**Explainable artificial intelligence in forensics: realistic explanations for number of contributor predictions of DNA profiles**

# Abstract

Using machine learning to determine the number of contributors (NOC) in short tandem repeat (STR) mixture DNA profiles has been shown to obtain good accuracy. However, the models used so far are not transparent to users as they only output a prediction without any reasoning for that conclusion. Therefore, we introduce eXplainable artificial intelligence (XAI) to help users understand why specific predictions are made. Where previous attempts at explainability for NOC estimation have relied upon using simpler, transparent models that achieve lower accuracy, we use techniques that can be applied to any machine learning model. Our explanations incorporate SHAP values and counterfactual examples for each prediction into a visualization. Existing methods for generating counterfactuals have not attempted to handle correlated features, causing those techniques to find examples that are impossible given their feature combinations. Since the features derived from STR data for NOC estimation are highly correlated, such counterfactual methods are inappropriate. For this reason, we have implemented a new counterfactual method, ReCo, which generates realistic counterfactual explanations for correlated data. We show that ReCo outperforms state-of-the-art methods on traditional metrics, as well as on a novel realism score. A user evaluation of the visualization demonstrates the opinions of end-users, which is ultimately the most appropriate metric in assessing explanations for real-world settings.

# Introduction

## Number of contributor estimation

Deriving the Number of Contributors (NOC) from Short Tandem Repeat (STR) profiles is a challenging task due to occluding factors such as allele sharing between donors, or allelic drop out [1-9]. This becomes increasingly difficult when the number of contributors rises. However, most probabilistic genotyping software that is used for weight of evidence calculations does require the NOC to be entered by the user [10, 11], which can influence the height of the likelihood ratio [2, 11-16].

Valuable steps have been made to develop methods that can more accurately predict the NOC than relying on the Maximum Allele Count (MAC)-method which involves taking the locus with the most alleles, dividing by two and rounding up [17]. The improvement mainly corresponds with incorporating more information such as for example the Total Allele Count (TAC), peak heights, drop out and stutter rates, the distribution of allele counts, and population allele frequency [3, 5, 8, 9]. Others use more complex techniques like Bayesian networks [4]. From the multitude of models to estimate the NOC, machine learning models have shown to outperform standard methods on both accuracy and speed [13, 18-20]. However, machine learning algorithms are often considered to be *black-boxes* [21-28], as the predictions they produce are made based on generalization from training data, but the exact mechanism is not easily understood. It is important for DNA experts to know which factors the algorithm or *model* used to make a prediction. In this way, the experts can decide whether or not to trust the outcome. Perhaps the model considered some information that the expert missed, or even made a decision on information that should not be relevant to determine the NOC. By delivering this transparency, predictions can be made more understandable and more informed decisions can be made.

A method using a decision tree was presented as a more transparent way to use machine learning to estimate the NOC [29]. However, using a simple model such as a decision tree leads to less accurate predictions; they reported a decrease in accuracy of over 10% as compared to a random forest model. The method also relies heavily on filtering of artefacts, for which another decision tree is used. Furthermore, the data used in this study is also derived from a small number of donors, which means that there is little diversity and less complexity in the data. If more complex data is used, the performance of a simple model decreases even further. More complex predictors are more suited to handle such data.

It seems that there exists a trade-off between accuracy and transparency. However, none of the previously mentioned techniques have explored the field of eXplainable Artificial Intelligence (XAI). XAI has emerged to provide explanations for any type of machine learning model, since users want to know *why* a certain prediction is made [21-28]. The European Commission recently underlined the importance of explainability in a proposal for rules on AI systems in higher-risk settings such as law [30]. Though NOC estimation does not directly cause decisions without the involvement of human experts, these experts should be well-informed about the system that they might let influence their decision. We aim to provide some basic insight into XAI before diving in to how it can be used in the application of NOC estimation.

## eXplainable Artificial Intelligence (XAI)

Machine learning models roughly fall into two categories when it comes to how interpretable they are; transparent- and black-box models [21, 23-28, 31]. With transparent models, one can derive the exact steps taken to arrive from input features to an output within reasonable time [24, 25]. A decision tree could be considered a transparent model, since it shows each decision made for any input to reach a prediction. It starts at the top with the root node, and splits off to different branches based on conditions specified in each node, until a leaf node is reached which represents a prediction. This transparency is limited by the size, the complexity and components of the algorithm. In the example of the decision tree, it cannot be too large, make decisions based on complicated conditions, or use variables that are not easily understood [24, 25, 32]. If all of these conditions are violated, the model becomes a black-box. It then requires post-hoc explanations, which are generated after the underlying model has been optimized.

To achieve an explanation, we can choose to leverage some structures of the model, or create a model-agnostic explanation [23, 24, 27, 28]. The decision tree example is therefore a model-specific explanation, since it utilizes the structure of the tree to serve as the explanation. *Model-agnostic* explanations do not make any assumptions of the type of model they are explaining and thus can be applied to any machine learning model.

It is also important to determine the scope of the explanation. Either they refer to the entire model and its data such as the decision tree example (global), or to specific parts of it (local) [21, 23-28, 31]. A *local* explanation has the advantage that only information about the current decision is shown. In this way, an explanation can be more compact and simpler than attempting to portray the entire model. Conversely, the complete model could be more complex, as the explanation only contains a subset of the entire prediction space.

