Supplementary Material - Estimating the Risk of Individual Discrimination of Classifiers

Jonathan Vasquez[1,2], Xavier Gitiaux[1], and Huzefa Rangwala[1]

[1] George Mason University, Fairfax, VA 22030, USA
[2] Universidad de Valparaiso, Valparaiso, Chile
{jvasqu6,xgitiaux,rangwala}@gmu.edu
jonathan.vasquez@uv.cl

1 Proofs

Theorem 1. *For any unaware classifier g, $x \in \mathcal{X}$, we have:*

$$\delta_{mis}(x) = r(x) \left| I_{\{g(x)=0\}} - I_{\{g(x)=1\}} \right|$$

where $I_{\{g(x)=1\}}, I_{\{g(x)=0\}}$ are the characteristic functions of the sets $\{x \in \mathcal{X} | g(x) = 1\}$ and $\{x \in \mathcal{X} | g(x) = 0\}$. Moreover, for a deterministic unaware classifier, $\delta_{mis}(x) = r(x)$.

Proof. Let $\eta_s(x)$ denote $P(Y = 1|X = x, S = s)$. For any classifier g, we have

$$\begin{aligned} P(g(X) = Y|X = x, S = s) =& P(Y = 1, g(X) = 1|X = x, S = s) \\ &+ P(Y = 0, g(X) = 0|X = x, S = s) \\ \overset{(a)}{=}& I_{\{g(x)=1\}}\eta_s(x) + I_{\{g(x)=0\}}(1 - \eta_s(x)), \end{aligned}$$

with indicator function $I_{\{g(x)=i\}} = 1$ whenever $g(x) = i$, and (a) uses the fact that $g(X)$ is independent of Y conditional on X. Therefore,

$$\begin{aligned} \delta_{mis}(x) &= |I_{\{g(x)=1\}}(\eta_0(x) - \eta_1(x)) - I_{\{g(x)=0\}}(\eta_0(x) - \eta_1(x))| \\ &= r(x) \left| I_{\{g(x)=1\}} - I_{\{g(x)=0\}} \right|. \end{aligned}$$

Theorem 2. *For any unaware classifier g,*

$$\Delta_{mis}(g) \leq \frac{1}{2} E_{x \sim P_s}[r(x)] + \frac{1}{2} E_{x \sim P_{ns}}[r(x)] + TV(P_s, P_{ns}), \tag{1}$$

where $TV(P_s, P_{ns})$ is the total variation between the distribution P_s of X conditional on $S = s$ and the distribution P_{ns} of X conditional on $S \neq s$.

Proof. Let $\eta_s(x)$ denote $P(Y|X = x, S = s)$:

$$\begin{aligned} P(g(X) = Y|S = s) &= \int P(g(X) = Y|S = s, X = x) P_s(dx) \\ &= \int \left[I_{\{g(x)=1\}}\eta_s(x) + I_{\{g(x)=0\}}(1 - \eta_s(x)) \right] P_s(dx). \end{aligned}$$

Let $c(x)$ denote $I_{\{g(x)=1\}} - I_{\{g(x)=0\}}$. Therefore, we have

$$\begin{aligned}
\Delta_{miss} &= \left| \int c(x)\eta_s(x)P_s(dx) - \int c(x)\eta_{ns}(x)P_{ns}(dx) \right| \\
&= \left| \int c(x)\frac{\eta_s(x)+\eta_{ns}(x)}{2}\left[P_s(dx) - P_{ns}(dx)\right] \right. \\
&\quad \left. + \int c(x)\frac{\eta_s(x)-\eta_{ns}(x)}{2}P_s(dx) + \int c(x)\frac{\eta_s(x)-\eta_{ns}(x)}{2}P_{ns}(dx) \right| \\
&\overset{(a)}{\leq} \int \frac{r(x)}{2}P_s(dx) + \int \frac{r(x)}{2}P_{ns}(dx) + \int \left|P_s(dx) - P_{ns}(dx)\right|,
\end{aligned}$$

where (a) uses the fact that $c(x) \leq 1$ and $\eta_s, \eta_{ns} \leq 1$, the triangular inequality. The result in Theorem 1 follows from the definition of total variation.

