14	**0.00**	-0.01	**0.00**	0.01	0.01	-0.35	-0.47	-0.30	0.03	0.02	**0.01**	0.08	0.02
MNAR (3b)	0.67	0.53	0.67	**0.28**	0.35	0.60	0.52	0.27	0.25	0.16	0.14	0.02	**0.01**	**-0.01**	0.03	0.05	-0.27	-0.32	-0.29	**0.00**	0.03	0.02	-0.05	-0.03
MNAR (3c)	1.50	**1.42**	1.62	1.59	1.43	1.67	1.85	1.59	1.09	1.06	1.14	1.04	1.00	**0.98**	1.04	0.95	-1.01	-1.02	-1.13	-1.01	-1.01	**-0.87**	-0.97	-0.89
MNAR (3d)	2.71	2.44	2.53	2.20	**1.88**	1.95	2.11	1.71	1.59	1.47	1.56	1.35	1.28	**1.01**	1.03	1.00	-1.18	-1.32	-1.25	-1.20	-1.20	**-0.83**	-0.88	-0.90

Table 1. Imputation fairness on COMPAS recidivism dataset. Number of features (besides sensitive attributes) is 10, $L = 5$. Here imputation methods are encoded as: I_1: MICE; I_2: missForest; I_3: KNN; I_4: SoftImpute; I_5: OptSpace; I_6: Gain; I_7: misGAN. Results are average values over 50 repeated experiments. Var denotes the variance of missing data, VarD_g denotes the difference of missing value's variances between two gender groups and VarD_r denotes that between two race groups.

Figure 4.3: Table 1 in (Zhang and Long, 2021*b*)

conclude that for a fixed imputation method, different missing mechanisms lead to different prediction fairness. Hence prediction fairness is linked to the missing mechanism. In 'Observation 5', the authors conclude that three imputation methods: SoftImpute, OptSpace and Gain have smaller EOD consistently compared with other imputation methods or the results from complete cases and complete data. The trade-off here is that these 3 methods have lower prediction accuracy. They attribute this to the trade-off between accuracy and fairness in prediction. For any missing mechanism the prediction accuracy with imputations is lower than that from complete data, and EOD calculated on models with imputation are smaller than that on complete data. In Table 2 results on prediction fairness, compared to the model built on the complete data set (or complete cases), the imputations appear to make the predictive models fairer according to EOD. Similarly in Observation 5 it is said that most imputation models have smaller EOD compared to that on complete data. If it is the case that the imputations make the models fairer, then this goes against one of the main conclusions of their research, which they state on the first page under 'Our contributions' as 'Severe unfairness exists in both imputation and prediction after imputation'.

In the machine learning fairness literature, there are different techniques for improving the fairness of algorithms, so called bias mitigation algorithms. These approaches have been developed

Fairness in Missing Data Imputation

	Prediction Accuracy								EOD regarding gender								EOD regarding race							
Method	I_1	I_2	I_3	I_4	I_5	I_6	I_7	CC	I_1	I_2	I_3	I_4	I_5	I_6	I_7	CC_g	I_1	I_2	I_3	I_4	I_5	I_6	I_7	CC_r
MCAR (1a)	0.71	0.71	0.71	0.63	0.65	0.60	0.71	**0.72**	0.18	0.16	0.16	**0.05**	0.06	0.06	0.17	0.16	0.29	0.29	0.29	**0.03**	0.05	0.04	0.30	0.29
MCAR (1b)	0.68	0.70	0.68	0.63	0.65	0.58	0.70	**0.71**	0.17	0.16	0.14	0.05	0.06	**0.03**	0.17	0.16	0.27	0.29	0.26	**0.03**	0.04	0.05	0.31	0.29
MCAR (1c)	0.67	0.66	0.66	0.64	0.65	0.64	0.64	**0.70**	0.17	0.13	0.13	**0.06**	**0.06**	0.08	0.12	0.15	0.23	0.24	0.23	**0.04**	0.07	0.15	0.20	0.29
MAR (2a)	0.69	0.69	0.69	0.64	0.65	0.63	0.69	**0.70**	0.14	0.15	0.13	0.06	**0.05**	0.06	0.14	0.26	0.24	0.24	0.25	**0.03**	0.04	0.11	0.24	0.27
MAR (2b)	0.70	0.70	**0.71**	0.64	0.65	0.61	0.70	**0.71**	0.17	0.15	0.16	0.06	**0.05**	**0.05**	0.17	0.16	0.31	0.30	0.29	0.04	**0.03**	0.04	0.30	0.29
MAR (2c)	0.46	0.46	0.46	0.64	0.65	0.52	0.46	**0.71**	0.01	**0.00**	0.02	0.06	0.06	0.03	0.03	0.16	0.02	**0.00**	0.02	0.03	0.07	0.15	0.06	0.30
MAR (2d)	0.57	0.61	0.59	0.54	0.65	0.50	0.54	**0.71**	0.01	0.06	0.03	**0.00**	0.07	0.01	0.01	0.17	**0.00**	0.05	0.02	**0.00**	0.05	0.10	0.03	0.29
MNAR (3a)	0.68	0.66	0.62	0.64	0.66	0.61	0.64	**0.69**	0.12	0.12	0.09	0.05	0.06	0.04	**0.01**	0.14	0.32	0.24	0.16	**0.03**	0.35	**0.03**	0.08	0.25
MNAR (3b)	0.67	0.69	0.69	0.64	0.65	0.62	0.67	**0.71**	0.12	0.16	0.16	**0.05**	0.08	**0.05**	0.06	0.16	0.26	0.28	0.26	**0.03**	0.17	0.04	0.16	0.29
MNAR (3c)	0.70	0.69	0.68	0.64	0.65	0.57	0.69	**0.71**	0.17	0.13	0.12	0.06	0.06	**0.03**	0.09	0.16	0.31	0.26	0.25	0.04	**0.03**	0.05	0.18	0.28
MNAR (3d)	0.69	0.68	0.69	0.64	0.65	0.58	0.69	**0.71**	0.12	0.09	0.09	0.07	0.07	**0.03**	0.15	0.16	0.27	0.26	0.26	**0.04**	0.06	0.05	0.31	0.28

Table 2. Prediction fairness on COMPAS recidivism dataset. Number of features (besides sensitive attributes) is 10, $L = 5$. Here imputation methods are encoded as: I_1: MICE; I_2: missForest; I_3: KNN; I_4: SoftImpute; I_5: OptSpace; I_6: Gain; I_7: misGAN. CC: prediction using complete cases in the training set. Results are average values over 50 repeated experiments. When using complete data for prediction, prediction accuracy is 0.72, accuracy difference regarding gender is 0.16, accuracy difference regarding race is 0.29.

Figure 4.4: Table 2 in (Zhang and Long, 2021*b*)

on 'clean' data sets, whereas real data has many issues to make them 'dirty'. The authors of the next paper (Wang and Singh, 2021) address the issues of missing values and selection bias. For our research only the missing value issue is of interest so we focus on that for the review. In a nutshell, the authors propose in particular a pre-processing technique of bias mitigation, by altering an existing one to account for missing values.

In particular, the context is a binary sensitive attribute and non-sensitive categorical predictors only, with missingness created in only one of these categorical variables. In our research, we use both continuous and categorical predictors with missingness created in both, as this is a more realistic case, and missingness can be present in one or two variables. The authors propose a *reweighting* method for the problem of missing values in categorical predictors.

The two fairness metrics that are used to quantify the fairness are *p%-rule* and *error rate balance*. The *p%-rule* is defined as:

$$min\left(\frac{P(\hat{Y}=1 \mid S=1)}{P(\hat{Y}=1 \mid S=0)}, \frac{P(\hat{Y}=1 \mid S=0)}{P(\hat{Y}=1 \mid S=1)}\right)$$

The higher the value, the fairer the classifier is. Error rate balance is defined as balancing the false positive rate and false negative rate for both groups. In particular,

$$P(\hat{Y} = 0 \mid Y = y, S = 0) = P(\hat{Y} = 0 \mid Y = y, S = 1) \quad \forall y \in \{0, 1\}$$

When $y = 1$, the constraint equalizes the false negative rate (FNR) across the two sensitive groups. When $y = 0$, the constraint equalizes the false positive rate (FPR). For a fair classifier, the error rate difference should be small.

The algorithm that is used as a starting point learns a probabilistic transformation to change feature value labels in the data to reduce discrimination. This algorithm works with categorical data but does not account for missing values. The authors adjust the transform to include a new re-weighting scheme to account for missing values. The details of the approach are as follows: They weight observations with regard to missing values. When training a classifier, they want the classifier to learn more information from those observations which do not contain missing values. They assign a higher weight to observations without missing values and a lower weight to observations with missing values. This amounts to oversampling the higher quality observations.

With regards to the empirical work, they use a logistic regression model to predict the outcome. They create missingness in one of the categorical variables, but no details are given as to how the missingness was created. They use two real-world data sets (Adult income and COMPAS) and one synthetic data set. In the experiments that are most relevant to our research, they analyse the impact of different proportions of missingness and the three MDM's (Section 3.3) on accuracy and fairness. In particular, the effect of their reweighting scheme on MAR and MNAR missingness is investigated. We next discuss the results obtained on the Adult and synthetic data set with reference to the figures.

The authors give reweighting results for MAR and MNAR only because as shown by their other experiments MCAR has very little impact on fairness. Figure 4.5 and Figure 4.6 show the average of measurements from 5-fold cross validation, the standard error is not reported as it is very small.

The authors interpret the plots as follows: for MAR the re-weighting algorithm is effective when the percentage of missing values is greater than 5 - 10%. In those cases they can see a 5% to 20% improvement in fairness with little decrease in accuracy and F1 score (less than a 4% decrease). The impact on fairness is larger for all the data sets if the missing values are generated from the MNAR mechanism, as it is correcting a larger bias. In all figures there is a trade-off between fairness and performance. For example in the figures for MAR (Figure 4.5), in the subfigures showing accuracy and F1 score, the blue lines are higher than the red lines for a tradeoff in fairness scores.

The authors conclude that MNAR has the biggest impact on fairness and MCAR has the least impact. Hence all MDM's are not equal when considering fairness. The fixing algorithm for reweighting is able to mitigate the negative effects of the missing values on fairness with a small impact on accuracy.

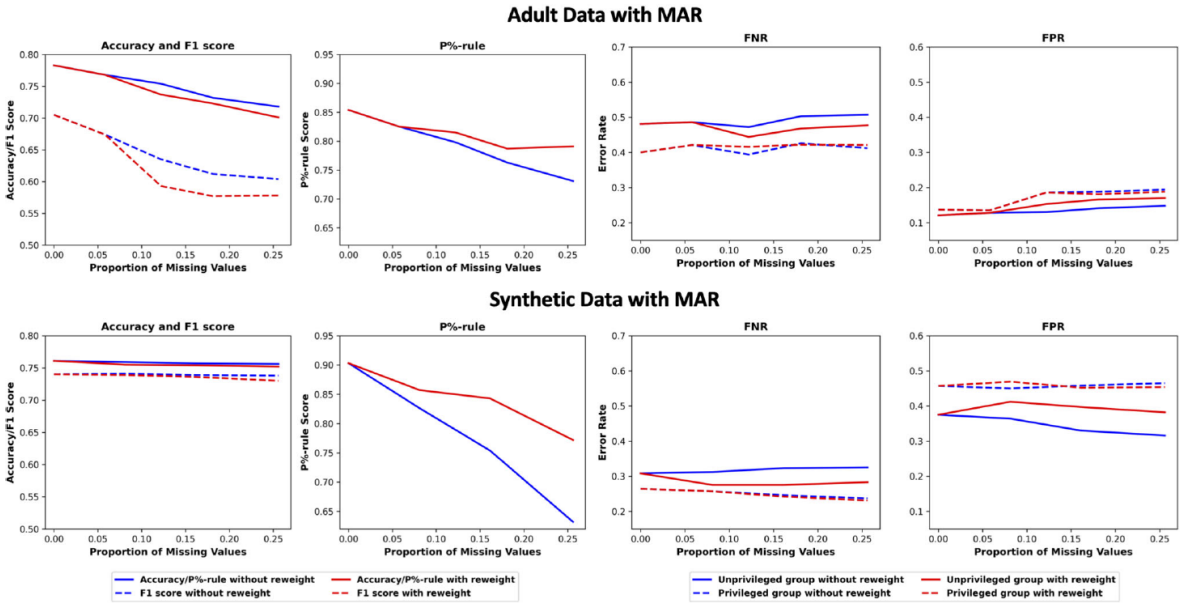

Figure 4.5: Accuracy, F1 score, and fairness measures with MAR before and after reweighting (Wang and Singh, 2021, p. 113)

Lastly, the work in (Zhang and Long, 2021*a*) is less relevant to our own research than the other work reviewed in this chapter. The authors provide theoretical results regarding fairness in the context of missing data. To this end, they define two 'data domains' which are related but also distinct: the 'complete case domain' which contains the complete cases of the data set once we have

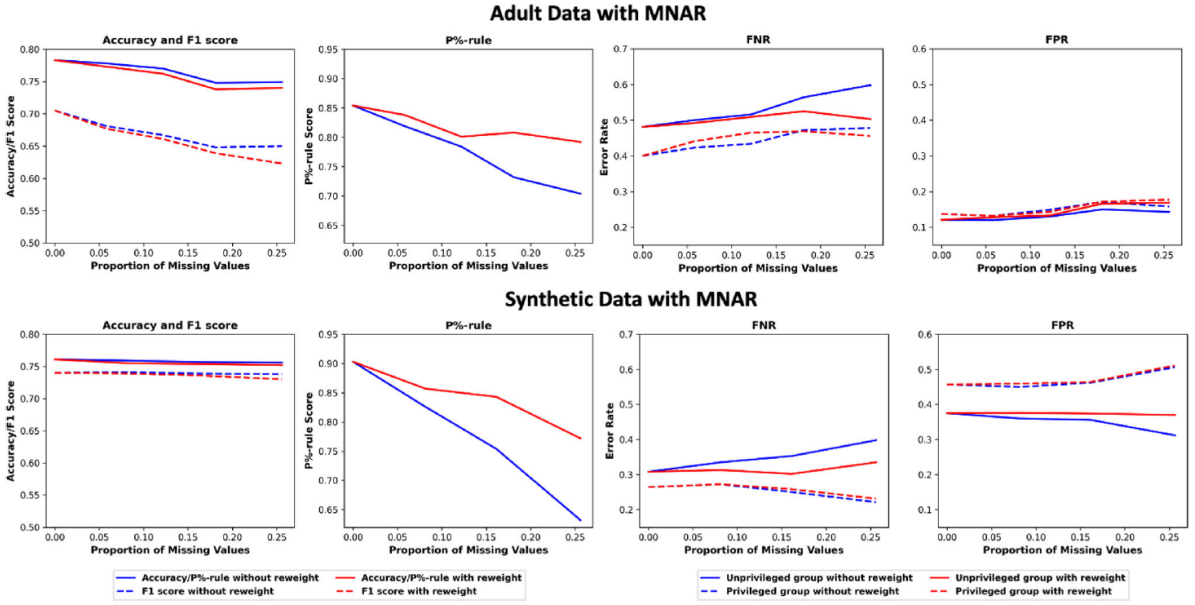

Figure 4.6: Accuracy, F1 score, and fairness measures with MNAR before and after reweighting (Wang and Singh, 2021, p. 114)

removed the incomplete cases, and the 'complete data domain', which refers to the entire data set including both complete and incomplete cases.

Depending on the nature of the missing data, i.e., the MDM, the data distributions in the complete case domain and the complete data domain may be different. In particular, if the MDM is MCAR, the distributions in both domains are the same, but in the case of MAR and MNAR the distributions could be very different, with the result that an algorithm which is fair in the complete case domain may be biased towards a sensitive group in the complete data domain.

The end goal is to measure fairness of an algorithm in the complete data domain. This fairness can be estimated by using the observations in the complete case domain. For their choice of fairness metric, the authors provide upper and lower bounds for the error that is introduced by the domain shift, when estimating fairness in the complete data domain by using observations in the complete case domain. The authors propose that their framework for bounding the fairness estimation error can be adapted to apply to other notions of fairness.

Lastly, we would like to compare the approaches discussed in this chapter to our approach, to point out where the similarities and differences lie.

Of the four papers reviewed in this section, the approach in Zhang and Long (2021*b*) is most similar to ours, but as mentioned earlier in the chapter, there are important differences between the approaches also. We recount these again here. In their research, they generate missingness only in numerical variables whereas we are working with missingness in both continuous and categorical variables. In data sets applicable to the real world, often there are many categorical variables with missing values (as these are used to capture personal data). Having missingness in both types of variables is a more realistic scenario.

The next difference is in how the missingness is generated, they generate it in a univariate manner whereas we are using `R` function `ampute` in package `mice`. Amongst other merits as mentioned before, this function allows one to generate missingness in multiple variables simultaneously, according to the missing data mechanisms. Using the multivariate amputation approach in ampute overcomes the problems of univariate amputation.

The next difference is in the missing data handling procedures used. The imputation methods they use are state of the art (including multiple imputation) and not widely known or used by most researchers. The effects of simpler single imputation techniques on fairness have not been studied as we do in our research. Often only one complete set of data is required by a researcher before they can begin analysis, so it is important to study the simpler techniques despite the superiority of multiple imputation.

The research of Fernando *et al.* (2021) does study the effect of listwise deletion or imputation with mean/mode on the fairness of machine learning models, but their methodology is entirely different; they do not study the above in the context of the MDM's.

In the study by Wang and Singh (2021), missingness in only one categorical variable is investigated. In our research, we use both continuous and categorical predictors with missingness created

in both, as this is a more realistic case, and missingness can be present in one or two variables. Like our study, they analyse the impact of the 3 missing data mechanisms (MDM) on fairness, but in particular the effect of their reweighting scheme on MAR and MNAR missingness is investigated.

A systematic study starting from the foundational questions of how the data are missing (i.e. MDM), how the missing data are dealt with and how this impacts the fairness metrics, calculated on the outcome of a few different types of machine learning algorithms, does not exist. Most researchers when dealing with missing data either delete listwise or tend to use the simpler methods of imputation versus the more complex ones (Eekhout *et al.*, 2012). The impact of these methods on the fairness of an algorithm has not been studied. In our research, we start at this point, in a sense from the beginning.

CHAPTER 5

EXPERIMENTS

5.1 OUTLINE AND AIM OF EXPERIMENTS

Broadly, the aim of this research is to understand the impact of missing data and missing data handling procedures on the fairness of machine learning algorithms. We categorise type of missingness by the MDM (see Section 3.3), of which there are three: Missing completely at random (MCAR), Missing at random (MAR) and Missing not at random (MNAR) (Van Buuren, 2018; Enders, 2022). We will create missingness in our data sets artificially according to these MDM's. Following this, missing data will be dealt with in the usual ways, either by listwise deletion or by imputation (see Section 5.5). Following imputation of the relevant training and test sets, classification models will be created and tuned on training data using 5-fold cross validation, predictions from these models will be calculated on the test data sets and the fairness metric distributions will then be obtained from these predictions. As a baseline to compare to, we will also obtain the fairness metric distributions from complete training and complete test sets, such that no amputation or imputation is performed on these data sets. Details follow in the sections below.

5.2 DATA SETS

We use the below real-world data sets (see Section 5.2.1 to 5.2.3) that are popular in the field of fairness in machine learning (Friedler *et al.*, 2019; Le Quy *et al.*, 2022; Pessach and Shmueli, 2020). We start with the assumption that the data sets are complete (i.e., they contain no missing values), and hence remove any rows with missing values for those data sets which contain missing values. The reason for adopting this approach is that our interest is in the effects of both the MDM and the imputation method on fairness, and hence we must restrict ourselves to situations in which the MDM is known (i.e., created artificially). Note that of the 3 data sets, only German Credit does not contain any missing data. See Appendix A for information on data set attributes.

5.2.1 German credit

The data set is publicly available on the UCI machine learning repository (Dua and Graff, 2017). This data set is comparatively small, containing only 1000 observations. There are 20 predictor variables, including both categorical and numerical variables. We can obtain two sensitive attributes from this data set: Sex, which can be obtained from the attribute *personal-status-and-sex*, and Age, which is considered as a protected attribute after binarisation into {*young*, *old*}. For Sex, males are the advantaged group and females are the disadvantaged group. For Age, above 25 are the advantaged group and ≤ 25 are the disadvantaged group. Once the protected attributes are created, the original versions of these variables are removed from the data set. The ratio of male:female is 690:310 (69%:31%). The ratio of old:young is 810:190 (81%:19%) (the cutoff age of 25 is used to binarise the Age variable). The outcome variable is whether an individual is a good or bad credit risk, with good credit risk the positive outcome. The ratio of bad:good credit risk individuals is 300:700 (30%:70%).

5.2.2 Adult income

The data set is publicly available on the UCI machine learning repository (Dua and Graff, 2017). It contains information about individuals from the 1994 U.S. census. It comes pre-split into a training and test set. We amalgamate both sets to get one data set. Once the rows with missing values are removed, we are left with 45222 observations. There are 13 predictor variables including both numerical and categorical. As in previous work (Le Quy *et al.*, 2022) we also discard the attribute *fnlwgt*. The sensitive attributes are *sex* and *race*. For sex, males are the advantaged group and for race, whites are the advantaged group. The ratio of male:female is 32,650:16,192 (66.9%:33.1%). We convert race into a binary attribute, race = {white, non-white}. The white:non-white ratio is 38,903:6,319 (86%:14%). The outcome variable is whether an individual makes less or more than \$50,000 in yearly income. We take the positive class as >\$50,000. The ratio of positive:negative outcomes is 1:3.03 (25%:75%).

