Business Conclusions

Summary

Our model for predicting if the vehicle has contraband evidence has been in production for one week. We had 1000 requests to our API made by Police Officers in order for them to know if they should search the vehicle or not. We received back the true outcome for half of these observations.

While our model was in production for the past week, we started suspecting that we were just receiving suspicious interventions because SearchAutorizationCode was almost never N, and by our interpretation, N is the value that appears in records that Police Officers don't think that the intervention is suspicious because there are just a few N values for this variable in searched vehicles. This is no surprise, we actually were expecting that this could happen, that Police Officers just send a request to our API when they have any kind of suspicious about the intervention. But having that in mind, we decided to implement a model that was "Bulletproof" so if police Officers start sending requests to our API with no criteria it still works without saying that interventions are suspicious by default.

Looking at the true outcome, we conclude that our model has the tendency to predict that the vehicle has contraband evidence. This happens because we are just receiving suspicious interventions and because we trained the model with suspicious and no suspicious data, the model has the tendency to classify suspicious interventions as having contraband evidence.

Our results were very different between our test set and data in production because of the mentioned reasons. The train and test set had suspicious and no suspicious interventions while our production data just had only suspicious data.

It is hard to build a model that is "Bulletproof" because it will not work well only on suspicious data. Our model is guessing if the intervention is suspicious or not, and that is the job of Police Officers. Our Api should help in the decision of going further in the searching process and not to do their job. So, instead of going to production with a "BulletProof" model that is trying to do Police Officers' job and doesn't work so well only with suspicious data, we should assume that Police Officers did their job well and build a model only trained with suspicious interventions that actually help Police Officers with their decisions.

Our requirement for model performance was to have at least a precision of 0.5. We actually improved the precision in production but that was because we received just suspicious interventions and we were saying by default that Police Officers should search the vehicle.

About the requirement of not harming protected groups, it is quite hard to measure the performance because we don't have a lot of data about each protected group in the production environment to evaluate that. But using a lower threshold than the one that we

used to evaluate the performance in the test set, we had worse results for this requirement than that we had in the test set. This can be a consequence of only having suspicious data in production.

Results Analysis

Model Performance

As a reminder, the requirement for model performance was to have at least 50% of Contraband evidence found for all the vehicles searched. This is the same as saying that we should have at least a precision of 0.5 in production.

Knowing the true outcome for some observations that we received in our API, we can evaluate the performance of our model in production and compare it with what we expect with the results we got with the test set.

Precision	Recall	F1-Score	Accuracy
0.36	0.98	0.53	0.37
True Negative	**False Positive**	**False Negative**	**True Positive**
108	3097	36	1758

Table 1: Evaluation of the deployed model performance with production data. Remember that 0 is the encoding for value False and 1 for True.

Precision	Recall	F1-Score	Accuracy
0.28	0.89	0.43	0.97
True Negative	**False Positive**	**False Negative**	**True Positive**
476125	12936	630	5038

Table 2: Evaluation of the deployed model performance with the test set. Remember that 0 is the encoding for value False and 1 for True.

We have the performance in table 1 for our model evaluated with production data and in table 2 evaluated with our test set. The overall result in production looks better but if we look closely at production data it looks like our model is predicting contraband evidence by default. This fact is reflected in the accuracy. Accuracy on table 1 is much lower than the one on table 2 and that is because of the mentioned fact, in production our model has the tendency to say that the intervention has contraband evidence.

With this results in production we started to suspect that our model that was supposed to be “BulletProof” (if Police Officers start sending observations with no criteria it still works), is predicting if the vehicle is suspicious or not instead of predicting contraband evidence. This is a consequence of training the model with all the data instead of just with searched

vehicles. Our model is probably not detecting the nuances in Suspicious data, between what is contraband or not because records are to close from each other.

SAC	VehicleSearchedIndicator	Percentage of number of Records
C	True	35.8
	False	0.01
I	True	20.3
	False	0.01
N	True	4.5
	False	99.3
O	True	39.4
	False	0.6

Table 3: Distribution of values for SeatchAuthorizationCode when VehicleSearchedIndicator was True and when it is False on all dataset (not production data).

