CS5785 Final Project Report

Eesha Khanna (ek542), Jenny Liu (jjl295), and Preksha Agarwal (pa293)

12/13/2021

MOTIVATION

New Yorkers have historically been subjected to police stop and frisk. Under the Bloomberg Administration, stop-and-frisk reached its height of 685,000 stops in a given year. [1] In 2013, a federal judge found overwhelming evidence of racial bias and racial profiling and ruled that NYPD's stop-and-frisk violates the Fourth Amendment prohibition of unreasonable searches and seizures.

While data shows that stop-and-frisk went down substantially after this point (overall stops for the CPW (criminal possession of a weapon) category went down from 301,513 between 2009-2010 to 13,031 between 2015-2019), NYPD continues to conduct stop-and-frisk. We are going to look at the latest stop-and-frisk data from 2015-2019 and predict which stops initiated due to suspicion of a weapon in possession actually result in weapons being found. We will be doing Dataset Exploration with this project, meaning we will be applying machine learning techniques to a new (most recent) dataset to understand if the 2013 legislation decreased discriminatory policing practices when conducting stop-and-frisk. Some initial data analysis quickly led us to our conclusion that even after the 2013 legislation, stop-and-frisk data indicates a violation of the Fourth Amendment—only 17% of stops result in finding a weapon on the suspect which means that at least some of the remaining 83% of stops violate the Fourth Amendment prohibition of unreasonable searches and seizures.

With this baseline knowledge, we further explore our data to predict the likelihood of a weapon being found, given a certain set of neighborhood, suspect, and time factors. In this manner, we also identify which factors are most important in the decision to conduct stop-and-frisk. Our study will attempt to both explore a new dataset to understand the implications of the 2013 legislation (which features are most important in stop-and-frisk decisions), and optimize predictive power to recommend an "ideal" stop rate to decrease crime (seize the same number of weapons) while also decreasing Fourth Amendment violations and police bias in policing tactics like stop-and-frisk. A use case of this model would be a screening app that the police can use to assess pedestrians they are observing and thinking of potentially stopping before they decide to stop them. An accurate predictive model will thus help reduce the number of 'unnecessary stops.'

CONTEXT

While the total number of stops are known to have decreased since the law was passed, we suspect that the stops were still conducted in a discriminatory manner, meaning that people from certain races were stopped at a disproportionately high rate relative to their representative population. Research also shows that nearly 90% of stopped-and-frisked New Yorkers are found innocent, and that stop-and-frisk disproportionately affects Black and Hispanic communities in New York City. [3] In the paper "Precinct or prejudice? Understanding racial disparities in New York City's stop-and-frisk policy", Sharad Goel, Justin M. Rao, and Ravi Shroff conduct a study using stop-and-frisk data from 2008-2012 to argue that the police conduct more stops that they need to, particularly in high-crime, predominantly minority, particularly public housing areas, and disproportionately on minority groups like Black and hispanic people relative to white people. And that the same results of weapon seizure could be achieved by conducting only 6% of the stops that are currently being conducted.

The study makes use of a logistic regression model using Stochastic Gradient Descent (SGD) followed by a random forest classifier to produce robust and accurate estimates of the features that accurately predict the likelihood of weapon recovery given a set of neighborhood-level and suspect features. The paper shows that Blacks and hispanics are still

stopped and frisked at a disproportionately higher rate than similarly situated whites. The paper also shows that by reducing the number of low "hit-rate" stops, which disproportionately affect minority communities, the same number of weapons can be seized while also reducing racial bias that may occur in policing tactics like stop and frisk. We hope to extend this study by examining more recent data from 2015-2019 after the 2013 legislation was passed and looking at all neighborhood-level/location characteristics, suspect characteristics (including age, race, as well as height, weight, and body type), and time factors (was the search conducting during the day/at nighttime, what day of the month, month of the year, etc.).

It is hard to analyze whether racial bias/discriminatory behavior influence a police officer's decision to conduct a frisk after a stop because of issues with omitted variables, so we assume that all stops resulted in a decision to frisk and by doing so we only focus on the outcome of the decision to frisk, which in this case is the probability of a weapon being found on the suspect after stop-and-frisk. This is a reasonable assumption because 83% of our data indicate that stops resulted in frisks. Further, we only consider stops where a pedestrian was stopped and frisked on suspicion of possession of a weapon. We then use a classifier (outcome-style-analysis) to predict the probability of actually finding a weapon for each of these stops. Stops that have a very low likelihood of finding a weapon violate the Fourth Amendment of unreasonable searches and seizures, and if these stops with low likelihood of finding a weapon can be accurately predicted we can enable police officers to avoid conducting these stops while still getting the same results.

