Business Conclusions **1**
Summary 1
Result Analysis **2**
Model Performance 2
Success on requirements 4
Population Analysis 6
Next Steps **8**
Deployment Issues **8**
Redeployment 8
Unexpected problems 9
What would you do differently next time 9

Business Conclusions

Summary

Awkward Problem Solutions^{TM} has delivered a model that has been in production for a week and a half, receiving a total of 8000 requests made by police officers in order for them to know if they should conduct a search or not. There have been two phases of this production step: the first phase where the model was initially deployed and after some days we received the true outcome for the 4000 observations; Secondly, the second phase of production, where we were allowed to retrain and improve the model, including this new test data and use it to increase the training data and redeploy the model.

While the model was in production, the app has not crashed since, dealing with any null value that was sent and behaviouring properly and as expected. If there is any missing column value an error is returned (since it should be null), and any extra information as columns is simply ignored since it was not present in the training set. The delivered model is also robust enough to deal with any missing data.

Looking at the predictions made by the model between a Successful Search and a Not Successful Search, encoded as 1 and 0 respectively, we can conclude that our model has a tendency to predict that a person should be stopped and searched having a performance barely better than a random classifier. This happens most frequently when the Object of search is controlled drugs, where the model deems the search as being always successful due to the bias introduced by the training set.

Our results were very similar between our production and train data. Moreover, fairness requirements were almost all fulfilled as we expected to. Concluding we find our model useful as a supportive tool for the police officers and will not replace their good senses when making a decision on whether to conduct a search or not, especially when there is visual aid for the decision making.

Result Analysis

Model Performance

By knowing the outcome for half of the observations that we received in our API, we can evaluate the performance of our model in production, calculate any needed metrics, and compare it to what we expected from the results obtained in our test set during training.

We have the corresponding performance of the model in the tables below. Table 1 has the performance details of the model when using the test set and Table 2 on the first phase of production data. The overall result in production looks somewhat similar when compared to the test data, showing that the model might be naive in terms of identifying a successful search, labelling often as it being successful, indicated by a high value of False Positives in both tables.

Precision	Recall	F1-Score	AUC Score
0.60	0.59	0.59	0.59
True Negatives	**False Positives**	**False Negatives**	**True Positives**
24418	25467	3233	9429

Table 1. Evaluation of the deployed model using test data

Precision	Recall	F1-Score	AUC Score
0.54	0.55	0.46	0.55
True Negatives	**False Positives**	**False Negatives**	**True Positives**
1031	1856	291	822

Table 2. Evaluation of the deployed model in the first phase of production.

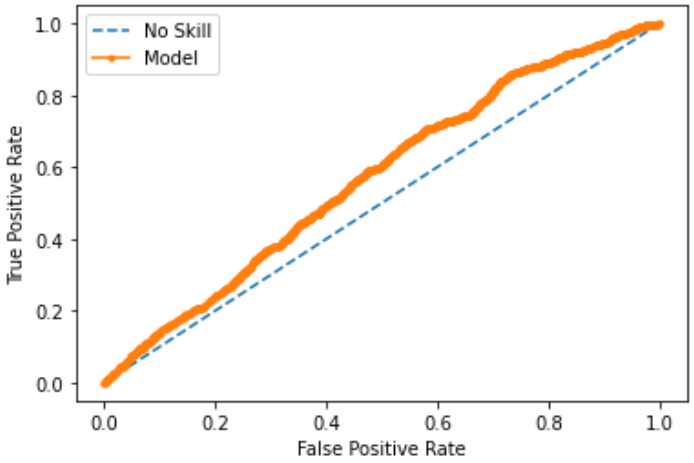

Figure 1. ROC curve of our deployed model with the phase one production data.

From Figure 1 we can see that the deployed model, in production, behaves somewhat better than a random classifier which just always answers the most frequent class (labelled in the plot as “No Skill”), having a AUC score of 0.55. The ROC curve presented is similar to the curve obtained when using the test data which had a AUC score of 0.59.

Regarding dealing with missing data in the requests, the API only received data that was already missing in the training set, being prepared for it. Parameters like *Latitude*, *Longitude* and *Part of a policing operation* are often missing from the requests, the latter being filled as False as previously indicated by the client. Moreover, this first phase of production only included four stations: Cambridgeshire, City of London, Nottinghamshire and Durham, which were distributed as follows in the Table 3 below.

Station		Cambridgeshire	City of London	Nottinghamshire	Durham
Successful Searches	**Predictions**	8	706	1699	250
	Real	**5**	**308**	**522**	**278**
Not Successful Searches	**Predictions**	23	566	504	244
	Real	**26**	**964**	**1681**	**216**
Total of Searches		**31**	**1272**	**2203**	**494**

Table 3. Number of observations per station for each class.

