Business Conclusions

Summary

In this report, we present an assessment of our deployed model based on the data it has collected during the deployment week. As we have anticipated in the previous report, model performance was very similar to the values we had achieved in the test set, both the dynamic performance and the score values at the model operating point. Although we have implemented the most basic approach to deal with the fairness requirement, the deployed model actually significantly overcomes what would have been the officers' fairness performance among all of the three protected classes. In this regard, we also suggest an alternative model that would further slightly improve the fairness behavior of the model. Nevertheless, further developments to address this concern should be attempted at a later phase, as suggested in report 1.

Results Analysis

Model Performance

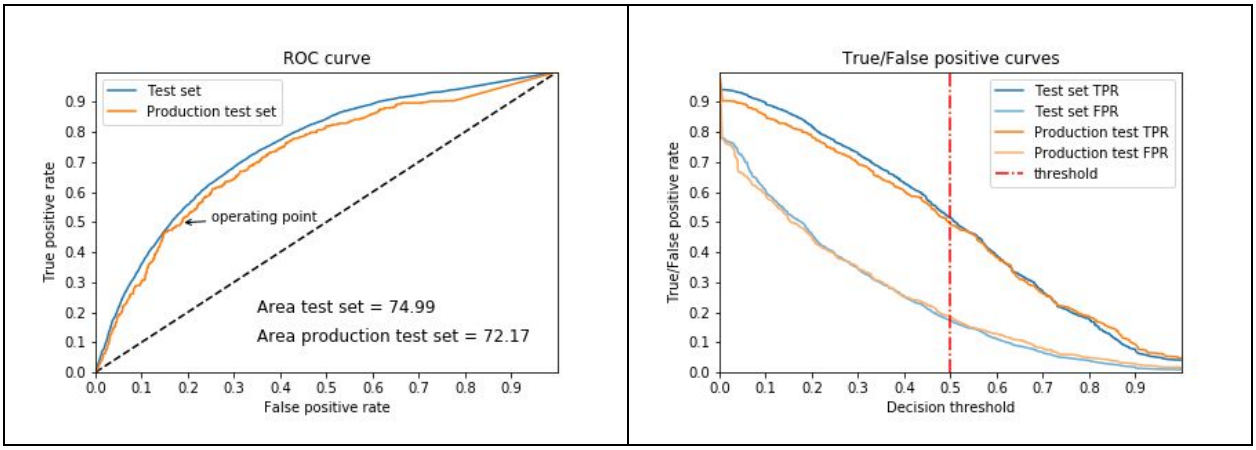
In this section, we compare the model performance on the test set with its actual performance on production (naturally, only for those observations for which we know the true class - we have named this subset "production test set"). The results are presented in the table below and officers' performance was also included in the last column as a baseline (i.e. what would have been the performance in case the model had not been deployed and all searches were performed). The recall is the most significant metrics to measure the model performance, as defined in the third requirement of the briefing. The precision score is a byproduct of setting the classification threshold at 50%, according to our interpretation of the first requirement. We also present the percentage of searches performed, as a measure of the cost-effectiveness of the model, and accuracy, since it is a standard metrics of common usage.

	Model performance		Officers' performance
	Test set (40% of the training set)	Production test set	Production test set
Recall	0.5144	0.4942	1.0
Precision	0.5998	0.6005	0.359
Searches [% of total]	29%	30%	100%
Accuracy	0.7231	0.7004	0.359

As we have anticipated in report 1, performance in production remained very close to its figures on the test set. Two main reasons led us to believe the model performance wouldn't significantly change in production:

1) Since the search selection process was going to be maintained, the first selection of subjects by the police officers, and only then a search decision provided by the model, we didn't expect the population characteristics to change much.
2) The test set on which the model performance was assessed before deployment had a considerable size (40% of the training set, amounting to 30,697 observations), which gave us confidence in the accuracy of the performance values on this set.

As we have also argued in report 1, because the first requirement of the briefing led us to define a specific classification threshold, the model dynamic assessment becomes less relevant in this case. Even so, on the figures below we compare the dynamic behavior of our model (ROC curve and True/False positives curves) on both the test set and production test set.



