A Beam-Search Based Method to Select Classification and Imputation Methods for Fair and Accurate Data Analysis

Dodavah Mowoh
Information Systems Department
University of Maryland-Baltimore County
Baltimore, Maryland 21250-0002
Email: dmowoh1@umbc.edu

Zhiyuan Chen
Information Systems Department
University of Maryland-Baltimore County
Baltimore, Maryland 21250-0002
Email: zhchen@umbc.edu

***Abstract*—Members from disadvantaged or minority groups are often more likely to have missing values in their record. Imputation is a common approach to deal with missing values before the data is being analyzed. Several studies have found interplay of imputation methods and classification methods with respect to accuracy and fairness: different combinations of imputation and classification methods will lead to different accuracy and fairness results. However, it is unclear how to choose the combination of imputation method and classification method to optimize the tradeoff between accuracy and fairness. An exhaustive search approach will be too expensive because it needs to check all combinations and measure both accuracy and fairness for every combination. This paper proposes a beam-search based method to select the optimal combination of imputation methods and classification methods. An empirical study was also conducted to compare the performance of the proposed method to exhaustive search. The proposed solution achieves the same result as the exhaustive search method but with much lower search cost.**

1. Introduction

Missing data is a common data quality issue as it can adversely impact the predicted outcomes of machine learning models, leading to predictions that are biased, missing context and untrustworthy. Members from disadvantaged or minority groups are often more likely to have missing values in their record because they are often more reluctant to provide sensitive information, or data acquisition may simply be less complete and systematic for such groups [1]. Imputation is a commonly used method to address this issue before data analysis. Several studies [1], [2], [3], [4] have found that there exists interplay between imputation methods and subsequent machine learning methods such as classification methods which leads to different accuracy and fairness tradeoff. For example, Fernando et al. [1] found that different imputation methods had different impact on classifiers' performance on different data sets.

However, it is unclear how to choose the combination of imputation method and classification method to optimize the tradeoff between accuracy and fairness. An exhaustive search approach will be too expensive because it needs to check all combinations of imputation method and classification method, and measure both accuracy and fairness for every combination.

This paper proposes a beam-search based method to select the optimal combination of imputation methods and classification methods. Beam search can be viewed as a pruned version of breadth-first search where it only stores a fixed number of best options in each level of the search tree. It is more efficient than exhaustive search in terms of both time and memory (only need to keep a fixed number of combinations).

This paper has made the following contributions:

1) We proposed a beam search method to find optimal combination of imputation method and classification method.
2) We conducted an empirical study to compare the performance of the proposed method to exhaustive search. The proposed solution achieves the same result as the exhaustive search method but with much lower search cost.

The rest of the paper is organized as follows. Section 2 summarizes related work. Section 3 presents the proposed method. Section 4 reports the empirical study and results. Section 5 concludes the paper.

2. Related Work

There has been a rich body of work on fairness [5], [6]. Techniques addressing fairness in machine learning can be divided into three categories: 1) pre-processing techniques that modify the data before model training to remove discrimination [7]; 2) in-processing techniques that modify the machine learning process to optimize both for accuracy and fairness [8]; 3) post-processing techniques that modify machine learning output to improve fairness [9]. This paper proposes a method to choose the combination of imputation methods and classification methods to optimize the balance between fairness and accuracy. This method does not modify the existing imputation or machine learning algorithms, so it differs from existing approaches.

The commonly used imputation methods include statistical methods such as mean/mode, regression, Expectation Maximization and machine learning based methods such as decision tree, k-nearest-neighbor (KNN), random forest [10].

Fernando et. al. [1] found that records with missing values often have fairer than the rest, so simply dropping such records may not be appropriate. They also studied the trade-off between accuracy and fairness when the rows with missing values are used through imputation. In general imputation improves accuracy, but its impact on fairness varies. Martinez-Plumed et. al. presented a similar study in [2]. Zhang and Long [3] conducted a theoretic study on assessing fairness using incomplete data. They argue that a fair algorithm on data without missing values may not be fair when data with missing values are included. They provided upper and lower bounds on the fairness estimation error. However, all these work have not specified how to select the combination of imputation and machine learning methods to optimize accuracy-fairness tradeoff.