From table 3, we can conclude that SearchAuthorizationCode is a good metric to evaluate if the intervention is suspicious or not because it is almost always N when the vehicle was not searched and not N when it was. Knowing this, we can evaluate the prediction and the true outcome in production, for each value of SearchAuthorizationCode (SAC), in order to understand if our model is just guessing if the intervention is suspicious instead of predicting contraband.

SAC	Contraband Indicator	Number of Records			
		Test Dataset		Production Dataset	
		Prediction	True Outcome	Prediction	True Outcome
C	True	5545	1386	1490	434
	False	0	4159	0	1056
I	True	3228	393	1272	182
	False	0	2835	0	1090
N	True	104	636	0	36
	False	476751	476219	144	108
O	True	9097	3253	2093	1142
	False	0	5844	0	951

Table 4: Distribution of Predictions and True Outcome for ContrabandIndicator by SearchAutorizationCode (SAC) for test dataset and production data obtained with the deployed model.

From table 4, we can conclude that our model in production is just guessing if the vehicle is suspicious or not and not predicting if it has contraband or not. Police Officers are just sending requests to our API if they think that the intervention is suspicious so with this model we are not helping them in their decisions to search the vehicle or not but just saying what they already know.

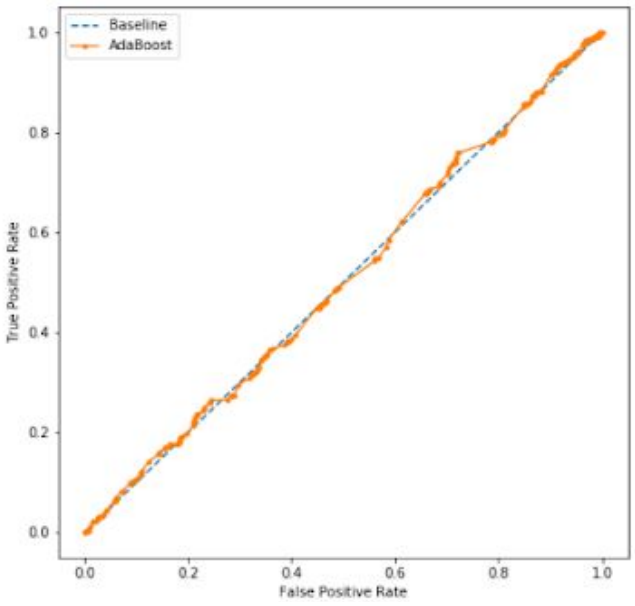

Figure 1: (left) ROC Curve for our model evaluated with Production data. The area under the ROC Curve is 0.5.

From figure 1, we can conclude that our deployed model, trained with all data, in production, is no better than tossing a coin. This curve is very different from the one we had presented in the last report for the model trained and tested with all data.

To sum up, our model is basically looking for the feature SearchAutorizationCode and guessing the outcome True if the value is not N and False if it is N. This is probably happening because this feature has the biggest correlation with target and because suspicious records are too close from each other when we are looking to all scenario (suspicious and not suspicious interventions). To conclude, our model is predicting if the car is suspicious or not and not if it has contraband or not.

We don't want to do the job of Police Officers, we want to help them in their decision of going further in the searching process when they already have some suspicious about the intervention. So we should take a step back and re-formulate some aspects of our model.

The most evident aspect that we should rethink about is the data that we are using to train the model. We did a quick evaluation of how our model would perform in production if we change the data used to train the model. Instead of training the model with all data, just training with searched vehicles (records where VehicleSearchedIndicator was True). The results are in the table below.

precision	recall	f1-score	Accuracy
0.50	0.58	0.54	0.64
true negative	**false positive**	**false negative**	**true positive**
2175	1030	753	1041

Table 5: Evaluation of model performance with production data with the same model as the one deployed but only trained with searched vehicles. Remember that 0 is the encoding for value False and 1 for True.

In table 5 we are presenting the evaluation of our model (we didn't change anything in the pipeline), trained only with interventions where the vehicle was searched by Police Officers, with production data.

From table 5, we can conclude that with our model trained with just searched vehicles, we achieve the requirement of having at least 0.5 of precision.

