Team 11: Fairness-aware AutoML

Zihao Guo, Yichao Shen, Nimi Wang, Haochen Wang
New York University
{zg866,ys3197,nw1212,hw1985}@nyu.edu

ABSTRACT

Machine learning algorithms are increasingly used by companies and governments to make decisions that have significant impacts on people's lives. Often, these predictions can affect different subgroups disproportionately. Therefore, the issue of fairness arose and caught great attention. Also, due to the success of ML in a wide range of applications and the burdens faced by non-experts to deploy ML models, the demand for automated machine learning systems that can be used by non-experts has soared over the past few years. This paper seeks to introduce a Fairness-aware AutoML system that incorporates debiasing algorithms into the whole automated machine learning framework to mitigate the issue of fairness without interventions from machine learning practitioners. We present the results of the system tested on 2 datasets to illustrate the effectiveness of the debiasing algorithms used by the system without much sacrifice of accuracy.

1 INTRODUCTION

As the scope of machine learning practice has rapidly widened in the past few years, our lives have been fundamentally changed, yet invisibly. Bias and controversies generated from various algorithms have been almost ubiquitous when the algorithm makes a decision using sensitive features, including race, sex, sexual orientation, etc. Therefore, it is critical to address this issue of fairness. Meanwhile, AutoML system has grown popularity because it requires a minimum amount of human efforts in terms of data engineering and algorithm choice. Therefore, the goal in this paper is to incorporate fairness into AutoML system so that we can get the best of both worlds. By using our system, both machine learning practitioners and non-experts can build fair machine learning models based on their demands easily.

To satisfy the demand for automated machine learning systems with fair predictions, we build a Fairness-aware AutoML system which consists of two parts: bias mitigation algorithms and automated machine learning framework. For the bias mitigation algorithms, the system chooses processing algorithms derived from the AIF360 toolkit which contains over 70 fairness metrics and 10 state-of-the-art bias mitigation algorithms. For the automated machine learning part, the system adopts the framework of Auto-Sklearn which frees machine learning practitioners from algorithm selection and hyperparameter tuning. By integrating bias mitigation algorithms, the AutoML system helps machine learning users develop their fair models without any extra work.

One of the biggest difficulty of the project is to find the balance of fairness-accuracy trade-off, namely maintaining a high accuracy while adhering to a fairness constraint. During the process of finding an optional solution, hyper-parameters of each bias mitigation algorithm can have a significant impact on fairness metrics value. In the mean time, different probability thresholds for classification have great influence on accuracy. Thus, instead of doing exhaustive grid search on both fairness metrics and accuracy, we want to maximize classifier metric given that the fairness metrics are in a "safe" range; a.k.a, the results are relatively fair.

To address the difficulty mentioned above, the system will maximize the accuracy of the prediction under the fairness-constraint. To be more specific, we can use the grid search to find the optimal threshold that maximizes the accuracy for classifiers that are valid under fairness-constraints.

There are three major achievements of our project.

- By incorporating bias mitigation algorithms, the sate-of-the-art autoML system improves the fairness of the classifier significantly.
- The simple APIs are very hands-on for users of all levels.
- The rationale of the whole system is very clear. The structure of the system is flexible to extend for future improvements.

2 PROBLEM STATEMENT & APPROACH

2.1 Problem Statement

The project main objective is to implement a fairness-aware auto machine learning solution for binary classification tasks. The system constructed should incorporate both the "fairness" element and the Auto-ML optimization element.

There are several goals the we try to accomplish through our fairness-aware Auto-ML system. Firstly, the system should be efficient, money-saving and automatic. It saves the users from repetitively and manually doing feature extraction and feature selection in classical machine learning settings. The second goal is that the system API should be clean and easy to use for users of all levels of machine learning proficiency. Lastly, there should be a strong logic behind the framework setup. The system should contain several discrimination tests which are used to determine if the data set fed into the system or the classifier trained has bias or not. If they are indeed

biased, the system should choose debiasing (fairness) algorithms to handle the bias while optimizing machine learning models to achieve high accuracy.