5.2.3 COMPAS

The data set is openly available in a GitHub repository (Store, 2016). It refers to data collected about the use of the COMPAS risk assessment tool in Broward County, Florida. Recidivism risk

scores are calculated based on defendants responses to the COMPAS survey. The data set contains 52 predictors, but we follow the process in Le Quy *et al.* (2022) and keep 11 of these. These 11 contain both numerical and categorical variables. The number of observations is 7214. The sensitive attributes are *race* and *sex*, with whites the advantaged group and non-whites the disadvantaged group and with females the advantaged group and males the disadvantaged group. The non-whites:whites ratio is 4760:2454 (66%: 34%). The male:female ratio is 5819:1395 (81%:19%). The outcome variable is *two year recid* indicating whether they were rearrested within two years after the first arrest. The positive outcome is not being rearrested, with ratio of negative:positive being 3251:3963 (45%: 55%).

5.3 DATA AMPUTATION

The task of data amputation in a complete data set is undertaken with the `ampute` function (Schouten *et al.*, 2018) in `R` package `mice` (Van Buuren and Groothuis-Oudshoorn, 2011; R Core Team, 2022). With `ampute` we are able to generate missing values in multiple variables, with different missing data proportions and missingness mechanisms. The merits of using the multivariate `ampute` function rather than the current practice of generating missingness in one variable at a time are detailed in Schouten *et al.* (2018) along with the pitfalls of the univariate amputation approach.

To summarise the multivariate amputation procedure, the schematic in Figure 5.1 is presented from Schouten *et al.* (2018). We will now explain briefly how the function works with reference to the schematic, and explain how we used the function for our experiments. We list the important arguments of the function, and explain the values we specified for them.

- **prop**: With this argument we can specify what proportion of cases will have missing values. The default missingness proportion is 0.5, and with no concrete reason to do so otherwise, we use the default proportion. A proportion of 0.5 may seem high, but this may allow us to see the effects of amputation more clearly.
- **patterns**: A missingness pattern is a particular combination of variables with missing values and variables remaining complete. If the number of patterns we have is k then our data set is randomly divided into k subsets, the sizes of these subsets can vary.

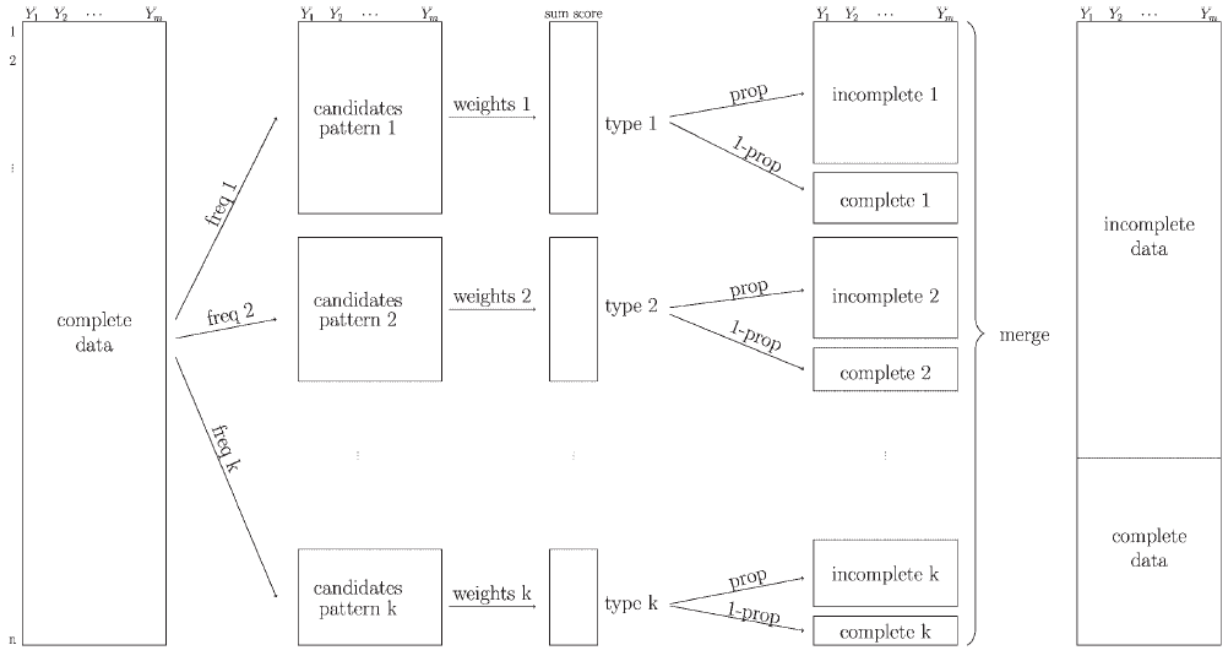

Figure 5.1: Multivariate amputation schematic from Schouten *et al.* (2018)

In our case, missingness will either be created in just one variable, in which case there is only 1 pattern: missingness created in that variable, or it will be created in two variables, in which case we specify 3 patterns: two patterns of missingness created in one variable each, and then another pattern where missingness is created simultaneously in both variables.

The variables chosen for creating missingness in will have one of these properties: strongly associated to the sensitive variable, strongly associated to the outcome variable, or strongly associated to both. The associations will be established by variable importance plots resulting from random forest models.

- **freq**: This argument allows us to vary the size of the subsets created previously through the patterns argument. The sum of the frequency values should always be 1 in order to divide all the cases over the subsets. In our case, we either have 1 pattern, in which case a frequency of 1 is assigned to this pattern, or we have 3 patterns: the patterns with missingness in one variable only are given frequency 0.4 each, and the pattern where missingness is created

simultaneously in both variables has frequency 0.2.

- **mech**: This argument allows us to set the MDM. For MAR, in our case we generate missingness dependent on the sensitive variable only, this is done for simplicity and consistency through the missingness scenarios. A higher proportion of missingness is created for the disadvantaged group, in line with the assumption that propensity of data being missing is often tied to socio-economic status or demographic characteristics of individuals, with the disadvantaged group being more reluctant to give certain information (Fernando *et al.*, 2021).

  If we generate MCAR missingness, this would be the last argument to specify, whereas if we want to generate MAR or MNAR missingness, there are two further arguments which need to be specified.

- **weights**: The weights that we specify in this argument are used to create weighted sum scores for MAR and MNAR, which determines whether a case receives missing values or not. Based on its weighted sum score, a case is allocated a probability of being missing. The weights are chosen by the user. In our case, for MAR we just assign a weight of 1 to the sensitive attribute. For MNAR, for patterns with one variable missing we assign a weight of 1 to that variable, and where both variables are missing we assign a weight of 1 to the first variable (although missingness is created in two variables, the probability of both being missing depends only on the first). In our case there is no particular reason or need to assign a weight other than 1 to a variable. This argument is used in conjunction with the *type* argument.

- **type**: This argument is used in order to allocate each case a probability of being missing, according to one of four possible logistic distribution functions which is applied to the weighted sum score. For instance, if a right-tailed (RIGHT) type of missingness is used, candidates with high weighted sum scores will receive a high probability of being missing. With a left-tailed (LEFT), centred (MID) or both-tailed (TAIL) missingness type, higher probability values are given to the candidates with low, average or extreme weighted sum scores respectively. See Figure 5.2.

The data are randomly divided into k subsets each time the amputation is run, for each data set we run the amputation i times. For instance, for the Adult data set, $i = 50$ whereas for the German

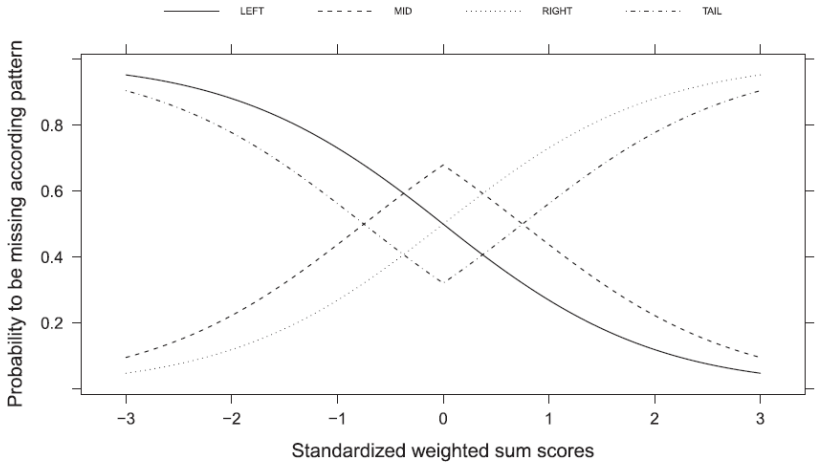

Figure 5.2: Four variants of the logistic distribution function (Schouten *et al.*, 2018)

credit data set, $i = 100$ (the difference in i here is because the Adult data set has about 50k rows, whereas the German credit data set has 1000 rows only, running the code on the former for $i = 100$ would be too time consuming). In brief, each of these i data sets is next divided into training and test sets, which are then imputed accordingly (see Section 5.5) and used to train the classification models, after which the fairness metric value is obtained from predictions on the test set.

5.4 TRAINING AND TEST SETS

Each of the amputed data sets is split up into an amputed training set and an amputed test set. Using the same indices, the complete data set is split up into a complete training set and complete test set. The test set is a third of the data set and the remaining is the training set. The complete training and test sets are used to provide the baseline distributions of the metrics. The amputed training sets are next processed according to an imputation method or listwise deletion. Regarding the results, these will be calculated on either the complete test sets or the imputed test sets.

To clarify the reasons for a complete test set (contains no missing values) and an imputed test set (this test set started off with missing values but was imputed according to the imputation model on the training set): On the imputed test set we only compare the imputation techniques to each other and the baseline. On the complete test set we compare listwise deletion (LD) to the other imputation methods and the baseline. Having the comparison of LD with the imputation

techniques is a relevant comparison because of the frequency with which people perform LD despite its obvious limitations. Having a test set with missing values is more realistic, but with this we cannot compare LD as then we would have to delete those test rows with missing values, which would not be done in reality and even if it was, we then would not have the same size test sets to compare LD with the imputation techniques. Having a complete test set allows us to compare LD to the imputation methods.

The training set is used to perform 5-fold cross-validation to choose the best parameters for each type of classification model. The 'best' model is defined as either one that maximises accuracy, or one that maximises Matthew's correlation coefficient (MCC). The latter is used for data that is imbalanced in the outcome classes (Chicco and Jurman, 2020). Once the best model has been selected through cross-validation, it is fitted on the full training set and predictions of the model are obtained from the test set. More on model tuning will follow later.

5.5 DATA IMPUTATION

The missing data methods that will be investigated are listwise deletion and three single imputation techniques: mode imputation (rather than mean imputation, because missingness will be created in categorical variables as well as numerical/continuous variables), regression imputation and k-NN imputation. The latter two will be implemented using the `R` package `VIM` (Kowarik and Templ, 2016). The first two are straightforward to program manually.

Despite there being a large array of `R` packages which implement single imputation methods, we chose the package `VIM` over other packages for two reasons: First, the variables in our choice of data sets are a mixture of continuous and categorical. Most other such packages are not designed to deal with mixed data sets. Second, imputation methods like k-NN imputation using a generalised distance function (which we require) are not available in these packages. `VIM` solves both these issues (Kowarik and Templ, 2016). To calculate distance between two observations it uses an extension of the Gower distance function.

As a note, in our study both regression imputation and k-NN imputation will use the sensitive

variable when creating the regression model or calculating distance in k-NN (the imputation model is not using the sensitive attribute to differentiate between the outputs so it is reasonable to do this). The test sets which are imputed are done so by using the imputation model on the corresponding training set.

5.6 CLASSIFICATION MODELS

Model selection is performed on the training set using 5-fold cross validation. The best model is either one which maximises accuracy or one which maximises MCC. Regarding the range of models, we chose to implement some of the more popular ones. Logistic regression gives a linear separation boundary, but is not very flexible, whereas random forests are flexible and a good fit for a non-linear separation situation. Boosting (trees) again gives a very flexible model. SVM is robust to outliers and the radial kernel method gives a flexible model. The focus of this research is not on the classification models themselves, we are implementing the standard ones that are available (James *et al.*, 2013).

Parameter tuning and model selection:

- **Logistic regression**: The probabilities that result from the model are converted to class membership with the classification threshold. The threshold which classifies an observation to outcome 0 or 1 is varied between 0.05 to 0.95 inclusive.
- **Random forests**: Most of the parameters are left at their default, for instance the number of variables considered at each split remains at $\sqrt{p}$ where p is the number of predictors. The number of trees grown is also left at its default value of 500. The 'cutoff' argument in the `R` function used, `randomForest`, is varied. This parameter has a similar effect as varying the threshold in logistic regression.
- **Boosting**: The number of trees to fit is left at the default value of 100. The interaction depth is also left at the default of 1. The shrinkage parameter λ is tuned over 4 possible values: 0.001, 0.01, 0.1, 0.2. As for the logistic regression case, we also tune over the classification threshold value.
- **Support vector machines**: The cost argument is the default value of 1. Model selection

includes a choice between a linear or radial kernel. The value of γ for the radial kernel is left at the default of $\frac{1}{number\ of\ predictors}$. As per default the data are scaled. We tune the *class.weights* parameter which is used for asymmetric class sizes.

Note that the classification models will not use the sensitive variable under consideration as a predictor.

5.7 FAIRNESS METRICS

Per data set two sensitive attributes each will be considered. We will calculate the following fairness metrics on our data sets: demographic parity, equality of opportunity, predictive equality and predictive parity (see Section 2.4 for details of the metrics).

5.8 SUMMARY

The following steps are repeated for the number of iterations required (for example, for the German credit data set we run 100 iterations):

1. Ampute the data set according to an MDM, where the MDM $\in$ {MCAR, MAR, MNAR}.
2. Create training and test splits of the amputed and complete data set, using the same indices, with two-thirds training and one-third test.
3. Impute the amputed training set where imputation models $\in$ {listwise deletion, mode, regression, k-NN}. Impute the amputed test set with the same model that is used to impute the training set. So now we have the following sets: complete training, complete test, imputed training and imputed test.
4. The complete sets are used to calculate baseline results. Regarding the training sets, we perform 5-fold cross validation for model selection for the following classification models: {logistic regression, random forests, boosting and support vector machine}. For each model, the parameters which maximise accuracy (or maximise MCC) are chosen.
5. The chosen model is fit on the entire training set for both complete and imputed training sets. Predictions from the model are calculated on the complete and imputed test sets. The fairness metric values are calculated from the predictions.

Repeating the above steps for the number of iterations gives us a distribution of fairness metric values.

5.9 COMPUTATIONAL ISSUES

Regarding application of the imputation models and/or classification models discussed above, it must be noted that due to limitations on time and available computing resources, some of these may not be applied to a particular data set. For example, the Adult income data set has around 50,000 rows, we ran all the code including imputation and classification models on this data set for one iteration only, and it took around 8 hours[1]. To run it for 50 or so iterations (to get a large enough sample size) for the total of 6 scenarios would take in excess of 3 months, which was not viable. See Appendix B for further details.

[1] A virtual machine was used, with dual Intel Xeon Gold 6136 processors and 256 Gb RAM.

CHAPTER 6

RESULTS

6.1 PRELIMINARIES

As explained previously, the aim of this research is to understand the impact of missing data and missing data handling procedures on the fairness of machine learning algorithms. We will create missingness in our data sets artificially according to the MCAR, MAR or MNAR MDM. Following this, missing data will be dealt with in the usual ways, either by listwise deletion or by imputation. In our results, we will focus on the following three metrics: demographic parity, equality of opportunity and predictive equality. Throughout, 'baseline' results refers to the fairness metric values obtained on the complete training and complete test sets. These will be displayed on the plots in red, as median and interquartile range of the baseline metric values, to allow useful comparison with the boxplots.

When we refer to variable/s being 'strongly related' to another variable, this means strong in a relative sense to the other predictor variables in the data set. The strength of the relationship has been inferred from variable importance results from application of random forest predictive models. We use random forest variable importance to choose variables in which we create missingness, by seeing how important they are for the outcome variable, the sensitive variables or both.

Again we remind the reader of the reasons for a complete test set (contains no missing values) and an imputed test set (this test set started off with missing values but was imputed according to the imputation model on the training set): On the complete test set we compare LD to the other imputation methods. On the imputed test set we only compare the imputation techniques to each other. Having a test set with missing values is more realistic, but with this we cannot compare LD as then we would have to delete those test rows with missing values, which would not be done in reality and even if it was, we then would not have the same size test sets to compare LD with the imputation techniques. Having a complete test set allows us to compare LD to the imputation methods. We also use the complete sets to calculate the baseline results.

Next, we will report on a subset of the total set of results from all 3 data sets which we believe to be interesting or instructive. We see if we can observe any commonalities in metric behaviour across the data sets. We will only look at results from models which maximise accuracy, as the fairness results do not differ much between maximising accuracy or maximising MCC models, which is a noteworthy observation in itself. Note, for descriptions we will at times generate one instance of an amputed then imputed vs complete training/test set, displaying the variable/s which are being amputed and imputed, to gain insight into why the fairness metrics behave as they do.

We introduce some shorthand notation which is displayed on the plots to follow. The shorthand for the models is as follows: 'lr' is logistic regression, 'rf' is random forests, 'b' is boosting and 'svm' is support vector machine. Listwise deletion will be denoted 'ld'. With regards to imputation, we use 'knn' to denote k-NN imputation, 'reg' is for regression imputation, and 'mode' to denote mode imputation. Plot headings will be given as 'imputation method.classification model'. So, for instance, if a plot heading says 'reg.rf' then the imputation technique used is regression imputation, and the classification model fitted is random forests. It will be clear from the context whether the test sets are also imputed (with the same technique as the corresponding training sets). Regarding the fairness metrics, demographic parity will be denoted 'dem.parity', equality of opportunity will be denoted 'eq.opp' and predictive equality will be denoted as 'pre.eq'.

Because of the manner in which the fairness metrics are determined, there are numerous possible combinations of effects which could result in the fairness levels being the same under different scenarios (e.g. baseline and ampute then impute). For example, if demographic parity is the same under the baseline scenario and under amputation/imputation then it could be that neither of the proportions of favourable outcomes for the different sensitive groups changed substantially, or it could be that both decreased by a similar amount, or both increased by a similar amount. Throughout the following discussion we will attempt to identify the simplest explanation for the results we observe, and so in anticipation begin by providing a few preliminary explanations which apply to multiple of the sets of results which follow, so as to avoid repetition:

1. Complete test set: if the fairness results on the complete test sets and the baseline results (which are calculated on the complete test sets) are similar, then one possibility is that the

models on the complete and imputed training sets are similar, which could happen because the two data sets themselves are similar. This would mean that the imputation is reasonably faithful to the true data values. Denoted **statement 1**.

As an example, Figure 6.1 shows COMPAS data with sensitive attribute race, equality of opportunity metric for the random forest model across listwise deletion, mode or regression imputation. The baseline distribution and the metric distribution on the complete test set are similar irrespective of MDM, and statement 1 would be applicable here. It is a point of interest that with listwise deletion, we would expect that MCAR missingness distribution is most likely similar to the baseline, as we can expect the remainder of the observations after listwise deletion to represent a simple random sample of the training data in this case, but the same pattern is shown for MAR and MNAR also.

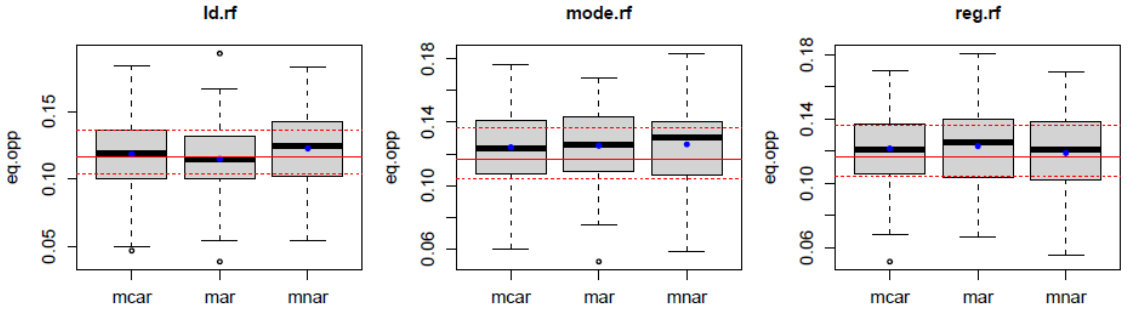

Figure 6.1: COMPAS data, sensitive attribute race, equality of opportunity

2. Imputed test set: if the fairness results on the imputed test sets and the baseline results are similar, then one possibility is that the models on the complete and imputed training sets are similar, and that the imputed test sets are similar to the complete test sets. This could happen because the complete and imputed data sets themselves are similar, including the test sets. This would mean that the imputation is reasonably faithful to the true data values, both for the training and test sets. Denoted **statement 2**.

As an example, Figure 6.2 shows COMPAS data with sensitive attribute race, the 3 fairness metrics with regression imputation and random forests model on the imputed test sets, where this statement would be applicable. In all 3 plots, the metrics distribution and the

baseline distribution overlap, irrespective of the MDM. In this case we may say something like 'For the COMPAS data set with sensitive attribute race, for regression imputation for all 3 metrics and all 3 models, irrespective of MDM, statement 2 holds.