Note that our analysis draws inspiration from Goel et al. (2016), but focuses more on machine-learning methods and applying different techniques to process data, select features, and perform the prediction [2].

METHODS

The Goel et al. (2016) study establishes a baseline showing that 43% of the stops conducted between 2009-2012 have a less than 1% likelihood of weapon seizure, and that the same number of weapons could be recovered by conducting only 6% of the total stops. However, because we used a newer dataset (2015-2019 data) we build our first classifier using Logistic Regression to form a baseline prediction for this new dataset. We used scikit-learn's implementation of Logistic Regression and started with a simple, non-regularized model that seeks to minimize the log loss. Given the lower number of stops (13,031) between 2015-2019 relative to the 301,513 stops that the previous study filters for, we did not feel the need to use stochastic gradient descent, which is suitable for very large models. Instead, we experimented with a few different solvers within scikit-learn and found that the liblinear solver, which uses coordinate descent, was most appropriate for our model. In the next iteration, we experimented with various regularization methods and optimized the regularization strength hyperparameter (described in the Setup section). We also experimented with dimensionality reduction using Principal Component Analysis (PCA) as a pre-processing step to the Logistic Regression model. While our data was not very high-dimensional to begin with (18 features), one-hot-encoding the categorical features significantly increased the dimensionality. Moreover, we anticipated at least a few features to be correlated to each other and therefore we were interested in exploring dimensionality reduction.

After a certain point, experimentation with various optimization methods for Logistic Regression did not result in any significant performance improvement. Therefore, we decided to experiment with a kernelized Support Vector Machine (SVM) model. We anticipated the possibility of our data not being linearly separable and therefore wanted to compare a linear SVM model with a Radial Basis Function (RBF) kernel SVM. We expect that an optimized linear SVM would provide a slight improvement compared to the Logistic Regression model because it reduces the risk of error by finding the optimal margin separating hyperplane. Additionally, we anticipated that the decision boundary of our data may be non-linear and therefore expected that an RBF kernel SVM model could potentially improve model performance. Furthermore, we expand upon the Logistic Regression and SVM models and introduce a Random Forest Classification (RFC) to further allow for better performance on. Approaching our problem conscientiously, we implement RFC to potentially improve the performance of our classification. RFC is considered to be one of the best statistical methods for large-scale classification. [4] The RFC model uses many decision trees, performs bagging on the dataset to improve error, and injects randomness by randomly selecting attributes to split on in its trees. Additionally, RFC does not require

rigorous preprocessing, which results in a simpler implementation.

Our decision to go from one model to the next is based on our goal to strengthen our model performance which determines the strength of our conclusions from this study. We repeat our analysis with SVM and RFC to measure performance against the baseline that we set with the Logistic Regression model. In the last section of this report, we compare performance with the three different models and use the best performing model to predict the likelihood of a weapon seizure. We believe that these results can subsequently be used by experts to consider the tradeoffs between missing someone with a weapon and Fourth Amendment violations to ultimately decrease the number of unnecessary stops in stop-and-frisk.

SETUP

To compute successful stops that result in weapon recovery, we look at 13,031 CPW stops between 2015 and 2019 to determine the probability that a suspect has a weapon at the moment a police officer decides to stop and search the individual (considering all the information that is available to the officer at the time) for criminal possession of a weapon.

Prior to setting up the machine learning models, we first clean the data from the NYPD SQF database by selecting important features based on literature reviews. While Logistic Regression and SVM required very similar pre-processing steps including one-hot-encoding categorical variables and standardizing numerical variables (using scikit-learn's Standard Scaler function), the RFC only required one-hot encoding the data. We had a few thousand rows where just one of the columns had a missing value, and therefore we experimented with two different methods for imputing these missing values - Simple Imputation (which uses the mean/mode of a column) and Iterative Imputation (which fits a linear regression by modeling each feature with missing values as a function of other features). The performance of the imputed data was benchmarked against the simple solution of dropping any rows with null values based on model performance. Additionally, we divided our pre-processed data set into training, validation, and test sets.

Our feature selection process was inspired by Goel et al. (2016). We include attributes such as suspect's attributes, location-specific features, and time-specific attributes which would impact the likelihood of a weapon being found. The location-specific features are include an indicator for the precinct in which captures both local crime rates as well as local enforcement standards for a given precinct [2]. We also include continuous variables for year, month, day, and the suspect's race, height, weight, age, and body build type, as well as the time for which the officer observed the suspect before stopping them.

