The Effects of Missing Data Imputation on Bias in Classification

Sida Zhu and Sucheta Soundarajan
Syracuse University
szhu28@syr.edu, susounda@syr.edu

Abstract

In recent years, a great deal of research has focused on bias in machine learning. In areas such as criminal sentencing, such bias may result in catastrophic consequences. In real-world applications, data for training is often incomplete. To fill the missing values, different data imputation methods are used before training a model. In this paper, we explore the effect of data imputation on classification bias. We combine 3 data imputation methods with 7 classic machine learning algorithms, and demonstrate that data imputation techniques can result in large reductions in classification bias with only small reductions in accuracy.

1 Introduction

Over the last few decades, machine learning algorithms have evolved rapidly, and have penetrated into almost all aspects of society. As was famously discussed by ProPublica in [Angwin *et al.*, 2016], criminal sentencing is one such area. In particular, states across the U.S. have adopted the COMPAS machine learning recidivism prediction algorithm, and use this tool for post-conviction criminal sentencing [Angwin *et al.*, 2016]. The algorithm takes as input a convicted criminal's basic information such as age, charge degree, and charge description, and predicts a risk score indicating whether the individual will reoffend or not.

However, according to ProPublica's analysis, even though race was not used by the algorithm, the resulting output was biased against African Americans [Larson *et al.*, 2016].[1] ProPublica concluded this on the basis of the classification confusion matrices for African Americans and Caucasians: in particular, African Americans were more likely to wrongly be predicted as being of high risk of reoffense; while Caucasians were more likely to be judged as unlikely to reoffend when they actually did reoffend. Such bias is especially devastating in the area of criminal sentencing.

[1]NorthPointe, the company that created COMPAS, uses a different definition of fairness to dispute ProPublica's findings [Larson and Angwin, 2016]. For purposes of this study, we use ProPublica's definition.

As is well known in the fair machine learning research community, such bias often originates from bias in the training data. In real world applications, datasets are often incomplete. Thus, when training a machine learning model, *data imputation* is often necessary. Such techniques 'fill in' missing values in a dataset so that a classifier can then be trained.

Our goal in this work is to *examine the effect of data imputation strategies on bias in classification.*. We demonstrate that on the ProPublica COMPAS dataset [Larson *et al.*, 2016], use of common data imputation algorithms tends to decrease output bias, sometimes by over 50%.

The rest of this paper is structured as follows: first, we give an overview of related work. We then provide background on the machine learning algorithms and data imputation methods used in our study, as well as other necessary background. We then present results of our analysis, and end with a discussion of possible future work.

2 Related Work

In recent years, topics related to fairness in machine learning have received a great deal of attention. The work in this paper is based on the well-known analysis by ProPublica, which used confusion matrices to analyze results from the COMPAS recividism prediction software and concluded that the analysis was biased against African-Americans [Angwin *et al.*, 2016]. However, there are competing definitions of fairness [Flores *et al.*, 2016]. Kleinberg et al. showed that several notions of fairness cannot simultaneously be satisfied unless the underlying distributions across the protected classes are the same [Kleinberg *et al.*, 2017]. Much work in fair machine learning has attempted to design algorithms for mitigating bias in classification. Such algorithms generally either modify the input data, modify an existing algorithm itself, or modify the output of some classification algorithm [Mehrabi *et al.*, 2019].

Because real datasets often have missing values, data imputation is an important and active research area in statistics and machine learning. Simple techniques for data imputation include filling in missing values by the mean of the column (feature) or finding rows that are similar with respect to other features and taking the average of the missing feature value from those rows [Leeuw, 2001]. The current state-of-the-art is *multiple imputation*, which fits the data to a distribution, and draws from that distribution to fill in missing values, thus generating multiple complete datasets [van Buuren and Groothuis-Oudshoorn, 2011].

By far the closest work to ours is that of Martinez-Plumed et al., who study fairness and missing values [Martínez-Plumed *et al.*, 2019]. However, their work focuses primarily on analyzing fairness with respect to which values are missing, and while they explore data imputation and fairness, they only examine the very simple data imputation method of filling in by the column mean.