For NOC estimation, DNA experts look at one prediction at a time and would like the most accurate description of how a single profile is processed. As we mentioned earlier, more complex machine learning models perform best, so it might be difficult to explain the entire decision-space of the algorithm. Instead, local explanations of how individual DNA profiles are predicted seems more fit. A model-agnostic approach is preferable, since the application of machine learning models to this problem is still in development [13, 18-20]. Local, model-agnostic explanations are generally one of two types; *feature importance* or *counterfactuals*.

Feature importance methods highlight the values of the input that have driven the model to make a certain prediction [21, 23-25, 27, 31]. This effectively answers the question *“Why did the model predict A?”*. An established method for arriving at such explanations is SHAP. SHAP calculates Shapley values that show how much certain input features have contributed to a prediction, in comparison to the average prediction [33]. These Shapley values have a solid background in game theory to produce consistent explanations. For the exact method and techniques used to calculate the Shapley values, we refer to Lundberg et al. (2017) [33]. SHAP has been implemented in real-life cases such as predicting hypoxia based on clinical data [34], and predicting the most fitting eye-surgery type [35]. They seem to have obtained valuable information about which factors the ML models based their prediction on.

Counterfactual explanations are example data points which have a different prediction from the input data point [21, 23, 24, 27]. From such a counterfactual the audience could derive how the original instance could have been predicted differently if certain input features had different values. As such, they answer the question *“Why did the model not predict B?”*. This way of reasoning is underpinned by the social sciences to be effective, as humans seek contrastive explanations [21, 22]. This field is in active development and no method has been proven to work well for all sorts of applications. With this paper, we present a counterfactual method that is suitable for the NOC prediction domain. As such, counterfactuals will be covered in more detail.

## Counterfactuals

A counterfactual is an example instance that is similar to the instance we want to explain, but has a different prediction [36-51]. The differences in feature values between the input and counterfactual can give the user an impression about the local decision space of the model. More formally, a counterfactual can be described as follows [38]:

“*The model predicted outcome because input instance had values . If instead instance had values , and all other values had remained constant, the model would have predicted outcome ”*

This alternative outcome is often referred to as the *target* of the counterfactual [36-51]. To help the user relate this new prediction as a possibility for the original input, the counterfactual must be similar to the input. To find the most suitable counterfactual, there needs to be a definition of what ‘similar’ entails. Most commonly, this is measured by the distance from the input to the counterfactual [37-43, 45-48, 52]. Though some methods use or Euclidean distance [43, 47], or Manhattan distance appears to be the measure of choice as it does not blow outlier distances out of proportion as distance tends to do [37, 38, 40-42, 46, 48]. This is because with Euclidean distance, the differences in feature values are squared, while Manhattan distance takes the absolute differences. Alternatively, or additionally, similarity of a counterfactual can be measured by the number of differences in feature values in comparison to the input [36, 37, 40, 42, 44-48, 50, 51], sometimes referred to as distance.

In summary, the consensus is that a counterfactual should be:

* Valid: it has the *target* outcome.
* Proximal: it has minimum *distance* to the input.
* Sparse: it has minimum *feature differences* with regards to the input.

There are more aspects to optimize such as presenting a diverse set of counterfactuals [37, 39-42, 48], or providing counterfactuals that are actionable; meaning that the changes can be acted upon to reach that alternative outcome [37, 39, 40, 43, 44, 51]. This is useful when the input features can be changed in the future, for example by raising your income for a loan application.

To generate counterfactuals, they can either be chosen from the training data [45, 53], or can be artificially sampled [37-42, 46, 47, 49-51]. The main advantage of presenting a training data point, is that it is a real-life example. It is therefore inherently realistic. However, training sets can be thinly populated, which means that the most similar counterfactual might still be widely different from the input that you are comparing to. The sampling-based approaches usually do not suffer from this problem. They either create a dense area of sampled data [38, 39, 41, 46], or take the input and perturb its feature values until a different outcome is reached [37, 40, 42, 47, 49-51]. While most tackle sampling by randomly changing feature values from the input instance or the training data [38-40, 46, 47, 49, 51], some take a more sophisticated route by using a genetic algorithm [37, 41, 42, 50]. Genetic algorithms generate instances from a starting ‘population’ such as the training data, or the input instance. These are then ‘evolved’ through crossover, mutation and selection. Crossover refers to combining feature values from two individuals, while mutation randomly changes an arbitrary feature value. By selection, only the samples with the best fitness score are kept. This fitness score is usually defined by the distance to the input. Some approaches also consider other objectives, such as sparsity [37, 40, 46]. This is usually implicitly incorporated by the previously mentioned methods that start from the input instance and adjust features one at a time until the target prediction is reached. This will keep the counterfactuals inherently sparse. Others have implemented sparsity as a constraint [44-46]. The difficulty with the latter is how to define beforehand how many differences between the input and counterfactual are allowed or even plausible. This can vary strongly between various domains, datasets, and users. Another approach is to edit produced counterfactuals back to the input instance until the target prediction no longer holds [40]. The risk here is that there is no guarantee the counterfactual can be made more similar to the input.

One method has identified that the search for counterfactuals can be tackled by using Multi-Objective Optimization (MOO), such that several scores can be optimized simultaneously [37]. In this way, the multiple objectives do not have to be enforced through summing them together, adding constraints or filtering steps, but can be included to find a Pareto optimum set of solutions. This set consists of instances with different trade-offs between the scores, and are non-dominated. What this entails is that for each of these instances in the set, there exists no better alternative; there cannot be an improvement for one objective, without decreasing the score for another objective.