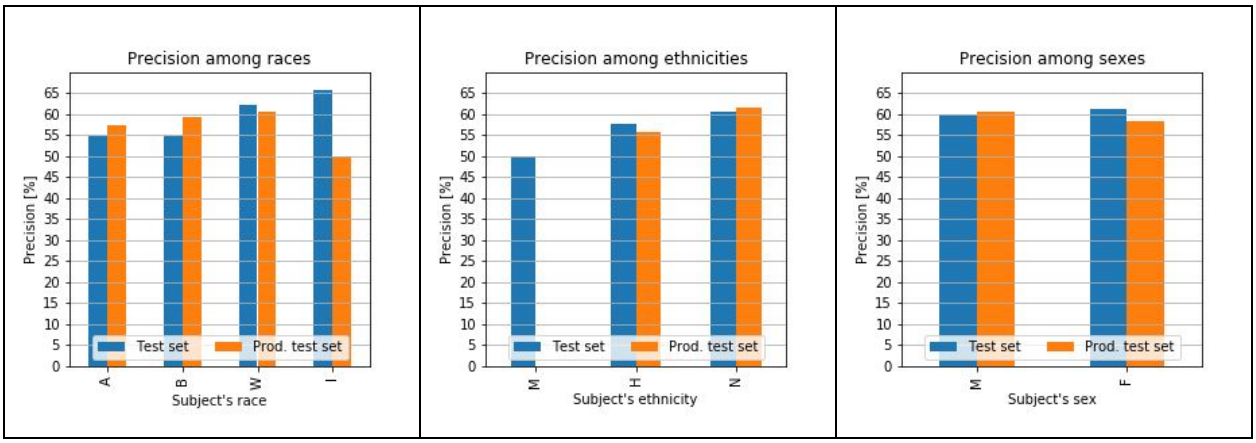
Fairness

In report 1 we have presented and discussed the question of fairness in the model decision and we have accepted the approach proposed by the client in the briefing: aim at limiting the precision score discrepancy between protected classes to 'acceptable' levels. We have also suggested two adjustments to this criterion.

1) To measure the 'acceptable' level in relative terms (i.e., maximum difference of 5 percent), instead of using an absolute value (5 percentage points).
2) To measure the discrepancy at a global level, instead of measuring it at a department level, as has been requested.

In this section, we assess the model behavior in terms of fairness, as defined by the options and metrics that have been chosen, and compare it with the expected results, based on its performance on the test set.

In the plots below, we present the model precision score among protected classes. We start by noticing that there are only 46 observations with class ‘Asian/Pacific’ and 10 with class ‘Indian American’ for which we know the true class. Among these, 14 ‘Asian/Pacific’ and 4 ‘Indian American’ individuals have received a positive prediction by the model. This means that it is prudent to disregard these classes when assessing the fairness requirements since we could have an extreme value for the precision among these classes just by chance (as indeed happens with ‘Indian American’). We also notice that there is no observation belonging to the class ‘Middle Eastern’, whereas the training set had 876 individuals.



From the plots, we also notice that the model seems to have improved its behavior in terms of fairness when compared to the test set (except among different ethnicities), since the classes previously with higher precision score have decreased this metrics and the classes previously with higher score have increased it, reducing in this way the discrepancy between classes.

In the following tables, we repeat the information presented in the plots, so that we can evaluate the exact values and compute the discrepancies (3th and 5th columns of the tables). We also compare our model behavior in terms of fairness with the officers’ performance in the same sets (i.e. what would have been the fairness behavior if the model had not been in operation and all search requests were performed).

