This is a manual to create your own machine learning tool for predicting the number of contributors (NOC) in a DNA sample. Based on the methods used in - C.C.G. Benschop, J. van der Linden, J. Hoogenboom, R. Ypma, H. Haned. Automated estimation of the number of contributors in autosomal short tandem repeat profiles using a machine learning approach. Forensic Sci. Int. Genet. 43 (2019) 102150.

Addition: The feature minimum NOC is explained in the paper as: ‘*Allele count of the locus with the largest number of alleles / 2, rounded up to 0 decimals’*, but the minimum NOC is ‘*Allele count of the locus with the lowest number of alleles / 2, rounded up to 0 decimals*’

To create a model for the NOC prediction several things are needed

Requirements and the versions which were used:

* DNA profiles with known number of contributors
* Python (version 2.7.6)
* Sklearn library (version 0.19.1)
* Numpy (version 1.16.3)
* Json (version 2.0.9)
* Scipy (version 1.2.1)
* Matplotlib (version 2.1.0)

To split up the data it is possible to use ‘**shuffle.py’**.

This python script will shuffle the given profiles and puts them randomly in the train, test and holdout set, taking into account that each NOC will be in balance in each set. Distribution of the number of samples in the train, test and holdout set will be around 60/20/20.

Input: ‘*nocs.txt’*, contains the sample names and the known number of contributors separated by a tab. No headers. The sample names can be anything. See example file (Example-nocs.txt).

Output: “*shuffled.txt*”, a file containing the sample names and their respective set (train, test or holdout) and what the NOC is of each sample. This file can be used to split the data after the features have been created with the use of the next python files.

‘**Features.py’** will calculate the features of given samples.

The script is made generic for each delimiter, and is based on the possibility for some different headers.

“Allele”, “Height”, “Size” or “Allele#1”, “Height#1”, “Size#1”, etc.

The first two headers of the file have to be either “File Name” & “Marker” or “Sample Name” & “Marker” or “Name” & “Marker”. Examples of sample files are can be found in GitHub.

Input/variables:

* ‘*lociMarkers’* is a variable list with the loci, which are being used. This list is adjustable in the python script.
* Folder where the data is. Needs to be adjusted in the python script.
* Allele frequency file / Population file. Fill in the name of the used allele frequency file. Example is present and is from *A.A. Westen, T. Kraaijenbrink, E.A. Robles de Medina, J. Harteveld, P. Willemse,S.B. Zuniga, K.J. van der Gaag, N.E.C. Weiler, J. Warnaar, M. Kayser, T. Sijen, P. de Knijff, Comparing six commercial autosomal STR kits in a large Dutch population sample, Forensic Sci. Int. Genet. 10 (2014) 55–63.*
* Stochastic Threshold. The stochastic threshold is used to calculate some features. Right now it is set to 800, but could be change to your preference.
* MatchProbability . To calculate the match probability some variables are needed. In the python file the variables are set which were used in our study. Please adjust them to fit your data.
  + q: theta correction.
  + pop: population size of the calculated allele frequencies.
  + sb: size bias.

Output: “*Features.txt”* A Json transformed to a txt file. The features dictionary is dumped with Json to the file, so it can be reused.

‘**SplitIntoSets.py’** will split the created samples and their features into the train, test and holdout set, based on the “*shuffled.txt*”

Input:

* “*shuffled.txt*” or your own file with each sample name and in which set each sample should be and their respective NOC. (Example available)
* “*Features.txt”*
* Output: The training, test and holdout features sets will be outputted separately in txt files. “*FeaturesTraining.txt”*, “*FeaturesTest.txt”* & “*FeaturesHoldout.txt”*

After the samples have been split up into the three different sets, it is time to make a selection out of the features, which can be done with Partial Correlation.

For the partial correlation it is possible to only use the training set or use the training and test set. Therefore there are two different python scripts. In our case, we used only the training data.

‘**PartialCorrelationTrain.py**’ & ‘**PartialCorrelationTrainTest.py**’

In our case we put MAC & TAC as the first two features to be used in the partial correlation, therefore it is still set in the two python files.

Input: “*FeaturesTraining.txt”* & “*FeaturesTest.txt”*

Output: “*partialCorrelationResultsTrainTest.txt”* or “*partialCorrelationResultsTrain.txt”*. A tab separated file containing: Feature that has been added, the partial correlation of that feature and the list of features at that time.

**‘GridSearch.py’** will perform a gridsearch, which can optimize the algorithm parameters.

Input: “*partialCorrelationResultsTrainTest.txt”* or “*partialCorrelationResultsTrain.txt”.*

The number of cores which is used by gridSearch is also adjustable (n\_jobs) is now set to 3.

Output:

* “*NoGridResults.txt”* and “G*ridResults.txt”*. A tab separated file containing: model name, training accuracy, test accuracy, number of features and the parameters of the algorithm in case of a gridsearch.
* A plot (\*.png), for each combination (algorithm, number of features and gridsearch or not), will be generated of True NOC vs Predicted NOC.
* For each combination a tab-seperated report will be generated with Sample name, True NOC and Predicted NOC.
* “scaler.pkl” ,
* A \*.pkl file of the trained model for each combination.

**‘input.py’** is a script to use the saved model (\*.pkl). Will calculate the features of given samples and predict the NOC.

Change the file names in the python script and then the python script can be used in the command line as: >> python input.py profileFile

The script is made generic for each delimiter, and is based on the possibility for some different headers.

“Allele”, “Height”, “Size” or “Allele#1”, “Height#1”, “Size#1”, etc.

The first two headers of the file have to be either “File Name” & “Marker” or “Sample Name” & “Marker” or “Name” & “Marker”. Examples of sample files are can be found in GitHub.

ProfileFile can contain multiple samples.

Input/variables: “*model.pkl”* & “*scaler.pkl”* .

* ‘*lociMarkers’* is a variable list with the loci, which are being used. This list is adjustable in the python script.
* Folder where the data is. Needs to be adjusted in the python script.
* Allele frequency file / Population file. Fill in the name of the used allele frequency file. Example is present and is from *A.A. Westen, T. Kraaijenbrink, E.A. Robles de Medina, J. Harteveld, P. Willemse,S.B. Zuniga, K.J. van der Gaag, N.E.C. Weiler, J. Warnaar, M. Kayser, T. Sijen, P. de Knijff, Comparing six commercial autosomal STR kits in a large Dutch population sample, Forensic Sci. Int. Genet. 10 (2014) 55–63.*
* Stochastic Threshold. The stochastic threshold is used to calculate some features. Right now it is set to 800, but could be change to your preference.
* MatchProbability . To calculate the match probability some variables are needed. In the python file the variables are set which were used in our study. Please adjust them to fit your data.
  + q: theta correction.
  + pop: population size of the calculated allele frequencies.
  + sb: size bias

Output: SampleName + predicted NOC + probability.