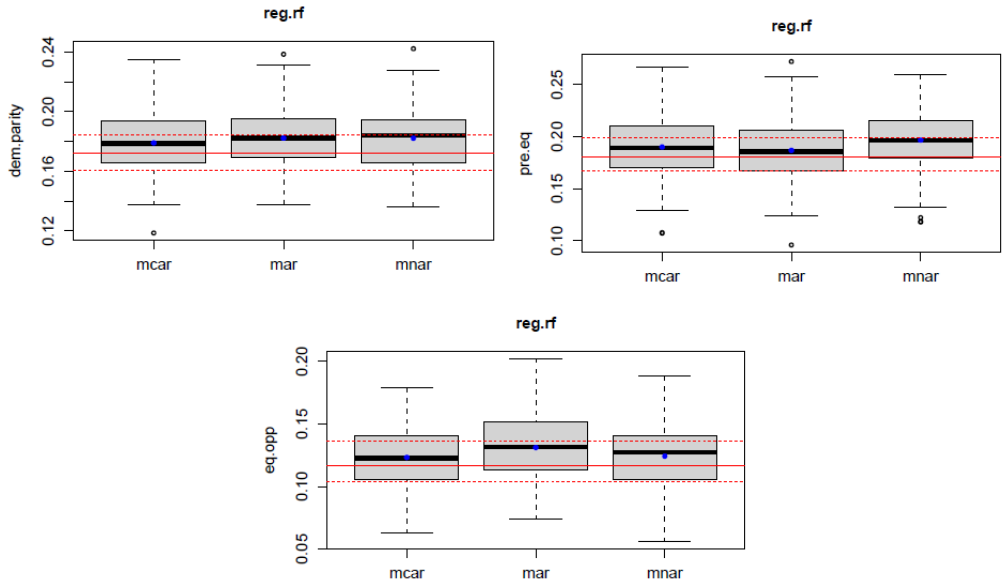

Figure 6.2: COMPAS data, sensitive attribute race, regression imputation and RF model

In the subsequent cases if statement 1 or 2 above is applicable to any results, for brevity we will just refer the reader to **statement 1** or **statement 2**.

6.2 COMPAS DATA SET

For the COMPAS data set, we only create missingness in variable/s strongly related to **both** the sensitive attribute and outcome variable, as the importance plots for either relationship separately gave overlapping variables for both. For reasons explained in Chapter 5 we will not fit svm or perform k-NN imputation on this data set.

6.2.1 Sensitive attribute: Race

6.2.1.1 Missingness created in variable/s strongly related to both sensitive attribute and outcome variable

- **Imputed test set**: Regarding mode imputation and the MAR MDM, there is a clear trend of reduction in discrimination on average from the baseline values for all 3 metrics. This is demonstrated for the logistic regression model in Figure 6.3. We obtain similar boxplots for the random forest and boosting models. The two variables in which missingness is created are 'age' and 'score text'.

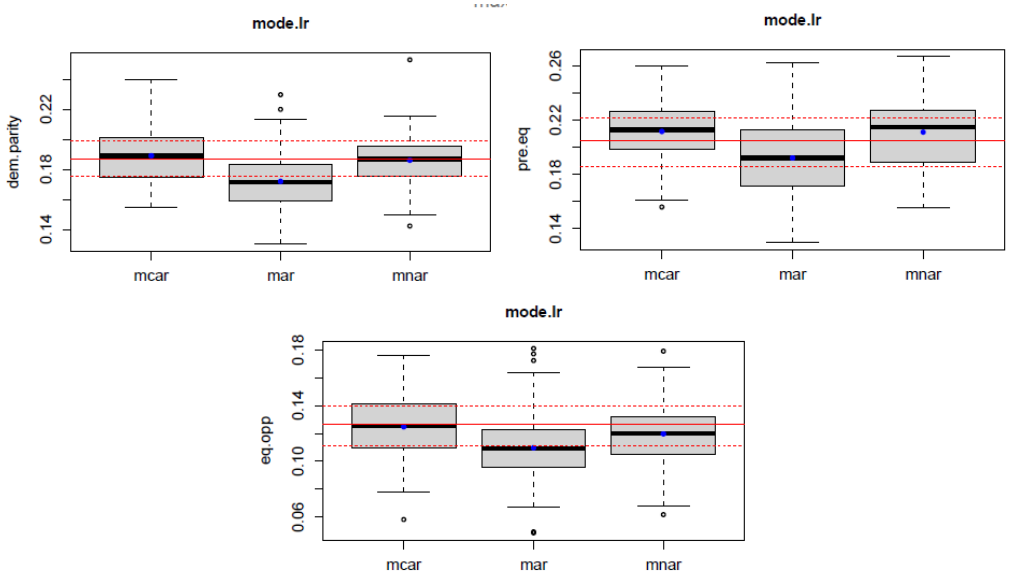

Figure 6.3: COMPAS data, sensitive attribute race, mode imputation, LR model

One possible explanation for Figure 6.3 is as follows: from Figure 6.4 we see that, with amputation by MAR, then imputation by mode, on the imputed training set, the distributions of variable 'age' with respect to outcome 0 or 1 are now less discernible than on the complete training set. This would likely lead to lower accuracy of the models fitted. With MAR, more missingness is created for the disadvantaged group, i.e., non-whites, which possibly leads to higher inaccuracy for them after imputation, and potentially more of the 0 predictions now being predicted as 1.

We can similarly explain the 'score text' variable, see Figure 6.5. The actual categories of this

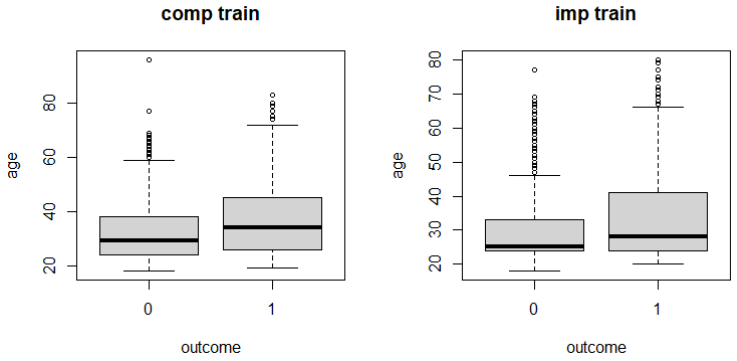

Figure 6.4: COMPAS and race: age distribution by outcome in imputed training set

variable are not important in this case, but we look at the overall pattern of the distributions. The distributions of score text with respect to outcome 0 or 1 are now less discernible than on the complete training set.

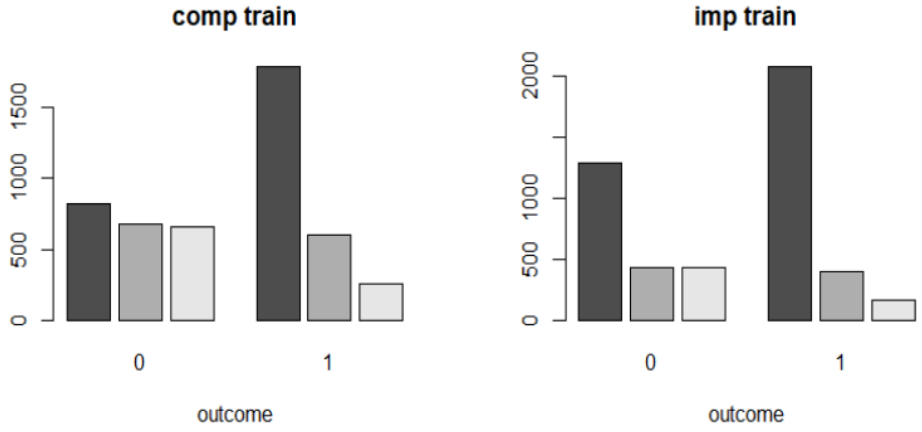

Figure 6.5: COMPAS and race: score text distribution by outcome in imputed training set

For regression imputation for all 3 metrics and all 3 models, irrespective of MDM, statement 2 holds.

- **Complete test set**: For all 3 metrics, all 3 models, listwise deletion and mode and regression imputation, statement 1 holds.

6.2.2 Sensitive attribute: Sex

6.2.2.1 Missingness created in variable/s strongly related to both sensitive attribute and outcome variable

- **Imputed test set**: For mode imputation, for all three models and metrics, compared to the baseline, for MCAR and MNAR the discrimination increases on average, and for MAR the discrimination sometimes increases and sometimes remains within the baseline on average. See Figure 6.6 where we show the results for demographic parity. Interestingly, the increase from the baseline is smallest for the random forest model.

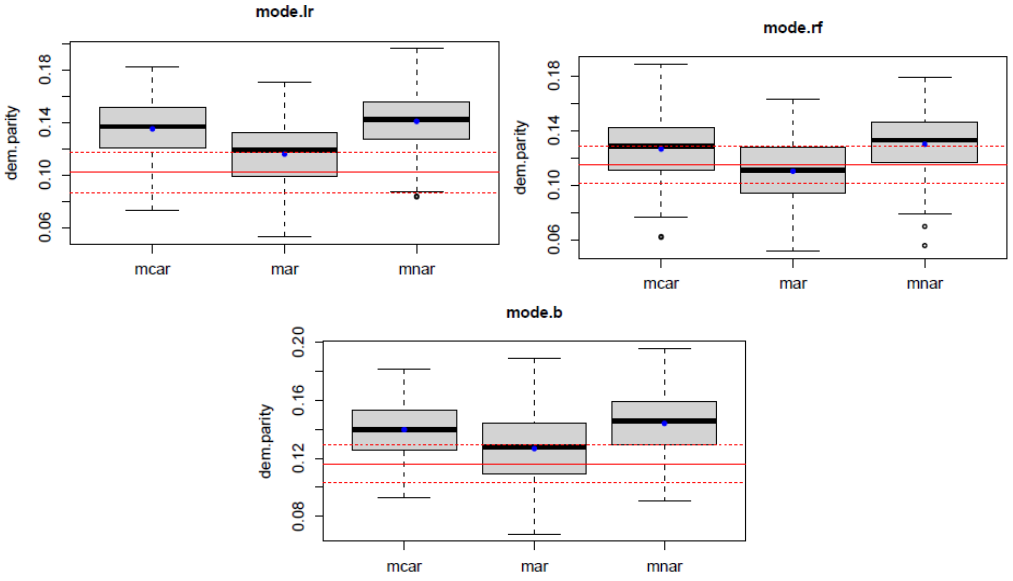

Figure 6.6: COMPAS data, sensitive attribute sex, mode imputation, demographic parity

As a possible reason, we see that on the imputed training set, the distributions of 'age' and 'score text' with respect to outcome 0 or 1 are now less discernible than on the complete training set. As an example, see the age and score text variable distributions from MNAR in Figures 6.7 and 6.8 (Again for the score text variable the actual categories of the variable are not important here). This would likely lead to lower accuracy of the models fitted and misclassifications in both directions for the sensitive groups.

For regression imputation for all 3 metrics and all 3 models, irrespective of MDM, statement 2 holds.

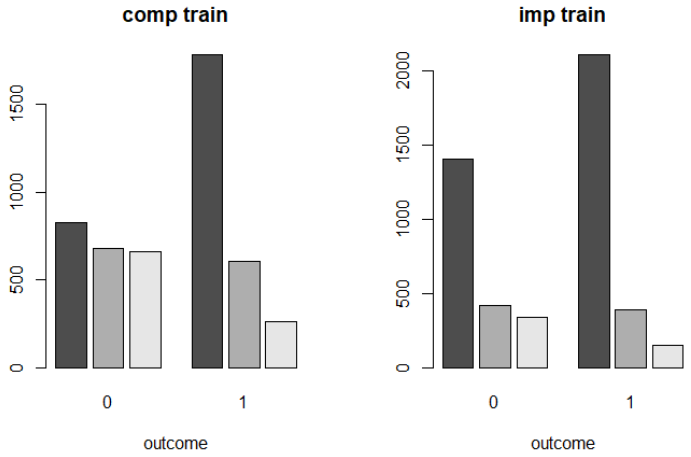

Figure 6.7: COMPAS: score text distribution by outcome in imputed training set, mode imputation

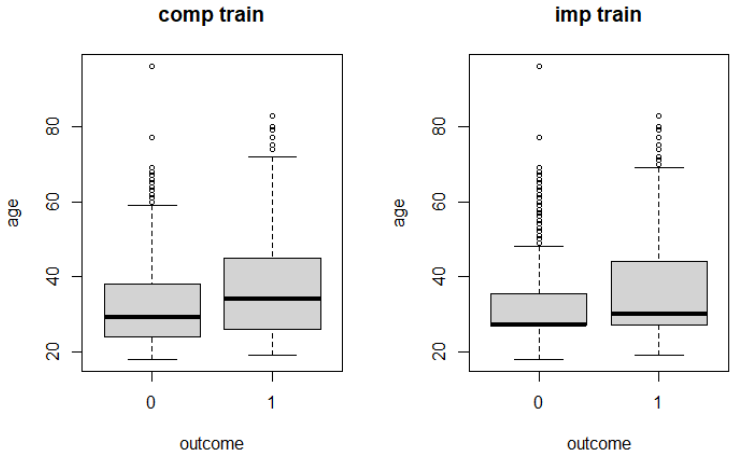

Figure 6.8: COMPAS: age distribution by outcome in imputed training set, mode imputation

- **Complete test set**: For all 3 metrics, all 3 models, all imputation techniques (including listwise deletion), statement 1 holds.

6.3 GERMAN CREDIT DATA SET

All classification models and imputation models are fit for this data set as it is of comparatively small size.

6.3.1 Sensitive attribute: Sex

6.3.1.1 Missingness created in variable/s strongly related to sensitive attribute

- **Complete test set**: For all 3 metrics, all 4 classification models and all 4 imputation techniques, statement 1 holds. See Figure 6.9 displaying demographic parity as an example. It shows demographic parity for random forests model and all 4 imputation techniques. We can see that irrespective of the MDM the fairness metric distribution is similar to the baseline. It is interesting to see that the metric distributions for listwise deletion are similar to those of the imputation techniques.

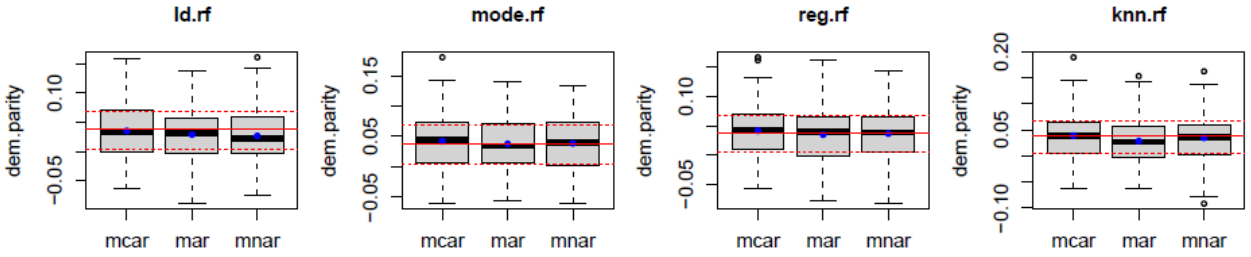

Figure 6.9: German data, sensitive attribute sex, complete test set, RF model

- **Imputed test set**: For all 3 metrics, all models and all imputation methods, statement 2 holds. See Figure 6.10 as an example, which shows the equality of opportunity metric for the SVM model.

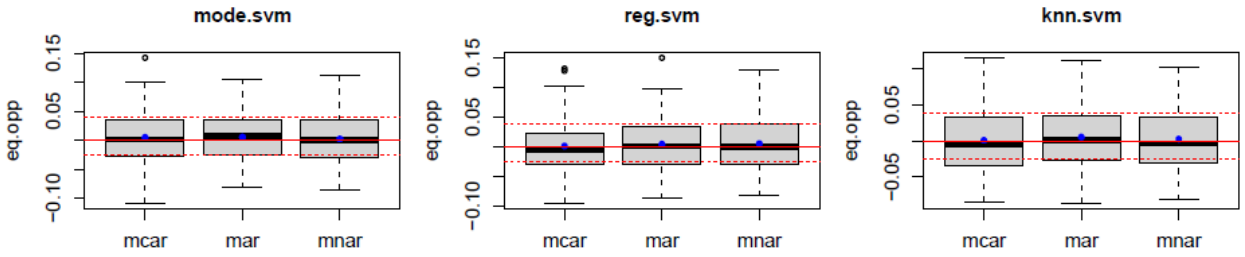

Figure 6.10: German data, sensitive attribute sex, imputed test set, SVM model

6.3.1.2 Missingness created in variable/s strongly related to outcome variable

- **Complete test set**: For all 3 metrics, all 4 classification models and all 4 imputation techniques, statement 1 holds.

- **Imputed test set**: In the case of mode and MAR, we see a reduction on average in all 3 metric values from the baseline. See Figure 6.11 as an example, which shows all 3 metrics for the boosting model.

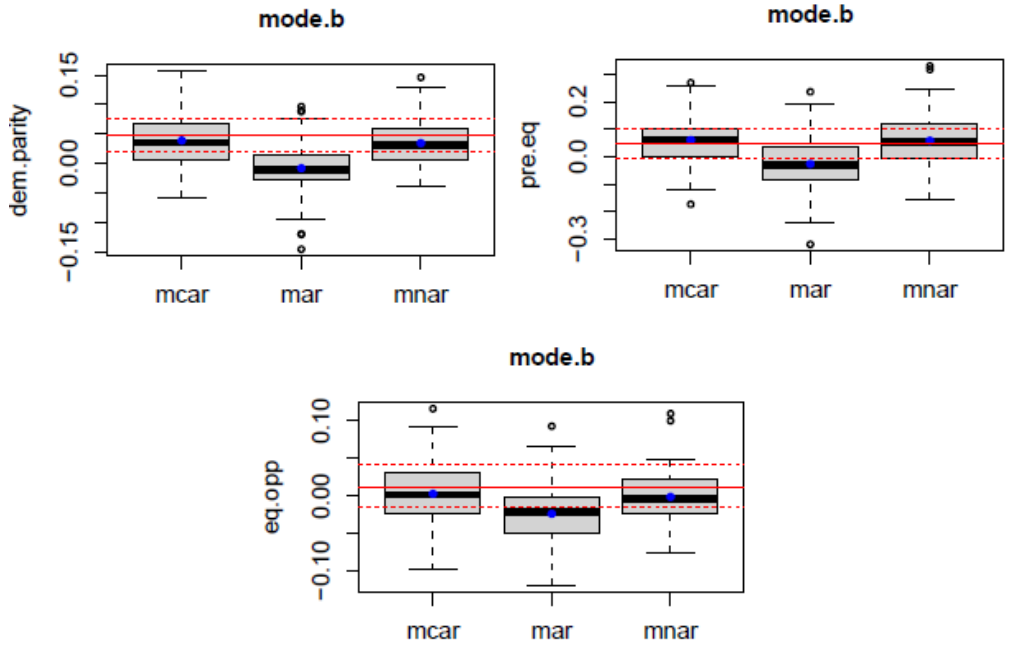

Figure 6.11: German data, sensitive attribute sex, mode imputation, boosting model

As a possible explanation, see Figure 6.12; the variable in which missingness was created is 'checking account status'. In the imputed training set the distributions with respect to outcome of this variable are slightly less discernible than in the complete training set. This would increase the inaccuracy of the predictive models. Despite the reduced discernibility of outcome 0 from 1, it still shows that the modal category, category 4, is more strongly linked to outcome 1 (the last bar in each bar plot is category 4). In MAR, more missingness is created for the disadvantaged class (females), and the missing values are replaced with the modal value 4. The mode is related to outcome 1, so it is possible that in the disadvantaged class many observations which were first being predicted as 0 are now predicted as 1. This would reduce the discrimination.

Interestingly, for k-NN imputation and predictive equality, for all 4 classification models we see a reduction in discrimination for MAR. See Figure 6.13.

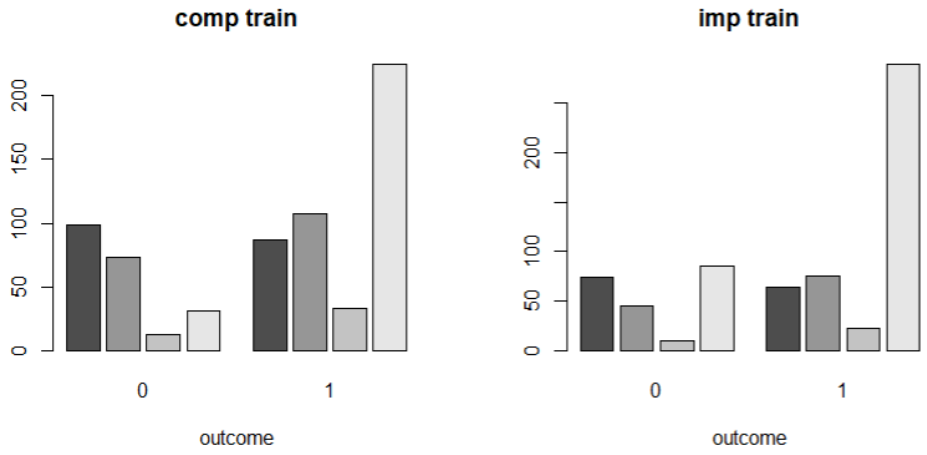

Figure 6.12: German data, sensitive attribute sex, checking account variable

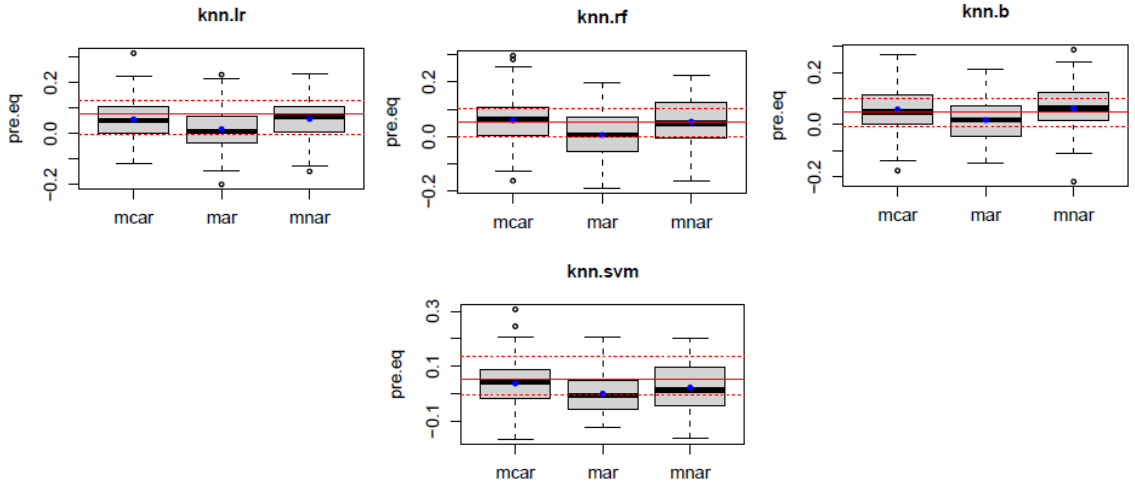

Figure 6.13: German data, sensitive attribute sex, k-NN imputation

6.3.1.3 Missingness created in variable/s strongly related to both sensitive attribute and outcome variable

- **Complete test set**: For all 3 metrics, all 4 classification models and all 4 imputation models, statement 1 holds.
- **Imputed test set**: For all 3 metrics, all 4 classification models and all 3 imputation methods, statement 2 holds.