Some approaches have leveraged the power of SHAP values to create counterfactuals [36, 49]. By only changing the features from the input instance that have negative SHAP values that work against the target prediction B, a counterfactual could be found [49]. This approach suffers from the fact that by only changing features with negative SHAP values, they limit the range of possible feature changes and therefore produce counterfactuals that are generally further away. In a similar approach, features with the highest SHAP values for the predicted class A were iteratively set to zero, until the target class is reached [36].

An aspect of generating counterfactuals with sampling-based methods that is largely overlooked or handled poorly is realism. As the samples are often generated by randomly changing feature values, or by combining instances, they might be infeasible. For example, a generated instance in the context of loan applications might be a 20-year-old person with 15 years of working experience as an ideal candidate for a loan. This example obviously does not represent a real-life situation. A counterfactual example must be a plausible data point to make the user see its real-life value. Note that this issue does not frequently occur with counterfactuals derived from the training data, which are inherently realistic.

There have been some attempts to create plausible counterfactuals. These mostly rely on the assumption that features are independent. For example, to give a general impression of the relation to the training data, the distance to the closest training data point can be measured [37]. By taking this score into account, found counterfactuals are generally closer to the training data. It is also possible to look for counterfactuals that lie in dense, connected areas of the training data [43]. This ensures that the query instance can be transformed to take on the target output, which is relevant for actionable settings. Similar to these approaches, the range of feature values can be limited [39, 41]. Either based on the training data, or inputted by the user. When considering our previous example, there are most likely plenty of 20-year-old people, and also people with 15 years of working experience in the training data. However, the issue with this example is that age and working experience are correlated, and the combination of the feature values is highly unlikely. None of the previously discussed techniques take correlation into account.

Some efforts have been made to handle correlated data, though these mostly leave the responsibility to the user. For example, the user can supply causal graphs between features to model certain feature correlations [40]. These graphs are then applied to filter the already-generated counterfactuals to remove any that do not comply. This could mean that no counterfactuals remain, as the filtering happens after the generative process is completed. An implementation called GeCo has shown promise by limiting feature combinations to the ones made in the training data, though again the user needs to supply each of these relations manually [42].

One method derives counterfactuals from training instances, which relies on the assumption that there are inherently sparse counterfactuals in the training set [45]. As they point out themselves, this will most likely fail on more real-life datasets as there are often more feature differences than they deem fit (< 2).

Though several studies have brought up the issue there should be a way to handle correlated features [39, 40, 54, 55], no method has been published that inherently adapted this in a way that is viable for real-life data, without the need to manually model feature relationships. To the best of our knowledge, we are the first to develop a method that is intrinsically suitable for real-life datasets with correlated features.

Finally, improving the visual presentation of counterfactuals is regarded helpful to the users. Most counterfactual methods for tabular data present the comparison of the input and counterfactual in a table [37, 40-42, 44, 45, 47, 53, 56]. This does not clearly communicate the magnitude of the feature value differences between these instances without the user having to do arithmetic. Similarly, this effect is also apparent in explanations from a conversational statement or natural language [48, 57, 58]. With a visual approach, these magnitudes can be communicated better. [59]. Though some previous visualizations were developed for counterfactuals [46, 51], it was unclear for which audience these were fit and how well they worked for those users. Furthermore, no visualization has incorporated feature attributions with counterfactuals, which could be beneficial to form a complete picture of the prediction [51, 59].

## Contribution

With this paper, we aim to demonstrate the value of XAI to the field of forensic science by

applying it to a practical issue. We generate explanations for individual predictions of the Number of Contributors (NOC) to a DNA profile, which can be applied to any type of machine learning model. These explanations consist of SHAP values and a counterfactual example in a compound visualization which we have found to be the first explanation that unifies these techniques. We also implemented a new method for finding realistic counterfactuals (ReCo), which to the best of our knowledge is the first technique that handles correlated data automatically. Lastly, we have created a new realism metric that scores counterfactuals on the plausibility of their feature combinations.

# Materials and methods

## Data analysis and sampling

The used dataset originates from a previous study that the Netherlands Forensics Institute (NFI) performed [18]. It initially consisted of 590 PowerPlex® Fusion 6C (PPF6C) profiles, either from a single donor, or from a mixture of up to 5 donors. The mixtures were formed from 1174 different single donors that were mixed in various proportions and using various amounts of DNA to create profiles that are representative of real casework. The ground-truth NOC was therefore available. Each profile was translated into 19 featuresconsisting of allele counts, allele frequencies and peak heights such that . These are all numeric variables which can be found in more detail in Supplementary Table 1.

The original dataset was expanded with 5000 samples to create a less sparse feature space. In a development version of the statistical library DNAStatistX [60], realistic profiles can be generated by using the same model that is used for calculating weights of evidence. Note that DNAStatistX implements an algorithm to calculate the Maximum Likelihood Estimate which is largely based on the source code of the probabilistic genotyping system EuroForMix [12]. This program was used to generate factors such as peak height, degradation, and mixture proportions within ranges derived from the original dataset. Note that elevated stutter peaks were not simulated. However, the probability of drop-in was set quite high at 0.05 by which the simulated DNA profiles could include additional peaks, not belonging to one of the donors, as can occur under casework circumstances. In Supplementary Table 2, the exact parameters can be found. With these parameters in place, the genotypes are generated randomly based on Dutch population frequencies [61]. To ensure that all donors have at least some of their alleles observed in the generated profile, we chose to set the requirement that each donor must have an LR of at least 1000 when computed using DNAStatistX.