3 Background

In this section, we discuss the 7 machine learning algorithms and 3 data imputation techniques from our study, as well as various other details of the workflow. We then discuss the technique used by ProPublica to compute bias.

3.1 Machine Learning Algorithms

Here, we present the 7 machine learning algorithms used in this study, and describe various other aspects of our implementation. For all algorithms, we use implementations from https://scikit-learn.org/.

1. The **K-Nearest Neighbors (KNN)** is a classic machine learning algorithm. In this algorithm, each unlabeled data point is assigned a label by taking a majority vote over the K nearest labeled data points. K is defined by the user [Cover and Hart, 1967].
2. **Support Vector Machine (SVM)** is an algorithm that maps the data into a high-dimensional space, and attempts to find the boundaries between different classes in that space. The *Scikit-Learn* implementation uses the Radial Basis Function (RBF) kernel from the the *libsvm* library [Chang and Lin, 2011]
3. **Linear Support Vector Machine (LinearSVM)** is similar to the Support Vector Machine algorithm, but uses a linear kernel instead. Linear Support Vector Machine produces linear boundaries, instead of the non-linear boundaries from Support Vector Machine using RBF kernel. The *Scikit-Learn* implementation is based on the *liblinear* library [Fan *et al.*, 2008].
4. **Logistic Regression (LogReg)** uses a statistical approach to compute the probability of a label for a given data point. The original algorithm can only produce binary classifications, since its output is between 0 and 1. However, it can also be adapted to multi-label classification by using the one-to-rest method [Kleinbaum *et al.*, 2002]. *Scikit-Learn* by default applies regularization.
5. The **Decision Tree (Tree)** algorithm generates a tree-like internal structure that directs each sample to a predicted label. The leaves at the end of the tree are the target labels, and the internal nodes are the conditions that decides which branch the prediction will go [Safavian and Landgrebe, 1991].
6. **Random Forest (Forest)** uses ensembles of multiple decision trees, and produces an output from the the average of outputs from the decision trees. By ensembling different random decision trees, Random Forest better avoids over-fitting [Liaw *et al.*, 2002].
7. **Multi-layer Perceptron (MLP)** is a neural network approach to classification problems. The model creates a neural network that has multiple layers of perceptrons: one input layer, one output layer, and several hidden layers. Each perceptron is a binary function that controls data flow through the network. The model will be trained with an optimizer that adjusts the weight on each node of the network, in order to best fit the input data.

For all algorithms, because the dataset is imbalanced, we use **SMOTE** to balance the classes[Chawla *et al.*, 2002]. In this research, we use the *SVMSMOTE* implementation from the *imbalanced-learn* library, which uses SVM to select samples for data re-sampling.

Parameter searching is implemented using the *GridSearchCV* function from *Scikit-Learn*. The function tries all combinations of the given parameters, and runs the predefined machine learning algorithm with each combination, on the given dataset, using cross-validation. The function then outputs the parameter combination producing the highest accuracy on the dataset.

Label encoding for categorical features is implemented using the *LabelEncoder* function from *Scikit-Learn*, which converts each categorical value to a unique index number.

3.2 Data Imputation Methods

Here, we discuss the data imputation techniques that we consider in our study. Multiple imputation (discussed below) is most sophisticated of the considered methods, and we include the others for the sake of comparison.

Fill by Average is a simple method that fills each missing data entry by the average of that column.

Fill by Similar Data finds the K most similar data samples in the dataset, and fills in the missing value by the average of those samples. The method in this research uses the *KNNImputer* implemented by *Scikit-Learn*. The similarity between 2 data samples is computed using the features where both are not empty. The implementation uses K-Nearest Neighbors to find the most similar samples. In this research, K is set to 5.

Fill by Multiple Imputation produces multiple complete datasets by applying a multivariate imputer to the missing values multiple times. The imputer used in this research is the *IterativeImputer* from Scikit-Learn. The idea is based on the Multivariate Imputation by Chained Equations (MICE) algorithm [van Buuren and Groothuis-Oudshoorn, 2011]. To use this method in our study, we produce multiple complete datasets, perform classification on each, and report the majority of the resulting predictions.