6.3.2 Sensitive attribute: Age

6.3.2.1 Missingness created in variable/s strongly related to sensitive attribute

- **Complete test set**: For equality of opportunity and predictive equality, for all 4 classification models and all 4 imputation techniques, statement 1 holds. In the case of Demographic parity and listwise deletion, we see a reduction on average in the metric value for MAR for some models. See figure 6.14.

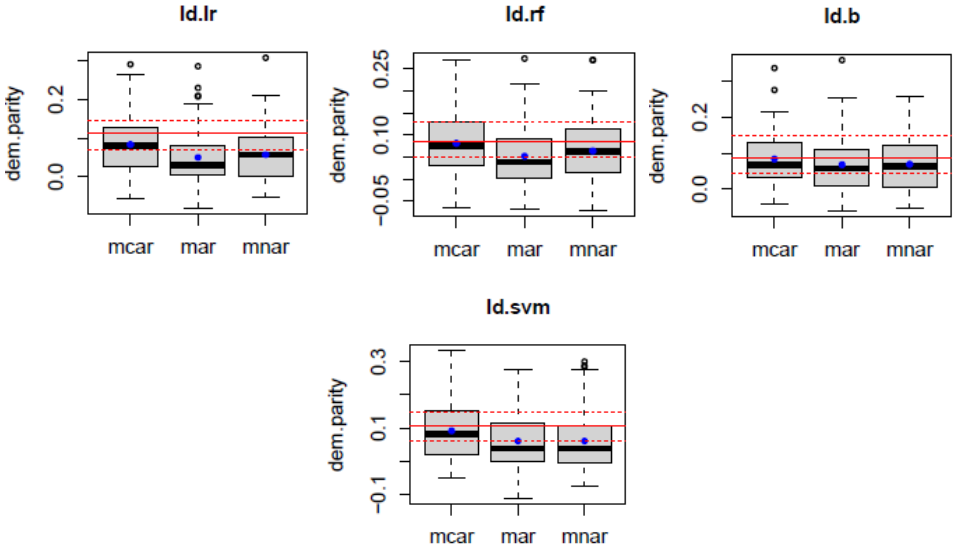

Figure 6.14: German data, sensitive attribute age, LD and demographic parity

A possible explanation: The variables which have missingness are 'employment since' and 'housing'. For MAR, more missingness is created in the disadvantaged class (young). With listwise deletion, these rows are removed. This means that compared to the complete training set, there will now be very little training data for the young group, see figures 6.15 and 6.16. Again the categories of the variables are not important, rather we are showing the reduction in data amount for the disadvantaged group. It is likely that the model trained on this data is less accurate for the disadvantaged class. As an example, see Tables 6.1, 6.2 and 6.3, from an instance of MAR amputed data, where the outcome variable by age has the given proportions remaining after listwise deletion.

Given the removal of a large amount of the data for the disadvantaged (young) class, the

	out	
age	0	1
0	59	63
1	157	388

Table 6.1: Complete training set

	out	
age	0	1
0	6	6
1	102	247

Table 6.2: Listwise deletion training set

predictions are more likely to match the outcome proportions in the advantaged group. Hence it is possible that now a higher proportion of 1's are predicted for the disadvantaged group, leading to the reduction in demographic parity value for MAR.

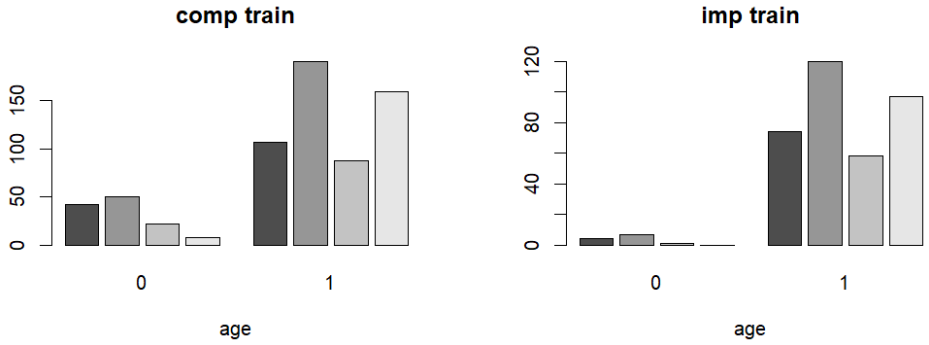

Figure 6.15: German credit data: variable employment since, LD

- **Imputed test set**: For all 3 metrics, all 4 classification models and the 3 imputation techniques, statement 2 holds.

	out	
age	0	1
0	0.102	0.095
1	0.650	0.637

Table 6.3: Proportions remaining

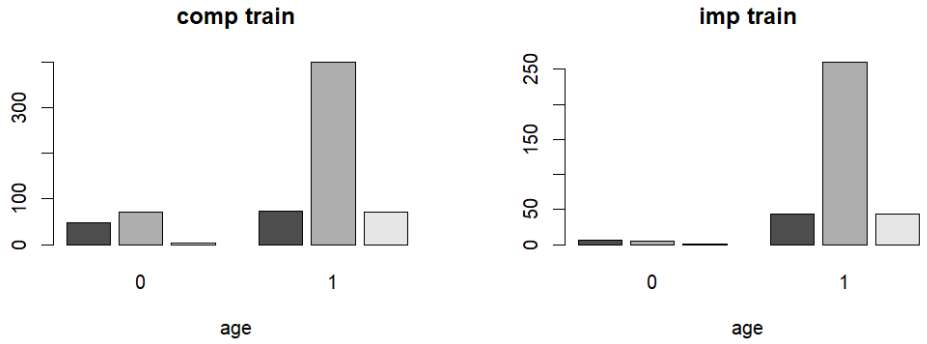

Figure 6.16: German credit data: variable housing, LD

6.3.2.2 Missingness created in variable/s strongly related to outcome variable

- **Complete test set**: For equality of opportunity and predictive equality, for all 4 classification models and all 4 imputation techniques, statement 1 holds. In the case of demographic parity and listwise deletion, we see a reduction on average in the metric value for MAR for some models, see Figure 6.17. Reason analogous to previous section 'Complete test set' (see section 6.3.2.1).

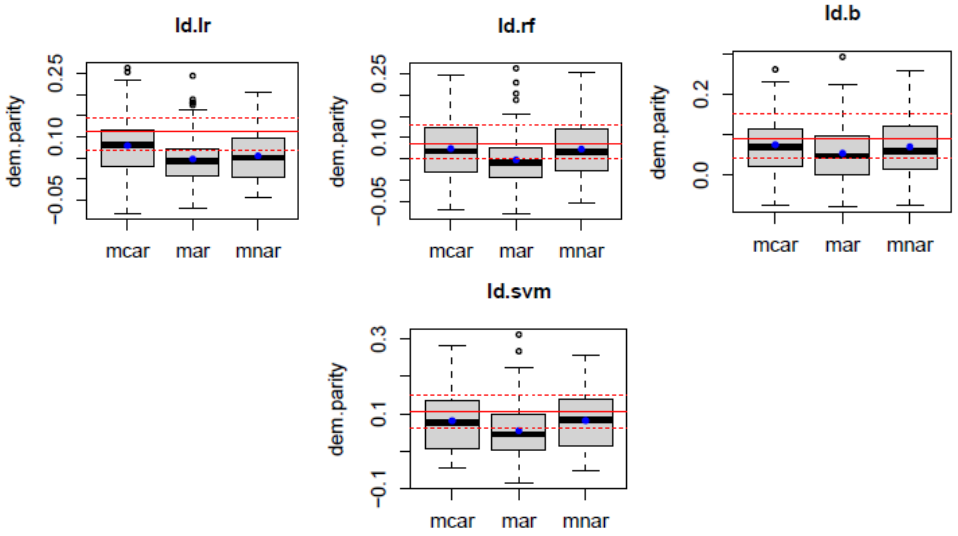

Figure 6.17: German data, sensitive attribute age, LD and demographic parity

- **Imputed test set**: For demographic parity and predictive equality, we see reduction on average from the baseline in the metric value for MAR and mode imputation. See Figure

6.18. Possible explanation: One of the variables where missingness is created is checking account status. From Figure 6.19 we see that the distribution of this variable with respect to outcome becomes slightly less discernible after imputation with mode, which will likely lead to reduction in accuracy of the models, but we can still conclude that the modal value 4 is linked to an outcome of 1 (the last bar in the bar plots is category 4).

From Figure 6.20, we see that the distribution of the variable checking account status with respect to sensitive attribute changes: with MAR a higher proportion of missingness is created for the young group, which is then imputed with the modal value 4, with the last bar in the bar plots representing category 4, the mode (We assume that the checking account status variable complete test distribution is similar to the complete training set distribution). This along with decreased accuracy could lead to a higher proportion of predictions of 1 for the young group, reducing the discrimination.

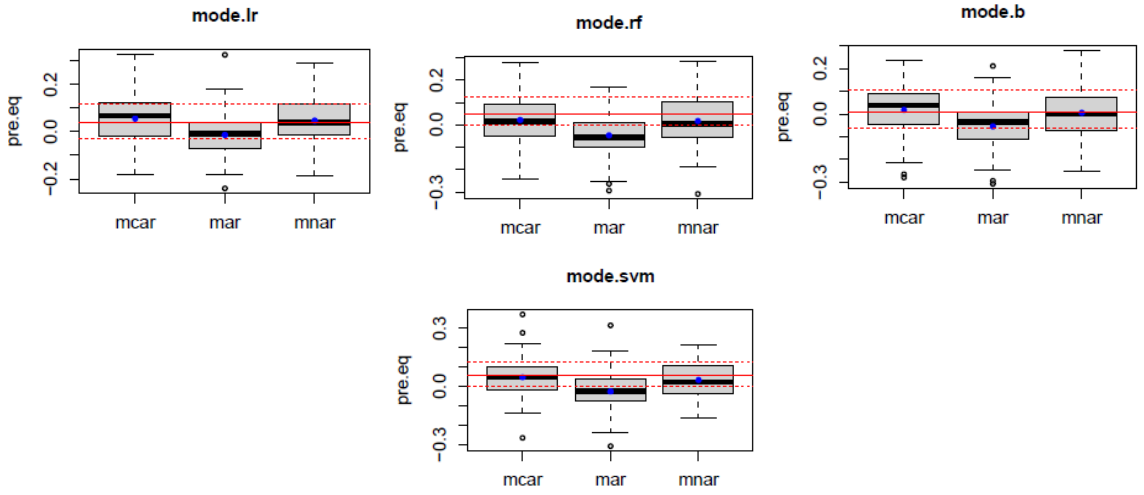

Figure 6.18: German data, sensitive attribute age, predictive equality, mode imputation

6.3.2.3 Missingness created in variable/s strongly related to both sensitive attribute and outcome variable

- **Complete test set**: For demographic parity and equality of opportunity, for listwise deletion and MAR, we see a slight reduction on average from the baseline distribution. See figures 6.21 and 6.22.

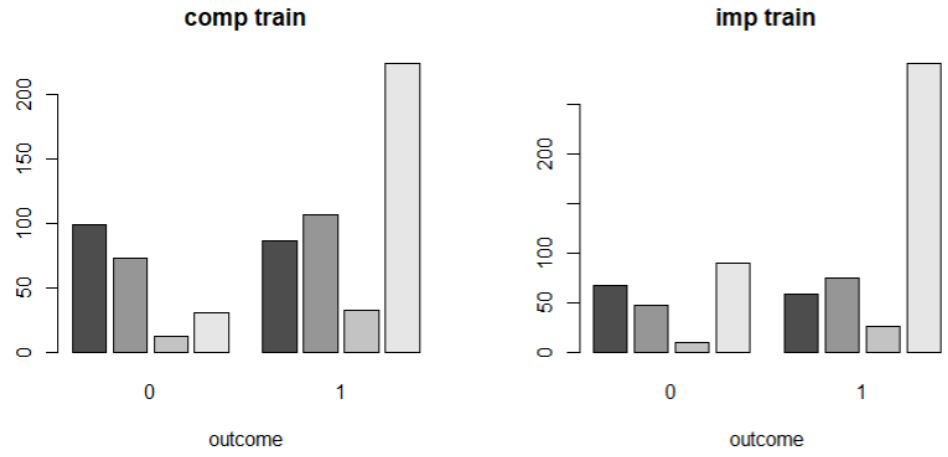

Figure 6.19: German data, checking account status by outcome after MAR and mode imputation

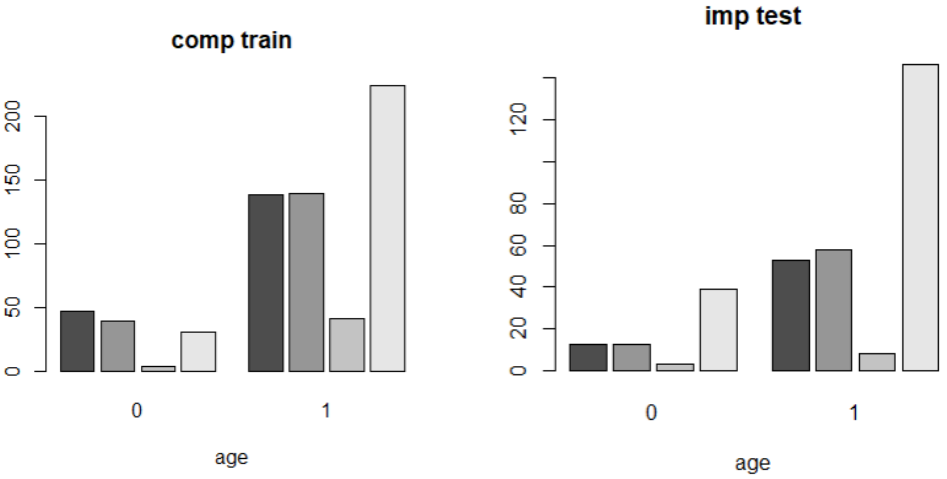

Figure 6.20: German data, checking account status by age after MAR and mode imputation

Possible explanation: For MAR, more missingness is created for the disadvantaged class (young). With listwise deletion, these rows are removed. This means that compared to the complete training set, there will now be very little training data for the young group. It is likely that the model trained on this data is less accurate for the disadvantaged class. In particular, from an instance of a MAR amputation and listwise deletion, the outcome variable by age has these proportions remaining after listwise deletion, shown in Tables 6.4, 6.5 and 6.6.

Given the removal of a large amount of the data for the disadvantaged (young) class, the

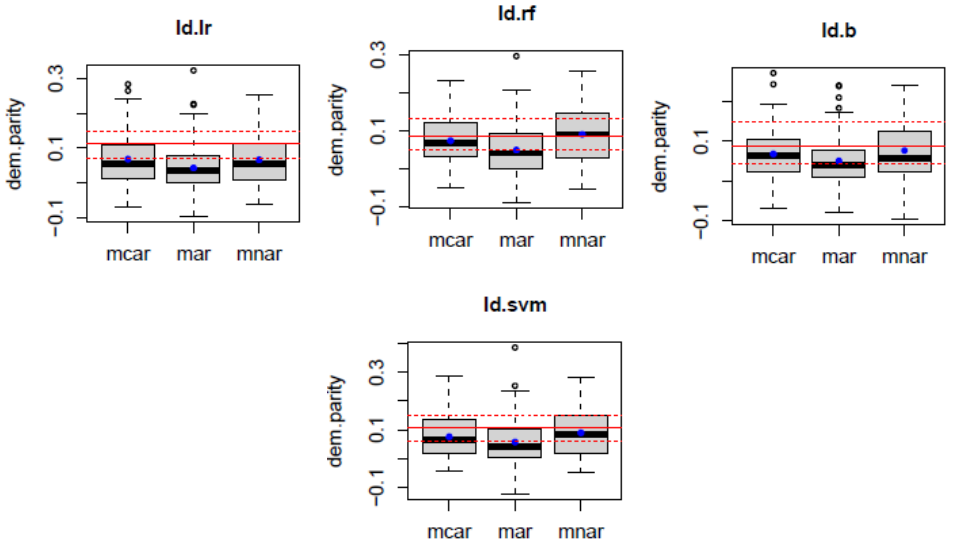

Figure 6.21: German data, sensitive attribute age, LD and demographic parity

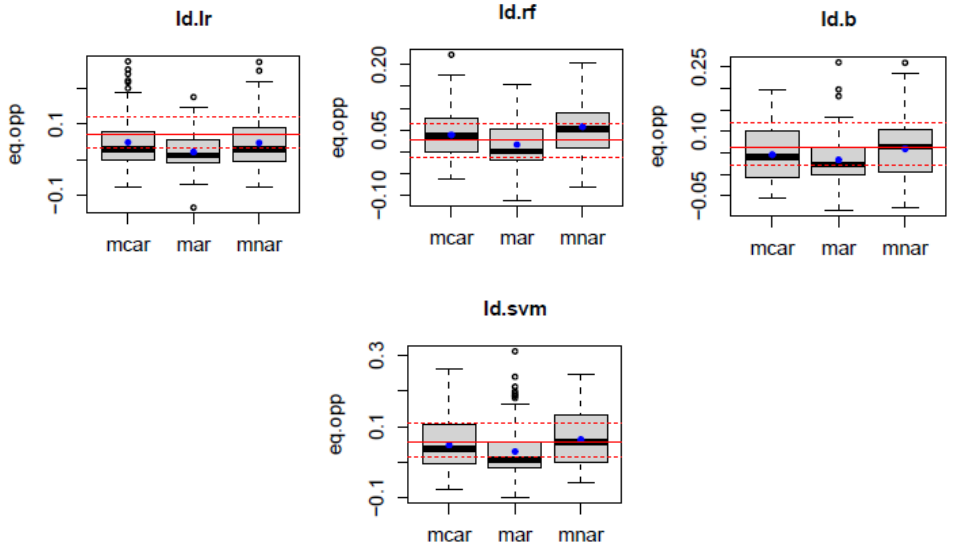

Figure 6.22: German data, sensitive attribute age, LD and equality of opportunity

predictions are more likely to match the outcome proportions in the advantaged class. It is possible that now a higher proportion of 1's are predicted for the disadvantaged class, leading to the reduction in metric value for MAR.

- **Imputed test set**: For all 3 metrics, all 4 classification models and all imputation techniques, statement 2 holds.

	out	
age	0	1
0	59	63
1	157	388

Table 6.4: Complete training set

	out	
age	0	1
0	4	11
1	106	229

Table 6.5: Listwise deletion training set

6.4 ADULT INCOME DATASET

The adult income data set is the largest of the 3 data sets we investigate. It also shows the more interesting boxplots. On this data set, as explained in Section 5.9, we do not fit random forests and svm models, and we only use listwise deletion, mode and regression as imputation methods, leaving out knn.

Before we go on, we would first like to point out something interesting which is easily observed in the Adult data set, which is that the shape of the distributions of the metrics demographic parity and predictive equality are very similar in this data set. Apart from the similarity in distributions for demographic parity and predictive equality, we also see some interesting patters in the metric distributions. See figures 6.23 and 6.24 for an example with boosting.

In the case of the Adult income dataset and referring back to the definitions of the metrics, perhaps the similarities in Demographic parity and Predictive parity can partially be explained by the fact that in the outcome variable the majority class is 0 with proportion 75%. Due to this, the dominating or underlying patterns seen in Demographic parity are from false positives, which could

	out	
age	0	1
0	0.068	0.175
1	0.675	0.590

Table 6.6: Proportions remaining

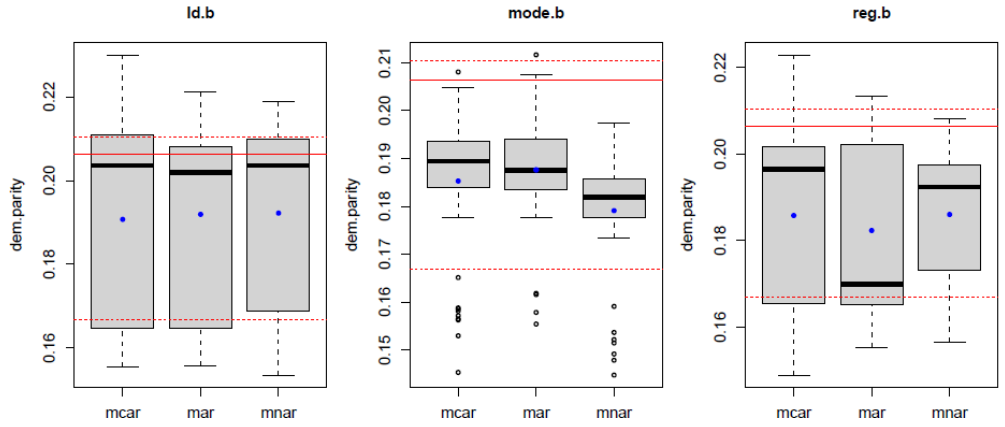

Figure 6.23: Adult data, demographic parity, sensitive attribute sex, boosting model

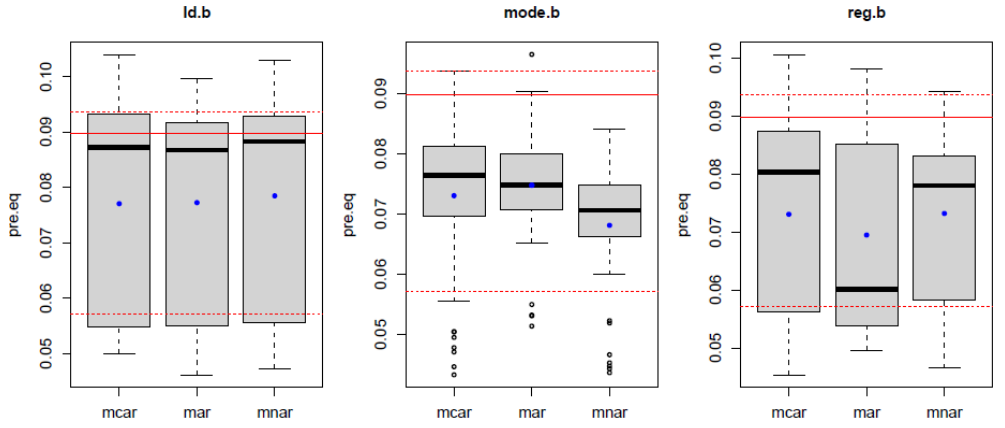

Figure 6.24: Adult data, predictive equality, sensitive attribute sex, boosting model

account for why it looks similar to predictive equality.