Object of search	**Search Outcome**	**Prediction**	**True Outcome**
Used in theft	Successful	0	86
	Not Successful	490	404
Controlled drugs	Successful	2669	825
	Not Successful	12	1856
Stolen goods	Successful	0	128
	Not Successful	363	235
Offensive weapons	Successful	0	61
	Not Successful	384	323

Table 4. Distribution of Predictions and True Outcome for Object of search for data of production phase 1.

Furthermore, Table 4 shows the distribution of the model’s predictions and true outcomes for the different objects of search for the production data. Notice that the displayed categories are the ones that were presented in majority during production. As it is possible to observe, only

when the object of search is Controlled Drugs, the search is predicted as being Successful, all others are predicted as Not Successful. This mostly happens due to the high quantity of successful cases in the training set when the object of search was Controlled Drugs, leading to a bias of the model on this specific object of search, where all others are always deemed as not successful.

Success on requirements

The requirements regarding the model performance was that the average success rate per station, that we previously determined as being the precision, should not be larger than 10 percentage points and that it should not also vary for more than 5 percentage points between the population sub-group deemed as (station, ethnicity and gender). Moreover, the discovery rate, also known as recall, should be similar between ethnicities per station and per search objective.

Analysing the first stated requirement, the precision score for all stations that were present in the first phase of production can be seen in the table below. From this we can see that all stations, except Durham, fulfilled the requirement. Furthermore, by analysing Table 3, we also noticed that contrary to what was expected during training, Durham had more observations that were successful (or True - 1) searches. However, what we were expecting based on the training set is that successful searches are fewer than not successful, which is what happens in other stations. Furthermore, we know that the model is naive, indicating more often than not that a search will be successful, especially if the object of the search column is controlled drugs. This might explain the increase of the precision score in Durham station.

When compared to what was expected in the first report, we reported that around 55% (21 in 38) of stations successfully fulfilled the requirement of the precision scores. However, here we only have 4 stations where 3 are successful, showing an improvement. However, take into consideration that we have a smaller sample of stations when compared to the training data.

Station	Cambridgeshire	City of London	Nottinghamshire	Durham
Precision (%)	21,7	26,9	27,5	64

Table 5. Precision score for every station present in the production phase 1.

Secondly the difference of precision scores between population subgroups can be seen in Figure 2, note that station Cambridgeshire is not present due to the low number of observations. As we have previously mentioned in the previous report, a minimum of 50 observations are needed to draw any meaningful conclusions for a station. This way, by analysing the figure, it is possible to see that the difference between subgroups should not surpass the 5 percentage points. This requirement is successfully met in the stations present in production, except for Durham, indicating a successful performance for gender and ethnicity fairness.

When compared to the training data and what was reported in the first report, this requirement was met for 23 out of the 38 stations that were analysed. This leads to a percentage of 60%. When compared with the production data, the model successfully met the requirement in 2 out of 3 stations, leaning into an improvement in terms of what was expected. But once again, we need to beware of the small quantity in stations before drawing any hard conclusions.