3.3 Bias in Classification

The literature contains several ways to calculate the fairness of classification outputs. Such fairness is generally computed with respect to so-called *protected groups* defined on the basis of features like race, sex, etc. Kleinberg et al. famously proved a fairness impossibility theorem showing that it is impossible to simultaneously satisfy three particular definitions of fairness [Kleinberg *et al.*, 2017]. In this work, we follow the measure used by ProPublica [Larson *et al.*, 2016], and use confusion matrices computed from the algorithm's output.

For binary targets such as 0 and 1, a confusion matrix contains four values: True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). We can then compute the false positive rate (FPR) and false negative rate (FNR) by dividing these values by the total number of rows with same true label. In this research, the FPR and FNR for each race group, FPR_AA for example, are computed by using the confusion matrix output values from that specific race.

$$\mathtt{FPR} = \frac{\mathtt{FP}}{\mathtt{FP} + \mathtt{TN}}.$$

$$\mathtt{FNR} = \frac{\mathtt{FN}}{\mathtt{TP} + \mathtt{FN}}.$$

.

The positive class represents those who reoffend, and the negative class is those who do not reoffend. A false positive thus represents an individual who was misjudged as likely to reoffend, when he or she actually did not reoffend; and a false negative is an individual who got the benefit of the doubt, and did reoffend even though the opposite was predicted.

ProPublica computed the FPR and FNR for African-Americans and Caucasians, and observed that the FPR for African-Americans was substantially higher than for Caucasians, and vice versa for FNR. ProPublica thus concluded that the classification results were biased.

4 Experimental Results

4.1 Dataset and Pre-processing

We use the ProPublica COMPAS dataset (named ”compas-scores.csv”), which can be retrieved from the official *Github* page of ProPUBLICA. This dataset contains information for 11757 defendants from Broward County, Florida. The dataset contains many features that are not relevant to our analysis, and we select the following 11 features:

Feature	Description
`age`	Numerical Age
`age_cat`	Age Category
`c_charge_degree`	Charge Degree
`c_charge_desc`	Detailed Charge Description
`priors_count`	Prior Offenses Count
`juv_misd_count`	Juvenile Offenses Count
`juv_fel_count`	Juvenile Offenses Count
`juv_other_count`	Juvenile Offenses Count
`days_b_screening_arrest`	Days Before Arrest from Charge Date
`c_jail_in`	Date into Jail
`c_jail_out`	Date out from Jail

Table 1: Feature Descriptions

For the target variable, we use the `is_recid` feature, which tells us whether the individual actually recidivates within two years of the COMPAS offense.

When choosing the features for training, we deliberately avoid identities, names, and those that comes from the COMPAS machine scores, such as `decile_score`. Additionally, we avoid personal protected features such as `sex` and `race`, which a judge should not take into account.

Before feeding the data into the machine learning algorithms, we pre-process the dataset, following the guide of the *COMPAS Analysis notebook* from ProPUBLICA , and perform the following steps:

1. We limit the `days_b_screening_arrest` to the range of [-30, 30], since if the interval is longer than 30 days, the information is likely incorrect.
2. We remove the rows whose `c_charge_degree` is ”O”, which represents traffic offenses.
3. We remove the rows where `is_recid` is -1, which means no COMPAS case was found for the defendant.
4. We combine `c_jail_in` and `c_jail_out`, and create a new feature `length_of_stay`, which is the days between the date into jail, and the date out from jail. This step is inspired from the *Responsibly* package.

After applying a Label Encoder to each categorical feature, the dataset is successfully prepared for training.

4.2 Measuring Bias

In this research, we focus our bias analysis on two races: ”African-American” (AA) and ”Caucasian” (C), which are the major races represented in the dataset. There are 4878 `AA`, and 3371 `C` among all defendants.