METRICS We want to approach this with the goal of maximizing prediction accuracy—because of this, we think about evaluation metrics before diving into any modelling. An important consideration that guided our analysis was that our dataset was highly imbalanced in the outcome, i.e. only 17% of the data points belong to the positive class and the rest 83% of the data points belong to the negative class. Given this imbalance, accuracy (typical classification metric) was not an ideal metric for us to measure performance because the model could achieve a high accuracy by simply predicting the negative class for all persons. Since the aim of our model is to reduce the number of stops by accurately predicting the likelihood of a person carrying a weapon, we want to find a balance between minimizing false negatives (i.e. minimizing the number of people who have a weapon but are misclassified as not having a weapon) and minimizing false positives (i.e. minimizing the number of people who don't have a weapon but are misclassified as having a weapon. Expert knowledge is needed to evaluate the trade-offs between minimizing false negatives and minimizing false positives as these forces act in opposite directions in our model. We considered using AUC as our overall evaluation metric as it is a good measure of how well a model is able to distinguish between classes, but decided to not use it because of the clear disparity between the cost of false negatives and false positives. Instead, we used weighted accuracy as our overall guiding metrics for choosing hyperparameters as well as our overall evaluation metric. Weighted accuracy allows us to address the outcome imbalance by introducing weights that are inversely proportional to the number of data points for each class. The formula for weighted accuracy is shown below:

Additionally, we also looked at recall (ratio of number of observations classified as class 1 to the number of observations

$$\frac{\left[\frac{1}{frac\ of\ class\ 1} * num\ of\ TP\right] + \left[\frac{1}{frac\ of\ class\ 0} * num\ of\ TN\right]}{\left[\frac{1}{frac\ of\ class\ 1} * num\ of\ class\ 1\ observations\right] + \left[\frac{1}{frac\ of\ class\ 0} * num\ of\ class\ 0\ observations\right]}$$

that actually belong to class 1) and specificity (ratio of number of observations classified as class 0 to the number of observations that actually belong to class 0) while selecting class weights and assessing the classification threshold in an effort to understand the trade-off between minimizing false negatives and false positives respectively.

LOGISTIC REGRESSION Since our dataset was imbalanced and we wanted our model to prioritize class 1 prediction accuracy, we decided to adjust the class weights. We used the first iteration of the logistic regression model to find the optimal weights, which were inversely proportional to the number of observations for each class in the test set. Class 0 was assigned weight=1, while class 1 was assigned weight=5. We preferred to use this method as opposed to over-sampling the minority class as this method allowed us to use the full dataset and was easier to tune. After tuning the class weights, we iterated on the logistic regression model: starting with a non-regularized model and then experimenting with L1 and L2 regularization. After choosing the combination of the type of regularization and the solver that performed best on our dataset, we chose the optimal regularization strength by plotting a validation curve, and finding the regularization strength value that maximized validation weighted accuracy. Next, we experimented with PCA as a pre-processing step to reduce the dimensionality of the data. We passed in the standardized continuous features and the one-hot-encoded categorical features into the PCA model and plotted both the marginal and cumulative increase in variance explained as we increased the number of principal components. The results of the PCA are discussed in the next section.

Initially, we split our data as follows: 60% train, 20% validation, and 20% test. While we did not plot a learning rate curve for Logistic Regression, we found that increasing the size of the training set to 80% and reducing the size of the validation and test sets to 10% led to improved performance. Therefore, we decided to use the 80:10:10 split for the final Logistic Regression model as well as for the rest of the models.

SUPPORT VECTOR MACHINE The set up for the SVM models required the same data preprocessing steps as discussed above. Moreover, since we had already chosen the appropriate class weights, we focused the experiments for SVM models on improving weighted accuracy. We started with a linear SVM model to establish a baseline performance within the realm of SVM models (different from overall baseline - which is based on LR). Since we hypothesized that our data would be better separated by a non-linear boundary, we experimented with RBF and polynomial kernels. We found that the polynomial kernel SVM model was computationally cheaper than the RBF kernel model, but it had a lower accuracy. Therefore, we focused the rest of the experiments on the RBF kernel model. We experimented with the gamma and regularization strength hyperparameters in order to select optimal values that define the optimal decision boundary curvature and minimize overfitting (variance) respectively. We chose the final gamma value and regularization strength by plotting validation curves, and picking values that maximized the validation set's weighted accuracy.

