MODEL COMPARISON FOR DISCRIMINATION ON COUNTRY OF ORIGIN

Ning Hu, Tianxing Jiang, Zhiyu Lin, Kuiyu Zhu

nh553@georgetown.edu tj247@georgetown.edu zl281@georgetown.edu kz175@georgetown.edu

INTRODUCTION

Algorithmic fairness has a profound impact on societies as machine learning algorithms start to dictate decision making in almost every industry. As studies in the past few years have shown, a biased crime prediction algorithm could discriminate against black and Latino people (O'Donnell, 2019); a poorly designed employment algorithm could discriminate against women in the job market (Houser, 2019). This project aims on studying algorithmic discrimination on the country of origin, adding a new category to the conversation around algorithmic fairness on the race and gender dimensions.

Our speculation is that people who are born outside of the U.S. are likely to have a lower income than those who are born native, given when other income-related factors remain constant. There are a few reasons why we think income discriminates against people of foreign births. First of all, people who are born outside of the United States are immigrants or children of immigrants. Compared to natives who have accumulated wealth through inheritance, immigrants face a generational economic disadvantage. Moreover, U.S. immigration policies place both financial and legal burdens on employers who hire non-US citizens, eliminating a large amount of job opportunities from highly skilled immigrants. Third, undocumented immigrants can be excluded from taking certain salary competitive jobs and forced to settle with less rewarding ones due to their documentation status, which further deepens the income inequality gap around country of origin.

However, could this socioeconomic discrimination be picked up my machines? In other words, do machine learning algorithms implement discrimination against minority groups on the national origin dimension? (Calmon et al, 2017) In this study, we want to answer four questions: First, do classification and regression models discriminate against immigrants when predicting income level? Second, how much do these models discriminate against immigrants? Third, which models are less prone to discrimination? Fourth, what exactly do these better algorithms do to make them discriminate less?

We will perform two types of comparisons: One, we will compare vertically how different machine learning models perform on a protected feature and find out which models are more prone to discrimination in general. Two, we will compare horizontally among country of origin, race, and sex, and propose potential explanations for why we see stronger or weaker machine classification biases in one protected feature over another. Finally, we will look closely at one of the better performing algorithms, provide possible reasons for why they are less prone to discrimination by expanding on the optimization programs in their solvers.