To measure bias in output, we use the False Negative and False Positive Rates of these 2 races: FNR_{AA}, FNR_C, FPR_{AA}, FPR_C.

In this dataset, `FPR` represents how likely it is that a defendant will be wrongly predicted as not a recidivist, while actually recidivating. The `FNR` measures how likely it is that a defendant will be wrongly predicted as a recidivist, while actually not recidivating. Under this context, we define a metrics for evaluating a classifier's bias:

$$Bias = \left| \frac{\mathtt{FPR_{AA}}}{\mathtt{FPR_C}} - \frac{\mathtt{FNR_{AA}}}{\mathtt{FNR_C}} \right|.$$

The `Bias` represents the scale of how differently African-American and Caucasian defendants are being wrongly judged by the system. It computes the ratio of `FPR` and the ratio of `FNR` between two races, and measures the scale of bias against two races. As a result, the smaller the value, the less bias between two races the classifier has.

4.3 Procedure Details

Before computing the biases, we first perform a parameter search for each machine learning algorithm on the training data to find the best parameters for each classifier.

To fully evaluate the classifiers in terms of bias, we use 10-fold cross validation. For each 9:1 combination, we select 9 folds as training data, and the remaining 1 fold as test data. We first run the selected imputation method on it to fill in the missing values, and apply SMOTE to balance the training data. Then for each classifier, we fit a model on the training data, run a prediction on the test data, and compute a confusion matrix for each race (`AA` and `C`). From the confusion matrix values, we then compute two types of bias and also an accuracy on the two races. The final values for `Bias` and the accuracy are computed by taking the average of the 10-fold outputs.

For Multiple Imputation, where the output from the data imputation method is multiple complete datasets (5 in our implementation), we use the majority voting to combine the results. The training data and test data are imputed separately, and we get 5 versions of each. Then for each type of classifier, we train 5 classifiers on the training data, and run prediction on each test data, and get 25 predictions. To combine the predictions, we take the majority predicted value for each sample as the final prediction. The rest of the process is the same.

4.4 Imputation on Simulated Missing Values

The original COMPAS dataset has rows with missing values. Before beginning our experiments, we remove those rows. All analysis is performed on the remaining dataset. As a baseline, we compute the `bias` for American-African and Caucasian defendants on this dataset. We then generate three incomplete datasets with, respectively, 20%, 10%, and 5% of the feature values missing. We remove values uniformly at random.

In the following figures, the grey bars represent results on the original dataset. The red, green, and blue bars show, respectively, the results of applying `Fill by Average`, `Fill by Similar`, and `Multiple Imputation` methods on the incomplete datasets..

Figure 1: `Bias` before and after data imputation. Data imputation generally reduces the amount of classification bias.

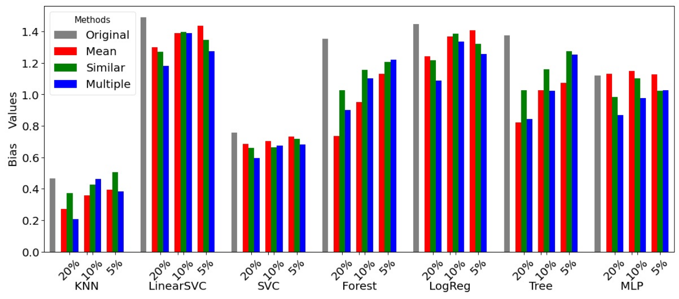
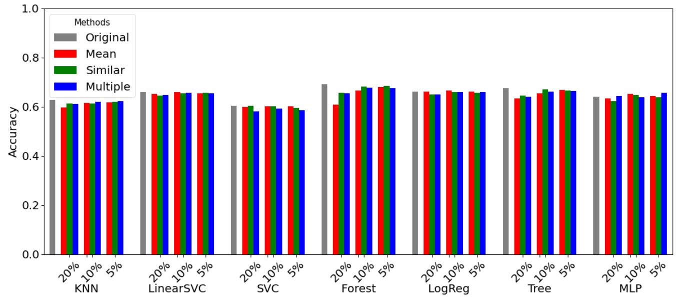


Figure 2: Accuracy before and after data imputation. Accuracy generally remains high after data imputation.



