When to Stop-and-Frisk

MADELINE SAVIANO, msaviano@stanford.edu
SARAH TIEU, stieu12@stanford.edu

This project was intended to demonstrate the bias that machine learning algorithms may learn. We wanted to explore the impact bias has on these algorithms and explore the methods that help correct these biases. Given data about stop-and-frisks instances, we ran several different models on the data and maximized the accuracies achieved by these models. After tuning the hyper parameters of the models, we decided to explore logistic regression, which allows us to manipulate the algorithm a bit more than the other models. One means of achieving algorithmic fairness is the idea of statistical parity, where the percentage of drivers of a certain demographic in our dataset is equal to the percentage of individuals of that demographic in the overall population. Overall, we were able to achieve statistical parity while maintaining a consistent accuracy. Thus we learned that we should be aware of the inherent bias in data regarding people and that algorithms can actually learn to be biased and that we, as members of a larger society, should attempt to correct these biases.

ACM Reference Format:
Madeline Saviano and Sarah Tieu. 2017. When to Stop-and-Frisk. 1, 1 (December 2017), 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

As machine learning algorithms continue to grow in popularity, their use often precedes ethical analysis on their impact. For example, machine learning algorithms that predict rates of recidivism are currently being used, regardless of the fact that some algorithms have been shown to have racially biased results. In the interest of analyzing potential bias created by algorithms around a controversial topic, we decided to work with data on stop-and-frisk. The input to our algorithm is a vector with numerical features. We then use several different machine learning algorithms to output a binary predicted stop outcome where a one corresponds to a non-negative outcome (i.e. the suspect having been involved in illegal activity that would have justified the stop).

2 Related Work

We did research on two disparate spheres. The first is statistical analysis on stop-and-frisk datasets. 10.1 and 10.2 analyzed which features tend to be the most predictive, which guided our thoughts on feature selection, and illustrated different ways of breaking down the dataset. The concept of a threshold test similar to the one we conducted on our models is also introduced in these papers, albeit with different models. 10.6 conducts a more thorough discussion of the threshold test, which we used to frame our setup of statistical parity through thresholding. The threshold test is introduced in 10.7 and is framed as a shift of probability distributions for each racial group to align in the name of fairness.

The second sphere of research we conducted was around algorithmic fairness and its mathematical and moral origins. 10.5 defines statistical parity, a type of algorithmic fairness we aim to achieve in our project, in a different context (recidivism) but we used that definition to shape our own: that an equal proportion of drivers are stopped in each race group. 10.4 furthered our understanding of the topic and drove our decision to focus on parity-based notions of fairness rather than preference-based as the former translates into algorithmic alteration more readily and suits our dataset. 10.3 introduces the concept of disparate mistreatment, another facet of fairness that applies to our dataset and provides more justification for the type of analysis we conduct by showing that our dataset is "unfair."

Finally, 10.8 is the article that sparked the idea for our project. It discusses the COMPAS algorithm, an algorithm used to predict recidivism rates, and uses statistical and error analysis to prove the algorithm's outcome (a risk score) is racially biased against black defendants. This algorithm had real world impact on many individuals before and after this analysis and the company that created it still disputes the analysis done. The article and its accompanying open source calculations prompted us to question whether algorithms trained on a potentially biased dataset would reflect the racially biases that the researchers in 10.8 found.

3 Dataset and Features

We used a dataset compiled by the Stanford Open Policing Project, from which we drew over 8 million examples from the state of Washington (80% to training data, 10% each to validation and test data). Each example contains all the information drawn from a single incident and corresponding police report. We used the outcome of the stop as the ground truth, which is positive if a ticket or violation was given.
From the raw input data consisting of 34 features (Id state, stop_date, stop_time, location_raw, county_name, county_fips,

Authors' addresses: Madeline Saviano, msaviano@stanford.edu; Sarah Tieu, stieu12@stanford.edu.

fine_grained_location, police_department, **driver_gender**, driver_age_raw, **driver_age**, driver_race_raw, **driver_race**, violation_raw, violation, **search_conducted**, search_type_raw, search_type, **contraband_found**, **stop_outcome**, is_arrested, **violations**, officer_id, **officer_gender**, **officer_race**, highway_type, road_number, milepost, **lat**, **lon**, contact_type, enforcements, drugs_related_stop), we removed duplicate features (ex. search_type_raw removed, search_type) and removed uncategorizable data (ex. county_name). We started with a set of ten features (bolded) and the predicted feature: stop_outcome, a binary outcome of 1 if the outcome was an arrest or a citation and 0 otherwise. Violations were manually mapped to a scale where more severe violations had a larger corresponding integer than minor violations and, for each example, the violation of highest severity was used as the input feature value. For each input feature, we added and removed the feature from our feature set and ran the accuracy; however, the accuracy decreased in all instances, and thus our final feature set consisted of those ten features.