RANDOM FOREST CLASSIFIER A Random Forest Classifier (RFC) is close to a machine learning model that mimics how a human would approach this problem. In our RFC, we choose to use the Gini impurity to decide the optimal split from a root node and subsequent splits. The most important hyperparameters of interest are the number of decision trees in the forest (n_estimators), the maximum depth of individual trees (max_depth), the minimum samples to split on at tree nodes (min_samples_split), the number of random features to include at each node for splitting, and the maximum number of leaf nodes (max_leaf_nodes). Due to the fact that our dataset classes are unbalanced, we set the class_weight parameter to "balanced" so that the classes are given different weights based on class frequencies. Firstly, the n_estimators hyperparameter represents the number of decision trees to be built. When this is larger, our aggregate model is more robust and model variance is reduced. However, the cost of training time increases, so we want to find a large enough value for n_estimators that is still computationally feasible. Secondly, the max_depth hyperparameter helps

the model by increasing more possibilities of combinations of features in the tree structure, which could potentially lead to some overfitting if it is too large, but should generally be feasible if it is a reasonable value with respect to the amount of features in our dataset.

The number of random features to consider at each split can help in minimizing bias in the model, as a higher number would help us to increase the chance that important features are taken into account when splitting at the nodes. This value is difficult to choose without experimentation, so we conduct hyperparameter tuning with grid search. We use GridSearchCV, a method that evaluates all combinations we define to tune our hyperparameters and optimize performance based on the development set's weighted accuracy. It runs through all the different parameters that are fed through the parameter grid, and produces the best combination of parameters. In our grid search, we fit 5 folds for each of 288 candidates, totalling 1440 fits. The hyperparameters we focused on are ones that have the greatest potential to improve performance: max_features, n_estimators, max_depth, min_samples_split, and min_samples_leaf.

OUTCOMES AND RESULTS

LOGISTIC REGRESSION Our baseline model was a non-regularized Logistic Regression model with equal class weights for both classes. Running this model yielded a weighted accuracy of 0.5742 on the training set and 0.5722 on the validation set. Adding in the appropriate class weights (as described in the setup) increased the weighted accuracy for the training set to 0.6542 and for the validation set to 0.6286. Note that we used the dataset where we dropped rows with any null values for these models. The alternative approach of imputing missing values using a simple mean/mode imputation as well as iterative imputation did not yield improved results. We anticipated that imputation did not help with improving model performance because a) the input data is not very systematic, and b) there were several missing values (20%) in 2 columns. Therefore we decided to use the dropped version of the dataset.

As we experimented with PCA to reduce the dimensionality of our data, we found that the data was not well explained in lower dimensions. The original data (after one-hot-encoding) had 118 features, and we found that we needed 80 principal components to explain 78% of the variance in the data, and 95 principal components to explain 90% of the variance (Figure 1). Running a logistic regression model on the dimensionally reduced data with 80 principal components yielded slightly worse results than the model with the non-reduced data (training set weighted accuracy - 0.6419 and validation set accuracy - 0.6209). Additionally, since adding a large number of principal components (within the ballpark of the original number of features) to accurately represent the data defeats the purpose of dimensionality reduction, we decided to not use the dimensionally reduced data. We found L1 regularization to be the most effective and chose a regularization strength of 1.612 (1/0.62) as that value maximized the validation weighted accuracy (as shown in the validation curve - Figure 2). Once the regularizer was optimized, the weighted accuracy for the training set was 0.6737, while the weighted accuracy on the validation set was 0.6446. After optimizing the hyperparameters, we experimented with changing from train:val:test split from 60:20:20 to 80:10:10, and while we did not see very large improvements, we found that the weighted accuracy improved by 2% on the validation set, thereby reducing bias to some degree. Therefore, we decided to use the 80:10:10 split for the final Logistic Regression model as well as for the rest of the models.

SUPPORT VECTOR MACHINE The results for the linear SVM model were comparable to the results for the Logistic Regression model but slightly better, with the training weighted accuracy being 0.6899, and the validation weighted accuracy being 0.6724. This model established our baseline performance under the SVM realm of models, and we further experimented with an RBF kernel SVM model, and optimized it's hyperparameters - gamma and regularization strength - as described in the previous section. The optimal gamma was found to be 0.0615, which seems to be reasonable and in-line with the general heuristic that gamma should neither be too small (so as to make the model too constrained) and neither too big (so as to lead to overfitting). The optimal regularization strength was 1, which means that the model was not overfitting to begin with. Figure 3 shows the validation curves for both the metrics. Since both the training and validation weighted accuracy was increasing as the value of the regularization parameter increased (i.e, regularization strength decreased), we thought it was reasonable to not regularize our model.