Wang and Singh [4] proposed pre-processing methods that use reweighing and resampling based upon the missing value generation process to reduce bias introduced by missing values and selection bias. However, this method requires modification of existing imputation methods. The solution proposed in this paper searches for the optimal combination of imputation method and classification method, without requiring changes to existing methods.

3. Methodology

Section 3.1 introduces some background and notations. Section 3.2 describes the proposed beam search approach.

3.1. Background

We first introduce some notations. Given a data set D consisting of n rows and m attributes, where attribute A_m is the target variable (e.g., whether a loan is approved). Each attribute A_i in a record $x \in D$ may have a value v or a NULL value which means the value is missing.

A set of attributes $A_1, \ldots A_P \in AP$ are protective attributes. For each protected attribute $A_i \in AP$, there is a set of protected values VP_i. If a row $x \in D$ has an attribute $A_i \in AP$ with value in VP_i, then x belongs to a protected group. For example, we can have race and gender as protected attributes, and if a record has race equals Black or gender equals female then the record belongs to the protected group.

Imputation Methods: Let $A_i(x)$ be the value of attribute A_i in record x. We assume that there is a set of imputation methods $IM_i \in IM$ where each IM_i is a function that takes D as input and returns a data set D' where there is a one-to-one mapping between each record $x' \in D'$ and a record $x \in D$ such that x' has the same value as x in attribute A_j if x has not null value on A_j, and if A_j's value in x is NULL, the method generates a value based on D, x, and A_j.

$$A_j(x') = \begin{cases} A_j(x), & \text{if } A_j(X) \neq \text{NULL}. \\ IM_i(D, x', A_j), & \text{if } A_j(x) = \text{NULL}. \end{cases} \quad (1)$$

We consider the following imputation methods in this paper.

1) Most frequent value: this method replaces missing values with the most frequent value in the column.
2) K-nearest neighbor imputation: for each record x which has a missing value in column A_j, this method finds k nearest neighbors in D that does not have missing values on A_j, and replaces $A_j(x)$ with mean of A_j in these nearest neighbors.
3) Deleting rows: this method simply drops any row with missing values. Strictly speaking this is not an imputation method but it is commonly used in dealing with missing values.

Problem Definition: We also have a set of classifiers $CL_i \in CL$ where each CL_i takes each imputed record $x' \in D'$ and predicts a target variable value $CL_i(x'), x' \in D$. We also use $CL_i(D')$ to represent the vector of predictions over all $x' \in D'$. We typically consider binary classification and one of the target variable value is considered positive (meaning favorable outcome) and the other value is considered negative (meaning unfavorable outcome).

We also assume that there is a set of accuracy measures $AM_p(D, IM_i, CL_j) \in AM$ and fairness measures $FM_q(D, IM_i, CL_j) \in FM$ such that these measures are computed over $D' = IM_i(D)$ and predicted values $CL_j(D')$.

We use F1-score in this paper which equals $\frac{Precision \times recall}{precision + recall}$. Precision equals $\frac{TP}{TP+FP}$ where TP is number of true positives, FP is number of false positives. Recall equals $\frac{TP}{TP+FN}$ where FN is number of false negatives.

We will use the following fairness measures [5] in this paper:

1) Equal opportunity: protected and unprotected groups have the same true positive rate or recall for the favorable class.
2) Predictive parity: the protected and unprotected groups have the same Positive Predicted Value or precision for the favorable outcome.

Both metrics are based on confusion matrix (true positive, true negative, false positive and false negative) so they can be computed along with the accuracy measures with little additional computation effort.

Since usually we do not have 100% fairness results, these measures usually report the ratio between the protected and unprotected group and a ratio of one means 100% fairness. We also bound the fairness measure by one as if the ratio is over one than there is discrimination against the unprotected group.