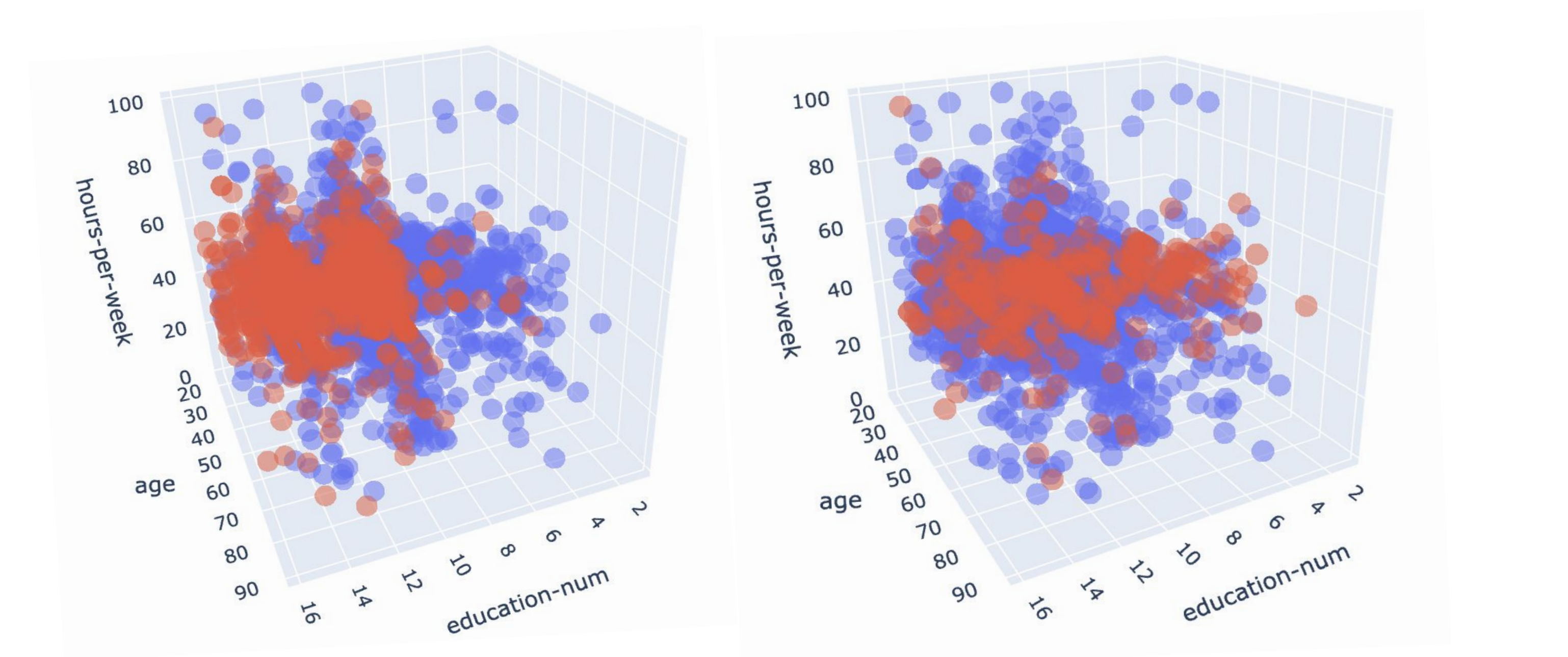

Fig.1. The plots above shows the 1:9 split test data from random sampling where the three axes represent age, years of education and hours an individual works per week. In the left plot, the red portion are individuals who make more than 50K a year. In the right plot, the red portion are individuals whose nation of origin is outside of the U.S. We will change the percentage of red groups in the right plot and see how the algorithms perform on predicting income.

DATA

We utilized the University of California Irvine Machine Learning Repository Adults data as our raw data. This data has 48842 observations where each observation is an individual and each category represents a demographic feature. To investigate 3 potential dimensions of discrimination, which are country of origin, sex, and race, three groups of data are generated based on almost the same processing.

Take country of origin for example. We selected five features: country of origin, income, age, education, and number of hours of work per week. Country of origin is our slicing variable. Income is our target, and the rest are predictors.

D_feature: **Country of origin**. Binary variable of value 1 or 0.
Y_feature: **Income**. Categorical variable of either >50K or <=50K.
X_feature: **Age**. The age of the individual, integer type from 17 to 90.
Education. The number of years of education the individual received, integer from 1 to 16.
Hours per week. The number of hours an individual work each week, integer from 1 to 99.

The reason we decided to choose age, education and hours per week as our predictor features is that we think that these are the most relevant ones when classifying for income. Other unnecessary columns are dropped, and then NA rows were dropped.

Since immigrants are minorities in this data set, we found that the ratio of US nationals and Non US nationals are about 1:9. Thus we separate all data by 1:9 ratio for immigrants versus native born Americans. We built 80% of the data into training set of the ratio 1:9, and randomly built two testing sets, one with original ratio and the other with a new ratio of 1:1. We will explain why we use these two test data sets later in the next section.

METHODOLOGY

Two testing sets are our trick to trick our selected models. The models are trained by dataset with original ratio. (In the case of country of origin, the ratio is **1:9**.) Then we use **testing set I** with original ratio to test the model and get accuracy I and precision I of each model. We do the same thing to **testing set II** with ratio of **1:1** and get accuracy II and precision II. Now, if discrimination exists, which means there is difference between **testing set I** and **testing set II**, our accuracy and precision are going to be different. The model will do its best on optimization referring to the training set, but it will not distinguish our trick of feeding another testing set to it. Vice versa, if there is trivial discrimination, our result of two testing set should be almost the same.