4 Methods

y denotes the stop outcome and x denotes the input features.

4.1 Multinomial Naive Bayes

Naive Bayes is a generative learning algorithm, an algorithm that tries to model $p(x|y)$ and $p(y)$. Naive Bayes classifies an example by calculating the probabilities of that example being each class, and selecting the class with the highest probability. Naive Bayes assumes that the input features are independent. The output class can be calculated by:

$$\begin{aligned} argmax_y p(y|x) &= argmax_y \frac{p(x|y)p(y)}{p(x)} \\ &= argmax_y p(x|y)p(y) \\ &= argmax_y p(y) \prod_{i=1}^{m} p(x^i|y) \end{aligned}$$

4.2 Logistic Regresssion

Logistic regression is an approach to a binary classification problem. The sigmoid function is the hypotheses $h_\theta(x)$ which outputs a value in the range between 0 and 1.

$$\text{sigmoid function} = h_\theta(x) = g(z) = \frac{1}{1+e^{-z}}$$

Because this is a binary classification problem, we can set the probabilities of each class as follows:

$$P(y=1|x;\theta) = h_\theta(x)$$
$$P(y=0|x;\theta) = 1 - h_\theta(x)$$
$$\text{Simplified to: } p(y|x;\theta) = (h_\theta(x))^y(1-h_\theta(x))^{1-y}$$

Maximizing the log likelihood we get:

$$\begin{aligned} \frac{d\ell(\theta)}{d\theta_j} = \frac{d \log L(\theta)}{d\theta_j} &= \frac{d \log\Big(\prod_{i=1}^m (h_\theta(x^{(i)})^{y^{(i)}}(1-h_\theta(x^{(i)}))^{1-y^{(i)}}\Big)}{d\theta_j} \\ &= \frac{d \sum_{i=1}^m y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log\Big(1-h_\theta(x^{(i)})\Big)}{d\theta_j} \\ &= (y - h_\theta(x))x_j \end{aligned}$$

Thus to maximize the log likelihood, we update θ as follows (until convergence):

$$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$$

α is the learning rate and $h_\theta(x)$ is the sigmoid function.

4.3 Support Vector Machine (SVM)

Support Vector Machines separates the data (given the features) into different classes, essentially creating a decision boundary where on one side, the data is categorized as 0 and the other 1. SVMs utilize the idea of margins, the minimal distance between a data point to the decision boundary. The geometric margin is the smallest margin of the individual training points. SVMs attempt to maximize the geometric margin because as the geometric margin increases, the separation of the two classes increases. That in turn allows for better classification accuracy.

$$\text{Decision boundary: } w^T x + b$$

$$\text{Geometric margin (single point): } \gamma^{(i)} = y^{(i)}((\frac{w}{||w||})^T x^{(i)} + \frac{b}{||w||})$$

$$\text{Geometric margin (training set): } \gamma = \min_{i=1,\dots,m} \gamma^{(i)}$$

(geometric margin is invariant to parameter scaling).
Thus SVM solves the following optimization problem:

$$\begin{aligned} &\max_{\gamma,w,b} \gamma \text{ s.t. } y^{(i)}(w^T x^{(i)} + b) \geq \gamma, i = 1,\dots,m \text{ and } ||w|| = 1 \\ &= \max_{\gamma,w,b} \frac{\gamma}{||w||} \text{ s.t. } y^{(i)}(w^T x^{(i)} + b) \geq \gamma, i = 1,\dots,m \\ &= \min_{\gamma,w,b} \frac{1}{2}||w||^2 \text{ s.t. } y^{(i)}(w^T x^{(i)} + b) \geq \gamma, i = 1,\dots,m \end{aligned}$$

4.4 Neural Network

Neural Networks is a form of 'deep learning'. Neural Networks allow for the creation of complex functions. Essentially a neural network processes the data, computes a function and passes it to the next layer. At the last layer, the output layer, the neural network returns the output. At each output, there is a weight matrix W and a bias matrix b that are learned, and an activation function g (such as the sigmoid function explained in 4.2) that is specified in the creation of the model. When running a Neural Network, it first computes the predictions and runs the data through the layers (forward prop)

and then calculates the error caused by each weight in both the weight and bias matrix and updates these weights accordingly (backward propagation).
$W^{[i]}, b^{[i]}$ denotes the weight, bias matrix of the i-th layer.
Forward Propagation:

$$\begin{aligned}\text{Layer 1: } z^{[1]} &= W^{[1]}x + b^{[1]}\\ a^{[1]} &= g(z^{[1]})\\ \text{Layer 2: } z^{[2]} &= W^{[2]}a^{[1]} + b^{[2]}\\ a^{[2]} &= g(z^{[2]})\\ &\dots\\ \text{Layer i: } z^{[i]} &= W^{[i]}a^{[i-1]} + b^{[i]}\\ a^{[i]} &= g(z^{[i]})\end{aligned}$$