We can combine both accuracy and fairness metrics into a single score TM as a weighted sum of all metrics:

$$TM(D, IM_i, CL_j) = \sum_{p=1}^{|AM|} w_p AM_p(D, IM_i, CL_j) + \sum_{q=|AM|+1}^{|AM|+|FM|} w_q FM_q(D, IM_i, CL_j) \quad (2)$$

Definition 1 (Optimize Imputation/Classification). Given a data set D with a set of protected attributes AP, a set of imputation methods IM, a set of classification methods CL, the goal is to find an optimal pair of imputation method and classification method pair (IM^*, CL^*) where $IM^* \in IM$ and $CL^* \in CL$ and $TM(D, IM^*, CL^*) \leq TM(D, IM_i, CL_j)$ for any $IM_i \in IM$ and $CL_j \in CL$.

3.2. Beam Search

To evaluate each combination of imputation method and classification method, one has to impute the data set (including both training and testing data set), train the classification model on the training data set, and then test it on the testing data set and compute accuracy and fairness metrics. This is quite time consuming and computationally expensive. So we want a more efficient search strategy than exhaustive search.

Hyperparameter tuning has been well studied in machine learning and deep learning [11]. The commonly used methods include 1) greedy search, which just keeps the local optimal setting for each parameter; 2) random search, which checks a number of randomly selected parameter settings; 3) Bayesian optimization, which determines the next hyper-parameter value based on the previous results of tested hyper-parameter values.

However, random search and Bayesian optimization are typically more suitable for very large search spaces where there are hundreds of possible parameters or parameter settings. In our setting we have a relatively small number of imputation methods and classification methods. So we use a variant of greedy search called beam search.

Algorithm 1 shows the detail steps of the proposed beam search method. The algorithm first splits the data set D into training D_{train} and testing D_{test} (line 1). It then randomly selects an imputation method (line 2). Without loss of generality, let it be IM_1. A set k-best-1 is used to keep track of the best k combinations at phase 1 of the search process and is initialized to an empty set (line 3). In the first phase of the search, the algorithm first freezes the imputation method (IM_1) but varies classification methods because accuracy metrics are often more important in practice. For each such combination (IM_1, CL_j) (line 4 to 11), the algorithm calls a function Evaluate (line 5), which imputes both D_{train} and D_{test}, trains a model M using CL_j, and applies the model on testing set, and computes both accuracy and fairness

Algorithm 1 Beam Search(D, IM, CL, k)

1: Randomly split the data set D into D_{train} and D_{test}
2: Randomly select an imputation method IM_1
3: Initialize k-best-1 combination to $\emptyset$
4: **for** each classification method $CL_j \in CL$ **do**
5: TM(D, IM_1, CL_j)=Evaluate($IM_1, CL_j, D_{train}, D_{test}$)
6: **if** k-best-1 has fewer than k combinations **then**
7: Add (IM_1, CL_j) to k-best-1
8: **else if** TM(D, IM_1, CL_j) is higher than TM of any combination in $x \in k$-best-1 **then**
9: replace x with (IM_1, CL_j)
10: **end if**
11: **end for**
12: Initialize k-best-2 to $\emptyset$
13: **for** each combination $IM_1, CL_b \in k$-best-1 **do**
14: **for** each imputation method $IM_i \in IM$ and $IM_i \neq IM_1$ **do**
15: TM(D, IM_i, CL_b)=Evaluate($IM_i, CL_b, D_{train}, D_{test}$)
16: **if** k-best-2 has fewer than k combinations **then**
17: Add (IM_i, CL_b) to k-best-2
18: **else if** TM(D, IM_i, CL_b) is higher than TM of any combination in $x \in k$-best-2 **then**
19: Replace x with (IM_i, CL_b)
20: **end if**
21: **end for**
22: **end for**
23: **return** the best combination in k-best-1 $\cup$ k-best-2

1: **function** EVALUATE($IM_i, CL_j, D_{train}, D_{test}$)
2: Impute D_{train} and D_{test} using IM_i
3: Train a model M using CL_j over imputed D_{train}
4: Apply M to predict target variable on D_{test}
5: Compute accuracy metrics and fairness metrics using results of prediction
6: Compute TM using Equation 2
7: **return** TM
8: **end function**