2.2 Approach

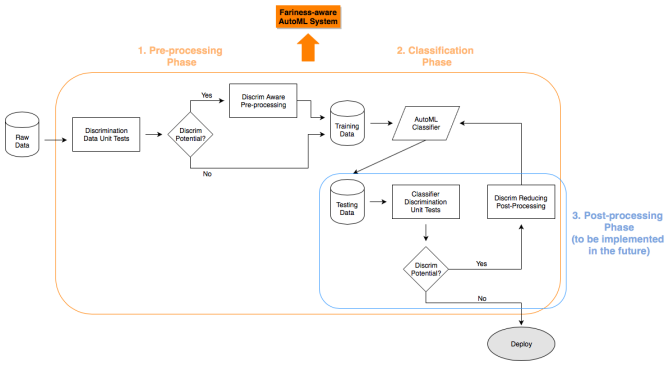

Figure 1: System Framework

Figure 1 shows the setup of the system which is consisted of three possible phases: the pre-processing phase, classification phase, and a post-processing phase (which will be implemented in the future). A raw data set with binary target values will be fed into the system by the user. After that, the data will be passed through an initial discrimination data unit test. The test indicates if the data set is biased towards privileged classes according to the data set metric and threshold of the data set metric by the user. For an unbiased data set, it will go directly to the classification phase where the system executes auto machine learning to fit the data set and learns a probability threshold that achieves the highest accuracy while the statistical parity difference (mean difference) of the classifier is between -0.1 to 0.1. However, for a biased data set, before classification phase, it will go through an additional fairness pre-processing step to debias the data. After training the classifier, we again arrive at another discrimination test checkpoint but designed for the classifier. It determined if the classifier will go through a post-processing phase. Until the fairness criteria is met, the trained model can be pushed into deployment. Due to the scope of the project, we have only implemented the pre-processing phase and classification phase. However, with the simplicity of the framework, it is very easy to add a post-processing phase according to instructions. In the next section, we will discuss more about available fairness processing algorithms, data set metrics, classifier metrics and how the system realizes the automation in details.

3 IMPLEMENTATION

Our Auto-ML system consists of four parts: initiate, fit, evaluate and predict. For the stage of initiating, the user should assigns various parameters for the system. Variables such as input/output columns, favorable classes, protected attributes, privileged/unprivileged groups have to be specified before any implements. The biased group is considered unprivileged class (group) and the protected attributes are the features which distinguishes the privileged group and unprivileged group. There are also some optional parameters can be specified, such as features to keep and features to drop. The users can freely choose the group of features to be considered before applying to the pipeline. Our Auto-ML system allows that the input dataset is not necessarily in the standard format needed for the training and testing, but the user should specify whether the dataset they feed into the system is valid or not through parameter is_valid. For the part

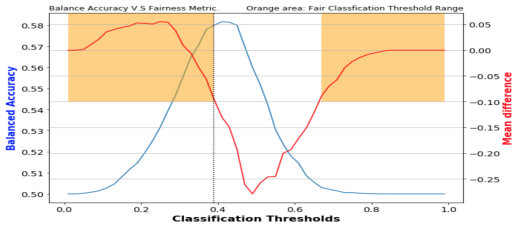

Figure 2: Mean Diff v.s Bal Acc on Law School Data

of fitting, the system checks the validity of input dataset, transforms the dataset into the standard format if necessary, and divide it into training and validation sets. Then, the system performs a pre-check on the dataset fairness using the dataset_metric and dataset_metric_threshold specified by users. There are several fairness metrics users can select, including mean difference (statistical parity difference) showing the difference in mean outcomes between unprivileged and privileged groups, disparate impact (the probability of favorable outcome for unprivileged instances / probability of favorable outcome for privileged instances), etc. If the fairness meets the criterion users set, the system will just proceed to perform fitting using standard Autosklearn. If not, a considerable difference between unprivileged and privileged groups exists. Then our system will grid search for hyper-parameters for each preprocessing method and figure out the best preprocessing configuration. We have 3 algorithms, namely optimized algorithm, disparate impact remover (DIR), and learning fair representations (LFR). For each iteration of grid search over all hyper-parameters for one preprocessing method, we used that preprocessing configuration to transform dataset, and then used autosklearn to fit the transformed data and predict the probability for validation set. Then, we looped through all classification thresholds to make decisions on the labels. Finally, we chose the threshold with highest balanced accuracy on validation set given that the fairness metric criterion we set in our

system is met. It's easier to understand this by looking at Figure 2. We need the mean difference in range [-0.1, 0.1] and orange area indicates the classification thresholds we can pick. The vertical dotted line shows the final threshold with highest balanced accuracy we select for this preprocessing configuration. Whenever a new preprocessing configuration yields a higher balanced accuracy, our system updated the preprocessing algorithm and classifier.

The final part is just to apply the trained model to predict the label of test dataset or evaluate the performance (both fairness and accuracy) of the model on test set. Users can use class method "plot" to generate the figure of Balanced accuracy and classifier metric against thresholds, like Figure 2 we mentioned above.