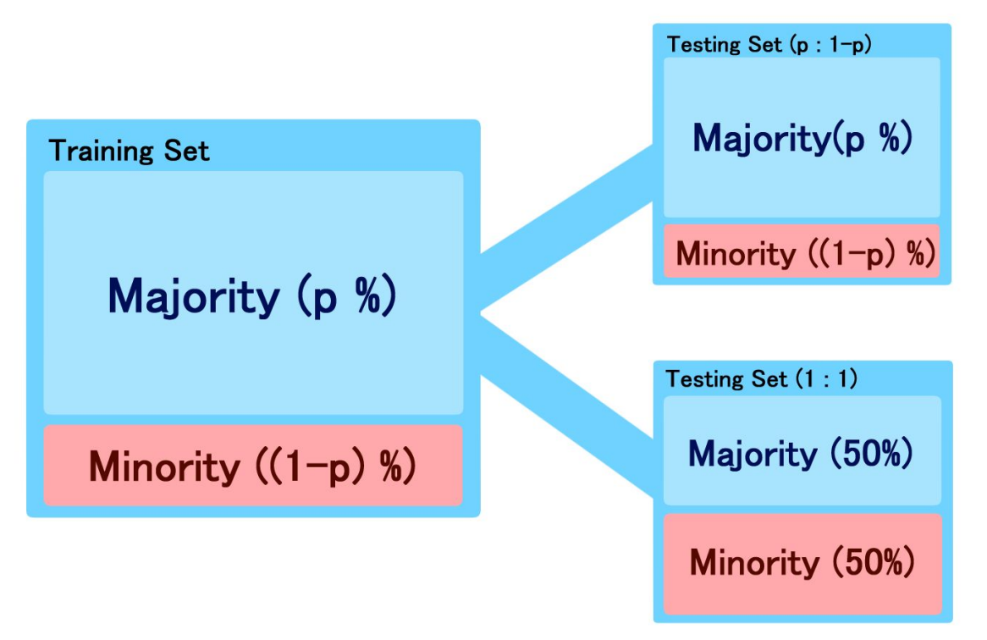

Fig. 2: Method for Separating the data

We selected a few classification models including: naive bayes, logistic regression, support vector machines, decision trees, and random forest.

We want to see how well the classification algorithms perform on predicting income, when we change the percentage of minority in the test group. In other words, we want to know the sensitivity of these algorithms in treating discrimination in a classification task. Once we have a better understanding of performance of each classification model, we will pick the best one and dive into its algorithm and the specific optimization program built in Sklearn, and propose potential explanations for why this classification model outperforms the others against discrimination.

RESULTS - Part I

Precision	National Origin		Race		Sex	
Naïve Bayes	0.6617	0.6313	0.656	0.5932	0.6225	0.5813
SVM	0.5847	0.5435	0.5339	0.4982	0.5402	0.5016
Logistic Regression	0.5918	0.5913	0.6353	0.5829	0.6151	0.5705
Decision Tree	0.5818	0.5639	0.5298	0.4831	0.5203	0.4754
Random Forest	0.5639	0.5612	0.5462	0.5054	0.5526	0.5078

National Origin	Race	Sex
-4.59	-9.57	-6.62
-7.05	-6.69	-7.15
-0.08	-8.25	-7.25
-3.08	-8.81	-8.63
-0.48	-7.47	-8.11

Table 1: Precision and precision difference for 5 classification models

Accuracy	National Origin		Race		Sex	
Naïve Bayes	0.8009	0.7965	0.8066	0.8267	0.7992	0.8129
SVM	0.7824	0.7699	0.7724	0.803	0.7783	0.7936
Logistic Regression	0.7795	0.7824	0.7982	0.8219	0.7936	0.8085
Decision Tree	0.7795	0.7758	0.7703	0.7987	0.7718	0.786
Random Forest	0.7743	0.7758	0.7766	0.8051	0.7825	0.7955

National Origin	Race	Sex
-0.55	2.49	1.71
-1.6	3.96	1.97
0.37	2.97	1.88
-0.47	3.69	1.84
0.19	3.67	1.66

Table 2: Accuracy and accuracy difference for 5 classification models

The table on the left shows accuracy and precision scores for five different algorithms with respect to national origin, race and sex. In each column of protected feature, the yellow bars are the scores tested on the data with 10% immigrants, and the blue bars are the scores tested on the data with 50% immigrants. The table on the right calculates the percentage of change in accuracy and precision for each model.

Comparing the performance of the five models, we can see that in general the classification precision decreases when the percentage of protected group in the test sample increases. Logistic regression has the least drop (much smaller than the rest) in precision when modeling for national origin. SVM and Naive Bayes have the least drop in precision when modeling for race and sex, respectively. Logistic regression also has one of the smallest changes in accuracy across the protected categories.