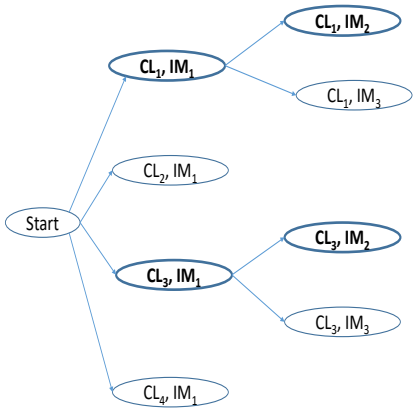

Figure 1. A beam search example.

metrics based on prediction results. The combined metric score TM is returned. At line 6, the algorithm checks whether the k-best-1 set has fewer than k combinations. If so, it directly adds the combination (IM_1, CL_j) to k-best-1 because this combination is among the best k so far. Otherwise, if the combined score of (IM_1, CL_j) is higher than any combination x in k-best-1, this means (IM_1, CL_j) is in top k. So the algorithm replaces x with (IM_1, CL_j) in k-best-1 (line 9). In this way, the k combinations with highest TM are preserved.

Figure 1 shows an example of running Algorithm 1. Let $k = 2$, in Phase 1 the algorithm checks four combinations and preserve the two combinations (CL_1, IM_1) and (CL_3, IM_1) because they have the best combined scores.

In Phase 2 of beam search, the algorithm first initializes a set k-best-2 to empty set, which will be used to store the top k combinations in this phase. For each of the k preserved combinations in k-best-1 (line 13), the algorithm freezes the classification method but varies the imputation methods for each imputation method that is in IM but is not the imputation method IM_1 which is selected in Phase 1 (line 14). Again, for each considered combination (CL_i, IM_j), the method calls Evaluate function (line 15) to compute the combined score TM. If k-best-2 has fewer than k combinations, (CL_i, IM_j) will be added to the set (line 16 and 17). Otherwise, if (CL_i, IM_j) has higher combined score than an existing combination x in k-best-2, x will be replaced with (CL_i, IM_j) (line 18 and 19). At the end k-best-2 will have the best k combinations examined in Phase 2. The algorithm then returns the combination with the best combined score in k-best-1 union k-best-2 (line 23).

In Figure 1, from (CL_1, IM_1) the method will consider (CL_1, IM_2) and (CL_1, IM_3) because there are three imputation methods and IM_1 is already used in the preserved pair. The algorithm examines four more combinations in Phase 2 and (CL_3, IM_2) and (CL_1, IM_2) are the best combinations at Phase 2. The best combination in k-best-1 and k-best-2 is (CL_1, IM_1).

Beam search is more efficient than exhaustive search because it only preserves k best combinations at each step. A simple greedy search that keeps the best option for each parameter can be seen as a variant of beam search where $k = 1$. In general, higher the value of k, the more combinations beam search will check, and more likely it will find the best combination. On the other hand, the search cost also increases. In our setting we find that $k = 2$ leads to a good balance between quality of results and search cost.

4. Experiments

Section 4.1 describes setup of the experiments. Section 4.2 presents the results.

4.1. Setup

Data sets: we used three data sets: Adult [12], Titanic [13], and Recidivism [14]. These are all popular benchmark data sets used in fairness research.

Data set	Attribute	# NULLs (percentage)
Adult	Workclass	1836 (5.6%)
	occupation	1843 (5.7%)
	native country	583 (1.8%)
Titanic	age	263 (20%)
	fare	1 (0.0008%)
	cabin	1014 (77%)
Recidivism	Gang_Affiliated	2215 (8.6%)
	Supervision_Risk_Score_First	319 (1.2%)
	Percent_Days_Employed	313 (1.2%)
	Jobs_Per_Year	551 (2.1%)
	Supervision_Level_First	1193 (4.6%)
	Prison_Offense	2282 (8.8%)

TABLE 1. NUMBER OF MISSING VALUES IN EACH DATA SET

The adult census dataset consists of information about adults in the United States. It contains attributes such as age, marital status, gender, native-country and work class etc. The protected attributes are gender and race. The target variable is whether someone's income is over 50k. The dataset consisted of 32561 rows.