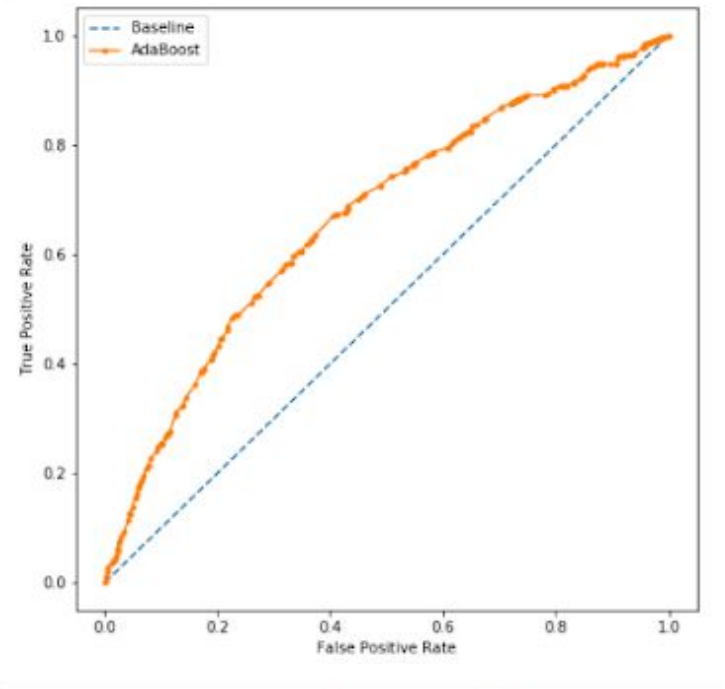

Figure 2: (left) ROC Curve for our model trained with just interventions where the vehicle was searched, evaluated with Production data. The area under the ROC Curve is 0.67.

Roc curve in Figure 2 has a much better result than the one shown in Figure 1. In table 4 we already had concluded that training the model with records where VehicleSearchedIndicator is True has better results in production.

Fairness

In the first report we evaluated the fairness of our model calculating the amplitude between precision for each protected group by police department. In order to achieve the fairness requirement, this amplitude should never be bigger than 5% for each protected group by police department.

In our first report we used a threshold of 20 on our test set to evaluate the fairness of our model. Our dataset in production has a much lower amount of records than the test set used in report 1 to evaluate the performance of our model (model trained with all data). So we need to use a lower threshold to evaluate fairness performance in production. We choose a threshold of 5 to evaluate fairness.