Comparing these three protected classes, race and sex demonstrate higher accuracy and precision change when we increase the percentage of the minority group in the test sample. National origin has the smallest overall precision and accuracy change, and it's variation can be both upwards and downwards in the accuracy table. We will discuss the implications of these results in the next section.

Comparing precision and accuracy scores, we can see that precision generally decreases when the percentage of minority group members increases in the test; whereas accuracy generally increases when minority groups receive more equitable representation.

DISCUSSION - Part I

As explained in the methodology section before, the change in classification accuracy and precision scores when we have only changed the percentage of the minority group in test samples indicates that these models treat minority groups differently than the privileged group.

Among these models, Logistic regression and Random Forest showed the lowest sensitivity toward the change in percentage of the protected group. SVM, Decision Trees, and Naive Bayes all show some higher level of sensitivity. This result means that Logistic regression and Random Forest are some of the best ones when it comes to classifying data with protected features. These are the least bias-prone and are not so easily affected when the training data does not provide enough representation for minority groups. SVM, Decision Trees and Naive Bayes are slightly worse, meaning that they are more heavily dependent upon equitable group representations in the training data.

Since Logistic regression performs pretty well in terms of showing relatively lower sensitivity to discrimination, we will dig into the specific algorithms and optimization programs built into the Sklearn.LogisticRegression package and compare their results.

RESULTS - Part II

National Origin	precision		accuracy	
newton-cg	0.6358	0.5909	0.7854	0.5909
lbfgs	0.6358	0.5909	0.7854	0.5909
sag	0.6358	0.5909	0.7854	0.5909
saga	0.6424	0.6012	0.7847	0.6012

Race	precision		accuracy	
newton-cg	0.6157	0.5574	0.7866	0.5574
lbfgs	0.6157	0.5574	0.7866	0.5574
sag	0.6157	0.5574	0.7866	0.5574
saga	0.6305	0.5765	0.7882	0.5765