	Precision score in the training set for feature *SubjectRaceCode*		Precision score in the prod. test set for feature *SubjectRaceCode*	
	All searches (complete set)	Our model (just test set)	All search requests	Our model
‘White’ [%]	34.68	61.99	36.75	60.48

‘Black’ [%]	29.97	54.82	34.51	59.29
‘Asian/Pacific’ [%]	29.11	54.76	n.a.	n.a.
‘Indian American’ [%]	29.54	65.71	n.a.	n.a.
Max. difference [%]	16.05	16.67	6.10	1.97
Max. difference [p.p.]	5.58	10.95	2.24	1.19

	Precision score in the training set for feature *SubjectEthnicityCode*		Precision score in prod. test set for feature *SubjectEthnicityCode*	
	All searches (complete set)	Our model (just test set)	All search requests	Our model
‘Not applicable’ [%]	34.99	60.64	37.98	61.38
‘Hispanic’ [%]	27.71	57.64	29.84	55.75
‘Middle Eastern’ [%]	27.97	50.00	n.a.	n.a.
Max. difference [%]	20.82	17.55	21.43	9.18
Max. difference [p.p.]	7.28	10.64	8.14	5.63

	Precision score in the training set for feature *SubjectSexCode*		Precision score in the prod. test set for feature *SubjectSexCode*	
	All searches (complete set)	Our model (just test set)	All search requests	Our model
‘Male’ [%]	33.54	59.73	36.97	60.55
‘Female’ [%]	31.96	61.13	31.87	58.20
Max. difference [%]	4.70	2.29	13.80	3.88
Max. difference [p.p.]	1.57	1.40	5.10	2.35

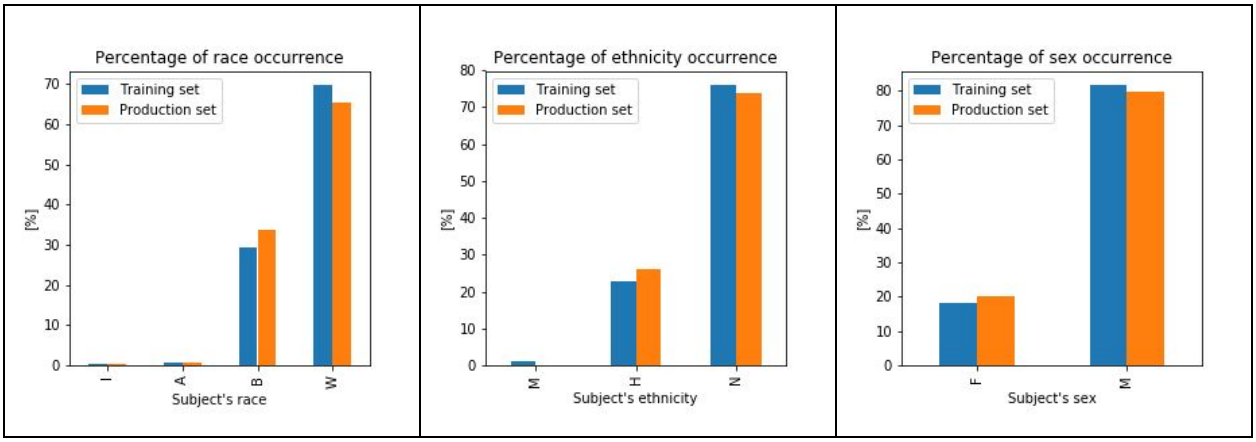
In the production test set, our model significantly improves what would have been the officers’ performance among all of the three protected classes. Our model implements the same decision threshold to all subjects, since it is blind (or unaware) to the protected classes. Officers are not, and this is one possible justification for the improved behavior attained by our model, which was also somehow visible on the test set (especially when we consider the discrepancy in

relative terms). When compared to the test set, the production results are quite similar. On the one hand, we have a clear improvement among *SubjectRaceCode* classes, but on the other hand, we have a slight deterioration among *SubjectSexCode* classes, although below the required discrepancy limit. Regarding *SubjectEthnicityCode*, the discrepancy among classes exceeds the required limit and even if the model seems to have improved its fairness behavior in production compared to the test set this is not the case when we disregard 'Middle Eastern' ethnicity, as we did for the production test set.