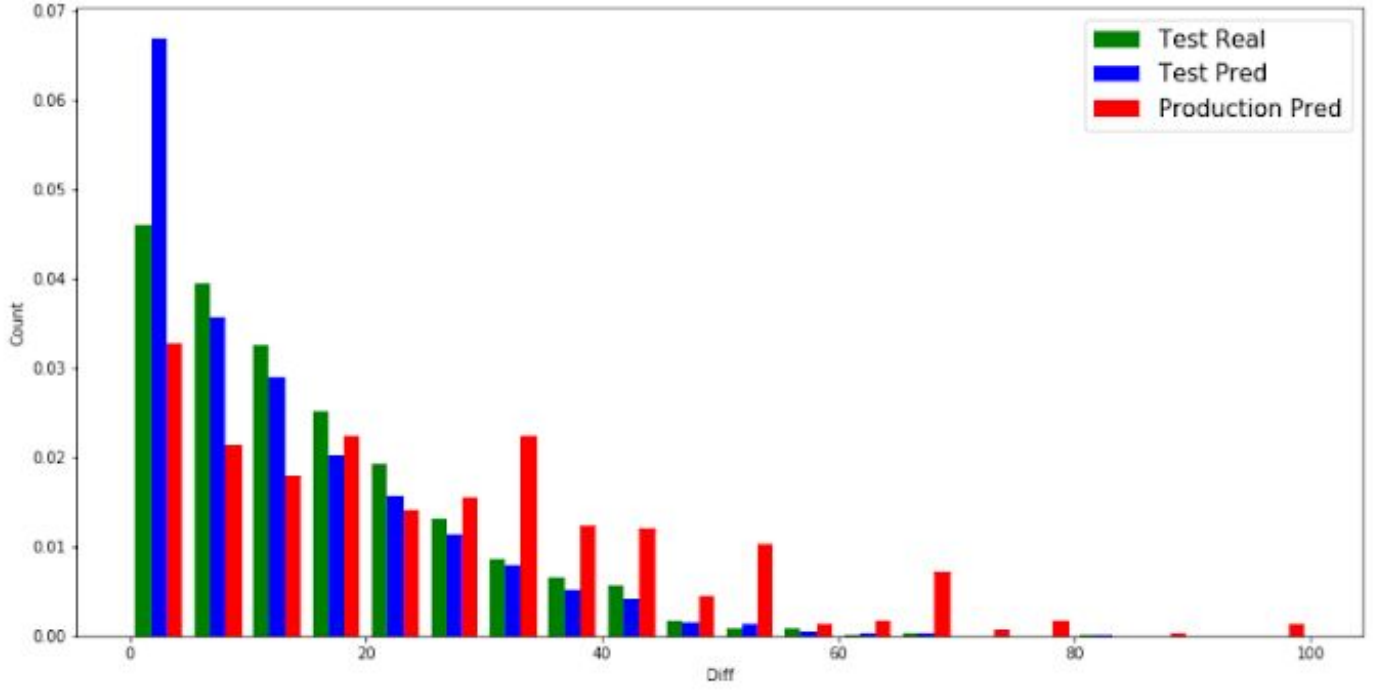

Figure 3: Frequency of differences in precision between protected groups from the same police department. All differences for protected groups (registered by the same police department) with more than 5 records, for every police department, are represented. Results are in percentage and there are represented predictions made by our model (model trained with all data) for the test set (test pred), police officers prediction for the test set(test real) and predictions made by our model (model trained with all data) in the production environment (Production Pred).

On figure 3 we can compare the performance of the deployed model, in terms of fairness, trained with all data (the one deployed in production, trained with all data) in production with the performance of the same deployed model calculated for test data and Police Officers performance also for test data. As we conclude on report 1, our model has a better performance on the test set than Police Officers in terms of fairness because the first bar, [0-5], is bigger for model predictions (Test Pred-blue bar) than for Police Officers prediction (Test Real-green bar). Red bar, that represents the performance of our model in production in terms of fairness, doesn't follow the same tendency as the blue bar. It is lower than the green bar between [0-15] and after that range is always bigger. So it follows the opposite tendency as our evaluation, of our model, for the test set.

This opposite tendency between test set and production data is probably a consequence of just having suspicious interventions in the production environment and suspicious and not suspicious interventions in the test set.