For both sensitive attributes in the Adult income data set, we will now focus on the cases where missingness is created in variable/s strongly related to both sensitive attribute and outcome variable.

6.4.1 Sensitive attribute: Race

6.4.1.1 Missingness created in variable/s strongly related to both sensitive attribute and outcome variable

- **Complete test set**: In the case of demographic parity and predictive equality, for mode imputation and both MCAR and MAR, we see an increase on average in the metric value from the baseline. See figure 6.25.

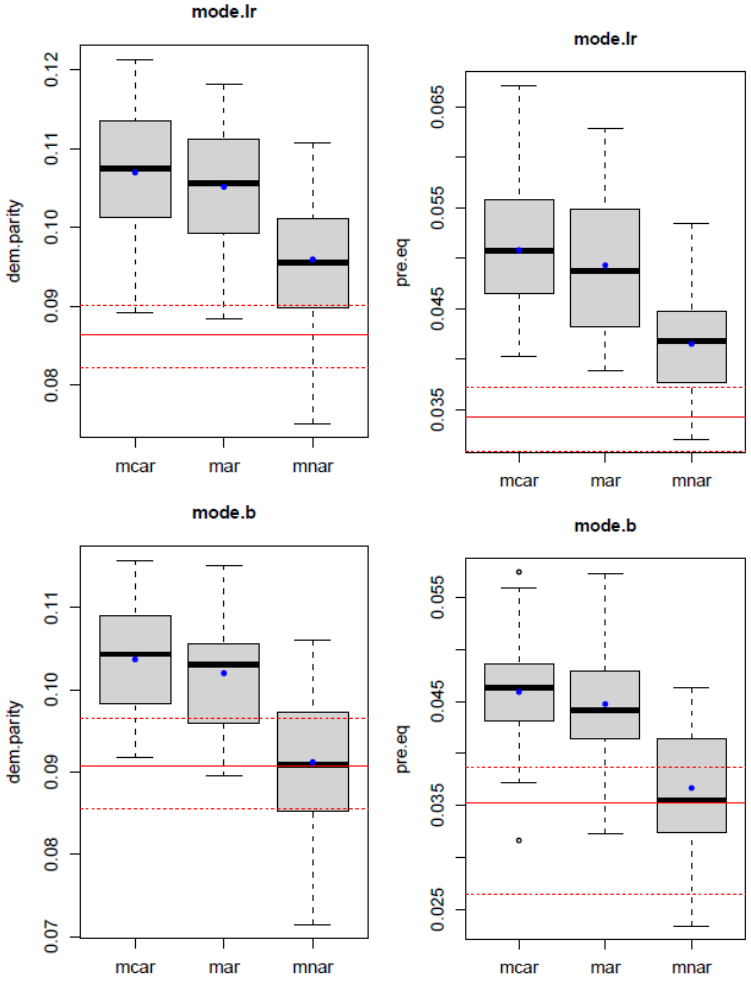

Figure 6.25: Adult data, sensitive attribute race, mode imputation

Possible explanation: As both demographic parity and predictive equality are increasing, it is reasonable to assume that the false positive rates for the advantaged class (whites) could

be increasing.

The variable that is amputed is 'educational num'. Looking at Figure 6.26, we see that after imputation the distributions of educational num (the categories of the variable are not important here) by outcome are more similar to each other than before, which could lead to reduction in classification accuracy. It is possible that this could lead to more false positives for the advantaged group (whites), so that some of the predictions will change from 0 to 1, leading to an increase in the metric value.

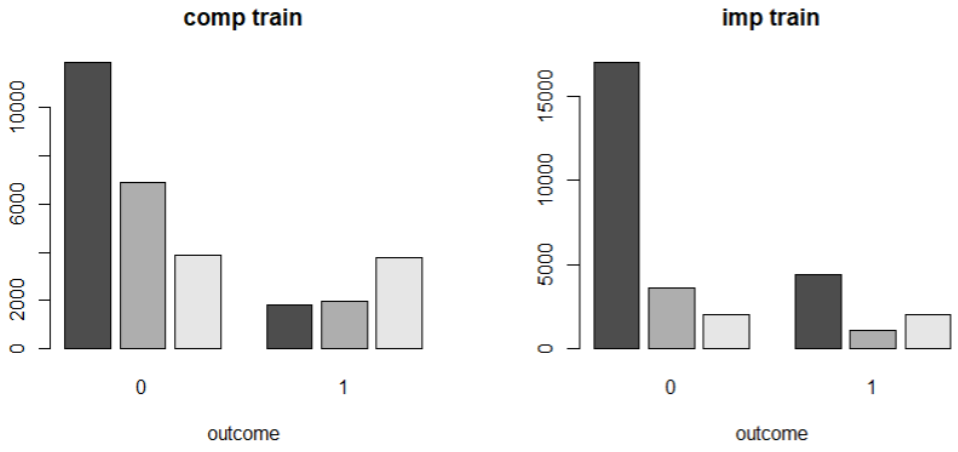

Figure 6.26: Adult income data: distribution of educational num by outcome

6.4.2 Sensitive attribute: Sex

6.4.2.1 Missingness created in variable/s strongly related to both sensitive attribute and outcome variable

- **Imputed test set**: In figure 6.27, we see that the boxplots for demographic parity and predictive equality look very similar. This suggests that the false positive rate is driving the changes in both these boxplots. We also notice the differences in distribution between the boxplots for logistic regression vs boosting. Even in their baseline we see differences: for the logistic regression model the baseline distribution is symmetric and narrow, whereas that for boosting is wider and left skewed. Specifically in the case of mode imputation and MAR, for demographic parity and predictive equality, the metric value increases on average from

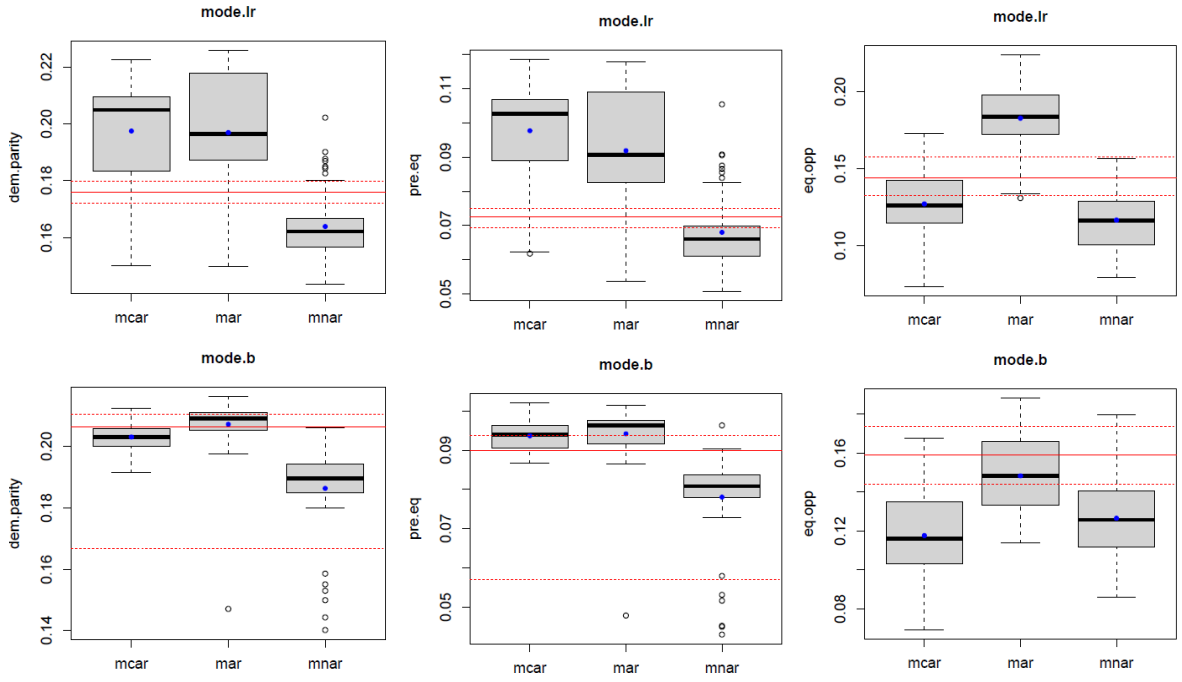

Figure 6.27: Metric distributions on imputed test set, adult income data

the baseline. Despite this commonality there are significant differences in the distribution of the metric for logistic regression and boosting, in the skewness of the distributions and the interquartile ranges. There are differences according to MDM also.

Possible explanation: The variable where missingness and then imputation is created is educational num. Again, the actual categories are not important. Looking at the distribution of the variable with respect to outcome, after an instance of amputation and imputation, see Figure 6.28, we see that the distributions for outcome 0 and 1 look more similar to each other than before. This reduced discernability of the outcome by the variable would likely lead to lower accuracy. Given that MAR creates more missingness for the disadvantaged group (females) we expect that this group will be affected more by the inaccuracy.

However, the boxplots suggest that although the false positive rate for both are increasing, the advantaged group (males) is increasing at a faster rate than for females, which leads to increase in the discrimination from the baseline and on average a positive metric value in predictive equality and demographic parity.

Looking at educational num distribution for males and females (the first bar in the bar plots is category 1), Figure 6.29, in the imputed train and hence imputed test set (the distributions are expected to be similar), the proportions of category 1 (the modal value) for both males and females increases as compared to the complete test set, but the proportion for the disadvantaged group (females) increases by a larger amount, as expected through MAR missingness. With the lower accuracy and the boxplots of demographic parity and predictive equality, it is plausible that more false positives are being predicted for both males and females, but it is unexpected that we get an increase in discrimination (higher false positive rate for the males) rather than a reduction, given that with MAR we expect the rate in the disadvantaged class to increase more.

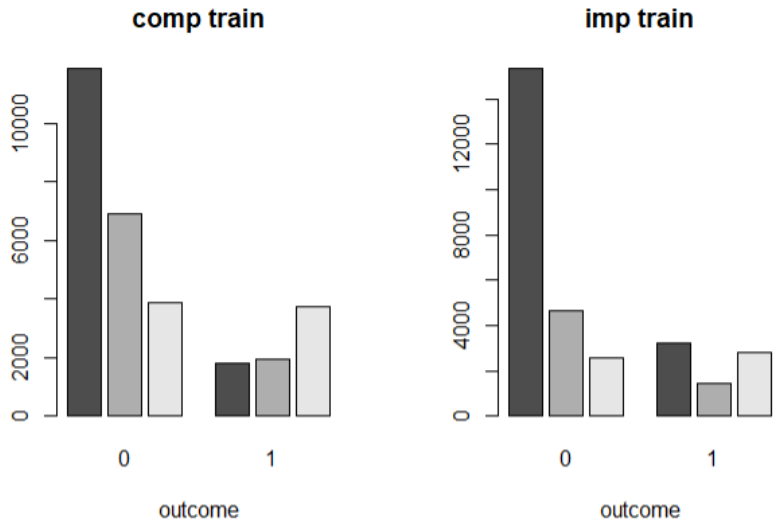

Figure 6.28: Distribution of educational num by outcome in mode imputed train set

- **Complete test set**: We see increase in discrimination on average compared to metric distributions on the imputed test set for demographic parity and predictive equality. See Figure 6.30, for mode imputation, logistic regression and boosting models. Possible explanation: In the case of imputed train and test set, we expect that the distributions of these two sets will be similar to each other, as the imputation model built on the training set is used to impute the test set, whereas in the case of the imputed training and complete test set, these two

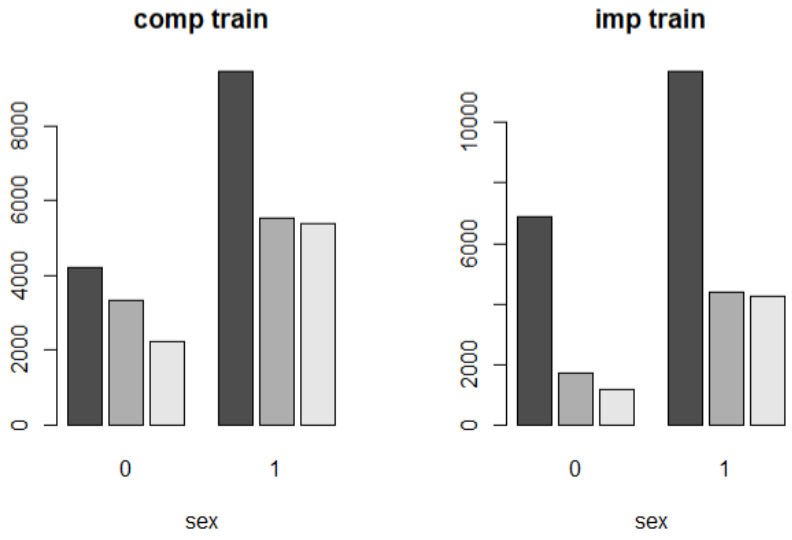

Figure 6.29: Distribution of educational num by sex in mode imputed train set

sets will be less similar to each other, leading to lower accuracy. This could translate to less unfairness than before, but from the plots we see it leads to an increase in discrimination compared to the imputed test set, rather than a reduction.

6.5 CONCLUDING OBSERVATIONS

From the fairness metric boxplots seen, especially for the Adult income data set, it became apparent that to untangle the influence of the missing data mechanisms, the imputation techniques and the classification models on the fairness metric distributions is a complex task. In future work, we need to perhaps create simpler scenarios, for example by generating a synthetic data set where we can control the attribute relationships, to gain clearer insight. Nevertheless, from the results we have seen, we can make some general comments:

- Missing data mechanisms and missing data handling procedures (listwise deletion or single imputation) influence the fairness of predictions obtained. This general statement is in agreement with the handful of previous works in the literature.
- Loss of data by listwise deletion for the disadvantaged group, resulting in much less data for this group compared to the advantaged group, can lead to increased fairness for this group, as predictions are more likely to match the outcome proportions in the advantaged group. This

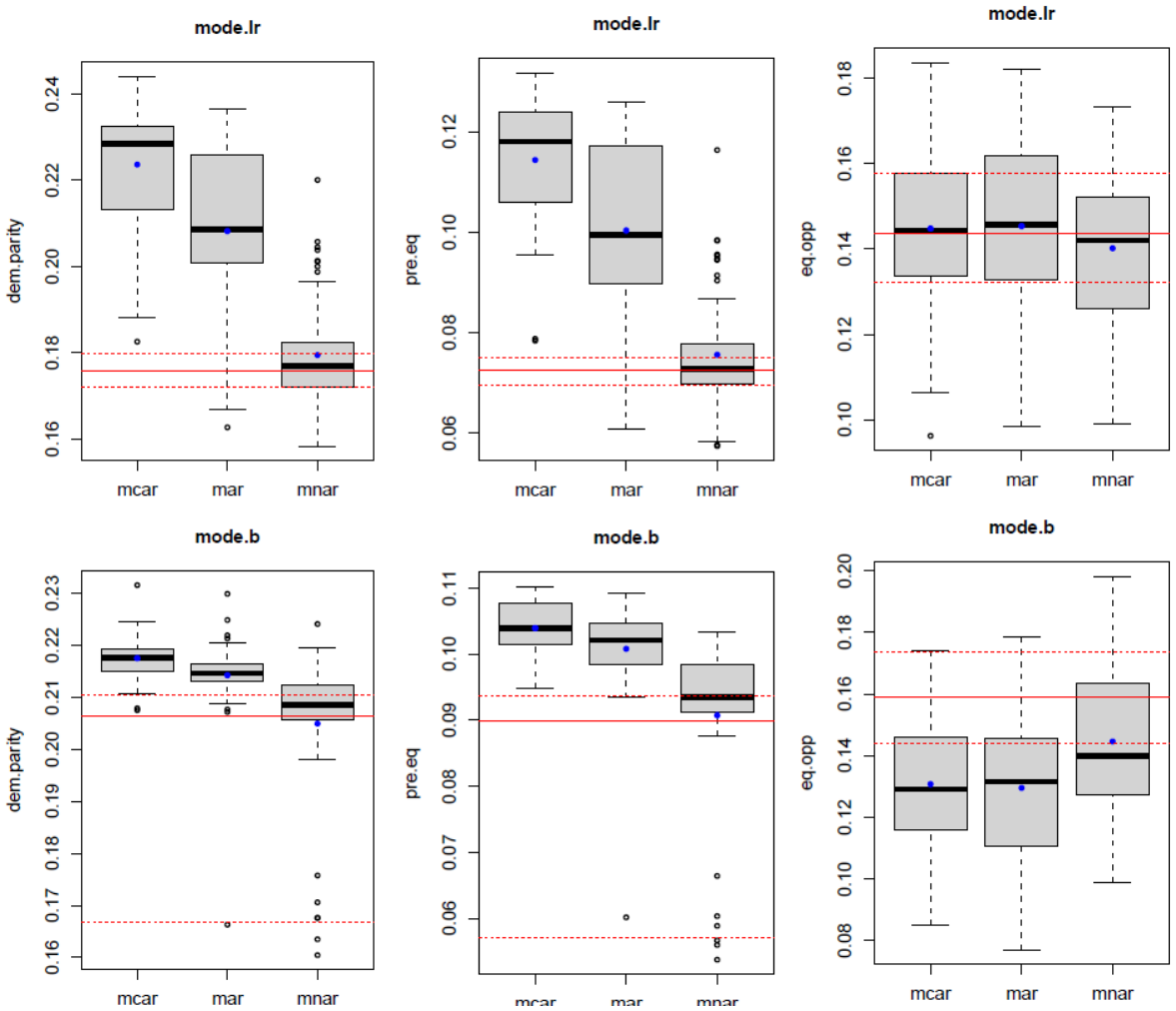

Figure 6.30: Adult data, metric distributions with mode imputation, all three metrics

is in line with Zhang and Long (2021*b*) observation 2, where they observe that imputation fairness can be influenced by the imbalance of missingness between the sensitive groups.

- In this study, the MAR missingness in a variable was created by dependence of missingness only on the sensitive attribute, in particular we implemented the assumption that observations from disadvantaged groups will contain more missing values. What we observed was that if missingness is related to the sensitive attribute in this way, the impact can be pronounced.
- For MAR missingness and mode imputation, if the mode of a variable is linked to the favourable outcome and there is more missingness in this variable for the disadvantaged group, then imputation by mode will tend to lead to less unfairness for the disadvantaged

group.

- From our study, the imputation techniques of regression and k-NN tend to be more faithful to the true data values than mode imputation, leading to similar fairness metric distributions as in the baseline model. If imputation accuracy is of importance to a researcher, then these methods are preferred over the very basic imputation by mode or mean.
- We created missingness in those variables which were either strongly related to the sensitive attribute or outcome variable, or both. These distinctions did not provide us with any obvious relationship between type of variable amputed and the effect on discrimination.
- Fairness metric results are data set dependent. For instance, we see much more variety of distributions in the Adult income data set than in the German credit data set. This could be due to the baseline levels of discrimination in the data sets, some data sets could be more biased than others. For the same data set, we have also seen that different sensitive attributes can produce different effects on the fairness. Based on the fairness metric and threshold, a model may be discriminatory with respect to one sensitive attribute but fair with respect to another.
- After imputation or deletion, there can be differences in metric distributions between different machine learning models. This could be due to different levels of trade-off between fairness and accuracy, and this should be investigated in future work. Although in this work we have not explicitly reported on accuracy results, by observing the imputation distributions of variables we have been able to comment on the likely direction of accuracy. Our findings are in line with previous studies of Fernando *et al.* (2021) and Zhang and Long (2021*b*) where they observe that increased fairness comes at the cost of reduced accuracy. There is an intimate connection between fairness and accuracy, and this relationship must be investigated in much more detail in the context of missing values.
- Fairness metric distributions can be strongly related, as we saw in the case of the Adult income data set for Demographic parity and Predictive equality.

CHAPTER 7

CONCLUSIONS, RECOMMENDATIONS AND FUTURE WORK

In this study, our aim was to investigate the impact of missing data mechanisms and the simpler missing data handling procedures like listwise deletion or single imputation, on the fairness of machine learning algorithms.

The field of fairness in ML is receiving increasing attention from researchers as more and more of these algorithms have a tangible impact in our lives. Audits are being conducted at various stages of the life cycle of algorithms to make them fairer. Existing literature combining the research of fairness of algorithms and missing data is limited to a handful of studies (Fernando *et al.*, 2021; Zhang and Long, 2021*b*; Wang and Singh, 2021; Zhang and Long, 2021*a*). Our research contributes to these studies.