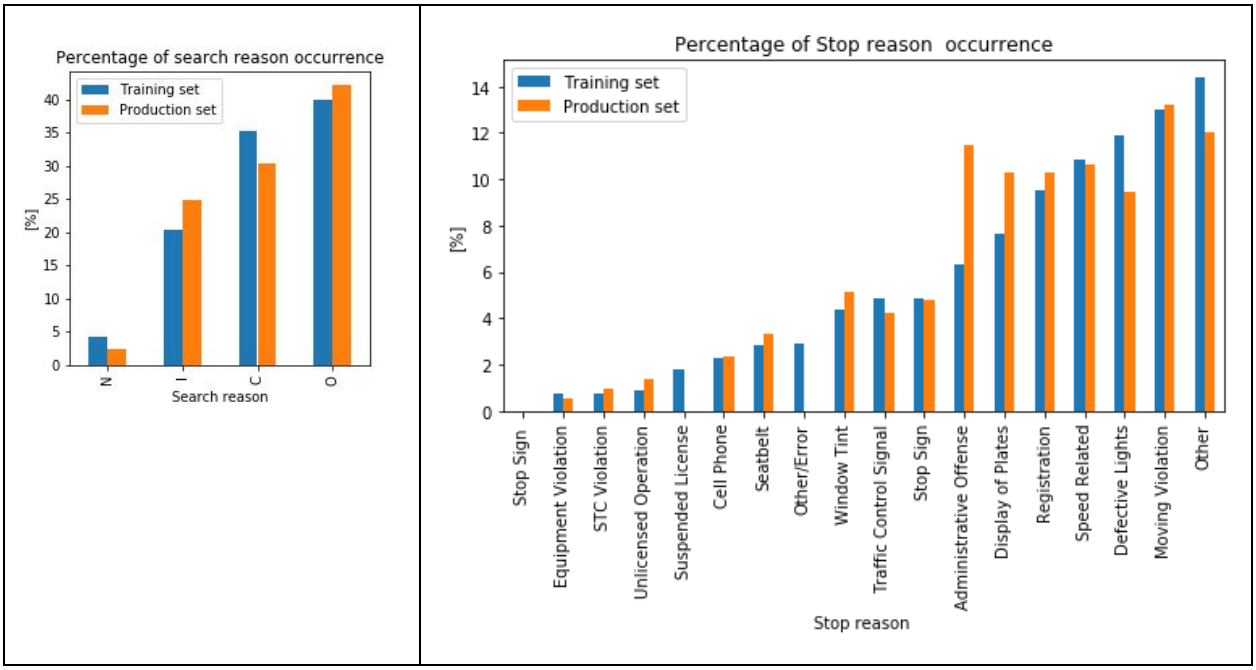
As we have mentioned, we have limited ourselves to the most basic approach in terms of fairness: the model is blind to protected classes and uses the same threshold value for all subjects. In report 1 we have argued this was not an ideal solution, we proposed alternatives to address this limitation in future model developments, but we were not able to provide a better model until the delivery deadline. In the section Next Steps of the present report, we will explore one alternative for improvement and assess the fairness performance of the improved model.

Population Analysis

In the three plots below we show the population distribution among protected classes comparing the training set (in blue) with the production set (orange). With the naked eye, we can see that the characteristics of the population don't differ significantly between the two sets when we consider these classes, or at least their presence in the training and production sets is quite similar. From a statistical perspective, however, we can't accurately say that the population is the same in both sets. To test this hypothesis - the population having the same characteristics in both sets - we may regard an observation's class, for example being 'Male', as a Bernoulli trial, i.e. for a given observation there is a probability p of being 'Male' and a probability $1-p$ of not being 'Male'. In this framework, the number of males in each set forms a Binomial distribution, and we can test if the probability p of being 'Male' is the same on both sets. This is done in the Annexes (for 'Male', 'Black', and 'White' as examples), and the result, with a great level of statistical significance, is that it is not, the probability of 'Male', or 'White', or 'Black' is different in each of the two sets. The same is true for the remaining classes (to avoid being exhaustive we limit the results to these classes). So, our conclusion is the following: although the populations in the training set and in the production set are very similar (as can be seen just by looking at the plots), from a statistical perspective we cannot say that it is the same population (since the probability for the occurrence of each class is different in the two sets, with a high level of statistical significance).