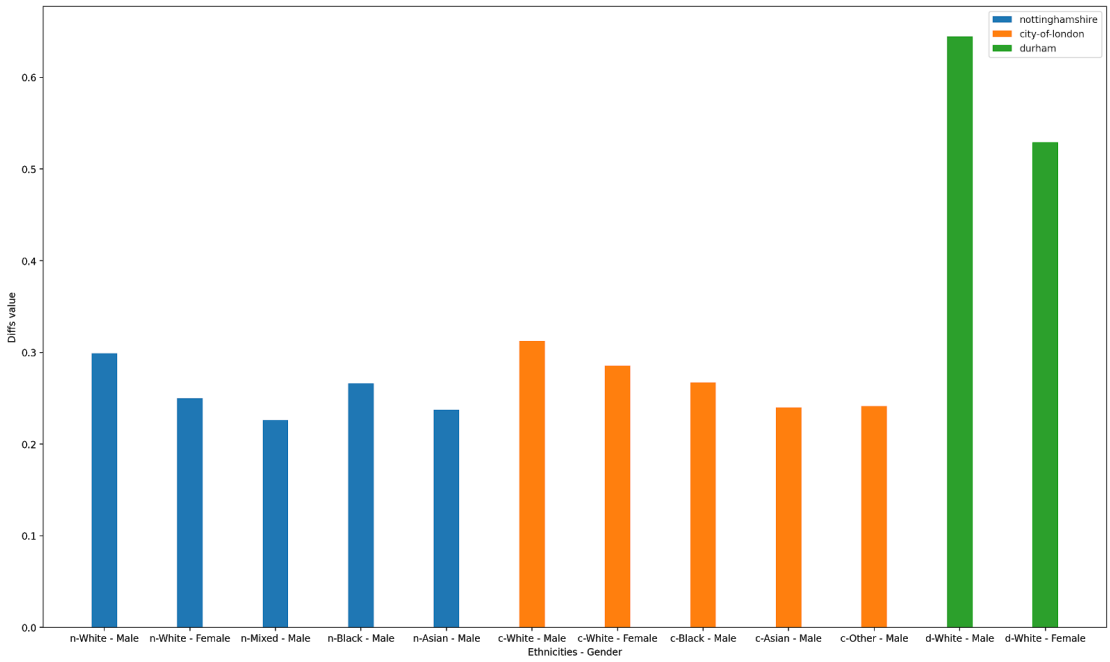

Figure 2. Differences of precision scores for each station between each subgroup (station, ethnicity and gender)

Finally, when it comes to levelling the discovery rate between ethnicities for every station and for every object of search, which was translated in the first report as being the recall score metric, it can be visualised in Figure 3. This figure, however, shows that for some categories in Object of Search, either the recall score is close to 1, meaning that almost all positive classes were found by the model, or on the other hand 0, meaning that no positive classes were found by the model, the same behaviour was previously seen in Table 4. Due to this, this particular requested requirement was failed by the model, this was also expected from the training data, where the model was also not successful in most cases for this requirement.

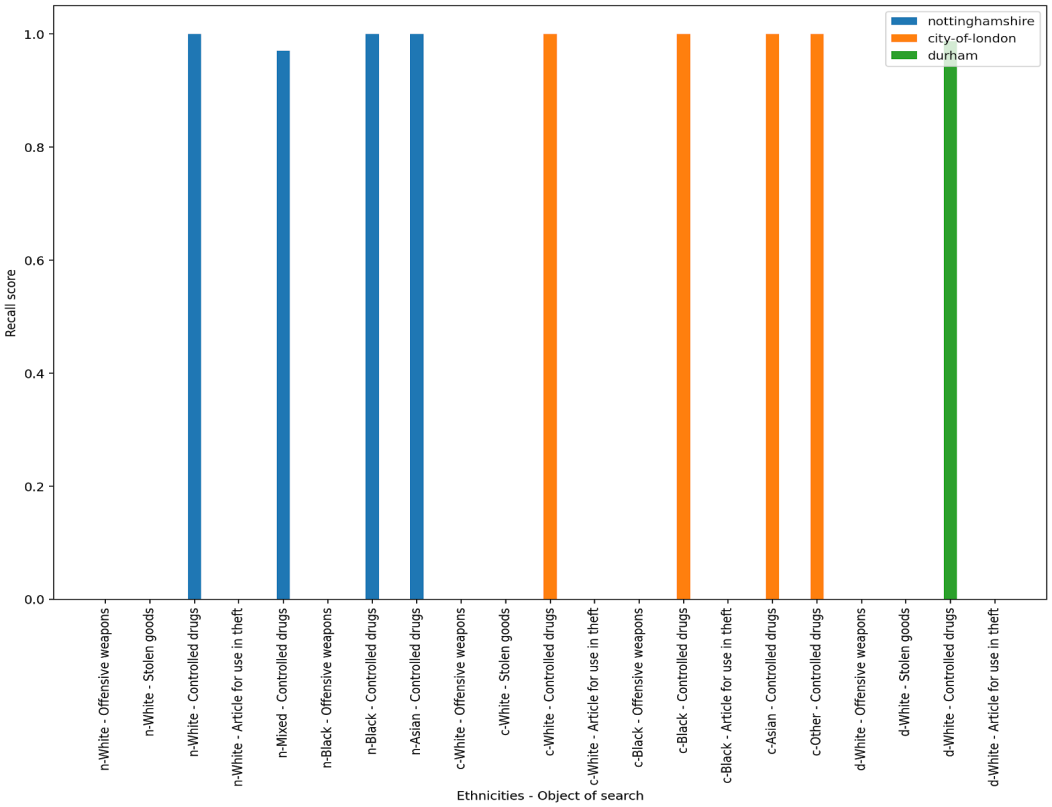

Figure 3. Recall score for every ethnicity for each station and object of search

Population Analysis

To evaluate differences in population distribution between test and the production environment, we calculated the percentage of each value, for each valuable class, in the dataset. Firstly, the training set had information about 42 different stations whereas in the production data there are only 4 stations present.