The titanic data set consists of records of Titanic passengers. The protected attribute is class of passenger and gender. The target variable is whether the passenger survived. Unlike other data sets, the female survival rate is significantly higher than that of male passengers. So we treat male as the protected group rather than female. Before imputation the entire dataset consisted of 1309 rows.

The recidivism data set consists of records of inmates. The target variable is whether the inmate commits recidivism. The protected attributes are race and educational level. This recidivism dataset consisted of 54 columns, however only 14 fields were useful for model training. Before imputation the entire dataset consists of 25835 rows.

Table 1 shows the columns with missing values in each data set and the number of missing values. It can be seen that Titanic has significant portion of rows with missing values while Adult and Recidivism have relatively lower portion of missing values.

Pre-processing: For each data set, we first convert all categorical attributes into numerical ones using one hot encoding because some of the classifiers such as K-nearest neighbor requires numerical input. One hot encoding will represent a category attribute with t distinct values as t binary attributes and the attribute representing its current value will be set to one and other attributes will be set to zero.

Imputation and classification methods: we considered three imputation methods: deleting rows with missing values, KNN imputation, and most frequent value. We considered the following classification methods: logistic regression, K-nearest-neighbor (KNN), decision tree, and support vector machine (SVM). We do not use deep learning methods here because these data sets are relatively small and deep learning methods work better for large data sets.

Search methods: we considered two search methods: 1) beam search, which is the proposed method; 2) exhaustive search, which checks all combinations of imputation meth-

ods and classification methods. As mentioned in Section 3, we use F1-score to measure classification performance and use equal opportunity and predictive parity to measure fairness.

In each data set, beam search will first pick an imputation method and then vary the classification methods in the first step. We set beam width $k = 2$. For weights in Equation 2, we set the weight for accuracy and fairness at the same importance, so the weight for F1-score is 0.5. We have two fairness metrics (Equal Opportunity and Predictive Parity) and two protected attributes in each data set. So we have four fairness measures (for each fairness metric and protected group combination), and the weight is 0.125 for each combination.

4.2. Results and Discussion

Search cost: Since search each combination takes about the same time, we measure the search cost with the number of combinations of imputation method and classification method. Since there are four classification methods, beam search will consider four combinations in Phase 1. We also assume that it selects deleting rows with missing values as the default imputation method at Phase 1. Since beam width $k = 2$, after Phase 2 two combinations with the highest combined score are stored. At Phase 2, beam search freezes the classification method of a stored combination, and varies the imputation methods. Since there are two unchecked imputation methods for each stored combination, Phase 2 will check four more combinations. So in total beam search will check eight combinations while exhaustive search will check all twelve combinations (3 by 4 as there are 3 imputation methods and 4 classification methods).

Results after Phase 1: Next we show the combinations at Phase 1, the F1 score, and fairness scores for each data set. Table 2 shows the results. DEL or Delete stands for deleting rows with missing values, F stands for most frequent value imputation, and KI stands for k-nearest-neighbor imputation. For classification methods, LR stands for logistic regression, KNN stands for k-nearest-neighbor, DT stands for decision tree classification, and SVM stands for support vector machine. EO stands for equality opportunity fairness measure, and PP stands for predictive parity. Since we compute a ratio between a protected group to an unprotected group, we also specify the protected attribute value for each group in our results. E.g., Black/White means it is the ratio between Black and White (race).

Figure 2 shows the combinations considered in each phase of beam search on Adult. Figure 3 and Figure 4 shows the combinations considered in each phase of beam search on Titanic and Recidivism, respectively. The combinations selected in each phase are boldfaced.