First, from Figure 2, we see that data imputation generally results in a very small drop in accuracy, even when up to 20% of the data is imputed. Figures 1 show the computed `Bias` values from each scenario. By analyzing and comparing the results, we make the following observations:

1. If we compare the results from the original data (grey bars) to those ones the imputed data (other bars), for each classifier, with rare exceptions, we see a large drop in bias after imputation. The drop is significant when the original bias is large. In cases such as `KNN` and `SVC`, where the original bias is already low, imputed data may create slightly more bias, but this change is small.
2. In many cases, the greater the amount of imputed data (e.g., 20% vs. 5%), the greater reduction in bias. This pattern is especially clear with classifiers such as `Forest` and `Tree`.
3. It is not clear which data imputation method results in the greatest drop in bias. Generally, `Fill by Similar` often produces the least reduction in bias, while `Fill by Average` and `Multiple Imputation` are similar.
4. Although our focus was not on the classifiers themselves, we note that the `KNN` classifier seems to result in the least bias compared to others, while `LinearSVC` and `LogReg` show high biases in their averaged outputs.

Our results generally support results in [Martínez-Plumed *et al.*, 2019]. However, while the authors in [Martínez-Plumed *et al.*, 2019] only considered fill-by-mean imputation, we demonstrate how multiple imputation and fill-by-similar imputation can also reduce bias, while maintaining high levels of accuracy. We also explored how the bias reduction changes according to the amount of missing data.

5 Conclusion and Future Work

In conclusion, we find that data imputation on random missing values helps to reduce the bias of most classifiers, while maintaining high accuracy. Specifically, we observe that `Fill by Average` and `Multiple Imputation` are the best of three imputation methods. Also, we find that introducing random imputated values to a biased dataset will help decrease the bias in most cases.

The process of this research involves many variables that may affect the final bias, such as the nature of the input data, the pre-processing steps, the generator of the missing entries, the imputation methods, the classifiers, and the bias metrics. In future work, we plan to explore why the various imputation techniques perform differently, analyze imputation with respect to other fairness metrics, theoretically characterize the datasets on which such results may occur, and develop data imputation techniques with the goal of reducing bias.

References

[Angwin *et al.*, 2016] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing, May 2016.

[Chang and Lin, 2011] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines.

ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[Chawla *et al.*, 2002] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, June 2002.

[Cover and Hart, 1967] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. January 1967.

[Fan *et al.*, 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[Flores *et al.*, 2016] Anthony Flores, Kristin Bechtel, and Christopher Lowenkamp. False positives, false negatives, and false analyses: A rejoinder to "machine bias: There's software used across the country to predict future criminals. and it's biased against blacks.". *Federal probation*, 80, 09 2016.

[Kleinbaum *et al.*, 2002] David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. *Logistic regression*. Springer, 2002.

[Kleinberg *et al.*, 2017] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. *Proceedings of the 8th Innovations in Theoretical Computer Science Conference*, 43:1–23, 2017.

[Larson and Angwin, 2016] Jeff Larson and Julia Angwin. Technical response to northpointe. https://www.propublica.org/article/technical-response-to-northpointe, July 2016.

[Larson *et al.*, 2016] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. How we analyzed the compas recidivism algorithm. https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm, May 2016.

[Leeuw, 2001] Edith De Leeuw. Reducing missing data in surveys: An overview of methods. 35:147–160, 2001.

[Liaw *et al.*, 2002] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[Martínez-Plumed *et al.*, 2019] Fernando Martínez-Plumed, Cèsar Ferri, David Nieves, and José Hernández-Orallo. Fairness and missing values. *CoRR*, abs/1905.12728, 2019.

[Mehrabi *et al.*, 2019] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *CoRR*, 2019.

[Safavian and Landgrebe, 1991] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[van Buuren and Groothuis-Oudshoorn, 2011] Stef van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45(3):1–67, 2011.