The advantage of sampling before applying any XAI technique is that the profiles are generated, not the derived features. In this way, validated software is used to generate as plausible as possible profiles from which features can be calculated afterwards. The features used are strongly correlated (see Supplementary Figure 5), which would make sampling in a later step more difficult.

Once the features were derived from the sampled data, about half of them appeared to have been drawn from a different distribution as compared to the original dataset of 590 instances (see Supplementary Figure 3 and 4 and Supplementary Table 3). For instance, the TAC and MAC values of the sampled data appear to be slightly higher, implying neater, easier to interpret data. On the other hand, the variation in allele counts and peak heights is larger, adding more diversity in the data. Because of these discrepancies, we tested the value of the simulated data in a benchmarking study, which demonstrated that the model actually performs better once trained on the combined dataset of the original and sampled datasets together (see Supplementary Table 4 and Supplementary Figures 8 and 9 in comparison to Supplementary Table 5 and Supplementary Figures 6 and 7).

## Machine learning model

Originally, the estimation of the NOC was treated as a classification problem by the NFI with the RFC19 model such that where is an input profile consisting of the 19 features [18]. The model is a random forest classifier (titled RFC19), which produces an output within five categories such that . They obtained a test accuracy of 82.5%.

Since the outputs of the model are ordinal, the problem could benefit from being tackled with a regression model. After a short benchmarking study with a default random forest regressor (see Supplementary Figures 6 - 9 and Supplementary Table 4), we concluded that a regression model has the potential to achieve more accurate predictions. The regression model in combination with the larger dataset even improved performance on the profiles that originated from the original dataset (see Supplementary Table 6). This shows that the model performs well on real profiles, and not just on the simulated instances.

Explanations can also benefit from using regression, as the outputs are not independent boxes, but lie on a relational scale. The model can then be defined to map the input profile to an output where . In this study, for the purpose of introducing XAI to a NOC machine learning model, we chose to continue with the regression model training on the combined dataset (RFR19\_merged) though the XAI method will be applicable independent of the type of machine learning model.

## Explanation goals

The most pertinent case for which explanations of the machine learning model are helpful, is when the DNA expert comes to a different conclusion than the model. It can be unclear why this discrepancy exists when only the model output is available. It is possible that the user missed some information that the model based its decision on, or perhaps the model made its prediction based on the wrong factors. Presenting explanations that provide a sense of thresholding values and how close the decision is, can help the expert make a more well-informed decision on which result to trust. Explanations can also be informative in clearer cases as a confirmation of the user’s own estimation of the NOC, with general information on the model’s focus. These scenarios line up nicely with the two questions that a good explanation of a single prediction should answer [21, 38, 39]:

1. *What were the main reasons for the model to reach the current prediction?*
2. *With which feature changes could the model have arrived at a different prediction?*

To answer these two questions, we consulted a study that has identified which types of

explanations work best for which types of questions [62]. From their definitions, the experts would like to have “WH-X” and “WH-NOT-Y” questions answered, which correspond to questions 1 and 2 listed above. These are respectively best answered with a factual explanation that denotes which factors in this profile is causing the prediction, and a counterfactual explanation demonstrating what would need to change in this profile to reach a different prediction.

To answer question 1, we determined that the use of SHAP values would be sufficient to give an impression of feature importance. We acknowledge that all perturbation-based feature importance methods arbitrarily split the impact of correlated features on the model [63]. The result of this issue is that the importance values for correlated contributing features are underestimated, in contrast to if their importance was left undivided. However, since the main goal of these values is to give an impression of the contributing factors to a prediction, the exact values are not a priority. This part of the explanation is to observe a general sense of which features contributed to the prediction in which direction. For this purpose, we deem SHAP adequate. For the second question, we have developed a new counterfactual technique, for which we performed a more in-depth analysis of the requirements.

## Desiderata counterfactual explanations

To develop the most suitable counterfactual method, we derived a list of desiderata that

it must accommodate. These requirements originated from the factors discussed in section 1.3, in combination with the explanation goals we defined in section 2.3. All desiderata are discussed below.

* Model-agnostic Can be applied to any model
* Interactive Target output can be chosen by the user
* Valid Target output must always be reached
* Sparse Minimal feature differences between input and counterfactual
* Proximal Minimal distance between input and counterfactual
* Realistic Plausible combinations of feature values in counterfactual

As we are looking to continue development on the machine learning model, a *model-agnostic* explanation method is preferred. In this way, the same explanations can be generated regardless of the underlying algorithm. We assume to have access to the predictions of the model.

Most existing methods assume a binary case, and thus can assume that the target output is the opposite of the current prediction. In this problem however, the range of possible values is 1-5. It is not always straightforward to pick the next-best option; different users determine different ranges of possibilities. We therefore let the user pick the target through an *interactive* prompt.