Back Propagation and Update Rule:

$$\begin{aligned}W^{[i]} &= W^{[i]} + \alpha\frac{\mathrm{d}L}{\mathrm{d}W^{[i]}}\\ b^{[i]} &= b^{[i]} + \alpha\frac{\mathrm{d}L}{\mathrm{d}b^{[i]}}\end{aligned}$$

α is the learning rate and L is the loss function.
In this case, we used a cross entropy loss and a softmax activation function.

$$\begin{aligned}\text{Actual Output } &:= y\\ \text{Predicted Output } &:= \hat{y}\\ \text{Cross Entropy } &:= -\sum_{j=1}^{k} y_j \log \hat{y}_j\\ \text{Softmax (k-dim vector): } &= \sigma(z)_j = \frac{e^{z_j}}{\sum_{i=1}^{k} e^{z_i}}\end{aligned}$$

5 Experiments

5.1 Multinomial Naive Bayes

For our baseline, we ran Multinomial Naive Bayes on only the violation type as the feature input. This achieved an accuracy of training 79.20% and testing 79.21%.
Running Multinomial Naive Bayes on the full set of parameters (except latitude and longitude), we tuned the hyper parameters of the Laplace smoothing operator using a form of grid search. The Laplace smoothing operator accounts for new values of features that have not been encountered in the training data. Given a new feature, the probability of the feature is equal to zero and because products are often calculated with sums of logs, logs of zeros would generate errors. The smoothing operator allows the probability of a new feature to be a small feature. For instance a value of one would allow the probability to be equal to $\frac{1}{1+\text{training dataset size}}$. Thus a larger smoothing operator corresponds to a larger probability of an example with a new feature. However, adding the training operator did not make much of a difference as we consistently achieved a training accuracy of 63.76% and a validation accuracy of 63.81%.

5.2 Logistic Regression

Running Logistic Regression, the main hyper parameter we tuned was the regularization parameter. Regularizing the data prevents the model from over fitting. However, tuning the parameter did not change the accuracy as much. The best training accuracy was about 68.37% and the best testing accuracy was 68.38%. Because we know that regularization parameter did not drastically increase the accuracy, we can infer that the training data examples are all generally similar. Thus it explains the mediocre accuracy percentages of the datasets because the data was similar across both classes, it would be hard to distinguish between the two.

5.3 Support Vector Machine

With SVM, we tuned the penalty parameter c of the error term. We ran different feature sets with different c values and based on our results set $c = 1.0$.

5.4 Neural Network

Given that our feature set only consists of a few features, we decided to use a one layer neural network with cross entropy loss and the softmax activation function. To tune the neural network to achieve the best accuracy, we tuned the learning rate and the batch size. Then we tuned the number of units in the hidden layer.

Tuning learning rate and batch size (training accuracy):

	lr = 0.01	lr = 0.005	lr = 0.001
batch_sz = 2^8	0.706178	0.77181	0.705942
batch_sz = 2^9	0.720356	0.767995	0.703547
batch_sz = 2^10	0.727951	0.707703	0.66787

Tuning number of hidden units (on the training set)

Num Hidden	8	9	10	11
Accuracy	0.7144	0.6958	0.7718	0.6941

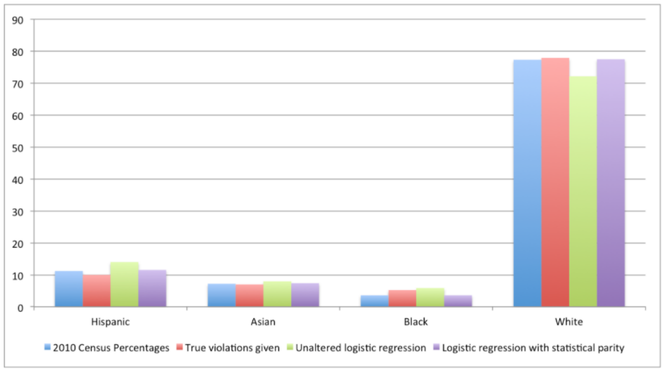
We ended up using a one layer neural network with ten hidden units, a batch size of 2^8 and a learning rate of 0.0005. Giving us a 77.18% training accuracy and a 77.39% testing accuracy.

6 Results

The following is the best accuracies achieved with each model after hyper parameter tuning.