From the results obtained from our experiments, we observed in general that missing data mechanisms and missing data handling procedures influence the fairness of predictions obtained. We also saw how fairness can be influenced by the extent of imbalance of the missingness between the sensitive groups. We observed the effects of the trade-off between accuracy and fairness, seeing that reduced accuracy can lead to increased fairness. These observations are in agreement with previous studies of the same nature.

In addition, we observed the following in our research, and based on these we can make some recommendations:

- Loss of data by listwise deletion for the disadvantaged group, resulting in much less data for this group compared to the advantaged group, can lead to increased fairness for this group, as predictions are more likely to match the outcome proportions in the advantaged group. However, in practice we cannot apply LD to a new test observation containing missing values as this would just remove the entire observation we are trying to make a prediction for.
- For MAR missingness and mode imputation, if the mode of a variable is linked to the

favourable outcome and there is more missingness in this variable for the disadvantaged group, then imputation by mode will tend to lead to less unfairness for the disadvantaged group. Regarding mode (or mean) imputation, despite this approach being very simple, it can lead to the effect of increased fairness for the disadvantaged group. If fairness of predictions is one of the concerns of a researcher then even this simple method could give the required effects, depending of course on how much they are willing to let the accuracy deteriorate.

- From our study, the imputation techniques of regression and k-NN tend to be more faithful to the true data values than mode imputation, leading to similar fairness metric distributions as in the baseline model. If imputation accuracy is of importance to a researcher, then these methods are preferred over the very basic imputation by mode or mean.

Regarding future work, the first aspect which needs to be investigated is the trade-off between accuracy and fairness, when data are missing according to a particular MDM, and a particular missing data handling procedure is applied. From our experiments and previous studies, it is evident that there is an intimate connection between an algorithm's performance and fairness, and this needs to be investigated in detail. Based on this relationship, the feasibility of developing an imputation algorithm that trades off between performance and fairness to impute missing data fairly should be undertaken. More advanced imputation methods such as multiple imputation and bias mitigation methods also need to be investigated in the same context.

REFERENCES

Barocas, S., Hardt, M. and Narayanan, A. (2017). Fairness in machine learning. *Nips tutorial*, vol. 1, p. 2.

Bellamy, R.K.E., Dey, K., Hind, M., Hoffman, S.C., Houde, S., Kannan, K., Lohia, P., Martino, J., Mehta, S., Mojsilovic, A., Nagar, S., Ramamurthy, K.N., Richards, J., Saha, D., Sattigeri, P., Singh, M., Varshney, K.R. and Zhang, Y. (2018 October). AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias.
Available at: `https://arxiv.org/abs/1810.01943`

Beretta, L. and Santaniello, A. (2016). Nearest neighbor imputation algorithms: a critical evaluation. *BMC medical informatics and decision making*, vol. 16, no. 3, pp. 197–208.

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

Calders, T. and Verwer, S. (2010). Three naive bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery*, vol. 21, no. 2, pp. 277–292.

Chicco, D. and Jurman, G. (2020). The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, vol. 21, no. 1, pp. 1–13.

Dua, D. and Graff, C. (2017). UCI machine learning repository.
Available at: `http://archive.ics.uci.edu/ml`

Dunkelau, J. and Leuschel, M. (2019). Fairness-aware machine learning. *An Extensive Overview*, pp. 1–60.

Eekhout, I., de Boer, R.M., Twisk, J.W., de Vet, H.C. and Heymans, M.W. (2012). Missing data: a systematic review of how they are reported and handled. *Epidemiology*, vol. 23, no. 5, pp. 729–732.

Enders, C.K. (2022). *Applied missing data analysis*. Guilford Publications.

Feldman, M., Friedler, S.A., Moeller, J., Scheidegger, C. and Venkatasubramanian, S. (2015). Certifying and removing disparate impact. In: *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 259–268.

Fernando, M.-P., Cèsar, F., David, N. and José, H.-O. (2021). Missing the missing values: The ugly duckling of fairness in machine learning. *International Journal of Intelligent Systems*, vol. 36, no. 7, pp. 3217–3258.

Friedler, S.A., Scheidegger, C., Venkatasubramanian, S., Choudhary, S., Hamilton, E.P. and Roth, D. (2019). A comparative study of fairness-enhancing interventions in machine learning. In: *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 329–338.

Hardt, M., Price, E. and Srebro, N. (2016). Equality of opportunity in supervised learning. *Advances in neural information processing systems*, vol. 29.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013). *An introduction to statistical learning*, vol. 112. Springer.

Jonsson, P. and Wohlin, C. (2004). An evaluation of k-nearest neighbour imputation using likert data. In: *10th International Symposium on Software Metrics, 2004. Proceedings.*, pp. 108–118. IEEE.

Kamiran, F., Calders, T. and Pechenizkiy, M. (2010). Discrimination aware decision tree learning. In: *2010 IEEE international conference on data mining*, pp. 869–874. IEEE.

Kamiran, F., Karim, A. and Zhang, X. (2012). Decision theory for discrimination-aware classification. In: *2012 IEEE 12th International Conference on Data Mining*, pp. 924–929. IEEE.

Kang, H. (2013). The prevention and handling of the missing data. *Korean journal of anesthesiology*, vol. 64, no. 5, pp. 402–406.

Kowarik, A. and Templ, M. (2016). Imputation with the r package vim. *Journal of Statistical Software*, vol. 74, pp. 1–16.

Kuhn, M. and Johnson, K. (2013). *Applied predictive modeling*, vol. 26. Springer.

Le Quy, T., Roy, A., Iosifidis, V., Zhang, W. and Ntoutsi, E. (2022). A survey on datasets for fairness-aware machine learning. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, p. e1452.

Little, R.J. (1988). A test of missing completely at random for multivariate data with missing values. *Journal of the American statistical Association*, vol. 83, no. 404, pp. 1198–1202.

Little, R.J. and Rubin, D.B. (2019). *Statistical analysis with missing data*, vol. 793. John Wiley & Sons.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K. and Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–35.

Nishanth, K.J. and Ravi, V. (2016). Probabilistic neural network based categorical data imputation. *Neurocomputing*, vol. 218, pp. 17–25.

Ntoutsi, E., Fafalios, P., Gadiraju, U., Iosifidis, V., Nejdl, W., Vidal, M.-E., Ruggieri, S., Turini, F., Papadopoulos, S., Krasanakis, E. *et al.* (2020). Bias in data-driven artificial intelligence systems—an introductory survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 10, no. 3, p. e1356.

O'Neil, C. (2017). *Weapons of math destruction: How big data increases inequality and threatens democracy*. Crown.

Pessach, D. and Shmueli, E. (2020). Algorithmic fairness. *arXiv preprint arXiv:2001.09784*.

R Core Team (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
Available at: `https://www.R-project.org/`

Schouten, R.M., Lugtig, P. and Vink, G. (2018). Generating missing values for simulation purposes: a multivariate amputation procedure. *Journal of Statistical Computation and Simulation*, vol. 88, no. 15, pp. 2909–2930.

Soley-Bori, M. (2013). Dealing with missing data: Key assumptions and methods for applied analysis. *Boston University*, vol. 4, no. 1, p. 19.

Store, P. (2016). Compas recidivism risk score data and analysis. *ProPublica Data Store*, vol. 2.

Van Buuren, S. (2018). *Flexible imputation of missing data*. CRC press.

Van Buuren, S. and Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, vol. 45, pp. 1–67.

Verma, S. and Rubin, J. (2018). Fairness definitions explained. In: *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)*, pp. 1–7. IEEE.

Wang, Y. and Singh, L. (2021). Analyzing the impact of missing values and selection bias on fairness. *International Journal of Data Science and Analytics*, vol. 12, no. 2, pp. 101–119.

Zhang, Y. and Long, Q. (2021*a*). Assessing fairness in the presence of missing data. *Advances in neural information processing systems*, vol. 34, pp. 16007–16019.

Zhang, Y. and Long, Q. (2021*b*). Fairness in missing data imputation. *arXiv preprint arXiv:2110.12002*.

Žliobaitė, I. (2017). Measuring discrimination in algorithmic decision making. *Data Mining and Knowledge Discovery*, vol. 31, no. 4, pp. 1060–1089.

APPENDIX A

DATA SETS

The following data set details are taken from Le Quy *et al.* (2022).

A.1 ADULT INCOME

Attributes	Type	Values	#Missing values	Description
age	Numerical	[17–90]	0	The age of an individual
workclass	Categorical	7	2,799	The employment status (private, state-gov, etc.)
fnlwgt	Numerical	[13,492–1,490,400]	0	The final weight
education	Categorical	16	0	The highest level of education
educational-num	Numerical	1–16	0	The highest level of education achieved in numerical form
marital-status	Categorical	7	0	The marital status
occupation	Categorical	14	2,809	The general type of occupation
relationship	Categorical	6	0	Represents what this individual is relative to others
race	Categorical	5	0	Race
sex	Binary	{Male, Female}	9	The biological sex of the individual
capital-gain	Numerical	[0–99,999]	0	The capital gains for an individual
capital-loss	Numerical	[0–4,356]	0	The capital loss for an individual
hours-per-week	Numerical	[1–99]	0	The hours an individual has reported to work per week
native-country	Categorical	41	857	The country of origin for an individual
income	Binary	{≤50K, >50K}	0	Whether or not an individual makes more than $50,000 annually

Figure A.1: Adult income data set attributes

A.2 GERMAN CREDIT

Attributes	Type	Values	#Missing values	Description
checking-account	Categorical	4	0	The status of existing checking account
duration	Numerical	[4–72]	0	The duration of the credit (month)
credit-history	Categorical	5	0	The credit history
purpose	Categorical	10	0	Purpose (car, furniture, education, etc.)
credit-amount	Numerical	[250–18,424]	0	Credit amount
savings-account	Categorical	5	0	Savings account/bonds
employment-since	Categorical	5	0	Present employment since
installment-rate	Numerical	[1–4]	0	The installment rate in percentage of disposable income
personal-status-and-sex	Categorical	4	0	The personal status and sex
other-debtors	Categorical	3	0	Other debtors/guarantors
residence-since	Numerical	[1–4]	0	Present residence since
property	Categorical	4	0	Property
age	Numerical	[19–75]	0	The age of the individual
other-installment	Categorical	3	0	Other installment plans
housing	Categorical	3	0	Housing (rent, own, for free)
existing-credits	Numerical	[1–4]	0	Number of existing credits at this bank
job	Categorical	4	0	Job (unemployed, (un)skilled, management)
number-people-provide-maintenance-for	Numerical	[1–2]	0	Number of people being liable to provide maintenance for
telephone	Binary	{Yes, None}	0	Telephone number
foreign-worker	Binary	{Yes, No}	0	Is the individual a foreign worker?
class-label	Binary	{Good, Bad}	0	Class

Figure A.2: German credit data set attributes

A.3 COMPAS RECIDIVISM

Attributes	Type	Values	#Missing values	Description
sex	Binary	{Male, Female}	0	Sex
age	Numerical	[18–96]	0	Age in years
age_cat	Categorical	3	0	Age category
race	Categorical	6	0	Race
juv_fel_count	Numerical	[0–20]	0	The juvenile felony count
juv_misd_count	Numerical	[0–13]	0	The juvenile misdemeanor count
juv_other_count	Numerical	[0–17]	0	The juvenile other offenses count
priors_count	Numerical	[0–38]	0	The prior offenses count
c_charge_degree	Binary	{F, M}	0	Charge degree of original crime
score_text	Categorical	3	0	ProPublica-defined category of decile score
v_score_text	Categorical	3	0	ProPublica-defined category of v_decile_score
two_year_recid	Binary	{0, 1}	0	Whether the defendant is rearrested within 2 years

Abbreviation: COMPAS, Correctional Offender Management Profiling for Alternative Sanctions.

Figure A.3: COMPAS recidivism data set attributes

APPENDIX B

R PROFILER OUTPUT

The following is the R profiler output, run on the Adult income data set, including all MDM's, imputation and classifcation models, for one iteration only. We can see that SVM seems to be taking up the bulk of the time, followed by k-NN and RF.

```
$by.total
```

	total.time	total.pct	self.time	self.pct
"comp_res"	28782.0	100.00	0.0	0.00
"kfolds_results"	25944.0	90.14	0.0	0.00
".C"	25444.2	88.40	25444.2	88.40
"svm.formula"	22682.3	78.81	0.5	0.00
"svm"	22682.3	78.81	0.0	0.00
"svm.default"	22676.8	78.79	0.5	0.00
"svm_tune"	18082.4	62.83	0.0	0.00
"predict"	12970.7	45.07	0.0	0.00
"predict.svm"	12739.6	44.26	1.4	0.00
"na.action"	11021.7	38.29	0.0	0.00
"svm_results"	6399.9	22.24	0.0	0.00
".Call"	2984.7	10.37	2984.7	10.37
"imp_n"	2835.7	9.85	0.0	0.00
"kNN"	2823.9	9.81	0.1	0.00
"dist_single"	2764.0	9.60	0.0	0.00
"gowerD"	2764.0	9.60	0.0	0.00
"gowerDind"	2763.8	9.60	0.0	0.00
"randomForest.formula"	954.7	3.32	0.1	0.00
"randomForest"	954.7	3.32	0.0	0.00
"randomForest.default"	951.5	3.31	4.2	0.01
"rf_results"	628.1	2.18	0.0	0.00
"rf_tune"	553.2	1.92	0.0	0.00

"gbm"	241.0	0.84	0.0	0.00
"gbm.fit"	239.6	0.83	0.8	0.00
"predict.randomForest"	225.8	0.78	4.2	0.01
"b_tune"	189.0	0.66	0.4	0.00
"apply"	80.4	0.28	9.0	0.03
"data.frame"	73.7	0.26	0.0	0.00
"b_results"	69.3	0.24	0.0	0.00
"as.data.frame"	68.8	0.24	0.0	0.00
"as.vector"	68.2	0.24	66.4	0.23
"as.data.frame.matrix"	63.2	0.22	0.1	0.00
"FUN"	58.3	0.20	7.2	0.03
"["	47.3	0.16	1.0	0.00
"aperm"	40.7	0.14	0.1	0.00
"do.call"	40.6	0.14	0.1	0.00
"[.data.table"	31.3	0.11	11.0	0.04
"aperm.default"	28.7	0.10	28.5	0.10
"factor"	27.1	0.09	21.0	0.07
"matrix"	23.7	0.08	23.6	0.08
"eval"	23.5	0.08	1.7	0.01
"integer"	21.1	0.07	21.1	0.07
"glm.fit"	15.5	0.05	4.2	0.01
"lr_tune"	15.0	0.05	0.1	0.00
"anyDuplicated.default"	14.8	0.05	14.8	0.05
"anyDuplicated"	14.8	0.05	0.0	0.00
"glm"	14.2	0.05	0.0	0.00
"na.omit.data.frame"	14.1	0.05	2.1	0.01
"[.data.frame"	13.7	0.05	3.3	0.01
"table"	13.5	0.05	2.8	0.01
"as.matrix"	13.3	0.05	0.0	0.00
"array"	12.6	0.04	12.4	0.04

"regressionImp"	11.8	0.04	0.0	0.00
"model.frame.default"	11.1	0.04	0.0	0.00
"checkMissing"	10.0	0.03	0.0	0.00
"double"	9.8	0.03	9.8	0.03
"as.matrix.data.frame"	9.7	0.03	0.0	0.00
"as.integer"	9.4	0.03	9.4	0.03
"summary"	8.9	0.03	0.7	0.00
"standardGeneric"	8.6	0.03	0.3	0.00
"summary.factor"	7.8	0.03	0.2	0.00
".External2"	7.6	0.03	1.2	0.00
"t"	7.4	0.03	0.2	0.00
"model.frame"	7.2	0.03	0.0	0.00
"prettyNum"	6.9	0.02	6.9	0.02
"format"	6.9	0.02	0.0	0.00
"format.default"	6.9	0.02	0.0	0.00
"lr_results"	6.8	0.02	0.0	0.00
"cbind"	6.6	0.02	0.4	0.00
"...elt"	6.3	0.02	0.0	0.00
"capture.output"	6.3	0.02	0.0	0.00
"multinom"	6.3	0.02	0.0	0.00
"withVisible"	6.3	0.02	0.0	0.00
"nnet.default"	6.2	0.02	0.0	0.00
".rowNamesDF<-"	6.0	0.02	0.0	0.00
"as.data.table.list"	5.9	0.02	0.8	0.00
"levels"	5.5	0.02	2.6	0.01
"scale_data_frame"	5.5	0.02	0.0	0.00
"as.data.frame.factor"	5.3	0.02	0.0	0.00
"scale"	5.0	0.02	0.0	0.00
"scale.default"	5.0	0.02	0.0	0.00
"<Anonymous>"	4.9	0.02	0.2	0.00

"as.factor"	4.8	0.02	0.1	0.00
"is.element"	4.3	0.01	0.2	0.00
"predict.gbm"	4.2	0.01	0.0	0.00
"stats::model.frame"	4.2	0.01	0.0	0.00
"vapply"	4.0	0.01	1.5	0.01
"[["	3.9	0.01	1.5	0.01
"NextMethod"	3.8	0.01	3.4	0.01
"rm"	3.8	0.01	0.5	0.00
"anyNA"	3.6	0.01	3.6	0.01
"as.matrix.data.table"	3.6	0.01	0.4	0.00
"[.factor"	3.6	0.01	0.1	0.00
"model.matrix"	3.6	0.01	0.0	0.00
"model.matrix.default"	3.5	0.01	0.0	0.00
"is.na"	3.4	0.01	3.1	0.01
"unique"	3.2	0.01	0.6	0.00
"cedta"	3.1	0.01	0.4	0.00
"as.double"	3.0	0.01	3.0	0.01
"order"	2.9	0.01	0.7	0.00
"match.arg"	2.8	0.01	0.8	0.00
"unlist"	2.7	0.01	2.7	0.01
"is.data.frame"	2.7	0.01	0.7	0.00
"sys.call"	2.6	0.01	2.6	0.01
"fn"	2.6	0.01	0.0	0.00
"nzchar"	2.5	0.01	2.5	0.01
"[[.data.frame"	2.4	0.01	0.7	0.00
"lapply"	2.4	0.01	0.3	0.00
"match.call"	2.3	0.01	0.4	0.00
"tryCatch"	2.3	0.01	0.3	0.00
"sample"	2.2	0.01	0.5	0.00
"recycle"	2.2	0.01	0.2	0.00

"dim"	2.1	0.01	1.6	0.01
"%chin%"	2.1	0.01	1.1	0.00
"try"	2.1	0.01	0.0	0.00
"unique.default"	1.9	0.01	1.6	0.01
"list.names"	1.9	0.01	0.6	0.00
"sweep"	1.9	0.01	0.1	0.00
"tryCatchList"	1.9	0.01	0.1	0.00
"as.character"	1.8	0.01	0.9	0.00
"tryCatchOne"	1.8	0.01	0.5	0.00
"sample.int"	1.7	0.01	0.8	0.00
"na.omit"	1.7	0.01	0.0	0.00
"replace_dot_alias"	1.5	0.01	1.5	0.01
"t.default"	1.5	0.01	1.5	0.01
"as.list"	1.5	0.01	1.1	0.00
"which"	1.5	0.01	0.8	0.00
"%in%"	1.5	0.01	0.4	0.00
"as.vector.factor"	1.5	0.01	0.2	0.00
"quantile"	1.5	0.01	0.0	0.00
"=="	1.4	0.00	0.7	0.00
"quantile.default"	1.4	0.00	0.6	0.00
"force"	1.4	0.00	0.5	0.00
"amp_n"	1.4	0.00	0.0	0.00
"mu.eta"	1.4	0.00	0.0	0.00
"doTryCatch"	1.3	0.00	0.5	0.00
"aic"	1.3	0.00	0.0	0.00
"ampute"	1.3	0.00	0.0	0.00
"sapply"	1.3	0.00	0.0	0.00
"parent.frame"	1.2	0.00	1.2	0.00
"nrow"	1.2	0.00	0.2	0.00
"data.matrix"	1.1	0.00	0.4	0.00

"formals"	1.1	0.00	0.2	0.00
"paste"	1.1	0.00	0.2	0.00
"row.names<-"	1.1	0.00	0.0	0.00
"row.names<-.data.frame"	1.1	0.00	0.0	0.00
"getOption"	1.0	0.00	1.0	0.00
"names"	1.0	0.00	1.0	0.00
"paste0"	1.0	0.00	0.8	0.00
"na.omit.default"	1.0	0.00	0.4	0.00
"getNamespaceImports"	1.0	0.00	0.2	0.00
"ifelse"	1.0	0.00	0.2	0.00
"perf_mets"	1.0	0.00	0.0	0.00
"as.character.factor"	0.9	0.00	0.9	0.00
"dev.resids"	0.9	0.00	0.9	0.00
"is.data.table"	0.9	0.00	0.9	0.00
"sum"	0.9	0.00	0.9	0.00
"validmu"	0.9	0.00	0.7	0.00
"%iscall%"	0.9	0.00	0.5	0.00
"is.na<-"	0.9	0.00	0.5	0.00
"sys.function"	0.9	0.00	0.5	0.00
"deparse"	0.9	0.00	0.3	0.00
"predict.lm"	0.9	0.00	0.1	0.00
"predict.glm"	0.9	0.00	0.0	0.00
"%*%"	0.8	0.00	0.8	0.00
"c"	0.8	0.00	0.8	0.00
"is.factor"	0.8	0.00	0.8	0.00
"stopifnot"	0.8	0.00	0.5	0.00
"length"	0.7	0.00	0.7	0.00
".unsafe.opt"	0.7	0.00	0.5	0.00
"assign"	0.7	0.00	0.5	0.00
"sort.int"	0.7	0.00	0.5	0.00