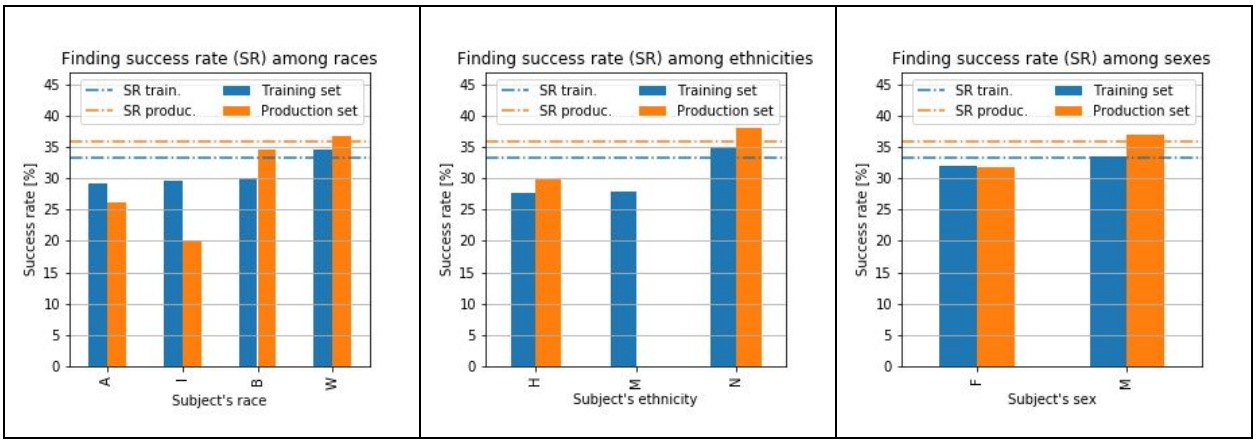


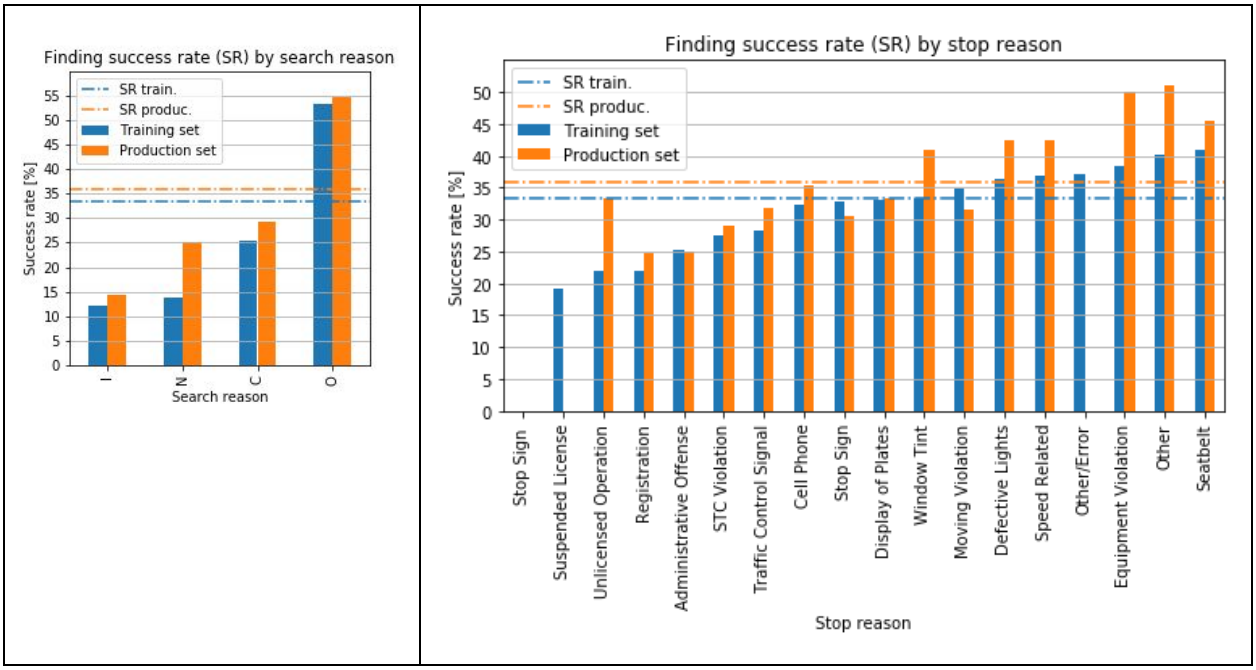
In this section, we think of population analysis in an extended sense: not only individual characteristics, which are relevant for fairness assessment, but also the distribution of the features used by the model, which may impact its performance. Thus, we replicate the previous plots for those features that have been included in the model. Below we show the plots for *SearchAuthorizationCode* and *StatuteReason* (to avoid being exhaustive we limit the results to these two features).



