# Model Training and Prediction

The outcomes of the police’s searches are divided into two categories: successful searches, where stops lead to further action, and unsuccessful searches where no further action is taken with the stopped individuals. The successful stop and searches variable is converted to a dummy variable with a successful stop equal to 1 and an unsuccessful outcome equal to 0.

## Logistic Regression

The prediction of the successful variable is a classification problem with the goal of correctly classifying the successful and unsuccessful outcomes with zeroes and nulls. To solve the binary classification problem the logistic regression model is used. The logistic regression evaluates the probabilities of positive/negative (1 or 0) outcomes by estimating a linear relationship between features (x values) and log-odds:

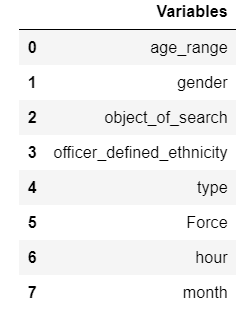
From the log-odds calculation the probability of the outcome being positive being positive/negative is estimated by applying the sigmoid function:

This probability is then used estimate the outcome based on a threshold (here 0.5).

When selecting the weights ,one needs to choose such that the likelihood function is maximized (or minimized if the negative likelihood function is used). This is done by choosing an algorithm that maximizes the log-likelihood function:

In this paper, the limited-memory BFGS method is used and applied in the scikit learn logistic regression function for the optimization problem.

The target value is the outcome of the stop and search and is captured by to the dummy variable “success”. As the goal is predict the outcome by using data on personal characteristics, time and geography, the following variables from the scraped data are applied in the machine learning model:



The person characteristics are described by the variables “age\_range”, “gender” and “ethnicity”. The variables “object\_of\_search” and ”type” characteristics of the search, while the geographical parameter is capture by the “Force” operating in the area of the stop and search. Finally, the “hour” and “month” seeks to capture the temporal influence on the outcomes. From the selected variables all missing values are dropped, so only observations with data for all of the categories above are included. This leaves 673,651 observations for the analysis. All the variables are categorical and before running the logistic regression these are converted to dummy variables, so that each feature is a category within the variables shown above. The list of all features is provided in appendix x.

## Training and Hyperparameter Optimization

The data is split into a development set and a test set to avoid contamination. The development set is used to optimize the regularization parameter . The regularization is optimized to control the complexity of the model and thereby the variance and bias. The variance and bias are measures of the variability in the estimates across samples and systematic errors in the predictions. When modifying the parameter by testing it across different values one can control the regularization and thus find an appropriate solution to the bias-variance trade off. In this estimation process the weight decay regularization method is used (in Scikit learn “l2”). This adds the following term to the cost function i.e. (the negative log likelihood function from before):

When this term is added, the weights will shrink. This ensures that the model can fit the data, without extreme weights and thus makes out of sample performance more robust. The regularization requires, that feature scales are comparable (Raschka & Mirjalili, 2017). Therefore, all features have been adjusted using scikit learn’s standardscaler function. In scikit learn’s logistic regression function, the is controlled by the “C” parameter, which is the inverse of . Therefore, decreasing the value of “C” will increase the regularization and vice verca. To select the optimal value of this regularization parameter, k-fold cross validation is applied. As mentioned, the hyperparameter tuning is done on the development set. This set is split into a training and validation set k times. This way the model can be fitted on the training sets and tested on the validations sets in each of the k iterations. In each iteration is used for training and is used for validation. In this paper the k in the k-fold cross-validation is set to 10, as empirical studies have shown that this gives a good bias and variance trade-off (Raschka & Mirjalili, 2017). This method is used to test which value of the hyperparameter “C” has the highest accuracy in its predictions and thereby should be used for fitting on the whole development set. The accuracy is defined as follows:

The accuracy measure shows the fraction of correct prediction out of the total predictions and thus tells how often one’s prediction is true. The hyperparameter “C”, which gives the best accuracy across the k-fold cross validations are fitted on the development set and then used to predict the target variables based on the features in the test set.

## Results

The results of the prediction are shown in the following confusion matrix, where the number of all correctly and incorrectly classifications of both positives and negatives can be seen.