".prepareFastSubset"	0.7	0.00	0.3	0.00
"topenv"	0.7	0.00	0.3	0.00
"exists"	0.7	0.00	0.2	0.00
"Ops.factor"	0.7	0.00	0.1	0.00
"linkinv"	0.7	0.00	0.0	0.00
"sort"	0.7	0.00	0.0	0.00
"sort.default"	0.7	0.00	0.0	0.00
"sum.scores"	0.7	0.00	0.0	0.00
".deparseOpts"	0.6	0.00	0.4	0.00
"asNamespace"	0.6	0.00	0.4	0.00
"getNamespaceName"	0.6	0.00	0.2	0.00
"[<-"	0.6	0.00	0.0	0.00
"data.row.names"	0.6	0.00	0.0	0.00
".getNamespaceInfo"	0.5	0.00	0.5	0.00
"dbinom"	0.5	0.00	0.5	0.00
"dim.data.table"	0.5	0.00	0.5	0.00
"isTRUE"	0.5	0.00	0.5	0.00
"levels.default"	0.5	0.00	0.4	0.00
"substring"	0.5	0.00	0.4	0.00
"copy"	0.5	0.00	0.3	0.00
"setDT"	0.5	0.00	0.2	0.00
"contrasts"	0.5	0.00	0.1	0.00
".Diag"	0.4	0.00	0.4	0.00
"as.list.default"	0.4	0.00	0.4	0.00
"is.na<-.default"	0.4	0.00	0.4	0.00
"sys.parent"	0.4	0.00	0.4	0.00
"[[<-.data.frame"	0.4	0.00	0.3	0.00
"[.table"	0.4	0.00	0.2	0.00
"terms"	0.4	0.00	0.2	0.00
"[<-.data.frame"	0.4	0.00	0.1	0.00

"rbind"	0.4	0.00	0.1	0.00
"[[<-"	0.4	0.00	0.0	0.00
"ctrfn"	0.4	0.00	0.0	0.00
"simplify2array"	0.4	0.00	0.0	0.00
"all"	0.3	0.00	0.3	0.00
"any"	0.3	0.00	0.3	0.00
"is.nan"	0.3	0.00	0.3	0.00
"isNamespace"	0.3	0.00	0.3	0.00
"match.fun"	0.3	0.00	0.2	0.00
"base::rbind"	0.3	0.00	0.0	0.00
"duplicated"	0.3	0.00	0.0	0.00
"duplicated.data.frame"	0.3	0.00	0.0	0.00
"NCOL"	0.3	0.00	0.0	0.00
"train_test_sets"	0.3	0.00	0.0	0.00
"unique.data.frame"	0.3	0.00	0.0	0.00
"vapply_1i"	0.3	0.00	0.0	0.00
".subset2"	0.2	0.00	0.2	0.00
"chmatch"	0.2	0.00	0.2	0.00
"is.finite"	0.2	0.00	0.2	0.00
"lengths"	0.2	0.00	0.2	0.00
"logical"	0.2	0.00	0.2	0.00
"make.unique"	0.2	0.00	0.2	0.00
"makepredictcall"	0.2	0.00	0.2	0.00
"mapply"	0.2	0.00	0.2	0.00
"nargs"	0.2	0.00	0.2	0.00
"seq_along"	0.2	0.00	0.2	0.00
"variance"	0.2	0.00	0.2	0.00
"all.vars"	0.2	0.00	0.1	0.00
"make.names"	0.2	0.00	0.1	0.00
"model.extract"	0.2	0.00	0.1	0.00

```
"predict.multinom"              0.2      0.00       0.1     0.00
"cmpfun"                        0.2      0.00       0.0     0.00
"compiler:::tryCmpfun"          0.2      0.00       0.0     0.00
"genCode"                       0.2      0.00       0.0     0.00
"is.na.data.frame"              0.2      0.00       0.0     0.00
"reallocate"                    0.2      0.00       0.0     0.00
"row.names"                     0.2      0.00       0.0     0.00
"row.names.data.frame"          0.2      0.00       0.0     0.00
"-"                             0.1      0.00       0.1     0.00
"$"                             0.1      0.00       0.1     0.00
".External"                     0.1      0.00       0.1     0.00
".Primitive"                    0.1      0.00       0.1     0.00
">"                             0.1      0.00       0.1     0.00
">="                            0.1      0.00       0.1     0.00
"as.numeric"                    0.1      0.00       0.1     0.00
"attr"                          0.1      0.00       0.1     0.00
"cb$patchlabels"                0.1      0.00       0.1     0.00
"cb$putcode"                    0.1      0.00       0.1     0.00
 [ reached 'max' / getOption("max.print") -- omitted 49 rows ]

$sample.interval
[1] 0.1

$sampling.time
[1] 28782
```