Theorem 3. *Suppose that there exists a sub-population $G \subset \mathcal{X}$ such that $P(G) > \gamma$, $P_s = P_{ns}$, and $P(Y=1|X=x, S=s) > P(Y=1|X=x, S\neq s)$. For any stochastic classifier $g : \mathcal{X} \to [0,1]$ such that $inf_{x\in G} g(x) > 1/2$, there exists $\kappa > 0$ such that*

$$\Delta_{sub-mis}(g,\gamma) > \kappa E[r(x)|x \in G]. \tag{2}$$

Proof. Let $2\kappa = inf_{x\in G} g(x) - 1/2 > 0$. Using derivations from the proof of Theorem 1 and the assumption that $P_0 = P_1$, we can show that

$$\begin{aligned}
\Delta_{mis}(g|x\in G) &= \left| \int_{x\in G} (2g(x)-1)(\eta_s(x) - \eta_{ns}(x)P(dx) \right| \\
&\overset{(a)}{>} \kappa \int_{x\in G} \left[\eta_s(x) - \eta_{ns}(x)\right] P(dx) \\
&= \kappa E[r(x)|x\in G],
\end{aligned}$$

where (a) uses that $\eta_s(x) \geq \eta_{ns}(x)$ on G.

2 Datasets

This section provides more information regarding the datasets used in the experiments. We prepared a public GitHub repository to share the code, experiments, and public datasets in the following link. The folder `data` contains the corresponding files of each data described in the following subsections.

2.1 Synthetic Data

The synthetic data is constructed by creating a ground truth risk. To this end, we denote instances as a triplet (X, S, Y). Feature input X is a two-dimensional space in $[0,1]^2$. The sensitive attribute S is one dimensional, where we assign 0 and 1 with $P(S=1) = P(S=0) = 0.5$. Finally, we make Y dependent of S, by assigning $y = 1$ when $s = 1$, and label $y = 1$ with probability equal to $1 - \frac{x_1+x_2}{2}$

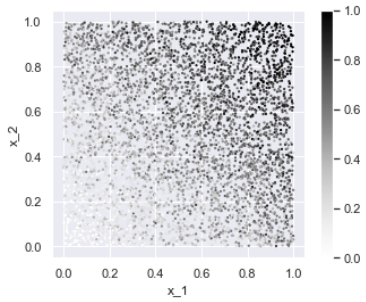

Fig. 1: Distribution of a sampled synthetic dataset. Values 1 (darker) and 0 (lighter) represent the highest and lowest levels of risk.

for $s = 0$. Furthermore, using the definition $r(x)$ in the article, the ground truth risk of synthetic data is equal to $r(x) = \frac{x_1+x_2}{2}$ for a given $X = x$.

Using this setting, we draw $5,000$ samples from this synthetic data 20 times using different seeds. Then we run FORESEE and BART to estimate $r(x)$ from it, so we can compute the mean and standard deviation and compare it to the ground truth. Figure 1 depicts the distribution of the resulting synthetic data and the distribution of true risk. Darker points represent a higher level of true risk obtained from the protocol.

2.2 Real world datasets

We use two benchmark data sets often explored in fair machine learning studies and a private dataset in educational context to evaluate our FORESEE algorithm. The former two sets are publicly available in ProPublica webpage[3] and UCI Machine Learning[4] repository. The latter is not publicly available, given privacy reasons regarding students' information. The final files can be found in the `data` directory. Description of each real datasets are provided below.