The results show the importance of selecting the best combination of imputation method and classification method. For example, the best combination for Titanic is (Delete, Decision Tree). It has a F1-score of 0.9, and fairness

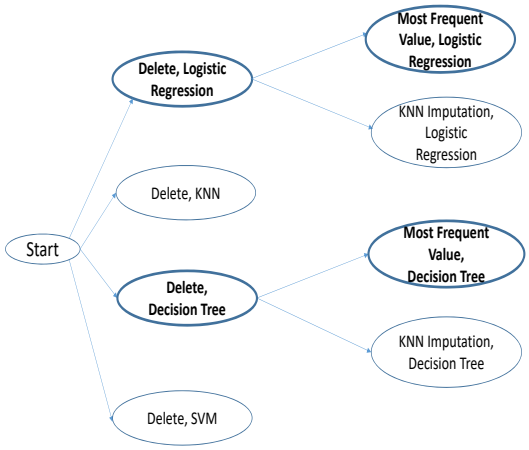

Figure 2. Beam Search on Adult Dataset

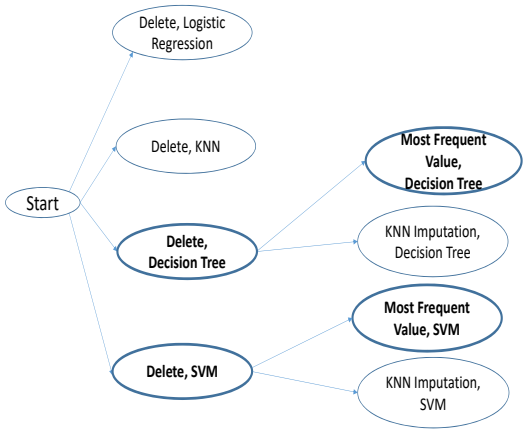

Figure 3. Beam Search Titanic Dataset

metrics for both gender and class of passengers are over 0.8. On the other hand, using the combination (Delete, Logistic regression) only has a F1-score of 0.78, and its fairness results are around 0.5-0.6 in three out of four measures.

The best two combinations in Phase 1 for Adult are (Delete, Logistic Regression) and (Delete, Decision Tree). Both combinations have the highest F1-score, and very high fairness scores ranging from 0.87 to 1.0. (Delete, Logistic regression) has slightly lower F1 score than (Delete, Decision Tree), but higher fairness scores and its combined score is also higher than that of the latter. The (Delete, KNN) combination has lower F1-score and Predictive Parity for Black group. The (Delete, SVM) combination has lower Predictive Parity score for female group.

The best two combinations in Phase 1 for Titanic are (Delete, Decision Tree) and (Delete, SVM). Both combinations have high F1-scores and fairness values. The difference

Data set	Combination	F1	Protected values	EO	PP	TM
Adult	**DEL, LR**	0.8	Female/Male Black/White	1.0 1.0	1.0 1.0	**0.9**
Adult	*DEL, DT*	0.81	Female/Male Black/White	1.0 0.95	1.0 0.87	0.88
Adult	*DEL, KNN*	0.78	Female/Male Black/White	1.0 0.97	1.0 0.71	0.85
Adult	*DEL, SVM*	0.80	Female/Male Black/White	1.0 1.0	0.82 0.98	0.87
Titanic	*DEL, LR*	0.78	Male/Female Third class/First class	0.54 0.94	0.62 0.59	0.72
Titanic	**DEL, DT**	0.9	Male/Female Third class/First class	0.85 1.0	0.87 1.0	**0.92**
Titanic	*DEL, KNN*	0.73	Male/Female Third class/First class	0.98 1.0	0.60 0.71	0.78
Titanic	*DEL, SVM*	0.89	Male/Female Third class/First class	1.0 1.0	0.75 0.96	0.91
Recidivism	**DEL, LR**	0.71	Black/White Less than High School/Some college	0.97 0.91	0.96 0.93	**0.83**
Recidivism	*DEL, DT*	0.69	Black/White Less than High School/Some college	0.97 0.92	0.96 0.88	0.81
Recidivism	*DEL, KNN*	0.60	Black/White Less than High School/Some college	0.98 0.96	0.94 0.88	0.77
Recidivism	*DEL, SVM*	0.57	Black/White Less than High School/Some college	0.99 0.87	0.69 0.99	0.73