Based on the report 2 guidelines we believe the previous analysis was expected. However, both for fairness and performance requirements, we don't think this is the most significant analysis to be done. Actually, we don't see how the population distribution is truly relevant for our purposes. We think that the prevalence of contraband among each of the features is the aspect that could have compromised both the performance and fairness requirements of our model, since the metrics that have been chosen - precision and recall - would be affected if the presence of

contraband among each feature value had changed (i.e. if the probability of contraband conditional on the feature had changed), but not if the distribution of those feature values happened to be different between the train and production sets (or at least we don't see how). In other words, even if the occurrence of different classes or values in a given feature changes significantly between the training and the production sets (i.e. even if the orange and blue bars in the plots above were very different), as long as the probability of finding contraband within each feature value doesn't change much, then we see no reason for or model not to preserve its performance (both from a fairness and a contraband finding perspective). Some examples of this alternative analysis are presented on the plots below.





Next Steps

In the previous report, as a way to improve the fairness behavior of the model, we have suggested a dynamic decision threshold instead of the fixed one at a 50% probability that we ended up implementing in the deployed model. This approach would have to be implemented in the web application since it requires an evaluation of the previously classified observations. As a matter of fact, there is an intermediate alternative between these two: we can define different decision thresholds for the different protected classes, to compensate for the discrepancies in terms of fairness metrics that we have found on the training set. The process would be as follows:

- ➔ Tuning the thresholds on the training set. For each protected class (*SubjectRaceCode*, *SubjectEthnicityCode*, and *SubjectSexCode*) repeat these steps:
  - ◆ Star with the same threshold of 50% for every element of the class (e.g. all races in *SubjectRaceCode* or all genders in *SubjectSexCode*). In report 1 we have argued that a minimum threshold of 50% was needed in order to meet client's first requirement.
  - ◆ Make the predictions for the training set using the 50% threshold and then compute the precision for each element of the class.
  - ◆ For the element of the class with higher precision leave the threshold at 50%.
  - ◆ For the remaining elements, progressively increase the threshold until the model's predictions on the training set result in similar precision values for all elements of the class.
  - ◆ Save the threshold values.
- ➔ Defining the threshold for new predictions. For each observation select the three corresponding threshold values based on the features *SubjectRaceCode*, *SubjectEthnicityCode*, and *SubjectSexCode*. Use the higher of those three threshold values to make the prediction.

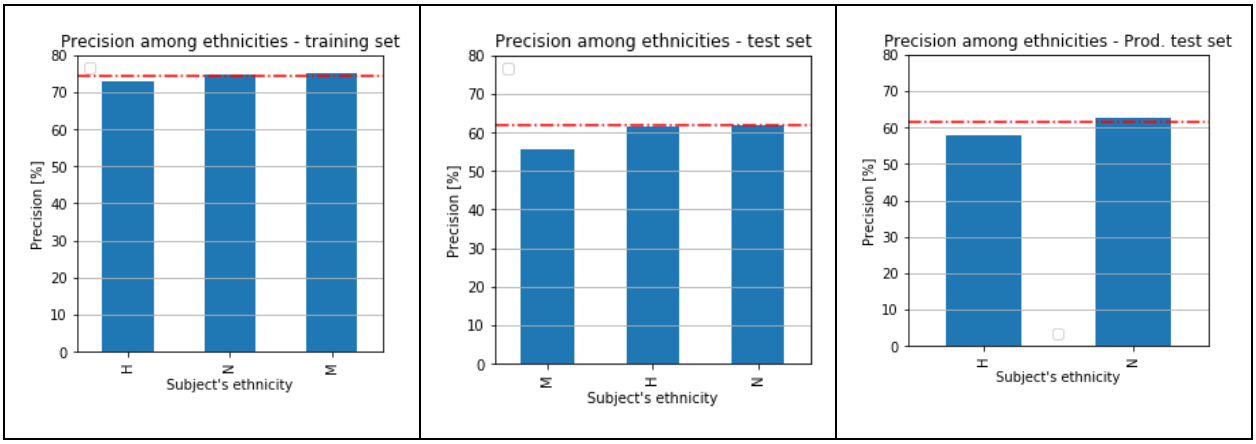
After tuning the thresholds on the training set we got the values described in the table below.