It should be possible to generate a counterfactual for any input. If the closest counterfactual example seems incomparable to the input profile, that shows a limitation of the dataset. This is not inherently bad; it could even provide the user some insight in how the model works. We have designed counterfactual targets to be integers between 1-5 to match directly with the NOC that DNA experts have to report. Since the current model uses regression, we consider instances with a rounded-off prediction that match the target to be *valid* counterfactuals.

*Sparsity* is encouraged to prevent users from experiencing cognitive overload. We know that humans pick explanations in a biased way [22], meaning that if many options are available, only a few will be selected. The number of different feature values between the input and counterfactual can be counted using norm as shown in Equation 1.

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

Where represents the number of features, the profile to be explained, and the counterfactual profile.

For the distance between the input and counterfactual, we first analyzed the underlying data. The choice of distance should be catered towards the problem [38]. As our dataset has outliers, and most features are not normally distributed (see Supplementary Figure 3 and 4), distance is more appropriate. With distance, outliers can get blown out of proportion. Though many counterfactual methods scale distance by the Median Absolute Deviation (MAD) [38, 40, 45, 46, 48], this is not appropriate for the current dataset because not all features are normally distributed. If a feature with much larger variation than the MAD were to be scaled this way, the distance score would be dominated by that feature. Therefore, we scale with each feature’s range to minimize the influence of different ranges, variations, and distributions [37, 39]. This is more robust for unscaled and unnormalized features with lots of outliers, which is the case in this dataset. This distance measure is shown in Equation 2.

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

Where represents the range of the -th feature, the number of features, the profile to be explained, and the counterfactual profile. It has an additional property that . It can also be used alongside categorical variables by replacing with .

Since none of the features of a DNA profile can be changed to reach an alternative prediction, actionability is not a goal of this method. Similarly, we do not strive to present a set of diverse counterfactuals as diversity is often encouraged for a similar purpose as actionability; to provide a user with multiple routes to reach the different outcome [38, 42, 48]. Moreover, presenting multiple, possibly contradicting examples does not seem like a user-friendly introduction to counterfactual explanations.

The desiderata discussed so far have been covered consistently in the literature. For realism, there is not such a proper definition. Within the problem of NOC estimation, it is essential to present the user with data that is plausible. None of the sampling methods discussed in section 1.3, are automatically suitable for datasets with correlated features, as they would produce unlikely feature combinations. For example, a TAC of 150 is impossible in combination with a MAC of 2 for this kit (from which 23 loci are used in this study), even though these are both normal feature values when looking at the feature distributions. Other approaches that place constraints on the sampled data are too time-consuming, as modelling the relationships between features is not trivial. Since the features might also change in the future, it is not practical to invest time in modelling their relations currently. Instead, we turn towards the training data which inherently consists of the most realistic instances to use. We therefore regard this a good starting point for our explanations.

## Realistic Counterfactuals (ReCo)

To fulfil all previously defined desiderata, we developed an algorithm called Realistic Counterfactuals (ReCo). Instead of generating data and then filtering instances that are infeasible with respect to the training data, ReCo starts with the training instances and forms them into sparser counterfactuals. ReCo therefore consists of two parts: First, the most suitable counterfactual training instance is found. Second, that counterfactual training instance is made sparser by applying a filter.

**Finding the most suitable counterfactual** **training instance:** From the input profile and its prediction , where can be any machine learning model, the user defines a target prediction . ReCo then finds all instances from the training data with the target prediction . This prediction must match with their ground truth NOC so that no incorrect predictions are presented as examples.

ReCo then finds the set of non-dominated instances with regards to sparsity and distance. By minimizing both objectives simultaneously, the obtained Pareto set of counterfactuals has optimal trade-offs between the two scores [37]. As we intend to present a single counterfactual, we select the median instance from this set which balances the two scores best as can be seen in Equation 3.

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

Where  and are defined in Equation 1 and 2 respectively. Additional objectives could be added if deemed important in the future, and the selection from the set can be adjusted if a certain score is preferred over another. Objectives can also be compared without any normalization as is required with for example a weighted sum where balancing scores is dependent on their variance and mean [64, 65].

The counterfactual instance is part of the training data, making it a realistic data point to present. However, such an instance has the following disadvantages:

* Lack of sparsity: the training instance has many different feature values as compared to the profile we want to explain.
* Lack of relevance: not all of these differences are informative to arrive at the target prediction.

ReCo tackles both of these issues by applying a filter to the found counterfactual instance, selecting only the most relevant feature value changes.

**Filtering the counterfactual training instance:** Filtering is defined by the following five steps. Table 1 shows a practical example.

1. Start by finding the set of features that have different values between the input and the counterfactual . The size of this set can be a maximum of , the number of features of which an instance consists. In Table 1, there are three features in this set.

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

1. Compute the SHAP values for both the input instance and the counterfactual instance, per feature in . Subtract the SHAP values of the input instance from the SHAP values of the counterfactual instance. This set is then sorted by the elements’ magnitudes. This gives us an impression of which changes in feature values from the input instance to the counterfactual instance have impacted the change in prediction the most. The biggest positive or negative SHAP changes have likely made the most impact on the change in prediction. In Table 1, the SHAP change of Feature 1 is largest, while it is the smallest for Feature 3.