TABLE 2. RESULTS FOR COMBINATIONS IN PHASE 1

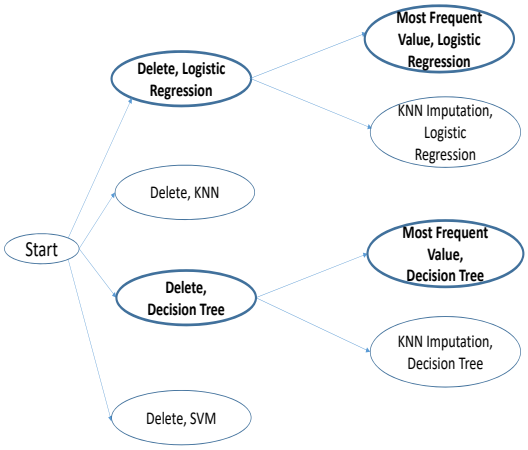

Figure 4. Beam Search on Recidivism Dataset

between these two combinations is quite small. The other two combinations have significantly lower F1 scores.

For the Recidivism data set, the best combinations in Phase 1 are (Delete, Logistic regression) and (Delete, Decision Tree). Both combinations have high F1-scores and fairness scores.

Results after Phase 2: Table 3 reports the two best combinations for each data set after Phase 2. We varied imputation methods for each combination selected in Phase 1. Please note that all the best two combinations in phase 2 happen to use most frequent value imputation.

In Phase 2, the best two combinations for Adult are (Most frequent value, Decision Tree) and (Most Frequent Value, Logistic Regression). But both combinations are no better than the two best combinations found in Phase 1. The best combination in both phases is (Delete, Logistic Regression) from Phase 1 with a combined score of 0.9.

In Phase 2, the best two combinations for Titanic are (Most frequent value, Decision Tree) and (Most frequent value, SVM). However, both combinations are no better than the two combinations stored in Phase 1. The best combination in both phases is (Delete, Decision Tree) with a combined score (TM) of 0.92.

For Recidivism, the best two combinations in Phase 2 are (Most frequent value, DT) and (Most frequent value, Logistic regression). The best combination overall is (Delete, Logistic regression) returned in Phase 1 with a combined score of 0.83.

We also observed that in our experiments the beam search always returns the same combination as the exhaustive search. So it does not compromise quality of results.

Discussion: The experimental results showed that different combinations of imputation methods and classification methods have different accuracy and fairness performance. So there is a need to find the optimal combination. The results also showed that the proposed beam search approach is quite effective. It finds the best combination but is more efficient than exhaustive search.

We also find that deleting rows with missing values is the preferred imputation method in all three data sets. A possible reason is that existing imputation methods may still introduce bias because they are either statistics based or machine learning based, and often rely more on records from unprotected groups because unprotected groups are often larger. Deleting rows with missing values will not introduce

Data set	Combination	F1	Protected values	EO	PP	TM
Adult	F, DT	0.82	Female/Male	1.0	0.82	0.87
			Black/White	0.89	1.0	
Adult	F, LR	0.81	Female/Male	1.0	0.93	0.88
			Black/White	0.88	1.0	
Titanic	F, DT	0.91	Male/Female	0.84	0.86	0.91
			Third class/First class	0.88	1.0	
Titanic	F, SVM	0.76	Male/Female	0.95	1.0	0.87
			Third class/First class	1.0	1.0	
Recidivism	F, DT	0.69	Black/White	0.98	0.96	0.81
			Less than High School/Some college	0.89	0.91	
Recidivism	F, LR	0.7	Black/White	0.96	0.97	0.82
			Less than High School/Some college	0.90	0.94	

TABLE 3. RESULTS FOR BEST TWO COMBINATIONS IN PHASE 2

Classification Method	Selected in Phase 1	Selected in Phase 2	Overall
Decision Tree	3	3	1
Logistic Regression	2	2	2
SVM	1	1	0
KNN	0	0	0

TABLE 4. NUMBER OF TIMES A CLASSIFIER IS SELECTED IN TOP k IN EACH PHASE OF BEAM SEARCH AND AS FINAL BEST COMBINATION.

more bias.