4 EVALUATION

4.1 Experimental Setup

Our AutoML system was built using Python 3 on macOS. We used jupyter notebook as the API to our system, which requires several libraries, including: aif360=0.2.2, auto-sklearn=0.5.2.

4.2 Datasets

We used 2 datasets to demonstrate our AutoML system. The first one is law school admission data and the second one is Adult Census Income data. For law school application data, we have sensitive features: sex, race, lgbt. However, race contains mostly missing values, and lgbt contains questionable values because it's a binary numerical feature (0, 1) with no missing values (Normally this kind of information has many missing values, so we cast doubt on this feature). Thus, we only used sex without many missing values as the protected feature. For AdultDataset, the protected feature is race.

4.3 Results

4.3.1 Law School Admission Data. Before running into analysis, we assumed male would be the privileged group and female be the unprivileged group. However, a mean difference = 0.06 and disparity impact = 1.13 shows that our data is actually biased against male applicants, so we switched the order of two groups. Other features of interest are GPA, LSAT, LSAT1, and school.

The new mean difference = -0.06 and disparate impact = 0.88, showing that our data is actually pretty fair, only slightly biased against male applicants. However, in order to figure out our system capability on relatively fair dataset, we implemented our system with a stricter fairness threshold so that our system will still use preprocessing methods.

First we set the mean difference = 0.05 as our dataset fairness metric checker. For classifier metric mean difference, our system uses disparate impact remover (DIR) at level = 0.1 as preprocessing method and the best classification threshold

Law School Admission Data		MeanDiff	Disparate Impact	Equal Oppor Diff	Theil Index
Mean Diff = 0.09	Preproc	DIR w/ repair=0.1	DIR w/ repair=0.1	DIR w/ repair=0.4	DIR w/ repair=0.1
	Best Threshold	0.39	0.35	0.35	0.39
	Bal Acc	0.58	0.59	0.57	0.57
Disparate Impact = 0.9	Preproc	DIR w/ repair=0.2	DIR w/ repair=0.1	DIR w/ repair=0.4	DIR w/ repair=0.1
	Best Threshold	0.33	0.37	0.37	0.39
	Bal Acc	0.57	0.58	0.58	0.59
Consistency = 0.5	Preproc	DIR w/repair=0.1	DIR w/repair=0.3	DIR w/repair=0.2	DIR w/repair=0.1
	Best Threshold	0.33	0.37	0.35	0.35
	Bal Acc	0.57	0.58	0.57	0.57

Table 1: Adult: Results on Law School Data

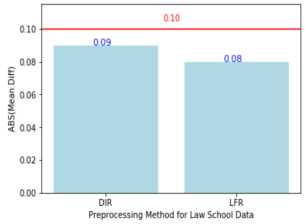

Figure 3: Mean Difference on Law

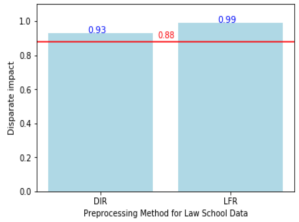

Figure 4: Disparate Impact on Law

is 0.39 with balanced accuracy on test set = 0.58. Results for other classifier metrics are interpreted the same way in Table 1, where other dataset fairness metric checkers, disparate impact and consistency, are also listed. The balanced accuracy is not ideal across all settings, showing that our features are not predictive. However, we're able to see that DIR shows up in all 3 dataset fairness metric checkers. The reason can be either DIR is a versatile preprocessing method from its nature or we set optimal range for Disparate Impact = [0.8, 1.2] a bit wide. Changing it into [0.9, 1.1] could be more ideal for this dataset, but we set it for the general purpose.

Then, we compared the classifier metrics (we used mean difference and disparate impact) for the test data after fitting raw train via autosklearn, and after fitting on transformed train via autosklearn. The results are shown in figure 3, and figure 4. The mean difference drops by 10% and 20%, and disparate impact increases by 6% and 13% for DIR and LFR, respectively. It's not a big amount due to the fact that this dataset is pretty fair. Also, LFR is more optimal in both scenarios, because it essentially fits a model on the dataset which can be more effective.