Adults. Also known as Census Income Data Set, this dataset can be found in the UCI Machine Learning repository. It is commonly used for benchmarking assessment of machine learning methods, particularly in the algorithmic fairness field due to the disparities in the outcome between males and females. It has a total of $48,844$ samples with 14 features plus the target variable, which is a binary variable indicating whether the income is larger than $50K$ or not. Table 2 enlists each feature and the corresponding range according to the repository. From the list, we use *sex* as the sensitive attribute. We remove the features *fnlwft* and *education* since both are considered useless for our classification task.

[3] ProPublica
[4] Census Income

Furthermore, *education* provides the same piece of information as *education-num*, and is highly correlated to it. Finally, we do not find null or empty values. Therefore, missing values handling is not needed.

Compas This dataset gathers information from $7,214$ individuals in the criminal justice system to assess their recidivism risk. It is publicly available by ProPublica through their github repository. From there, we retrieve *compas-scores-two-years.csv* file, which contains a total of $7,214$ samples and 53 features. Amongst these variables, we select the following 10: *sex*, *age*, *age_cat*, *race*, *juv_fel_count*, *juv_misd_count*, *juv_other_count*, *priors_count*, *c_days_jail*, *c_charge_degree*, and *two_year_recid*, where *race* is denoted as the sensitive attribute and *two_year_recid* as the target variable. As in Adult dataset, we do not find missing values to handle.

Dropout This datset is gently facilitated by a university. It contains information of $4,706$ undergraduate students from three bachelor programs, and 42 features plus the target variable representing whether the student dropped out within the first two years. Features include courses grades, participation in academic support programs, pre-matriculation information (e.g., type of high school and the GPA), and socio-demographic characterization (e.g., gender and family income). The sensitive attribute is the gender to which each student self-identifies to.

We remove useless and highly correlated features (e.g., year of entry) resulting in a dataset of 39 features. From this subset, 29 variables show some level of null values. Thus we implement the procedures described in the **Preprocessing and Transformation** section.

2.3 Preprocessing and Transformation

We implement a one-hot encoder to the categorical variables, and standardization to numerical features for all datasets. It is worthy to note that for *FORESEE* algorithm, one-hot encoder is not used for data fed to this algorithm, whereas for LR, RF, KNN, SVM, and MLP cases we implement both transformations. Furthermore, we use two imputers to handle missing values. In the case of categorical variables, we use a simple imputer by replacing null values with the most frequent one. On the other hand, for numerical variables, we use an iterative-multivariate that estimates missing values of a feature as a function of the others. We rely on the impute module of sklearn library [1] to implement the one-hot encoders and imputers.

3 FORESEE Algorithm

The FORESEE algorithm estimates risks by using two steps. The first one, called fitting, fits M decision trees on a training dataset (i.e., collection of trees

Feature	**Range**
age	continuous
workclass	Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
fnlwgt	continuous
education	Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
education-num	continuous
marital-status	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
occupation	Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspect, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
relationship	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
race	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
sex	Female, Male.
capital-gain	continuous.
capital-loss	continuous.
hours-per-week	continuous.
native-country	United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

Table 2: Feature and its range of Adults dataset.

$T_1, T_2, \ldots, T_M$) and stores for each one a tuple comprising the resulting models (T_m) and the corresponding leaves (L_m). Furthermore, we assume a pessimistic scenario for any groups not present in a leaf and set the maximum misclassification rate to that group (i.e., equal to 1). The second step, called estimation, aims to estimate the risk scores for a given set of samples $x \in D$ by using the outcome from the fitting step. To this end, it retrieves the tuples of each decision tree, finds the subset of T_m that satisfy the performance threshold, and computes the average of misclassification rate variation within the leaves where sample x belongs to ($L_m(x)$) along the subset of decision trees. Algorithm 1 depicts the pseudocodes of both fitting and estimate steps. Furthermore, it uses complementary methods, which are detailed in the following list:

- SUBSAMPLING(D, π, ϕ): this method returns a subsampling dataset from D_{train} by randomly sampling ϕ features and a portion of π samples.
- LEAVES(T_m): it gathers and returns the collection of leaves of a T_m. Note that $L_m(x)$ returns the leaf from T_m where sample x belongs to.
- PERFORMANCE(T_m): it computes and returns the performance of T_m.
- MV(L_m, x): it retrieves the misclassification rate variation across the groups for $L_m(x)$.