RANDOM FOREST CLASSIFIER The results from LR and SVM reinforce our initial belief that RFC might improve overall performance. The results of the RFC show a high weighted accuracy for the training set (0.8025) and slightly lower weighted accuracies for validation and test sets, indicating that there is overfitting in the model despite our hyperparameter tuning through grid search. Furthermore, we also manually reduced the depth of the trees, a common reason for overfitting, and found that the scores were not improving. We found that the values that led to the highest weighted accuracies for our validation set were where max depth = 15, the maximum number of features to look at each split (max_features) = 25, the minimum number of samples at each leaf node (max_samples_leaf) = 1, minimum number of samples to split a node = 3, total number of trees in the forest = 300, and with balanced class weights.

We also plotted a confusion matrix using all optimal RFC hyperparameters on the right, and a confusion matrix for an RFC with the same parameters but without balanced class weights on the left, to allow for us to better understand our classification model's performance and how class weights may improve our model (Figure 4).

The confusion matrix gives us insights into the recall and specificity of the RFC model. We see that the majority of people who do not have a weapon are also not classified as having a weapon in both models. Additionally, if our model is implemented, only 8% of people who do not have a weapon are being misclassified as having a weapon (with class weights) or 0.46% (without class weights). In comparison to previous studies, this is a significant improvement as 43% of people who were previously being stopped had a less than 1% likelihood of having a weapon using police judgement. [2] These are all generally still good indicators, as it implies our model would generate fewer false alarms if it were to be deployed in real life as a decision maker. However, we notice that 69% of people in our optimal classification model who do have a weapon are being mislabelled, and our recall is 31%, which is quite low. Arguably, this is one of our most important metrics to maximize, as we do not want to let any suspects go who are actually a threat to safety. We know that this is due to an imbalanced data set as well as a limited amount of data. However, by comparing the recall to the confusion matrix without balanced class weights, we see a 21% improvement. Our model with balanced class weights thus is not as biased towards classifying instances towards the majority class (innocent suspects).

To qualitatively extend beyond our many iterations, we also wanted to gain insight into the specific features that are most important for our classification in RFC (Figure 5), giving us a better understanding of which features were more useful when building the model. We utilized the mean decrease impurity (Gini importance), which calculates how much each feature decreases the Gini impurity of splits over all trees in the forest. Rather than focusing on the absolute values of the feature importances, this method allows us to interpret relative values of computed importances. The drawback of this method is that it tends to select numerical features and categorical features with high cardinality, but it relatively gives us a sense of what features are relevant to our model, and helps us to add a qualitative layer to what may potentially connect to a police officer's decision to stop-and-frisk someone.

OVERALL RESULTS The table in the Tables section shows the most optimal model and its corresponding weighted accuracy on the train, validation, and test sets for each classification algorithm that we modeled.

After iterating through various experiments with each of the models, we found that the RFC model had the highest weighted accuracy on the validation and test sets. While this was not a huge improvement compared to the optimal LR and SVM models, we think the RFC model works best on our data because it is non-linear and is able to capture the 'decision boundary' better than a linear model. Additionally, since RFC uses feature bagging, which helps to decrease correlations between decision trees, it helps to increase the average accuracy of predictions. We found that even the best performing model reached a performance ceiling after a certain number of experiments, as we were not able to improve performance beyond 73.81%. We did experiment with additional feature engineering by trying to find relationships between features, as well as by dropping certain sets of features, but those steps did not improve performance. Based on our explorations, we think that improving performance beyond this point would either require additional data or additional features. Given that the problem we are trying to solve—predicting whether someone has a weapon or not based a suspect's attributes, their location, and the manner in which they are perceived by the police—is very complex and nuanced, we think that our features do not capture the picture very well and that our models suffer from omitted variable bias. Perhaps having more domain-specific knowledge about policing and the heuristics that police officers use

when stopping someone on account of weapon possession would help with additional feature engineering or feature selection.

While the weighted accuracy of our model is only 73.81%, we think our model is still useful in achieving the goal of reducing the number of unnecessary stops because we have the ability to adjust the class weights as well as the classification threshold depending on how we prioritize recall and specificity. For instance, if we were to find the optimal balance between the two, we would use the 1:5 ratio for class weights (class 0: class 1). On the other hand, if we were to optimize only for recall, i.e. to minimize the number of false negatives, such that we almost never misclassify a person with a weapon as someone without a weapon, we can increase the class weight for class 1 or decrease the classification threshold (Figure 6). While we are not qualified to make the decision about the optimal class weights or threshold in line with the recall-specificity trade-off, we believe that our model could be used by a domain-expert such as a judge or a senior police officer to make this decision, and ultimately reduce the number of stops.