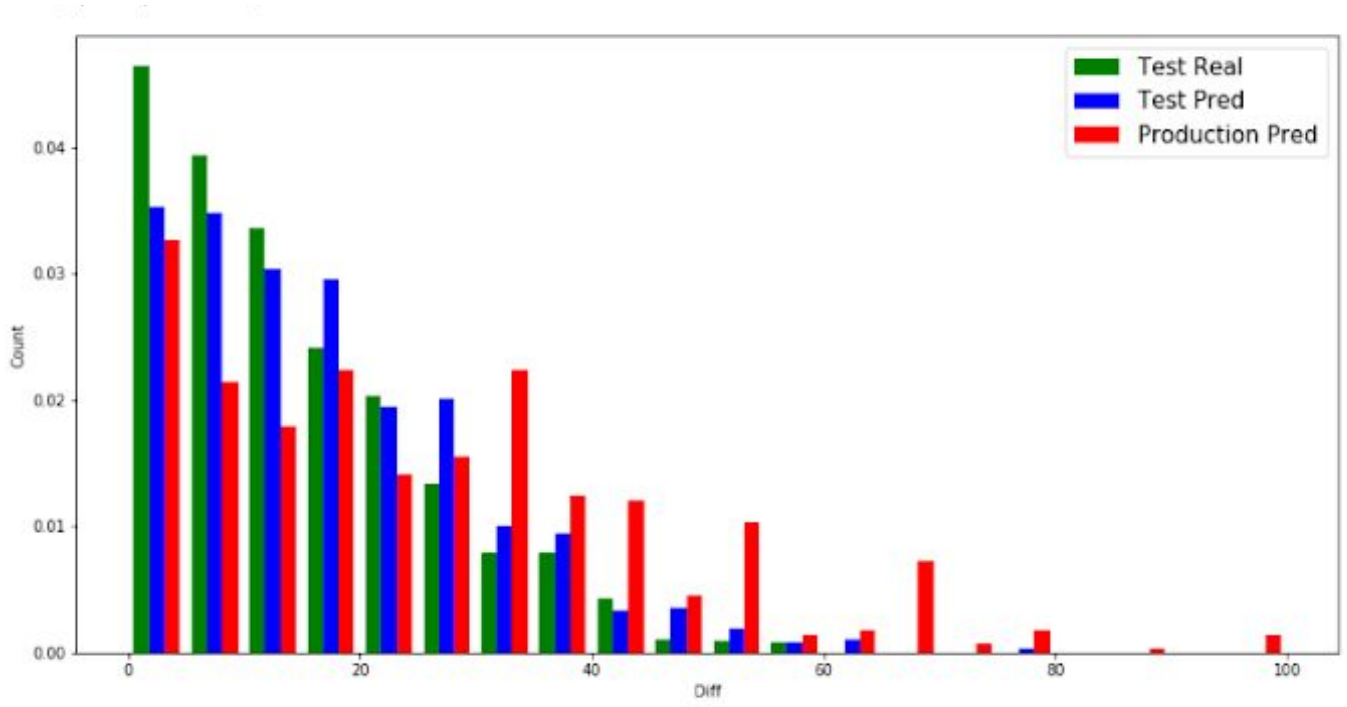

Figure 4: Frequency of differences in precision between protected groups from the same police department. All differences for protected groups (registered by the same police department) with more than 5 records, for every police department, are represented. Results are in percentage and there are represented predictions made by our model (model trained with data where VehicleSearchedIndicator is True) for test set (test pred), police officers prediction for test set(test real) and predictions of our model (model trained with data where VehicleSearchedIndicator is True) in production environment (Production Pred).

Again, if we had just trained exactly the same pipeline with just suspicious interventions (records where VehicleSearchedIndicator is True) the results in the test set and production would be the one shown in figure 4. From figure 4 we can conclude that predictions made by our model, trained only with suspicious records, evaluated on test and production set (blue and green bar respectively), are not better than predictions made by Police Officers on the test set (green bar). This means, if in future we deploy our model trained just with records where VehicleSearchedIndicator is True, we need to re-think about our pipeline because the actual one (pipeline used in production), trained with just suspicious data doesn't improve the performance in terms of fairness in comparison with predictions made by police officers.

Population Analysis

To evaluate differences in population distribution, between test and production environment, we calculated the percentage of each value, for each variable, in the dataset that has personal information about vehicle drivers. We did this analysis for test data and production data in order to compare the differences between the test environment and the production environment.

We did this analysis for all the test dataset because we used all this data to evaluate our model before deploying it to production, but we also did the same analysis, for records in test dataset where VehicleSearchedIndicator is True, in order to compare results of population

distribution in case if we had deployed the same pipeline but only trained with suspicious data. Results are represented in the table below.

Variable	Value	Percentage %		
		Test	Test (VSI==TRUE)	Production
SubjectEthnicityCode	N	84.9	75.6	74.4
	H	13.3	23.1	25.6
	M	1.8	1.2	-
SubjectRaceCode	w	81.6	69.4	66.7
	B	15.6	29.8	32.2
	A	2	0.6	0.9
	I	0.8	0.2	0.2
SubjectSexCode	M	63.2	82.1	79.0
	F	37.0	17.9	20.9
SubjectAge	Young Adult	47.9	73.3	70.4
	Adult	35.5	21.2	23.9
	Senior	15.3	3.7	4.1
	Youth	1.3	1.7	1.5

Table 6: Distribution of each value for each variable that represents personal information about vehicle driver.
This analysis is represented here for test records, test records filtered by VehicleSearchedIndicator (VSI) == True and production records (represented in this order, from left to right).