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

1. To make the counterfactual instance sparser as compared to the input instance, we need to remove the irrelevant feature differences. If the prediction goes down from the input to the counterfactual, or becomes more negative, we expect the features with negative SHAP change to be most relevant. On the other hand, positive SHAP changes are defined to be with the change in prediction in this case. This is listed in the bottom row of Table 1; the change in Feature 2 is . We also include very small SHAP changes such as for Feature 3. These feature differences are most likely not relevant to help reach the counterfactual prediction, and could therefore possibly be filtered from the counterfactual instance.

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

1. The next step is to check if the feature differences with SHAP change can be removed. ‘Removing’ in this context means that the feature value of the counterfactual is replaced with the feature value of the input instance . If the prediction of this filtered counterfactual stays the same as the target , it is labelled as .

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

1. Once removing the next feature difference causes a different outcome than the target prediction, filtering stops. All irrelevant features differences are filtered from the counterfactual so that the final counterfactual is defined as:

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Feature 1 | Feature 2 | Feature 3 |
| SHAP value in input | 0.300 | -0.200 | 0 |
| SHAP value in counterfactual | 0 | -0.150 | -0.001 |
| SHAP change | -0.300 | +0.050 | -0.001 |
| Candidate to be filtered from counterfactual? | No | Yes | Yes |

Table 1: Example of how a counterfactual is filtered. The input instance has a prediction of 4, and the counterfactual has a prediction of 3. Therefore, the direction of the change in prediction is negative. Features 1-3 are the features that differ between the input and counterfactual. Their SHAP values are calculated for both the input and the counterfactual. For Feature 1, the SHAP change is negative, matching the direction of the change in prediction. In contrast, the SHAP change in Feature 2 is positive, and the SHAP change in Feature 3 is small. These last two differences in feature values are therefore likely not relevant to the counterfactual prediction, and thus are candidates to be filtered.

Note that even though we directly use SHAP values to determine whether or not a feature value can be ‘removed’, we are aware that these SHAP values can be underestimated for correlated features. However, ReCo mainly relies on the direction of the SHAP value, so whether it positively or negatively contributes to the prediction. The SHAP values will not become negative while the true value is positive. Therefore, these inaccuracies are not as important to our method. Still, if a feature value difference is marked to be irrelevant though it was impactful for the model, ReCo always checks the prediction before removing it from the counterfactual.

Although the current implementation of ReCo is used for regression, it can be used for classification as well. In this case, we do not have to consider the direction of the change in prediction, we only determine if the change in feature value corresponds to more positive SHAP values for the counterfactual class. If that is the case, the counterfactual feature value is kept, otherwise the input value could remain.

## Realism score

We present a novel realism score which can be used to evaluate counterfactuals. This score assesses whether a generated counterfactual has feasible combinations of feature values in relation to the training data. It is calculated as follows:

1. When the dataset is loaded, a list is generated for each feature that ranks all other variables according to their correlation with the feature.
2. When a counterfactual is found, each feature that has a different value than the original instance is assessed. We will refer to this feature under investigation as .
   1. The feature’s top correlated variable is looked up from the list in step 1.
   2. We check that the value in combination with the value exists in the training data. If so, add 1 to the realism score. If not, add 0.
   3. If was also part of the set of features that differs between the original and the counterfactual instance, we return to step a. and pick the next most correlated feature with to be . In this way, the score is always grounded in the values of a real instance.
   4. The total realism score is normalized by dividing by the number of features that were scored.

Please refer to Figure 1 for an example. In this case, instances only consist of a TAC and a MAC value. The counterfactual only has a different TAC value from the original instance, so we need to check if that generated TAC value is plausible. The most highly-correlated feature to the TAC is the MAC. We assess if the combination of TAC = 30 (from the counterfactual) with MAC = 6 (from the original) exists in the training data. Since it does not exist, the realism score is incremented by 0. The MAC feature is not part of the differences between the counterfactual and the original, so the algorithm terminates. The final realism score for this counterfactual is 0.

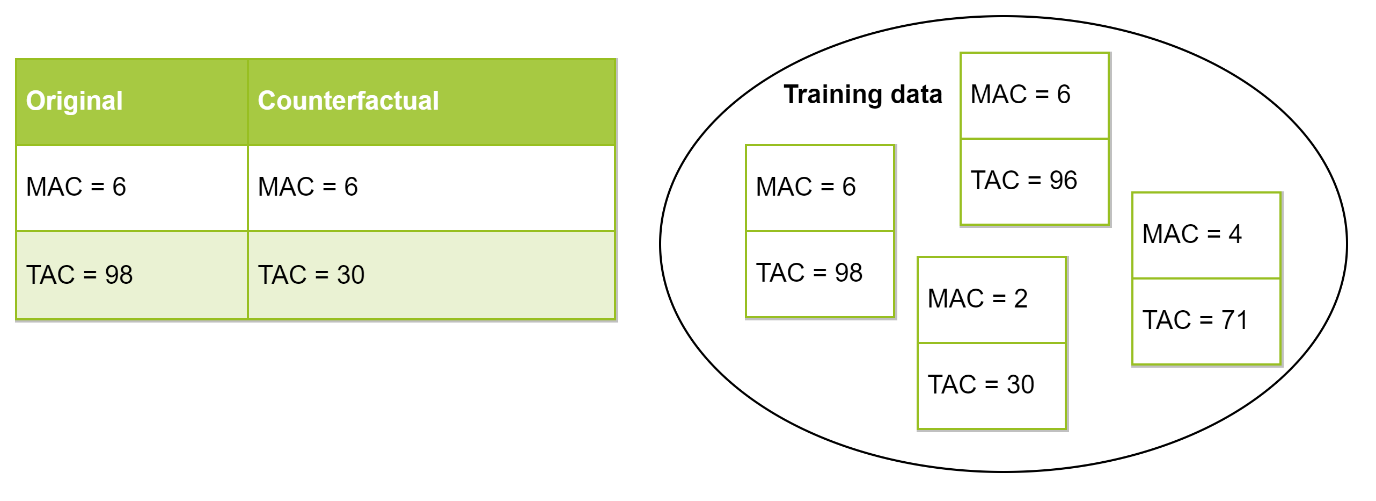


Figure 1: Example of a counterfactual that receives a realism score of 0; the proposed counterfactual contains a feature combination that does not occur in the training data.