By analysing Table 6 below, we can conclude that the distribution of values is very similar between the two datasets, minority classes such as gender and ethnicity are similarly distributed between the two sets which validates the similar performance in terms of fairness requirements obtained by the model between the training data and the production data. On the other hand, the column Part of a policing operation has a different distribution between the two datasets. The feature importance presented in the first report showed this latter column has a valuable column to make a prediction. Due to this change in distribution and a high dependence on this feature by the model, its change in distribution may lead to a difference in the overall performance of the application.

Class		Percentage %	
		Training set	Production
Age Range	Under 10	0,05	0,2
	10-17	19,9	10,7
	18-24	36,6	35,4
	25-34	23,9	29
	Over 34	19,4	24,6
Gender	Male	91,6	91,4
	Female	8,3	8,2
	Other	0,05	0,3
Ethnicity	White	57,2	63,2
	Black	26,4	16,4
	Asian	13	13,7
	Other	3	2,4
	Mixed	0,28	4,4
Type	Person search	76,2	74,4
	Person and Vehicle search	23,6	25,6
	Vehicle search	0,14	0
Part of a policing operation	False	96,8	54,9
	True	3,2	45,1

Table 6. Distribution of each class value in the training and production data

Next Steps

After looking at the results in production as well as the logs, we think that some more feature engineering, especially around the Location by using Latitude and Longitude, might improve drastically the model. If we were able to extract either the town or the city of the search operation by using the geolocator package, we could then use it as a sort of heatmap to understand where the most successful stops were taking place.

Moreover and due to the possible different values of Legislation as well as Object of search, we would like to analyse the data and understand if these two columns could provide any valuable information for decision-making, such as maybe grouping several values that might be similar or convey similar information as well as analysing this high dependency on controlled drugs by the model.

We also found that if we try to improve the overall performance of the model, often the fairness requirements start failing, meaning that we cannot build a bullet proof solution with the best possible performance as well as being the most fair. On the other hand, if we successfully fulfill all requests then the performance of the model decreases.

Deployment Issues

Redeployment

In order to improve the model after this first release, we conducted a search for the best model parameters, using a GridSearchCV, from scikit-learn, and found that the best parameters for our proposed RandomForestClassifier are: min_samples_leaf = 100, n_estimators = 100, max_depth = 9, min_samples_split = 2, class_weight = “balanced”.

Furthermore, the columns Legislation, Longitude and Latitude were also added as features but did not seem to improve the model, probably due to the high number of missing values in the data.

Moreover, and after this first phase of production, the true outcomes were received in our endpoint in the API, which we used to add with the already existing training data and retrain the model. The new metrics obtained can be seen in Table 7 below, where the performance remained somewhat constant when compared to previous iterations as well as production data.

Precision	Recall	F1-Score	AUC Score
0.58	0.62	0.51	0.61

Table 7. Performance of the model when retrained with production data added.

Unexpected problems

Deploying our API in production was an easy task. We already had some insights about the process of deploying solutions in production, database connection, saving feature pipelines and models into pickle files, etc. We had one error after the first redeployment but we were able to solve it fast.

The only problem that we came across was that, by using a tree classifier, we did not really need to encode missing values as these types of classifiers are prepared to deal with missing data natively. So when we added Legislation as a feature to the model and deployed it, the data type that the pipeline was expecting was of object and not null type, therefore when a null value was sent in column Legislation, even though the model was prepared for it, the pipeline broke as it could not convert the dtypes of a null value to object type.

We found it challenging when extracting the information from the PostgresSQL database programmatically. We added the database key to the code and even with that we were not being successful. Moreover we did not want to keep the key in the code as it may lead to a security leakage. To deal with this, we accessed the database UI to download all the data available as a CSV file so we could work locally.

Other than that and after our API has been officially launched to production, we have received all the requests made by the police officers and everything seems to have worked fine. We have been accompanying the logs of the application and so far haven't had any problem.

What would you do differently next time

It was the first time that we built a machine learning solution for the ground up. It was a very incredible experience and opportunity to learn and improve our skills. With this project we have developed skills about putting all pieces together of a machine learning project, we also had to wear several hats: data engineer, data scientist and project manager.

We didn't have any serious problems while trying to deploy the model as well as any problems in production due to badly parsed requests. Due to this, there is nothing specific that we would like to do differently next time. It would have been nice to build some unit testing to check written functions as well as an automatic pipeline to update and deploy the model every predetermined number of weeks while checking on performance. We also believe that in our next project things will run faster and smoothly since we have more experience.