Table 4 shows the number of times a classifier is selected as top k in each phase of beam search and as the final best combination.

The results showed that decision tree appears most often in top k combinations of each search phase. It is in the top 2 list in both phases and in all three data sets. One possible reason for the good performance of decision tree is that it is a very flexible model and does not make too much assumptions about data distribution. It has high F1-scores in all three data sets and also high fairness scores.

The other classification methods' performance seems to be more data dependent. Logistic regression works well in Adult and Recidivism data sets, and is selected in the best combination overall in both data sets. But it performs poorly in Titanic data set (it has the lowest combined score in Phase 1). One possible reason is logistic regression assumes that there is a linear relationship between independent variables and log-odds of the target variable. This assumption may hold in Adult and Recidivism but not in Titanic.

SVM has high F1 scores in Adult and Titanic and is selected as top k in both phases on Titanic data set, but has low F1 scores in Recidivism data set. KNN only has high F1 score in Adult data set, and has low Predictive Parity scores for Black. So it is never selected as top k.

As for the overall best combination, Logistic Regression is selected twice (in Adult and Recidivism data sets) and Decision Tree is selected once (in Titanic data set). However, the performance of Decision Tree seems to be more stable as the combination with Decision Tree just has a slightly worse combined score than the best combination in Adult and Recidivism. The combination with Logistic Regression on the other hand has much lower combined score in Titanic compared to the best combination.

5. Conclusion

Missing values are often ignored in research on fairness in machine learning. This paper proposes a beam search method to find the optimal combination of imputation methods and classification methods in both accuracy and fairness, without the need to modify the imputation or classification methods. We conducted an empirical study on three data sets and the results showed that the proposed method always found the optimal combination but was more efficient than exhaustive search.

We also found that simply deleting rows with missing values often lead to better accuracy fairness tradeoff, possibly because imputation methods often introduce new bias to the data. Decision tree works well in all cases in our study, possibly due to its flexibility. The other classifiers' performance vary more with the data sets. We plan to conduct experiments on more data sets in future.

References

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

[2] F. Martínez-Plumed, C. Ferri, D. Nieves, and J. Hernández-Orallo, "Fairness and missing values," *arXiv preprint arXiv:1905.12728*, 2019.

[3] Y. Zhang and Q. Long, "Assessing fairness in the presence of missing data," *Advances in neural information processing systems*, vol. 34, pp. 16 007–16 019, 2021.

[4] Y. Wang and L. Singh, "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, 2021.

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

[6] S. Caton and C. Haas, "Fairness in machine learning: A survey," *ACM Computing Surveys*, vol. 56, no. 7, pp. 1–38, 2024.

[7] Z. Zhang, S. Wang, and G. Meng, "A review on pre-processing methods for fairness in machine learning," in *The International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*. Springer, 2022, pp. 1185–1191.

[8] M. Wan, D. Zha, N. Liu, and N. Zou, "In-processing modeling techniques for machine learning fairness: A survey," *ACM Transactions on Knowledge Discovery from Data*, vol. 17, no. 3, pp. 1–27, 2023.

[9] M. Hort, Z. Chen, J. M. Zhang, M. Harman, and F. Sarro, "Bias mitigation for machine learning classifiers: A comprehensive survey," *ACM Journal on Responsible Computing*, vol. 1, no. 2, pp. 1–52, 2024.

[10] W.-C. Lin and C.-F. Tsai, "Missing value imputation: a review and analysis of the literature (2006–2017)," *Artificial Intelligence Review*, vol. 53, pp. 1487–1509, 2020.

[11] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020.

[12] B. Becker and R. Kohavi, "Adult," UCI Machine Learning Repository, 1996, DOI: https://doi.org/10.24432/C5XW20.

[13] T. C. Frank E. Harrell Jr., "Titanic dataset," oct 2017. [Online]. Available: https://www.openml.org/d/40945

[14] J. Larson, S. Mattu, L. Kirchner, and J. Angwin, "How we analyzed the compas recidivism algorithm," *ProPublica (5 2016)*, vol. 9, no. 1, pp. 3–3, 2016.