SubjectRaceCode thresholds [%]	
'W'	50.0
'B'	58.5
'A'	61.0
'I'	83.0

SubjectEthnicityCode thresholds [%]	
'N'	50.0
'H'	59.0
'M'	58.5

SubjectSexCode thresholds [%]	
'M'	50.0
'F'	52.5

In the plots below we show the fairness behavior of this new model in the training set, in the test set and in the production test set. For simplicity we present just one of the classes, *SubjectEthnicityCode*.



As expected, overall precision (red bar) is significantly higher on the training set, because we are making predictions for the data where the model was trained. In the same way, precision among different ethnicities also presents lower discrepancy on the training set because this was the set where thresholds were tuned. Notice that, even on the training set, precision among different ethnicities is not exactly the same. This happens because the model doesn't necessarily uses the threshold value that was tuned to reduce discrepancy among different ethnicities, it will use the higher threshold applicable to the specific observation. In the tables below we compare this new model performance in terms of fairness with our deployed model, both on the test set and on the production test set.

	Precision score in the test set for feature *SubjectRaceCode*		Precision score in the prod. test set for feature *SubjectRaceCode*	
	New model	Deployed model	New model	Deployed model
'White' [%]	63.34	61.99	61.46	60.48
'Black' [%]	56.46	54.82	62.05	59.29
'Asian/Pacific' [%]	59.02	54.76	n.a.	n.a.
'Indian American' [%]	71.43	65.71	n.a.	n.a.
Max. difference [%]	20.96	16.67	0.66	1.97
Max. difference [p.p.]	14.97	10.95	0.41	1.19

	Precision score in the training set for feature *SubjectEthnicityCode*		Precision score in prod. test set for feature *SubjectEthnicityCode*	
	New model	Deployed model	New model	Deployed model
‘Not applicable’ [%]	61.83	60.64	62.64	61.38
‘Hispanic’ [%]	61.53	57.64	57.78	55.75
‘Middle Eastern’ [%]	55.73	50.00	n.a.	n.a.
Max. difference [%]	9.87	17.55	7.76	9.18
Max. difference [p.p.]	6.1	10.64	4.86	5.63

	Precision score in the training set for feature *SubjectSexCode*		Precision score in the prod. test set for feature *SubjectSexCode*	
	New model	Deployed model	New model	Deployed modell
‘Male’ [%]	61.45	59.73	62.32	60.55
‘Female’ [%]	63.10	61.13	58.97	58.20
Max. difference [%]	2.61	2.29	5.36	3.88
Max. difference [p.p.]	1.65	1.40	3.35	2.35

The results seem to improve slightly with the new model. On the production test set this is the case for all features except *SubjectSexCode*, which suffers a small increase in the discrepancy. On the test set the performance is more similar between both models, even if the new one greatly improves the performance for *SubjectEthnicityCode*. Anyway, the results in terms of fairness remain far from brilliant and the model would require further improvements.

Deployment Issues

Re-deployment

We have considered the possibility of re-deploy the model with an improved version, especially in terms of the fairness requirement that we were not able to address. However, we don’t think that would have been the most honest approach. We got to the best possible model within the available time and we think it would have been a bit unfair to change it during the deployment

week, unless we had faced some unexpected behavior (e.g. web app error) or if the model was performing in a way significantly different from what we were expecting and had presented in the first report. We believe this report is the appropriate moment to suggest changes, as we have done in the section Next Steps. Besides, as it has been shown, the proposed improvement didn't result in a significantly better fairness performance, and for a re-deployment to be worth we would have to come up with something different. Additionally, once we had a working model running on heroku, any re-deployment would present some degree of risk and of losing observations.