APPENDIX C

R CODE

```
    ##-------------------------results from all models--------

comp_res <- function(data, n, p, f1, f2, f3, k, form, out, s, s_p,
                     b_form, type_mar, type_mnar, var_amp,
                     c_w = 0, outvar_idx, to_imp, reg_f, excl_vars){

  imp <- c("ld", "mode", "reg", "knn")

  mdm <- c("MCAR", "MAR", "MNAR")
  res_full <- list()
  for(j in mdm){
    #---list of n amputed datasets with mdm = j
    dats_amp <- amp_n(data, n, f1, f2, f3, s_p, mdm = j, var_amp, p,
                      type_mnar, type_mar)

    #---training and test sets------
    sets_train_test <- train_test_sets(data, dats_amp, n)
    train_comp <- sets_train_test$train_full
    test_comp <- sets_train_test$complete_test

    for(l in imp){
      lr_res_mincost <- list(c(), c())
      lr_res_maxmcc <- list(c(), c())
      lr_res_maxacc <- list(c(), c())
      rf_res_mincost <- list(c(), c())
      rf_res_maxmcc <- list(c(), c())
      rf_res_maxacc <- list(c(), c())
      b_res_mincost <- list(c(), c())
```

```
b_res_maxmcc <- list(c(), c())
b_res_maxacc <- list(c(), c())
svm_res_mincost <- list(c(), c())
svm_res_maxmcc <- list(c(), c())
svm_res_maxacc <- list(c(), c())

#---list of n imputed training datasets with imp = l
train_imp <- imp_n(sets_train_test$amputed_train, imp_meth = l, outvar_idx,
                   to_imp, reg_f, excl_vars)

# #---list of n imputed test sets with imp = l-------
if(l != "ld"){
  train_test <- list()
  for(a in 1:n){
    test_na <- sets_train_test$amputed_test[[a]]
    test_na[, var_amp] <- NA
    train_test[[a]] <- rbind(train_imp[[a]], test_na)
  }

  train_test_imp <- imp_n(train_test, imp_meth = l, outvar_idx,
                          to_imp, reg_f, excl_vars)

  test_imps <- list()
  for(b in 1:n){
    test_imp_1 <- train_test_imp[[b]][-c(1:round(dim(data)[1]*2/3)), ]
    rownames(sets_train_test$amputed_test[[b]]) <- rownames(test_imp_1) <-

    1:round(dim(data)[1]*1/3)
    test_imp <- sets_train_test$amputed_test[[b]]
```

```
    for(v in var_amp){
      test_imp[is.na(test_imp[,v]),v] <- test_imp_1[is.na(test_imp[,v]),v]
      class(test_imp[,v]) <- class(test_comp[[b]][,v])
    }
    test_imps[[b]] <- test_imp
  }
}
#--------------------------------
for(q in 1:n){
  if(l!= "ld") res_kfolds <- kfolds_results(train_imp[[q]], list(test_imps[[q]],

  test_comp[[q]]),
  k=5, form, b_form, out, s)
  else res_kfolds <- kfolds_results(train_imp[[q]],

  list(test_comp[[q]], test_comp[[q]]),
  k=5, form, b_form, out, s)

  for(jj in 1:2){
    lr_res_mincost[[jj]] <- cbind(lr_res_mincost[[jj]],
    t(t(res_kfolds[[1]][[jj]][,1])))

    lr_res_maxmcc[[jj]] <- cbind(lr_res_maxmcc[[jj]],
    t(t(res_kfolds[[1]][[jj]][,2])))

    lr_res_maxacc[[jj]] <- cbind(lr_res_maxacc[[jj]],
    t(t(res_kfolds[[1]][[jj]][,3])))

    rf_res_mincost[[jj]] <- cbind(rf_res_mincost[[jj]],
    t(t(res_kfolds[[2]][[jj]][,1])))
```

```
    rf_res_maxmcc[[jj]] <- cbind(rf_res_maxmcc[[jj]],
    t(t(res_kfolds[[2]][[jj]][,2])))

    rf_res_maxacc[[jj]] <- cbind(rf_res_maxacc[[jj]],
    t(t(res_kfolds[[2]][[jj]][,3])))

    b_res_mincost[[jj]] <- cbind(b_res_mincost[[jj]],
    t(t(res_kfolds[[3]][[jj]][,1])))

    b_res_maxmcc[[jj]] <- cbind(b_res_maxmcc[[jj]],
    t(t(res_kfolds[[3]][[jj]][,2])))

    b_res_maxacc[[jj]] <- cbind(b_res_maxacc[[jj]],
    t(t(res_kfolds[[3]][[jj]][,3])))

    svm_res_mincost[[jj]] <- cbind(svm_res_mincost[[jj]],
    t(t(res_kfolds[[4]][[jj]][,1])))

    svm_res_maxmcc[[jj]] <- cbind(svm_res_maxmcc[[jj]],
    t(t(res_kfolds[[4]][[jj]][,2])))

    svm_res_maxacc[[jj]] <- cbind(svm_res_maxacc[[jj]],
    t(t(res_kfolds[[4]][[jj]][,3])))
  }
}
tmp <- list(lr_res_mincost = lr_res_mincost, lr_res_maxmcc = lr_res_maxmcc,
            lr_res_maxacc = lr_res_maxacc, rf_res_mincost = rf_res_mincost,
            rf_res_maxmcc = rf_res_maxmcc, rf_res_maxacc = rf_res_maxacc,
            b_res_mincost = b_res_mincost, b_res_maxmcc = b_res_maxmcc,
```

```
                    b_res_maxacc = b_res_maxacc, svm_res_mincost = svm_res_mincost,
                    svm_res_maxmcc = svm_res_maxmcc, svm_res_maxacc = svm_res_maxacc)

      name <- paste0(j, "_", l)
      print(name)
      res_full[[name]] <- tmp
    }
  }
  list(res_full = res_full, train_comp = train_comp, test_comp = test_comp)
}

################################---baseline results----

baseline_res <- function(n, k = 5, form, b_form, out, s, results){

  mets <- c("dem.parity", "eq.opp", "pre.eq", "ppv.eq", "npv.eq")

  lr_res_mincost <- c()
  lr_res_maxmcc <- c()
  lr_res_maxacc <- c()
  rf_res_mincost <- c()
  rf_res_maxmcc <- c()
  rf_res_maxacc <- c()
  b_res_mincost <- c()
  b_res_maxmcc <- c()
  b_res_maxacc <- c()
  svm_res_mincost <- c()
  svm_res_maxmcc <- c()
  svm_res_maxacc <- c()
```

```
for(i in 1:n){
  #---run the kfolds function----
  full_res_kfold <- kfolds_results(results$train_comp[[i]], list(results$test_comp[[i]]),
                                   k, form, b_form, out, s)

  lr_res_mincost <- cbind(lr_res_mincost, t(t(full_res_kfold[[1]][[1]][mets,1])))
  lr_res_maxmcc <-cbind(lr_res_maxmcc, t(t(full_res_kfold[[1]][[1]][mets,2])))
  lr_res_maxacc <-cbind(lr_res_maxacc, t(t(full_res_kfold[[1]][[1]][mets,3])))
  rf_res_mincost <- cbind(rf_res_mincost, t(t(full_res_kfold[[2]][[1]][mets,1])))
  rf_res_maxmcc <-cbind(rf_res_maxmcc, t(t(full_res_kfold[[2]][[1]][mets,2])))
  rf_res_maxacc <- cbind(rf_res_maxacc, t(t(full_res_kfold[[2]][[1]][mets,3])))
  b_res_mincost <- cbind(b_res_mincost, t(t(full_res_kfold[[3]][[1]][mets,1])))
  b_res_maxmcc <- cbind(b_res_maxmcc, t(t(full_res_kfold[[3]][[1]][mets,2])))
  b_res_maxacc <- cbind(b_res_maxacc, t(t(full_res_kfold[[3]][[1]][mets,3])))
  svm_res_mincost <- cbind(svm_res_mincost, t(t(full_res_kfold[[4]][[1]][mets,1])))
  svm_res_maxmcc <- cbind(svm_res_maxmcc, t(t(full_res_kfold[[4]][[1]][mets,2])))
  svm_res_maxacc <- cbind(svm_res_maxacc, t(t(full_res_kfold[[4]][[1]][mets,3])))
}

#------quantiles-------------
lr_mincost <- apply(lr_res_mincost, 1, quantile)
lr_maxmcc <-  apply(lr_res_maxmcc, 1, quantile)
lr_maxacc <-  apply(lr_res_maxacc, 1, quantile)
rf_mincost <- apply(rf_res_mincost, 1, quantile)
rf_maxmcc <- apply(rf_res_maxmcc, 1, quantile)
rf_maxacc <- apply(rf_res_maxacc, 1, quantile)
b_mincost <- apply(b_res_mincost, 1, quantile)
b_maxmcc <- apply(b_res_maxmcc, 1, quantile)
b_maxacc <- apply(b_res_maxacc, 1, quantile)
svm_mincost <- apply(svm_res_mincost, 1, quantile)
```

```
  svm_maxmcc <- apply(svm_res_maxmcc, 1, quantile)
  svm_maxacc <- apply(svm_res_maxacc, 1, quantile)

  base <- list(lr_mincost = lr_mincost,
               lr_maxmcc = lr_maxmcc,
               lr_maxacc = lr_maxacc,
               rf_mincost = rf_mincost,
               rf_maxmcc = rf_maxmcc,
               rf_maxacc = rf_maxacc,
               b_mincost = b_mincost,
               b_maxmcc = b_maxmcc,
               b_maxacc = b_maxacc,
               svm_mincost = svm_mincost,
               svm_maxmcc = svm_maxmcc,
               svm_maxacc = svm_maxacc)

  base
}
#####-----------------k-folds results function-------------------
kfolds_results <- function(train, tests, k, form, b_form, out, s){
  #---k-fold cv on train----
  #---Create k equally sized folds----
  folds <- cut(seq(1,nrow(train)),breaks=k,labels=FALSE)
  #k-fold results structures
  lr_pars_sum <- matrix(0, 6, 19)
  rf_pars_sum <- matrix(0, 6, 10)
  b_pars_sum <- matrix(0, 7, 40)
  svm_pars_sum <- matrix(0, 8, 8)

  for(i in 1:k){
```

```
  #---create train and validation sets------
  #---validation set to choose the best parameters for each model-----
  valid_idx <- which(folds==i,arr.ind=TRUE)
  valid <- train[valid_idx, ]
  train_v <- train[-valid_idx, ]

  lr_tune_res <- lr_tune(form, out, train_v, valid)
  rf_tune_res <- rf_tune(form, out, train_v, valid)
  b_tune_res <- b_tune(b_form, out, train_v, valid)
  svm_tune_res <- svm_tune(form, out, train_v, valid)

  lr_pars_sum <- lr_pars_sum + lr_tune_res
  rf_pars_sum <- rf_pars_sum + rf_tune_res
  b_pars_sum <- b_pars_sum + b_tune_res
  svm_pars_sum <- svm_pars_sum + svm_tune_res
}
#----result averaged over k folds-----
lr_pars <- lr_pars_sum/k
rf_pars <- rf_pars_sum/k
b_pars <- b_pars_sum/k
svm_pars <- svm_pars_sum/k

#---choose best models according to the 3 objectives---
lr_best_pars <- best_pars(lr_pars)
rf_best_pars <- best_pars(rf_pars)
b_best_pars <- best_pars(b_pars)
svm_best_pars <- best_pars(svm_pars)

#---get results from best models on test set
lr_res <- list()
```

```
  rf_res <- list()
  b_res <- list()
  svm_res <- list()
  for(jj in 1:length(tests)){
    lr_res[[jj]] <- lr_results(train, tests[[jj]], best_mods = lr_best_pars, form, out, s)
    rf_res[[jj]] <- rf_results(train, tests[[jj]], best_mods = rf_best_pars, form, out, s)
    b_res[[jj]] <- b_results(train, tests[[jj]], best_mods = b_best_pars, b_form, out, s)
    svm_res[[jj]] <- svm_results(train, tests[[jj]],
    best_mods = svm_best_pars, form, out, s, c_w = 0)
  }

  list(lr_res = lr_res, rf_res = rf_res, b_res = b_res, svm_res = svm_res)
}
################################################
###-------amputation---------------
library(mice)
amp_n <- function(data, n, f1, f2, f3, s_p, mdm, var_amp, p,
                  type_mnar, type_mar){
  n_data <- list()
  if(length(var_amp) == 1){
    #---define missingness patterns-----
    pats <- matrix(1,1,dim(data)[2])
    pats[1, var_amp] <- 0
    #---weights for mar mech------
    wgts <- matrix(0,1,dim(data)[2])
    wgts[1, s_p] <- 1
    #----frequencies vector-----
    freqs <- f1

    for(i in 1:n){
```

```
if(mdm == "MCAR"){
  a <- ampute(data, prop = p, patterns = pats,
              mech = "MCAR", freq = freqs)
  n_data[[length(n_data) + 1]] <- a$amp
}
#---MAR or MNAR
if(mdm == "MNAR"){
  a <- ampute(data, prop = p, patterns = pats,
              mech = "MNAR", freq = freqs,
              type = type_mnar)
  dat <- data
  var1 <- a$amp[, var_amp]
  if(class(data[,var_amp])=="factor"){
    dat[ , var_amp] <- as.factor(var1)
  }else{
    dat[ , var_amp] <- var1
  }
  n_data[[length(n_data) + 1]] <- dat

}
if(mdm == "MAR"){
  a <- ampute(data, prop = p, patterns = pats,
              weights = wgts, freq = freqs,
              mech = "MAR", type = type_mar)
  dat <- data
  var1 <- a$amp[, var_amp]
  if(class(data[,var_amp])=="factor"){
    dat[ , var_amp] <- as.factor(var1)
  }else{
    dat[ , var_amp] <- var1
```

```
      }
      n_data[[length(n_data) + 1]] <- dat
    }
  }
  n_data
}else{
  #---define missingness patterns-----
  pats <- matrix(1,3,dim(data)[2])
  pats[1, var_amp[1]] <- pats[2, var_amp[2]] <- pats[3, var_amp[1]] <-
  pats[3, var_amp[2]] <- 0
  #---weights for mar mech------
  wgts1 <- matrix(0,3,dim(data)[2])
  wgts1[1, s_p] <- wgts1[2, s_p] <- wgts1[3, s_p] <- 1
  #---weights for mnar mech------
  wgts2 <- matrix(0,3,dim(data)[2])
  wgts2[1, var_amp[1]] <- wgts2[2, var_amp[2]] <- wgts2[3, var_amp[1]] <- 1
  #----frequencies vector-----
  freqs <- c(f1, f2, f3)

  for(i in 1:n){
    if(mdm == "MCAR"){
      a <- ampute(data, prop = p, patterns = pats,
                  mech = "MCAR", freq = freqs)
      n_data[[length(n_data) + 1]] <- a$amp
    }
    #---MAR or MNAR
    if(mdm == "MNAR"){
      a <- ampute(data, prop = p, patterns = pats, weights = wgts2,
                  mech = "MNAR", freq = freqs,
                  type = type_mnar)
```

```
  dat <- data
  var1 <- a$amp[, var_amp[1]]
  if(class(data[,var_amp[1]])=="factor"){
    dat[ , var_amp[1]] <- as.factor(var1)
  }else{
    dat[ , var_amp[1]] <- var1
  }
  var2 <- a$amp[, var_amp[2]]
  if(class(data[,var_amp[2]])=="factor"){
    dat[ , var_amp[2]] <- as.factor(var2)
  }else{
    dat[ , var_amp[2]] <- var2
  }
  n_data[[length(n_data) + 1]] <- dat
}
if(mdm == "MAR"){
  a <- ampute(data, prop = p, patterns = pats,
              weights = wgts1, freq = freqs,
              mech = "MAR", type = type_mar)
  dat <- data
  var1 <- a$amp[, var_amp[1]]
  if(class(data[,var_amp[1]])=="factor"){
    dat[ , var_amp[1]] <- as.factor(var1)
  }else{
    dat[ , var_amp[1]] <- var1
  }
  var2 <- a$amp[, var_amp[2]]
  if(class(data[,var_amp[2]])=="factor"){
    dat[ , var_amp[2]] <- as.factor(var2)
  }else{
```

```
          dat[ , var_amp[2]] <- var2
        }
        n_data[[length(n_data) + 1]] <- dat
      }
    }
    n_data
  }
}
#------plotting function--------------------
fair_boxplots <- function(ld = T, all = T, base_r, results, lctn){
  if(ld == T){
    imp <- c("ld","mode", "reg", "knn")
  }else{
    imp <- c("mode", "reg", "knn")
  }

  mod_names <- names(results$res_full$MCAR_reg)
  mincost <- mod_names[grep("mincost", mod_names)]
  maxmcc <- mod_names[grep("maxmcc", mod_names)]
  maxacc <- mod_names[grep("maxacc", mod_names)]
  objective <- c("mincost", "maxmcc", "maxacc")
  obj <- cbind(mincost = mincost, maxmcc = maxmcc, maxacc = maxacc)
  n <- dim(results$res_full$MCAR_reg$lr_res_mincost[[1]])[2]
  mets <- c("dem.parity", "eq.opp", "pre.eq", "ppv.eq", "npv.eq")

  for(m in mets){
    for(o in objective){
      if(all == T){
        f_n <- paste0(lctn, m, "_", o, ".pdf")
        pdf(f_n, 8, 7)
```

```
  par(mar=c(2.2,4,3.5,1))
  if(ld == T){
    par(mfrow = c(4,4))
  }
  if(ld == F){
    par(mfrow = c(4,3))
  }
}

for(i in obj[, o]){
  mod_name <- strsplit(i, split = "_")[[1]][1]
  if(all==F){
    f_n <- paste0(lctn, m, "_", o, "_", mod_name, ".pdf")
    pdf(f_n, 8, 5)
    par(mar=c(2.2,4,3.5,1))
    if(ld == T){
      par(mfrow = c(1, 4))
    }
    if(ld == F){
      par(mfrow = c(1, 3))
    }
  }
  for(j in imp){
    idx <- grep(j, names(results$res_full))
    imp_mdm <- names(results$res_full[idx])
    df <- data.frame(matrix(ncol = 3, nrow = n))
    x <- c("mcar", "mar", "mnar")
    count = 1
    for(k in imp_mdm){
      colm <- c()
```

```
  for(l in 1:n){
    ## if ld is true compare all on comp test
    ## if ld false compare all non-ld on imputed test
    if(ld == T) colm <- rbind(colm,
    t(t(results$res_full[[k]][[i]][[2]][,l]))[m, ])
    else colm <- rbind(colm, t(t(results$res_full[[k]][[i]][[1]][,l]))[m, ])
  }
  df[, count] <- colm
  count = count + 1
}
name <- paste0(j, ".", mod_name)

base_name <- paste0(mod_name, "_", o)

#---baseline-----------------
iqr_min <- base_r[[base_name]][2, m]
med <- base_r[[base_name]][3, m]
iqr_max <- base_r[[base_name]][4, m]
#---------------------

#--------------------------------
names(df) <- x
boxplot(df, ylab = m, main = name, cex.main = 1.1, cex.axis = 1.1,
        cex.lab = 1.1)
av <- apply(df, 2, mean)
names(av) <- x
points(av, pch=20, col="blue")
abline(h = med, col = "red")
abline(h = iqr_min, col = "red", lty = 2, lwd = .5)
abline(h = iqr_max, col = "red", lty = 2, lwd = .5)
```

```
        }
        if(all == F){
          mtext(text = o, side = 3, line = -1, outer = TRUE, cex = 0.9, col = "grey50")
          dev.off()
        }
      }
      if(all == T){
        mtext(text = o, side = 3, line = -1, outer = TRUE, cex = 0.9, col = "grey50")
        dev.off()
      }
    }
  }
}

#---function to create the various subsets-------
#---takes in amputed data set list----

train_test_sets <- function(data, amp_data, n){
  complete_test <- list()
  amputed_test <- list()
  amputed_train <- list()
  train_full <- list()

  for(i in 1:n){
    set.seed(i)
    test_idx <- sample(1:nrow(data), round(1/3*nrow(data)))
    #----complete test set----
    test_comp <- data[test_idx, ]
    complete_test[[i]] <- test_comp
```

```
    #----complete training set---
    train_full[[i]] <- data[-test_idx, ]
    #----amputed test set----
    test_amp <- amp_data[[i]][test_idx, ]
    amputed_test[[i]] <- test_amp
    #---amputed training set---
    train_amp <- amp_data[[i]][-test_idx, ]
    amputed_train[[i]] <- train_amp
  }
  list(complete_test = complete_test, amputed_test = amputed_test,
       amputed_train = amputed_train, train_full = train_full)
}

#----imputation fuction---
imp_n <- function(l_dats, imp_meth, outvar_idx, to_imp, reg_f, excl_vars){
  imp_dats <- list()
  for(i in 1:length(l_dats)){
    #----listwise deletion-----------
    if(imp_meth == "ld"){
      idx_comp <- complete.cases(l_dats[[i]])
      comp <- l_dats[[i]][idx_comp, ]
      imp_dats[[length(imp_dats) + 1]] <- comp
    }
    #-----knn imputation-----------
    if(imp_meth == "knn"){
      knn_imp <- kNN(l_dats[[i]], variable = to_imp, numFun = mean,
                     useImputedDist = FALSE, imp_var = FALSE,
                     dist_var = colnames(l_dats[[i]][, -outvar_idx])[-excl_vars])

      imp_dat <- knn_imp
```

```
      imp_dats[[length(imp_dats) + 1]] <- imp_dat
    }
    #---mode imputation------categorical vars only----
    if(imp_meth == "mode"){
      #---get the mode of the variable to be imputed---
      #---replace NA's by mode-------
      for(j in 1:length(to_imp)){
        if(class(l_dats[[i]][, c(to_imp[j])]) == "factor"){
        mode1 <- names(which.max(summary(na.omit(l_dats[[i]][, c(to_imp[j])]))))
        l_dats[[i]][which(is.na((l_dats[[i]][, c(to_imp[j])]))), c(to_imp[j])] <- mode1
        }
        else{
          mode1 <- Mode(na.omit(l_dats[[i]][, c(to_imp[j])]))
          l_dats[[i]][which(is.na((l_dats[[i]][, c(to_imp[j])]))), c(to_imp[j])] <- mode1
        }

      }
      imp_dats[[length(imp_dats) + 1]] <- l_dats[[i]]
      }
    #--regression imputation----
    if(imp_meth=="reg"){
      reg_imp <- regressionImp(reg_f, l_dats[[i]], imp_var = FALSE)
      imp_dat <- reg_imp
      imp_dats[[length(imp_dats) + 1]] <- imp_dat
    }
  }

  imp_dats
}
```

```
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

##----------------Logistic Regression-----------------
#-----training/validation function------
lr_tune <- function(form, out, train_v, valid){
  #-----fit on train_v--------
  lr <- glm(form, data=train_v, family="binomial")
  #----calculate model over 19 thresholds on valid------
  thresh <- seq(0.05, 0.95, length.out = 19)
  lr_valid <- predict(lr, valid, type="response")
  lr_results_perf <- c()
  for(j in thresh){
    class_valid <- rep(1 , nrow(valid))
    class_valid[lr_valid <= j] <- 0
    class_valid <- factor(class_valid, levels = c(0,1))
    cm_lr <- table(pred = class_valid, actual = valid[, out])
    perf_res_lr <- perf_mets(cm_lr)
    p_res_lr <- rbind(thresh = j, perf_res_lr)
    lr_results_perf <- cbind(lr_results_perf, p_res_lr)
  }
  lr_results_perf
}

#----test set results function-----
#---takes in best models as input----
lr_results <- function(train, test, best_mods, form, out, s){
  #----fit (on full train set) and predict (on test set) the best models
```

```
##------Logistic regression---------
lr_train <- glm(form, data=train, family="binomial")
lr_preds <- predict(lr_train, test, type="response")
####-------------####---------------###-----------------
res_lr <- c()
for(i in 1:3){
  test_preds <- rep(1 , nrow(test))
  test_preds[lr_preds <= best_mods[1,i]] <- 0
  test_preds <- factor(test_preds, levels = c(0,1))
  #--dem_parity table----
  tab <- table(y_hat = test_preds, s = test[, s])
  #---other metrics, 2 tables, for s = 0/ s = 1
  tab0<- table(pred = test_preds[which(test[, s]==0)],
               actual = test[which(test[, s]==0), out])
  tab1 <- table(pred = test_preds[which(test[, s]==1)],
                actual = test[which(test[, s]==1), out])

  #---fairness metrics values on test set-----
  fair_mets <- fair_mets_classifier(tab, tab1, tab0)
  #---performance metrics on test set----
  cm <- table(pred = test_preds, actual = test[, out])
  perf_res <- perf_mets(cm)
  perf_res[5, 1] <- perf_res[5, 1]/sum(cm)

  #-----combine fairness and performance results----
  best <- rbind(best_mods[1,i], perf_res, fair_mets)
  rownames(best)[1] <- "thresh"
  res_lr <- cbind(res_lr, best)
}
colnames(res_lr) <- c("min cost", "max mcc", "max acc")
```

```
  res_lr
}
##--------------Random Forests------------------
library(randomForest)

rf_tune <- function(form, out, train_v, valid){
  #----fit model over 10 cutoffs on train and validate on valid------
  p <- seq(0.05, 0.95, length.out = 10)
  rf_mods_valid <- c()
  for(i in p){
    #---fit on train----
    rf <- randomForest(form, train_v, cutoff = c(i, 1-i))
    #---validate on valid---
    rf_pred <- predict(rf, newdata = valid)
    cm <- table(pred = rf_pred, actual = valid[, out])
    perf_res <- perf_mets(cm)
    rf_mods_valid <- cbind(rf_mods_valid, rbind(cutoff = i, perf_res))
  }
  rf_mods_valid
}

rf_results <- function(train, test, best_mods, form, out, s){
  res_rf <- c()
  for(i in 1:3){
    rf_train <- randomForest(form, train,
                             cutoff = c(best_mods[1,i], 1-best_mods[1, i]))
    rf_preds <- predict(rf_train, newdata = test)
    #--dem_parity table----
    tab <- table(y_hat = rf_preds, s = test[, s])
    #---other metrics, 2 tables, for s = 0/ s = 1
```

```
    tab0<- table(pred = rf_preds[which(test[, s]==0)],
                 actual = test[which(test[, s]==0), out])
    tab1 <- table(pred = rf_preds[which(test[, s]==1)],
                  actual = test[which(test[, s]==1), out])

    #----fairness metrics values on test set-----
    fair_mets <- fair_mets_classifier(tab, tab1, tab0)
    #----performance metrics on test set-------
    cm <- table(pred = rf_preds, actual = test[, out])
    perf_res <- perf_mets(cm)
    perf_res[5, 1] <- perf_res[5, 1]/sum(cm)

    #-----combine fairness and performance results----
    best <- rbind(best_mods[1,i], perf_res, fair_mets)
    rownames(best)[1] <- "cutoff"
    res_rf <- cbind(res_rf, best)
  }
  colnames(res_rf) <- c("min cost", "max mcc", "max acc")
  res_rf
}

#####------------boosting -----------------
library(gbm)

b_form <- ifelse(as.integer(risk)==1, 0, 1) ~ . - sex
b_tune <- function(b_form, out, train_v, valid){
  #---fit model with different shrinkage on train_v-----
  #---validate the models on valid with different thresh----
  sh <- c(0.001, 0.01, 0.1, 0.2)
  th <- seq(0.05, 0.95, length.out = 10)
```

```
  b_mods_valid <- c()
  for(i in sh){
    #---fit on train----
    b <- gbm(b_form, data = train_v, distribution = "bernoulli", shrinkage = i)
    #---validate on valid---
    b_pred <- predict(b, newdata = valid, n.trees = 100, type = "response")
    for(j in th){
      class_valid <- rep(1 , nrow(valid))
      class_valid[b_pred <= j] <- 0
      class_valid <- factor(class_valid, levels = c(0,1))
      cm <- table(pred = class_valid, actual = valid[, out])
      perf_res <- perf_mets(cm)
      b_mods_valid <- cbind(b_mods_valid,
                            rbind(shrinkage = i, perf_res, thresh = j))
    }
  }
  b_mods_valid
}

b_results <- function(train, test, best_mods, b_form, out, s){
  res_b <- c()
  for(i in 1:3){
    b_mod <- gbm(b_form, data = train, distribution = "bernoulli",
                 shrinkage = best_mods[1, i])
    b_test <- predict(b_mod, newdata = test, n.trees = 100, type = "response")

    class_test <- rep(1 , nrow(test))
    class_test[b_test <= best_mods[7, i]] <- 0
    class_test <- factor(class_test, levels = c(0,1))
    #--dem_parity table----
```

```
    tab <- table(y_hat = class_test, s = test[, s])
    #---other metrics, 2 tables, for s = 0/ s = 1
    tab0<- table(pred = class_test[which(test[, s]==0)],
                 actual = test[which(test[, s]==0), out])
    tab1 <- table(pred = class_test[which(test[, s]==1)],
                  actual = test[which(test[, s]==1), out])

    #---fairness metrics values on test set-----
    fair_mets <- fair_mets_classifier(tab, tab1, tab0)
    #------performance metrics on test set----
    cm <- table(pred = class_test, actual = test[, out])
    perf_res <- perf_mets(cm)
    perf_res[5, 1] <- perf_res[5, 1]/sum(cm)

    #-----combine fairness and performance results----
    best <- rbind(best_mods[1,i], perf_res, fair_mets, best_mods[7, i])
    rownames(best)[1] <- "shrinkage"
    rownames(best)[12] <- "thresh"
    res_b <- cbind(res_b, best)
  }
  colnames(res_b) <- c("min cost", "max mcc", "max acc")
  res_b
}

##------------------svm-----------------------
library(e1071)

svm_tune <- function(form, out, train_v, valid, c_w = 0){
  gam <- c(2^-5, 2^-4, 2^-3, 2^-2)
  c <- c(0.1, 1, 10)
```

```
wgt <- c(1, 2, 3, 4)
svm_mods <- c()
for(i in c){
  for(k in wgt){
    #----fit models on train------
    #---validate on valid----
    c_wts <- c(k)
    names(c_wts) <- as.character(c_w)
    svm1 <- svm(form, data = train_v, scale = TRUE, kernel = "linear", cost = i,
                class.weights = c_wts)
    svm_pred1 <- predict(svm1, newdata = valid)
    cm1 <- table(pred = svm_pred1, actual = valid[, out])
    perf_res1 <- perf_mets(cm1)
    svm_mods <- cbind(svm_mods, rbind(svm_c = i, perf_res1, wght = k,
                                      gamma = NA))

    for(j in gam){
      #----fit models on train------
      #---validate on valid----
      svm2 <- svm(form, data = train_v, scale = TRUE, kernel = "radial",
                  cost = i, gamma = j, class.weights = c_wts)
      svm_pred2 <- predict(svm2, newdata = valid)
      cm2 <- table(pred = svm_pred2, actual = valid[, out])
      perf_res2 <- perf_mets(cm2)
      svm_mods <- cbind(svm_mods, rbind(svm_c = i, perf_res2, wght = k,
                                        gamma = j))
    }
  }
}
svm_mods
```

```
}

svm_results <- function(train, test, best_mods, form, out, s, c_w = 0){
  res_svm <- c()
  for(i in 1:3){
    if(is.na(best_mods[8, i])){
      #----linear kernel----
      c_wts <- c(best_mods[7, i])
      names(c_wts) <- as.character(c_w)
      svm_mod <- svm(form, data = train, scale = TRUE, kernel = "linear",
                     cost = best_mods[1, i], class.weights = c_wts)
    }else{
      #---radial kernel----
      c_wts <- c(best_mods[7, i])
      names(c_wts) <- as.character(c_w)
      svm_mod <- svm(form, data = train, scale = TRUE, kernel = "radial",
                     cost = best_mods[1, i], gamma = best_mods[8, i],
                     class.weights = c_wts)
    }

    test_svm <- predict(svm_mod, newdata = test)
    #--dem_parity table----
    tab <- table(y_hat = test_svm, s = test[, s])
    #---other metrics, 2 tables, for s = 0/ s = 1
    tab0 <- table(pred = test_svm[which(test[, s]==0)],
                  actual = test[which(test[, s]==0), out])
    tab1 <- table(pred = test_svm[which(test[, s]==1)],
                  actual = test[which(test[, s]==1), out])

    #---fairness metrics values on test set-----
```

```
    fair_mets <- fair_mets_classifier(tab, tab1, tab0)
    #----------performance metrics on test set----
    cm <- table(pred = test_svm, actual = test[, out])
    perf_res <- perf_mets(cm)
    perf_res[5, 1] <- perf_res[5, 1]/sum(cm)

    #-----combine fairness and performance results----
    best <- rbind(best_mods[1,i], perf_res, fair_mets)
    rownames(best)[1] <- "svm_cost"
    res_svm <- cbind(res_svm, best)
  }
  colnames(res_svm) <- c("min cost", "max mcc", "max acc")
  res_svm
}

#####------------###############------------#########
#----best models according to 3 objectives-----
#----same function for all models--------
#---takes in a matrix of all parameter values/models----
#---outputs best model according to objective---
best_pars <- function(pars_full){
  #----min cost model---
  min_cost_pars <- pars_full[, which.min(pars_full[6, ])]
  #---max mcc model----
  max_mcc_pars <- pars_full[, which.max(pars_full[5, ])]
  #---max acc model----
  max_acc_pars <- pars_full[, which.max(pars_full[2, ])]
  #----results----
  data.frame(min_cost = min_cost_pars, max_mcc = max_mcc_pars,
             max_acc = max_acc_pars)
```

```
}

#-----------fairness metrics-----------
#---outcome Y: 1 is positive outcome, 0 is negative outcome
#---sensitive attribute S: 1 is privileged class, 0 is unprivileged class

#-------------#----------------#-------------------#---------------
#---1. demographic parity--------
#---Can be calculated on a dataset or prediction output of a classifier----
#---takes a table as input------
#---table is of form, (rows are outcome Y/Y_hat and cols are sens. attr S)-----
#-----          S
#----         0   1  -----
#- Y_hat| 0   a   b
#-        1   c   d

dem_par <- function(tab){
  tab[2, 2]/(tab[1,2] + tab[2,2]) - tab[2, 1]/(tab[1,1] + tab[2,1])
}

#----the next metrics can only be calculated with predictions from a model----
#----require as input 2 tables, one for each binary sensitive attribute class
#---the tables are just the confusion matrices --------
#---table is of form, (rows are Y_hat and cols are Y)-----
#-----          Y
#----         0   1  -----
#- Y_hat| 0   a   b
#-        1   c   d

#--tab1 is for privileged and tab2 for unprivileged-----
```

```
#---2. Equality of opportunity/ equality of TPR-----
eq_opp <- function(tab1, tab2){
  tab1[2,2]/(tab1[1,2]+tab1[2,2]) - tab2[2,2]/(tab2[1,2]+tab2[2,2])
}

#---3. Predictive Equality/ equality of FPR-------
pre_eq <- function(tab1, tab2){
  tab1[2,1]/(tab1[1,1]+tab1[2,1]) - tab2[2,1]/(tab2[1,1]+tab2[2,1])
}

#---4. TNR/specificity equality-------
tnr_eq <- function(tab1, tab2){
  tab1[1,1]/(tab1[1,1]+tab1[2,1]) - tab2[1,1]/(tab2[1,1]+tab2[2,1])
}

#---5. FNR equality-----
fnr_eq <- function(tab1, tab2){
  tab1[1,2]/(tab1[1,2]+tab1[2,2]) - tab2[1,2]/(tab2[1,2]+tab2[2,2])
}

#---6. Overall accuracy equality----
ov_acc_eq <- function(tab1, tab2){
  (tab1[1,1]+tab1[2,2])/sum(tab1) - (tab2[1,1]+tab2[2,2])/sum(tab2)
}

#---7. PPV equality/ precision equality/ predictive parity-------
ppv_eq <- function(tab1, tab2){

  den1 <- tab1[2,1]+tab1[2,2]
```

```
  den2 <- tab2[2,1]+tab2[2,2]

  if(den1 == 0 && den2 == 0){
    0
  }else if(den1 == 0 && den2 != 0){
    -(tab2[2,2]/den2)
  }else if(den2 == 0 && den1 != 0){
    tab1[2,2]/den1
    }else{
  tab1[2,2]/den1 - tab2[2,2]/den2
  }
}

#---8. NPV equality-------
npv_eq <- function(tab1, tab2){

  den1 <- tab1[1,1]+tab1[1,2]
  den2 <- tab2[1,1]+tab2[1,2]

  if(den1 == 0 && den2 == 0){
    0
  }else if(den1 == 0 && den2 != 0){
    -(tab2[1,1]/den2)
  }else if(den2 == 0 && den1 != 0){
    tab1[1,1]/den1
    }else{
    tab1[1,1]/den1 - tab2[1,1]/den2
  }
}
```

```
############################
#-----Matthews correlation coefficient------
#----takes in confusion matrix/table as input where-----
#---predictions are rows and truth is columns-----
#----assume binary classes are ordered and called 0 and 1
matCC <- function(cm){
  cm <- matrix(as.numeric(cm), 2, 2)
  num <- cm[2,2]*cm[1,1] - cm[2,1]*cm[1,2]
  t1 <- cm[2,2] + cm[2,1]
  t2 <- cm[2,2] + cm[1,2]
  t3 <- cm[1,1] + cm[2,1]
  t4 <- cm[1,1] + cm[1,2]

  if(t1==0|t2==0|t3==0|t4==0){
    den = 1}
  else{
    den <- sqrt(t1*t2*t3*t4)
  }
  num/den
}

#----------############----------################-----------

#-------performance metrics result function--------
#---takes in confusion matrix/table as input, where-------
#-----predictions are rows and truth is in columns------
#----assume binary classes are ordered and called 0 and 1
perf_mets <- function(cm){
  acc <- (cm[1,1] + cm[2,2])/sum(cm)
  pos_acc <- cm[2,2]/(cm[2,2] + cm[1,2])
```

```
  neg_acc <- cm[1,1]/(cm[1,1] + cm[2,1])
  mcc <- matCC(cm)
  cost <- cm[1,2] + 5*cm[2,1]
  mets <- rbind(accy = acc, pos_accy = pos_acc, neg_accy = neg_acc, MCC = mcc,
                cost = cost)
  mets
}

#-####------------######------------######--------------
#----fairness metrics results function-----
#--on classifier outputs, all 5------
#---tab1 is counts of Y_hat and S for dem_parity---
#---tab2 and tab3 are confusion mats for privileged and unprivileged respectiv.
#---form of tables is described in fair_mets.r-------
fair_mets_classifier <- function(tab1, tab2, tab3){
  f_mets <- rbind(dem_par(tab1), eq_opp(tab2,tab3), pre_eq(tab2, tab3),
                  ppv_eq(tab2, tab3), npv_eq(tab2, tab3))
  rownames(f_mets) <- c("dem.parity", "eq.opp", "pre.eq", "ppv.eq", "npv.eq")
  f_mets
}
```