From table 6, we can conclude that the distribution of values is a bit different between the test dataset and production dataset. On the test dataset, minority classes have a higher occurrence than in production. This fact can be the source of the difference between fairness results from test dataset to production represented in figure 3. On the other side, the distribution of values between records on the test dataset, where VehicleSearchedIndicator (VSI) == True, is very close to data in production.

Next Steps

After looking at the results in production, we think that it is a better option to go with the solution where the model is trained only with records were vehicles were searched. We can also use observations in production that we know the true outcome to add to our previous dataset so we have more data to train the model.

With this new approach, of using just interventions where vehicles were searched to train the model, we will have a different distribution of values in our training and test set. Because of that we need to take a step back and doing again an exploratory data analysis, try again different approaches to transform the dataset, new features, evaluating what features are influencing the target and the model itself.

Another approach that we thought about for our model is to build two different models. One to apply to suspicious records (records with SearchAuthorizationCode not N) and another model to not suspicious records (records with SearchAuthorizationCode == N). In that way our model could be “BulletProof”, if Police Officers start sending requests with no criteria it still has a good performance on suspicious data, without saying that police officers should go further in the searching process without any criteria. We actually tried this approach without changing our pipeline but results were not so good as the results that we got with the same pipeline, trained with just suspicious data. We can spend some more time in this approach, testing other features, transformations and models for each of these models, the one to use in suspicious data and the other one in not suspicious data. If this works, we can accomplish our goal of building a “BulletProof” model.

Like we saw in figure 4, our model, just trained with records where VehicleSearchedIndicator is true, is not accomplishing the fairness requirements at all. It is true that we don’t have enough records to evaluate with precision if fairness requirements are being accomplished or not. We used a threshold of 5 to compare production results with the test dataset. This means that interventions for protected groups with more than 5 records per police department were evaluated. This value is to low to do this kind of statistic, in the first report, when we did the analysis for all the dataset, we used a threshold of 100. Besides that, we also evaluated the performance of police officers in terms of fairness with this threshold, and from figure 4 we can conclude that our model has worst performance than Police Officers’ predictions. Because of this, we need to dedicate some more effort into exploring techniques in order to achieve fairness requirements. One idea that can be explored is finding a way of penalizing records for protected groups searched by police departments that have lower precisions.

Next time we can spend some time tuning the model. We trained the model with all data, we were not able to tune the model on our machines because we went out of memory.

Deployment Issues

Unexpected problems

Deploying our API in production was an easy task. We already had some insights about the process of deploying an API in production, database connections, docker files, pickle files and environment files. Every time we had an error we were able to solve it fast.

The first problem that we had when we tried to deploy our API in production was that our project was not a git project. We add a git project to our folder and the problem was solved.

The next problem that we had was that we were using more imported packages that the ones that had by default in our environment.yml file. This file was not updated with the dependencies that our conda environment had imported. In order to solve that, we generated a new environment.yml from our conda environment and replaced the file in our git project.

The last problem that we had was that we changed our table structure in our local environment. After changes in our local database, API was running on our local machine but when changes went to production it crashed. What happened was that the remote database had the old structure and not the updated structure that we had on our local. To solve that we used the credentials from our remote database that were on Heroku web page, we connected to it and we applied a query to the table in order to change the structure of it.

When our API officially was deployed to production, after publishing the URL so police officers can send requests to it, we had no problems. We have been checking logs on the Heroku web page and so far we haven’t had any reported problem.

What would you do differently next time

It was the first time that we built a machine learning product from A to Z. It was a huge opportunity for us to learn and improving our skills. We learned who to put an already trained machine learning model into production and also who to work with Heroku. With this project we developed skills about putting all pieces of a machine learning project together.

Now we know some possible issues that we can have in production, like unseen data (we took care of this possible scenario in our pipeline so we did not have this problem in production), out of date environment files, database problems…

We didn't have serious problems while trying to deploy the model and also we didn’t have any kind of problems in production after we had published the URL of our API to Police Officers. Because of that, there is nothing specific that we would like to do differently next time.

Next time we will already have some experience in this kind of process so things will probably be much faster to put it working in a production environment.