	Train Accuracy	Test Accuracy
Baseline	0.7920	0.7921
Naive Bayes	0.6376	0.6381
Logistic Regression	0.6837	0.6838
SVM	0.7970	0.7779
Neural Network	0.7718	0.7726

The algorithms did similarly well on both train and test datasets because the dataset came from one source and nothing distinguishes one data point from being in the training set and the testing set. The baseline did the best, however, it was learning solely from the violations. While this may seem 'fair', to only look at the violation of each incident, it is still important to acknowledge that there may be information that can be learned from other features of each incident, such as the bias from gender and race. Naive Bayes and Logistic Regression performed similarly well, probably because they are less capable of learning complex relationships between the features, unlike SVM and Neural Networks.



After training the models, we added constraints to our models to achieve statistical parity, which is that an equal proportion of drivers are stopped in each race group. We first analyzed our dataset to look at proportions of racial groups stopped, then compared those percentages with the outcomes of our models. An unexpected twist found here is that each algorithm wildly changes the resultant percentages. For instance, logistic regression was very racially biased, stopping far more hispanic drivers than the police did (see above graph for comparison of the logistic regression model as thresholding was implemented) and fewer white drivers. However, naive bayes was also very racially biased, but in the opposite direction, stopping far more white drivers and far fewer asian and hispanic drivers. We then implemented thresholding on our models, where we used a different threshold (to decide positive or negative outcome) for each racial group in order to ensure that the percentages of drivers stopped in each racial group were equivalent to the percentages present in the 2010 census for Washington state. Using this type of tuning, we were able to mimic the 2010 census percentages quite closely without decreasing the training accuracy more than a percentage or two. Thus, the use of race-specific thresholds is clearly a viable technique to increase "fairness" and reduce the spread of probability distributions for each racial group.

7 Discussion

As we discovered after training multiple models, our data contains a relatively high level of bias, which could be explained by bias in using police reports generated by the officers themselves (general human error or self-serving justification to provide paperwork to support a stop) or inherent bias in our dataset not possessing enough or the right features to explain the complexity of a stop-and-frisk incident. The fact that our models were more racially biased (although variable in the type of racial biasing) in their labeling than the true violations given by officers was surprising. Furthermore, the act of manually adjusting thresholds based on race feels wrong – should a single software engineer decide that one race must meet a higher level of suspicion to be pulled over? On the other hand, since unaltered model can themselves be incredibly biased, software engineers, regardless of whether or not they acknowledge it, have a large impact on potentially devastating social impact issues.

8 Future Work

We would love to take test accuracy a step further by running our best models (SVM and Neural Network) on a dataset from another state; however, given that each state has slightly different features, this step would require a large amount of synthesizing between models. We would also attempt to find less biased data by consulting different stop-and-frisk datasets or training on multiple different states from the Stanford Open Policing Project to compare results.

Algorithmic Fairness as a whole is a relatively new area. Every paper we referenced was published in the last few years, with the threshold test only created this year, so each paper using the technique adds valuable contributions to the field. Although we implemented statistical parity, there is another way to achieve algorithmic fairness: predictive equality. Predictive equality occurs when the percentage of drivers of a certain demographic cited or arrested is equal to the percentage of that demographic in the population. It would be interesting to see how prediction accuracy differs between predictive equality and statistical parity. We also explored algorithmic fairness in terms of racial bias but we could also explore gender bias, whether or not the driver or officer genders were features that were learned in a biased manner.

9 Contributions

We split the models amongst ourselves and tuned the hyper parameters on our own. The report, poster, and other submissions were done together.

10 References

(1) E. Pierson, C. Simoiu, J. Overgoor, S. Corbett-Davies, V. Ramachandran, C. Phillips, and S. Goel. 2017. "A large-scale analysis of racial disparities in police stops across the United States".
(2) Sharad Goel, Justin M. Rao, and Ravi Shroff. "Precinct or Prejudice? Understanding Racial Disparities in New York City's Stop-and-Frisk Policy." March 2, 2015. Annals of Applied Statistics, Vol. 10, No. 1, 365-394, 2016.
(3) Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. 2017. "Fairness Beyond Disparate Treatment and Disparate Impact: Learning Classification without Disparate Mistreatment." In Proceedings of the 26th International World Wide Web Conference.
(4) Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, Krishna P Gummadi, and Adrian Weller. 2017. "From Parity to Preference-based Notions of Fairness in Classification."
(5) Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. 2017. "Algorithmic Decision Making and the Cost of Fairness." In Proceedings of KDD '17, August 13-17, 2017, Halifax, NS, Canada, 10 pages. DOI: 10.1145/3097983.3098095
(6) Emma Pierson, Sam Corbett-Davies, and Sharad Goel. 2017. "Fast Threshold Tests for Detecting Discrimination." Under review.
(7) Camelia Simoiu, Sam Corbett-Davies, and Sharad Goel. 2017. "The Problem of Infra-marginality in Outcome Tests for Discrimination."
(8) Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. 23 May 2016. "Machine Bias." ProPublica. Online.
(9) TensorFlow
(10) SciKit-Learn