Stop-and-frisk is a highly contentious topic and the implications of a machine learning algorithm misclassifying an innocent person as "likely to carry a weapon" can be dangerous. This policing tactic has been heavily criticized for being discriminatory against Black and hispanic comminities only for a marginal reduction in crimes. [5] On the one hand, machine learning algorithms, such as the one we are proposing, have the potential to remove human bias completely from the decision to stop-and-frisk and the promise of an improved and efficient decision-making process. On the other hand, actually deploying these predictive algorithms in policing can encourage racial profiling—after all the algorithms learn from data that over-represent Black and hispanic populations and in some capacity are the result of (potentially discriminatory) decisions made by the police. Improving stop efficiency will not only reduce Fourth Amendment violations but also help temper the public's reaction to policing strategies like stop-and-frisk, including distrust and resentment of the police.

REFERENCES

[1] "Stop-and-frisk in the de Blasio Era." New York Civil Liberties Union. March 2019.

[2] Goel, S., Rao, J. M., amp; Shroff, R. (2016). "Precinct or prejudice? Understanding racial disparities in New York City's stop-and-frisk policy." The Annals of Applied Statistics, 10(1). https://doi.org/10.1214/15-aoas897

[3] Gelman, Andrew, Jeffrey Fagan, and Alex Kiss. "An Analysis of the New York City Police Department's "Stop-and-Frisk" Policy in the Context of Claims of Racial Bias - Columbia University." Accessed November 14, 2021. http://www.stat.columbia.edu/ gelman/research/published/frisk9.pdf.

[4] Fernández-Delgado, M., Cernadas, E., Barro, S. And Amorim, D. (2014). "Do we need hundreds of classifiers to solve real world classification problems?" J. Machine Learning. Res. 15 3133– 3181. MR3277155

[5] Matteo Tiratelli, Paul Quinton, Ben Bradford, "Does Stop and Search Deter Crime? Evidence From Ten Years of London-wide Data", The British Journal of Criminology, Volume 58, Issue 5, September 2018, Pages 1212–1231, https://doi.org/10.1093/bjc/azx085

TABLES

Model	Hyperparameters	Training Set Weighted Accuracy	Validation Set Weighted Accuracy	Test Set Weighted Accuracy
Logistic Regression (L1 Regularization)	Regularization parameter = 0.62, class_weight = 'balanced'	0.6937	0.6606	0.6789
SVM with RBF kernel	Gamma = 0.0615, Regularization parameter = 1, class_weight = 'balanced'	0.7821	0.7079	0.7367
Random Forest Classifier	max_depth=15, max_features=25, min_samples_leaf=1, min_samples_split=3, n_estimators=300, n_jobs=-1, class_weight='balanced'	0.8025	0.7192	0.7381

FIGURES

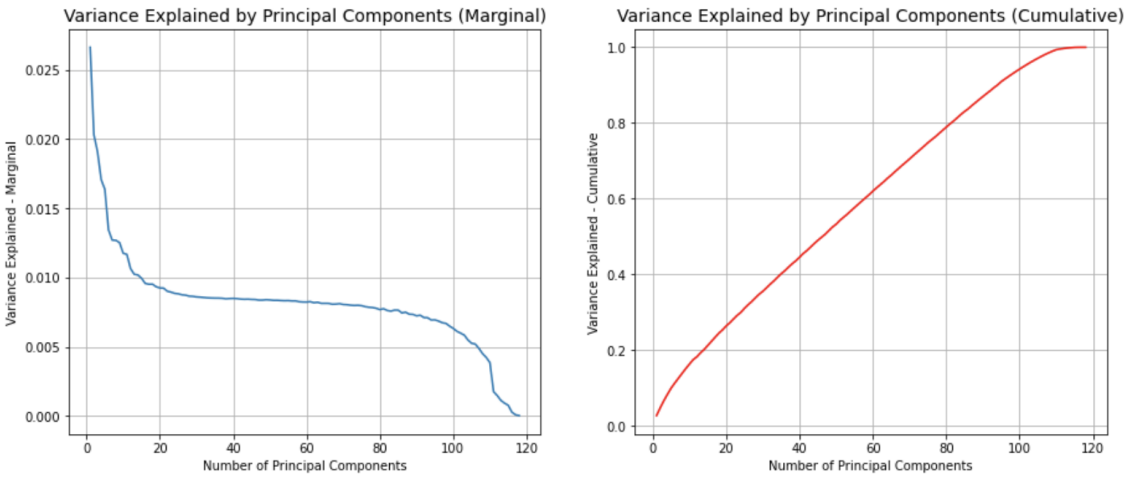

Figure 1: Plot of variance explained using different number of principal components