Unexpected problems

During the deployment week everything went smoothly and we didn't encounter any unexpected problems or unexpected data. In the days before deployment, however, every problem was unexpected, because it was the very first time we were working with a web application and it revealed to be something not easy to understand in some hours or a couple of days. We needed to spend 4 or 5 days working on the app before we had a working model, this was the greatest unforeseen problem. Additionally, package 'gunicorn' included in the Anaconda environment (file 'environment.yml') that was not available for Windows operating system. This little detail was the cause of a lot of errors and a big headache.

What would you do differently next time

Next time I will be working on a linux operating system.

Annexes

Considering a class, e.g. being 'Male' or being 'White', the presence of this class on a given observation may be interpreted as Bernoulli trial with probability p, i.e. the class occurs in the observation with probability p and does not occur with probability $1-p$. Repeating the trial n times, the number of occurrences of this class becomes a Binomial distribution X with parameters n and p. That is, the number of occurrences of this class, X, is a random variable represented as

$$X \sim Binomial(n, p).$$

If we denote the training set with the subscript 1 and the production set with the subscript 2, then the occurrence of this class in each of the sets is given by

$$X_1 \sim Binomial(n_1, p_1)$$

and

$$X_2 \sim Binomial(n_2, p_2),$$

where n_1 is the size of the training set and p_1 is the probability of class occurrence for each of the observed outcomes of the training set (equivalent for n_2 and p_2 in the production set).

The most natural estimator for p_1 (actually, the maximum likelihood estimator) is

$$\hat{p_1} = \frac{X_1}{n_1}$$

and the same for $\hat{p_2}$.

Having computed the estimates $\hat{p_1}$ and $\hat{p_2}$ we can test the hypothesis that p_1 is equal to p_2, i.e. that the probability of class occurrence on the training set is the same as in the production set. We will test

$H_0 : p_1 = p_2$ against $H_1 : p_1 \neq p_2$.

With this purpose in mind, we take advantage of the fact that when the sample size n is large enough, as we consider to be the case, then the Normal distribution can be a reasonable approximation to the Binomial distribution of X, i.e.

$$X \sim \mathcal{N}(np, np(1-p)).$$

Using this approximation, the distributions for X_1 and X_2 become

$$X_1 \sim \mathcal{N}(n_1 p_1, n_1 p_1 (1-p_1)),$$

$$X_2 \sim \mathcal{N}(n_2 p_2, n_2 p_2 (1-p_2))$$

and, under the null hypothesis H_0 that $p_1 = p_2$, we can build the test statistic TS which follows a standard normal distribution

$$TS = \frac{\hat{p_1} - \hat{p_2}}{\hat{p} \cdot (1-\hat{p}) \cdot (1/n_1 + 1/n_2)} \sim \mathcal{N}(0,1)$$

where

$$\hat{p} = \frac{n_1 \hat{p_1} + n_2 \hat{p_2}}{n_1 + n_2}.$$

Now, for each class, we just have to compute the value of TS (based on the occurrence of this class in the training and test sets) and see how likely this value is to happen under the null hypothesis (which led us to accept the standard normal distribution of TS). If it is very unlikely, then we reject the null hypothesis and accept that $p_1 \neq p_2$.

Class	$\hat{p_1}$	$\hat{p_2}$	TS	$p-value$	Test result
'Male'	0.8168	0.7986	4.4135	0.000	Reject H_0
'White'	0.6974	0.6534	8.9766	0.000	Reject H_0
'Black'	0.2932	0.3362	-8.8417	0.000	Reject H_0

Definitions:

Historical dataset: complete dataset collected by the police department and provided with the briefing.

Training set: Subset of the historical dataset containing observations for which feature *VehicleSearchedIndicator* equals 'True'.

Test set: Subset of the training set that was used to assess model performance.

Production set: complete dataset resulting from collected data by the web app during the week the deployed model was in production.

Production test set: Subset of the production set for which we know the true outcome.