1. A screenshot of a cell phone

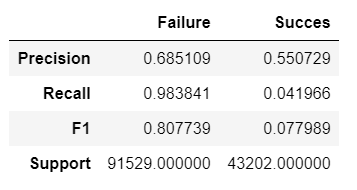
   Description automatically generated

The accuracy at first sight seems to be at an acceptable level of 0.6820. However, when comparing this value to what accuracy one would get by just classifying all stops and searches as unsuccessful (0.68), the model does not do a significantly better job. To asses the quality of the model it is necessary to take other performance measures into account. From the confusion matrix it is seen that the model mainly predicts the stops and searches to be unsuccessful. This is problematic as the bottom left part of the matrix shows that there is a huge number of false negatives. The right part of the matrix reveals that the model very rarely predicts that a stop and search will be successful, despite approximately 30% of the dataset is classified as successful searches. To examine these predictions further the metrics precision, recall, f1 and support are taken into account.

## Performance Metrics

In the formulas below true positive is denoted as TP, false negative as FN and so on.

These metrics are computed for the failures (negatives/0) and successes (positive/1) and are shown along with support in the table below. Support shows the actual number of failures and successes in the test set.



From these metrics it can be seen that the model does very poorly on the success side. The model does prove to have a precision above 0.5, meaning that when it predicts a successful outcome, there is a higher chance of this being true. However, the model does terrible at capturing all of the successful outcomes, as it very rarely predicts that the outcome is successful. This leads to a very low recall on the success side of 0.0429, indicating that the majority of successful stop and searches are misclassified as failures in the model. This low recall also results in a very low score when the combination of precision and recall is measured in the F1 which is equal to 0.07966 for successes. For the failure side, the metric scores are much higher. However, this is mainly a result of the model generally classifying the outcomes as failures. This leads to an extremely high recall of 0.9837 and the model does capture most of the failed stops and searches. The model mainly predicts unsuccessful outcomes and therefore does poorly on another important factor: capturing the successful outcomes. Of course, one can modify the hyperparameter to be measured by the scoring of other measures such as for example F1. However, this has been tested and does not yield significantly different results. Another improvement which could have altered the model, would be changing the threshold in the logistic regression. However, this should have been done in the tuning of the hyperparameters, so one does not just overfit on the estimated model using test set information post-learning. Although one can to some degree evaluate the model over different threshold by plotting the ROC curve and seeing whether the model is better than random guessing. The ROC curve plots the true positive rate and false positive rate across different threshold for the classification. For the estimated model this yields the following graph:

A screenshot of a cell phone

Description automatically generated

With

In the plot, the stippled line is the results of random guessing. Here it is worth noticing that the model does better than this and the area under the curve is 0.61 compared to 0.5 for random predictions. Decreasing the threshold increases the true positive rate and thereby more of the successful outcomes are predicted. Although this is at the expense of increasing the amount of false successful outcome predictions. Despite the better than guessing performance shown in the graph, ROC curves can sometimes be misleading for imbalanced datasets such as this (Davis & Goadrich, 2006). Instead a precision-recall curve can be used to examine the model’s abilities. To illustrate the trade off for varying thresholds, graphs with recall and precision is constructed below - here also with a graph showing the thresholds on the x axis. This shows how precision is highly affected when wanting to capture more of the successful outcomes.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Here it is shown that recall is very low at the high precision levels and decreases rapidly when increasing the threshold and thereby the precision. However, the precision decreases rather slowly when lowering the threshold, which means the threshold can be altered without seeing significant changes in this. For the precision-recall curve, the AUC is only 0.4080, and the model is not particularly good at capturing the outcomes nor highly precise when predicting them.

Overall the model does a bad job in predicting the outcome when keeping the imbalanced data set in mind as the accuracy is evaluated. This is highly visible from the low recall and F1 scores, which tells that the model captures few of the successful stops and searches. Of course, this could have been altered by hyper tuning the threshold, so that more focus was put on higher recall to catch all criminals. However, this was first apparent in the post modelling descriptive analysis. One parameter where the model does a slightly acceptable job is in its precision. When the model actually predicted that the stop and search would be successful it proved right 55% of the time, whereas classifying all outcomes as successful would only have had a precision of approximately 30%.