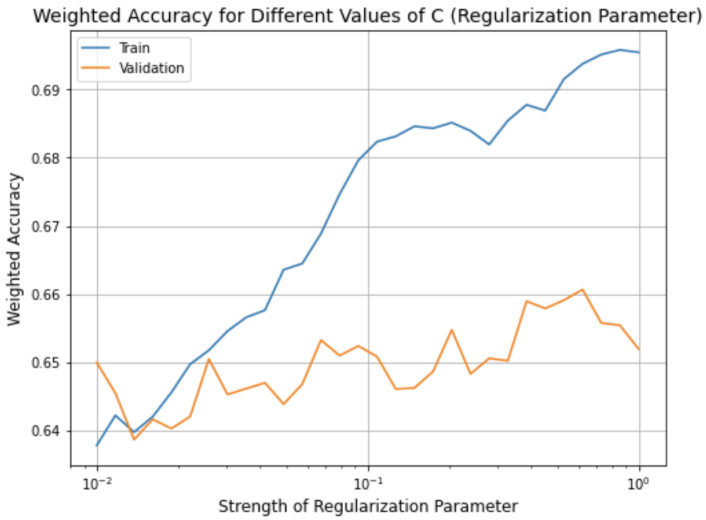

Figure 2: Validation curve showing regularization strength and weighted accuracy

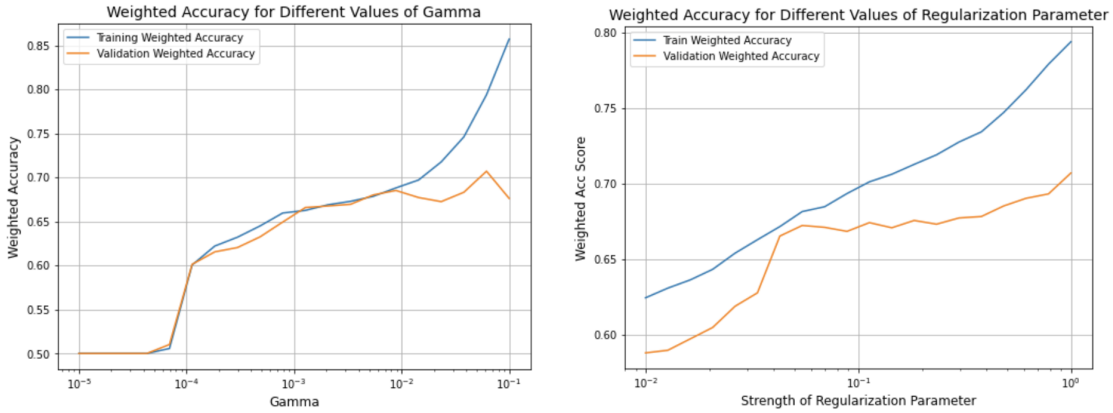

Figure 3: Validation curves showing hyperparameter selection for RBF kernel SVM model