An implementation of these pseudocodes can be found in the directory `Code/source/estimators` in the Dropbox repository found in this link.

4 Experiments Setup

The experiments are implemented in a `python 3.8` environment. We use `jupyter notebooks` to perform the data preprocessing, estimates, and analysis. All classes, objects, and functions are implemented in python modules. Both notebooks and python modules can be found in `Code/source` and `Code/experiments` directories. For running experiments, we use a machine with a 2 Ghz Quad-Core Intel Core i5 processor, and a 16 GB 3733 MHz LPDDR4X memory.

Table 3 enlists the libraries and corresponding versions used in the implementation and experiments. It is worthy to note that `bartpy` and `causalml` are needed for training and testing the SBART model.

Table 4 lists the set of hyperparameters we use for hyperparameter tuning. Additionally, we use 200 decision trees for **Adult** dataset, and 400 for **Dropout** and **Compas** datasets when we train FORESEE algorithm. The reason of these parameters is related to the dataset size. Finally, for the sake of reproducibility, we set `random_state` seed equal to 0.

We also implement a greedy searching procedure to find the highest β provided to the estimation step of FORESEE. This process consists in increasing progressively the β such that the final number of leaf contributing to the $r_{FORESEE}(x)$ computation is as much the t-top percentile regarding performances. We set this threshold equal to .9, therefore, we are removing from the estimation at least the 10% decision trees with the worst performance. After implementing this procedure we find that for Adult, Dropout, and Compas datasets

Library	Version
`bartpy`	0.0.2
`causalml`	0.12.0
`matplotlib`	3.5.1
`numpy`	1.4.3
`pandas`	1.4.3
`pickle`	4.0
`scipy`	1.7.3
`seaborn`	0.11.2
`sklearn`	1.1.1
`statsmodels`	0.13.2

Table 3: List of library requirements for experiments.

the highest β are 0.61, 0.34, and 0.57, respectively. Recall that for Compas we use accuracy instead of F-1 since we do not find issues of imbalanced class in its target variable. More specifically, the positive/negative ratio class is approximately 0.5 in the training and testing sets of Compas dataset.

Model	**hyperparameters**
Decision Tree	`criterion`: entropy `splitter`: random `min_samples_leaf`: 2 `class_weight`: {balanced, None}
Logistic Regression	`C`: $[0.01, 0.1, 1]$ `fit_intercept`: {True, False} `solver`: {liblinear, lbfgs} `max_iter`: 100000 `class_weight`: {balanced, None}
Random Forest	`n_estimators`: $\{10, 50, 100\}$ `criterion`: {gini, entropy} `max_depth`: {None,5, 10, 15} `class_weight`: balanced
`k`-Neirest Neighbors	`weights`: {uniform, distance} `n_neighbors`: $\{5, 10, 15, 20, 25, 30\}$
Support Vector Machines	`kernel`: {rbf, sigmoid, poly} `probability`: True `C`: $\{0.001, 0.01, 1, 10, 100\}$ `class_weight`: {balanced, None}
Multi Layer Perceptron	`number_hidden_layer`: 2 `hidden_layer_sizes`: $\{10, 100, 500\}$ `learning_rate_init`: $\{0.00001, 0.0001, 0.001, 0.01, 0.1, 1\}$ `max_iter`: 1000 `tol`: $1e-4$
SBART	default
FORESEE	`M`: $[200, 400]$ β: $\{0.61, 0.34, 0.57\}$

Table 4: List of hyperparameters for each machine learning model used in the experiments.

References

1. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)