## Set-up quantitative evaluation ReCo

To determine the quality of ReCo, we have assessed it on the metrics defined by the desiderata described in section 2.4. As our method is model-agnostic and valid by design, and interactivity is a built-in feature, we chose to focus on the three remaining metrics of sparsity, proximity and realism. Sparsity and proximity seem to be a standard for evaluation of counterfactuals [36, 37, 39, 40, 42, 44-46], whereas the metric for realism is not as clearly defined. Proximity to the training data is often used as a score of realism, though we argue that our realism metric defined in section 2.6 reflects this purpose better. We will present both of these for comparison. To re-iterate:

* Sparsity concerns the number of feature differences between the input and counterfactual and is measured using distance (Equation 1).
* Proximity relates to the distance between the input and counterfactual. We measure this according to range-normalized distance (Equation 2).
* Proximity to the training data is the distance between the counterfactual and the closest training instance. We measure this according to range-normalized distance (Equation 2).
* Realism measures if the feature combinations of the counterfactual are present in the training data. It is measured according to the realism score proposed in section 2.6.

These metrics are used to compare ReCo against three other counterfactual methods [40]. As constraints, we have chosen methods that are model-agnostic, suitable for regression, and suitable for numeric tabular data. *WhatIf* is our own implementation of Google’s What-If tool for searching the closest counterfactual from the training data [53]. We implemented it with our range-normalized distance. *DiCE random* is a sampling approach that generates counterfactuals from the input by randomly sampling its feature values [40]. It starts from the input instance, and randomly picks a feature to be given a sampled value until the target prediction is reached. For this implementation, we used default parameters and set the target prediction between and . The algorithm automatically takes the minimum and maximum values of each feature into account. Lastly, we compare with *DiCE genetic* which implementation is similar to GeCo [42], as it generates counterfactuals from using a genetic algorithm. The algorithm starts from training instances with the target prediction, and *evolves* them to form new samples. When generating a new instance, two training instances are used as its *parents*. This means that for each feature, it can either take the value of instance 1, instance 2 (*crossover*), or a random value is assigned (*mutation*). Through selection of the best instances with respect to sparsity and proximity, a counterfactual is found. We also used the default implementation for DiCE genetic.

For all these methods, the target is set to the second most likely prediction so that the process runs automatically. For instance, if the test prediction is 3.2, the counterfactual target is set to 4.

## Set-up visualization

We incorporated both SHAP values, and the counterfactual example generated by ReCo into a single figure so that the user can understand the main reasons for the original prediction, along with how a different outcome could have been achieved. We believe this is the first visualization to unify counterfactuals with SHAP values. The following requirements were considered from conferring with the consulted DNA experts from the NFI in addition to some desiderata already expressed in the literature [66, 67].

First of all, the visualization is *consistent*. Each profile is presented in the same format; SHAP values on the left and a counterfactual example on the right. The same features will always be on the same location as well. This consistency helps users reach some level of familiarity with the visualization over time as it allows for comparison between profiles.

The feature values are also plotted on a normalized scale to get a visual representation of how large a value is compared to the range of possible values. For normalization, we used a quantile transform as this maps all feature values between 0-1 while spreading out the most frequent values [68]. As we have observed in Supplementary Figure 3 and 4, many features have a skewed distribution and contain outliers. For the visualization, that would make some values difficult to distinguish since the scale is warped by outliers. The quantile transform smooths the relationship between the observations, making the variation between the more common values more evident.

Secondly, the explanation is *contextualized* with informative text about the current prediction, and the conditions of the two parts of the explanation. In this way, the user understands for which conditions the explanation holds. By encoding the two separate explanations with different color palettes, a distinction is made between the SHAP values and the counterfactual. Only the counterfactual differences will be shown with arrows as they indicate changes. The used color palettes are specifically chosen to be *accessible* as they are distinguishable to the color-blind [69].

Lastly, some *interactivity* is introduced by letting the user choose the counterfactual target.

## Set-up user study

It was important to evaluate the visualization from the perspective of the end-users, an aspect often brushed over in XAI studies [56, 59]. The explanation was specifically designed for DNA experts within the context of NOC estimation, so we invited DNA experts from the NFI to participate in a user study.

We did not use this survey to see if users can more accurately determine the NOC as this is the experts’ initial introduction to any XAI implementation, and as such require more training and experience to properly use it as a decision-making tool. The data on which the explanation is based is also not fully understandable as many of the features are too abstract for users to see how they relate to NOC estimation. Instead, the evaluation was set up around two simpler aspects; the first was to see if users can gain insight into the predictions of the model, and by extension, if that information helps regulate the users’ trust. The second aspect concerned how user-friendly the explanation is.