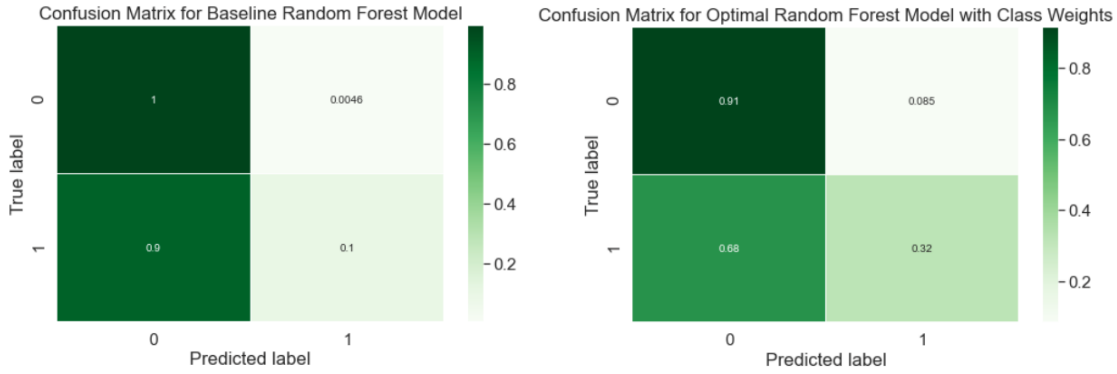

Figure 4: Confusion matrix for baseline and optimized RFC models

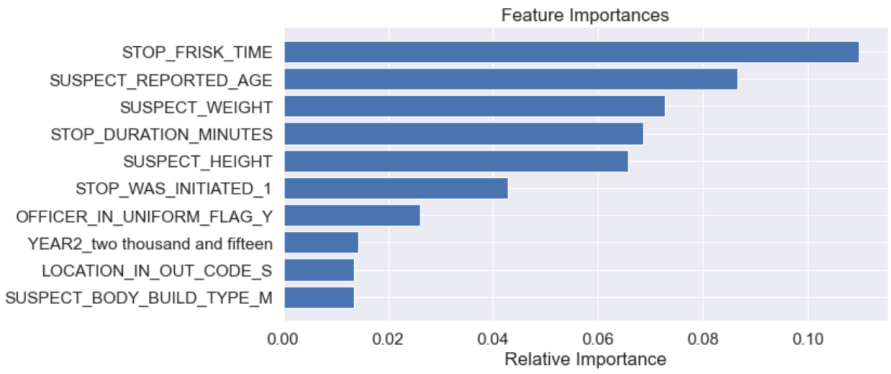

Figure 5: Feature importance graphs from RFC model

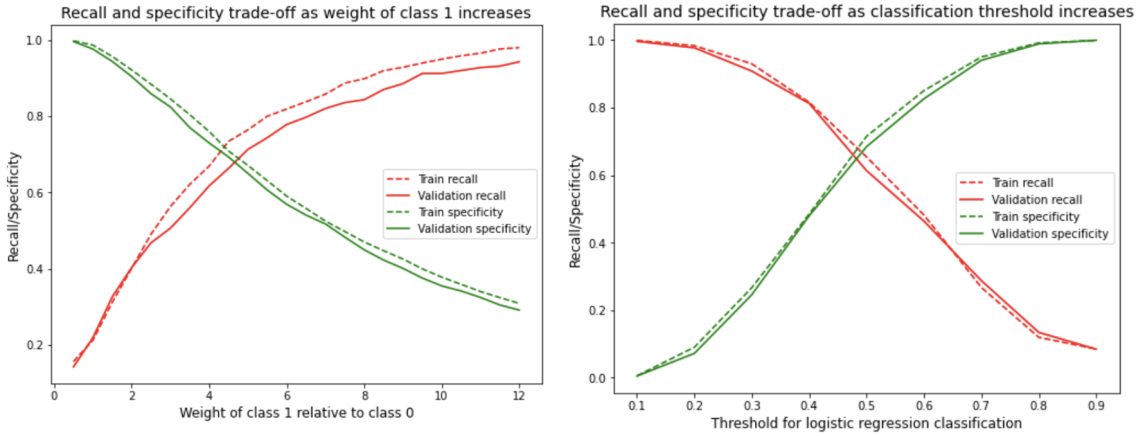

Figure 6: Recall-specificity trade-off with respect to classification threshold and class weights

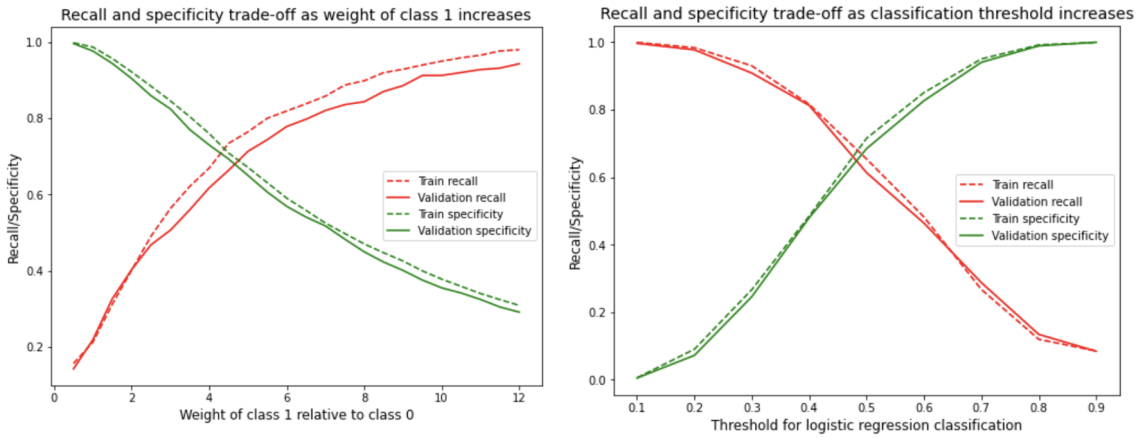

Figure 7: Recall-specificity trade-off with respect to classification threshold and class weights

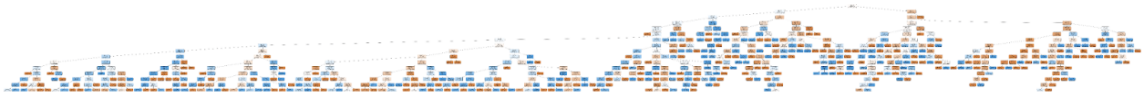

Figure 8: Visualization of a tree in the RFC (we be-leaf in our model!)