Table 3: Sklearn.LogisticRegression solvers for national origin and race

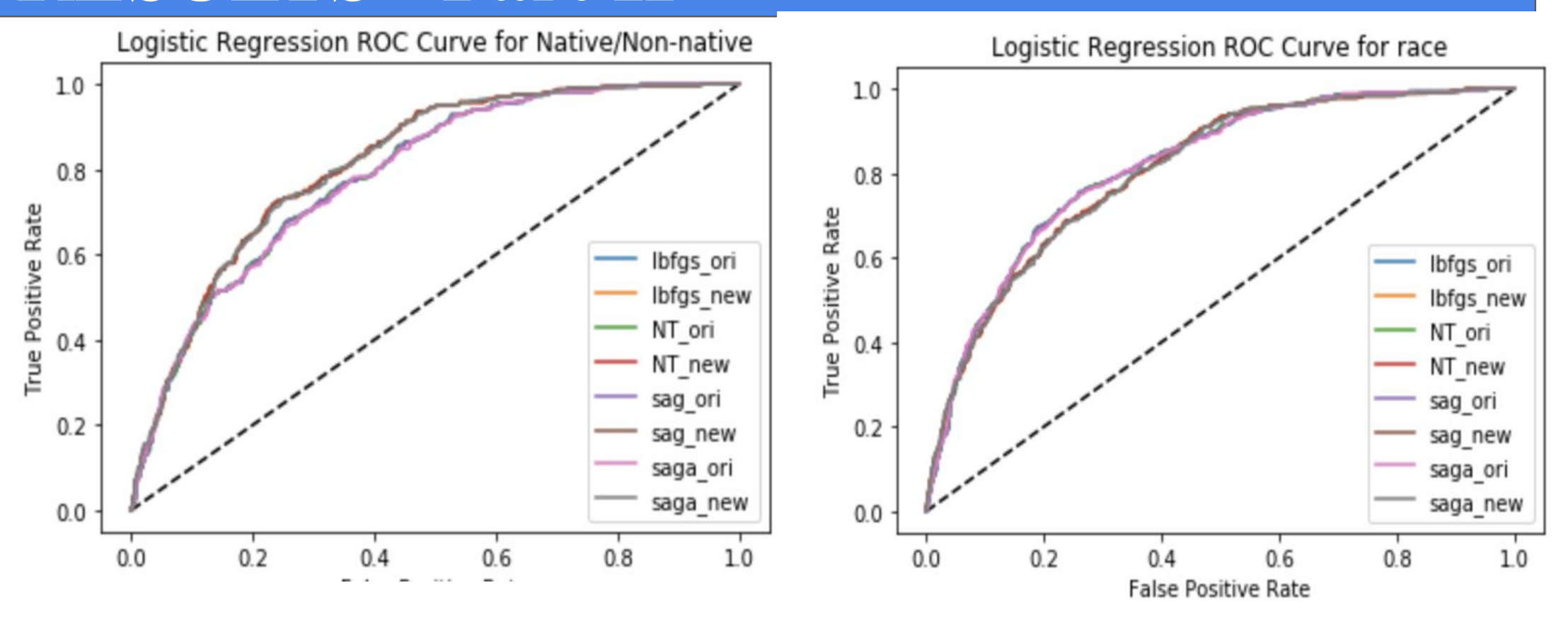

Fig. 3: ROC curves for four different LogisticRegression solvers on the regular 1:9 test set and 1:1 test set, for feature: National Origin and Race

The top three solvers (Newton-CG, L-BFGS, and SAG) return the same precision and accuracy scores in both tables. The last solver: SAGA shows slightly higher precision and accuracy results. This is because these algorithms incorporate different optimization procedures. The ROC curves show pretty good classification results; again, we do not see a huge difference among these algorithms. In the next section, we will explain the optimization programs behind them. We also tested the run time for these four solvers. They are: 0.102, 0.040, 0.278, and 0.464, respectively.

DISCUSSION - Part II

All four solvers by Sklearn.LogisticRegression incorporate convex optimization programs.

Newton-CG:

This solver utilizes Newton's Method when searching for the minimizer. If the target class is binary, then the Hessian is calculated with double derivatives. Otherwise if there are multiple classes in the target (multi_class = 'multinomial'), then the Hessian is calculated in a smart way (R{.}) where double derivatives are avoided. This R{.} method takes advantage of the relationship between the gradient and the Hessian (Pearlmutter, 1994).

$$\nabla_{\mathbf{w}}(\mathbf{w}+\Delta\mathbf{w}) = \nabla_{\mathbf{w}}(\mathbf{w}) + \mathbf{H}\Delta\mathbf{w} + O(\|\Delta\mathbf{w}\|^2)$$

(1) Expanding the gradient of w+dw

$$\mathbf{H}\mathbf{v} = \frac{\nabla_{\mathbf{w}}(\mathbf{w}+r\mathbf{v}) - \nabla_{\mathbf{w}}(\mathbf{w})}{r} + O(r).$$

(2) Rewriting (1) for the Hessian

$$\mathbf{H}\mathbf{v} = \lim_{r\to 0} \frac{\nabla_{\mathbf{w}}(\mathbf{w}+r\mathbf{v}) - \nabla_{\mathbf{w}}(\mathbf{w})}{r} = \frac{\partial}{\partial r}\nabla_{\mathbf{w}}(\mathbf{w}+r\mathbf{v})\Big|_{r=0}$$

(3) Compute the derivative of the Hessian

$$\mathcal{R}_{\mathbf{v}}\{f(\mathbf{w})\} \equiv \frac{\partial}{\partial r} f(\mathbf{w}+r\mathbf{v})\Big|_{r=0}$$

(4) Rewriting (3) where $\mathbf{H}\mathbf{v} = \mathcal{R}_{\mathbf{v}}\{\nabla_{\mathbf{w}}(\mathbf{w})\}$.

After calculating the gradient and the Hessian using the methods above, Sklearn LogisticRegression uses Newton's Conjugate Gradient (what Newton-CG is short for) to find the minimizer. This algorithm has an inner and an outer loop (5). The outer loop performs the Newton iteration to find the search direction and the inner loop uses conjugate gradient to iteratively solve *fhess_p . xsupi = fgrad* for *xsupi*. Conjugate Gradient simply means that when we are solving systems of equations Ax=b, in order to avoid inverting matrix A, we iteratively find $\alpha_k = \frac{\langle p_k, b\rangle}{\langle p_k, p_k\rangle_A}$ where pk represents the k th conjugate vector with respect to matrix A.