4.3.2 Adult Dataset. The popular AdultDataset has been known to be biased on race. We are able to use optimized algorithm for this dataset since the distortion function is given. Running similar analysis as the Law School data, we first get mean difference = -0.13, and disparate impact = 0.6, showing that this data is unfair and we proceeded with our system. The results are in Table 2, we can see that disparate impact remover at level = 0.4 is deployed as preprocessing methods

Adult Data		MeanDiff	Disparate Impact	Equal Oppor Diff	Theil Index
Mean Diff = 0.1	Preproc	DIR w/ repair=0.6	DIR w/ repair=0.6	DIR w/ repair=0.4	DIR w/ repair=0.4
	Best Threshold	0.35	0.37	0.35	0.35
	Bal Acc	0.73	0.72	0.71	0.74
Disparate Impact = 0.8	Preproc	DIR w/ repair=0.4	DIR w/ repair=0.4	DIR w/ repair=0.4	DIR w/ repair=0.8
	Best Threshold	0.23	0.19	0.27	0.27
	Bal Acc	0.72	0.72	0.71	0.74
Consistency = 0.5	Preproc	DIR w/repair=0.8	Optimized	DIR w/repair=0.4	DIR w/repair=0.4
	Best Threshold	0.41	0.45	0.43	0.31
	Bal Acc	0.73	0.7	0.71	0.76

Table 2: Adult: Results on Adult Data

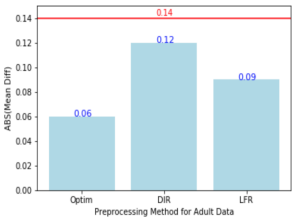

Figure 5: Mean Difference on Adult

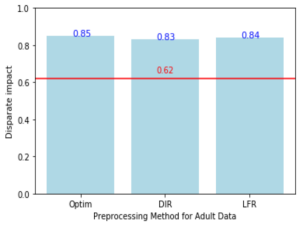

Figure 6: Disparate Impact on Adult

in most settings, showing the strong power of it and again the potential wide optimal range we set for it. The balanced accuracy is around 0.73. Then, looking at (consistency, Theil index) pair, we can see that a balanced accuracy = 0.76 is achieved via DIR with repair level = 0.8. Since consistency and Theil index are both individual fairness metrics, we can see that this high accuracy comes from shaping individual similarity instead of group fairness.

Again, we compared the classifier metrics for the test data after fitting raw train via Autosklearn, and after fitting on transformed train via Autosklearn. The results are shown in figure 5, and figure 6. The mean difference drops by 16%, 56%, and 133%, and disparate impact increases by 34%, 35%, and 35% for Optimized, DIR, and LFR, respectively. Obviously, Optimized algorithm changes fairness the most in both scenarios, but it will inevitably hurt accuracy. That's why our system doesn't pick it as the preprocessing method often. The reason for LFR less piked is the same. Disparate impact remover, due to our wide optimal range and adequate adjustment power, shows its power in elevating accuracy while maintaining relatively fair results.

5 DISCUSSION

To summarize, our Auto-ML system allows users to train a dataset with attention on both fairness and accuracy. By specifying a couple of parameters, our users can use the system to automatically transform, fit and predict the final labels. The final result of the system is a combination of a graph and summary of all statistics such as accuracy and classifier metrics. It is fairly easy for people not familiar with ML to understand and use our system. However, there are also some restrictions for our system. For example, reweighing preprocessing doesn't work in our system because reweighing changes the instance weight but autosklearn doesn't take instance weight into its account. Also, since autosklearn is so packed and uniform, it's hard to incorporate in-processing into it. Finally, the post-processing method requires more researches and will be implemented in the future.

6 DETAILED CONTRIBUTIONS

6.1 Zihao Guo

He implemented the fit and plot methods, and restructured evaluate and predict methods for the Fair AutoML system. He integrated all pieces and finalized the system by debugging, optimizing and implementing Grid Search over hyperparameters for various preprocessing methods. He applied AutoML to both datasets to demonstrate the effectiveness and efficiency of the system. For the paper, he completed evaluation part and revised implementation part.

6.2 Yichao Shen

He wrote and debugged the fit method together with Zihao Guo. Moreover, he finished the code for plotting and testing the correctness of the final output. He also provides a version of Fairness-aware AutoML system using sklearn package (logistic regression) for users not accessible to autosklearn. In terms of project paper writing, he completed the third and fifth section, implement and discussion.

6.3 Nimi Wang

She set up the code base (the entire FairAutoML class) and implemented the flowchart and framework of the system. She also debugged and tested for the fit method of the Fair AutoML class on both datasets with Zihao Guo. She also designed the slides for the final presentation and gave the final presentation. She completed problem statement and approach sections for the paper, and proofread before submission.

6.4 Haochen Wang

He worked with the team to design the Fair AutoML framework, in terms of the structure and APIs. Specifically, he implemented, debugged, and tested the predict and evaluate methods of the Fair AutoML class. Also, he wrote the README with detailed instructions. Lastly, he created the Fair AutoML package and published the package through PyPI. He completed the abstract and the introduction sections for the paper.