L-BFGS (Limited-memory BFGS):

This method is generally the same as Newton-CG, but using an approximate inverse of the Hessian to approach the optimization. In order to achieve limited memory usage, L-BFGS uses only a few vectors to represent the inverse. Thus, if the objective data is not very large, this algorithm can perform perfectly but not as expensive as the Newton-CG solver. In our tests, L-BFGS turns out to be holding the same result as Newton-CG. This is because our data only contain three features, so L-BFGS and Newton-CG are essentially both using Newton's method.

SAG (Stochastic Average Gradient) and SAGA:

Both SAG and SAGA only update the immediate gradient, not for the whole sample, so they save memory space. (6) SAGA is an extension of SAG that allows for L1 regularization, with better theoretical convergence rates. L1 shrinks the less important features' coefficient to zero, which works well for feature selection in case we have a huge number of features. From a variance reduction perspective, while SAG uses a biased step size $\alpha = \frac{1}{n}$, SAGA uses an unbiased step size where $\alpha = 1$ (7). While the variance of the SAG update is 1/n^2 times that of SAGA, it happens at the expense of having zero bias. Below is the step update for SAG and SAGA.

$$x^{k+1} = x^k - \gamma\left[\frac{f_j'(x^k) - f_j'(\phi_j^k)}{n} + \frac{1}{n}\sum_{i=1}^{n} f_i'(\phi_i^k)\right]$$

(5) SAG step update

$$x^{k+1} = x^k - \gamma\left[f_j'(x^k) - f_j'(\phi_j^k) + \frac{1}{n}\sum_{i=1}^{n} f_i'(\phi_i^k)\right]$$

(6) SAGA step update

CONCLUSION

Minorities need enough representations in training machine learning algorithms. If the data has low representation for a minority group, use logistic regression and random forest first because these tend to discriminate less than the other classification algorithms.

Algorithmic discrimination on national origin shows a weaker trend than race and sex. It is unclear why this happens, although intuitively we get the general sense that societal discrimination based on these two features are much more prominent than on national origin. Future studies on algorithmic interpretations for this result could be interesting.

Comparing the 4 different solvers in Sklearn.LogisticRegression, SAGA shows slightly lower level of discrimination than the other three. This might be due to the fact that SAGA only performs partial update on the immediate gradient compared to Newton's Method, and it utilizes an unbiased step size alpha = 1 compared to SAG.

REFERENCES

O'Donnell, Renata M. 2019. Challenging Racist Predictive Policing Algorithms Under The Equal Protection Clause. New York University Law Review. Vol:94.544. https://www.nyulawreview.org/wp-content/uploads/2019/06/NYULawReview-94-3-ODonnell.pdf

Houser, Kimberly A. 2019. Can AI Solve the Diversity Problem in the Tech Industry? Mitigating Noise and Bias in Employment Decision-Making. P324. https://law.stanford.edu/wp-content/uploads/2019/08/Houser_20190830_test.pdf

Calmon, Flavio P., Wei, Dennis, Ramamurthy, Karthikeyan N., Varshney, Kush R., 2017. Optimized Data Pre-Processing for Discrimination Prevention. Cornell University, ArXiv: 1704:03354 https://arxiv.org/pdf/1704.03354.pdf

Pearlmutter, Barak A., 1993.Fast Exact Multiplication by the Hessian. Neural Computation. Vol 6-1. 147-160. http://www.bcl.hamilton.ie/~barak/papers/nc-hessian.pdfa

Tavory, Ami, Azulay, Shahar, Raveh-Sadka,Tali. 2017. Source code for sklearn.linear_model.logistic https://ibex.readthedocs.io/en/latest/_modules/sklearn/linear_model/logistic.html

Schmidt, Mark, Le Roux, Nicolas, Bach, Francis. 2015. Minimizing Finite Sums with the Stochastic Average Gradient. Cornell University, ArXiv: 1309: 2388 *https://arxiv.org/pdf/1309.2388.pdf*

Defazio, Aaron, Bach, Francis, Lacoste-Julien, Simon. , 2014. SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. Cornell University, ArXiv: 1407:0202. *https://arxiv.org/pdf/1407.0202.pdf*