For the exercise on trust, we selected two exemplary profiles for two use-cases. Profile 1 was fairly simple for the model to predict, where we intended the explanation to increase trust in the model prediction. Profile 2 was difficult for the model, leading to an erroneous prediction. In this case the explanation was meant to make the user doubt the model prediction. We measured trust with two questions:

1. Which number(s) of contributors do users consider?
2. Do users think that the prediction is correct?

As a baseline, we ask these questions when users are only presented with the prediction.

Then we ask them once again after a state-of-the-art explanation, and once after our visualization. In this way, we compare against readily available explanations. If users trust the prediction more after seeing the explanation, we expect them to be able to pinpoint the NOC more, and think the prediction is (more) correct. For profile 1, we compared our visualization against a SHAP force plot [70]. As SHAP is designed for users to understand “why a model makes a certain prediction”, we deemed it fit for the goal of increasing trust. For profile 2, we compared our visualization against a counterfactual table, as this representation is common for counterfactuals [37, 40-42, 44, 45, 47, 53, 56]. As counterfactuals show how a different prediction can be reached, it can decrease trust in the original prediction if that change seems small. To keep the survey short, we did not compare the visualization against SHAP and the counterfactual (CF) table for both profiles.

Before these questions were asked, all explanations were introduced with a video, visualization and bullet points to ensure that the participants understand the presented information. With a qualification test, we checked that the participants had completed the introduction.

Within the section about user-preference, we asked users to pick their favorite explanation based on three aspects: *ease of use* (how easily users could find the relevant information), *appeal* (how nice users thought it was to use), and *completeness* (how well users could form a total picture of the prediction). The aim was to determine if the participants had an absolute preference for any of the explanations they had seen.

# Results and discussion

This work presents two distinct products; a new counterfactual method ReCo, and a visualization combining the results from ReCo with SHAP values. We show the results of the objective evaluation of ReCo, after which we present the visualization and the corresponding user study results. As both of these aspects specifically target NOC estimation, only the dataset described in section 2.1 was used for the evaluation.

## Quantitative evaluation ReCo

The obtained scores on the test data for the four methods can be found in Figure 2.

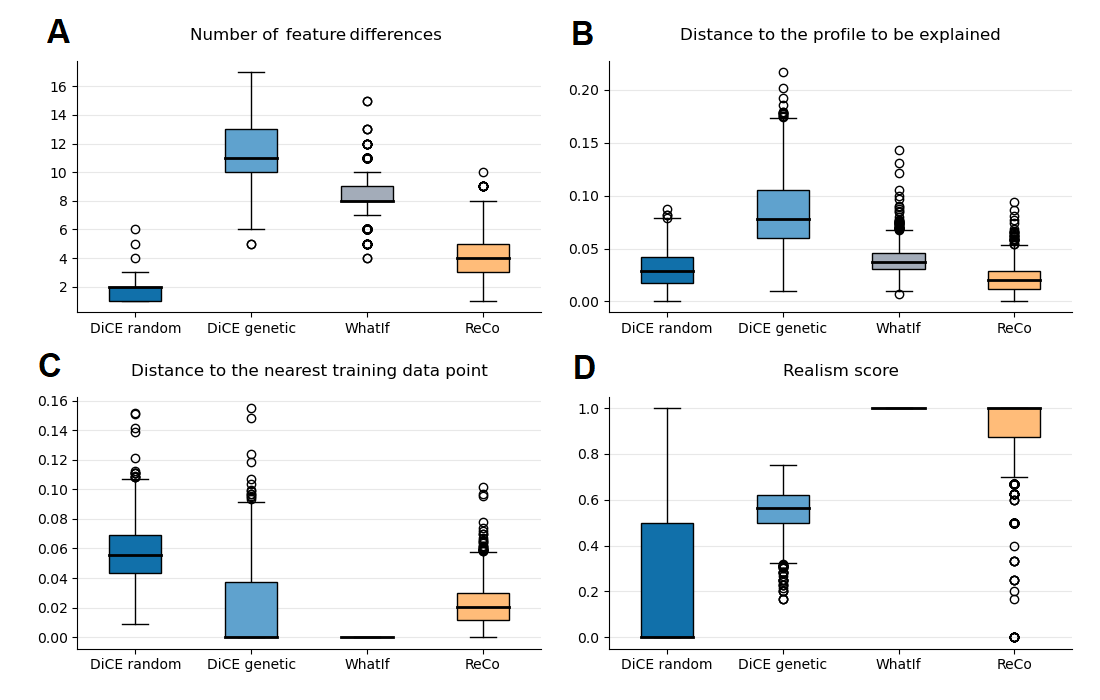


Figure 2: Quantitative evaluation of ReCo in comparison to WhatIf, DiCE random and DiCE genetic on four different metrics; the number of feature differences (A), the distance to the input (B), the distance to the training data (C), and realism (D).

The WhatIf method could be seen as a baseline, using only existing training examples as counterfactuals. Its realism score and distance to the training data are therefore perfect, but it suffers from many feature differences and a higher distance score due to the sparsity of the training data.

While DiCE random performs best in terms of the number of feature differences, and quite well on distance, it performs poorly on realism and is the furthest away from the training data. This is because DiCE random starts from the original instance, and perturbs a random feature until the target prediction is reached. This strategy helps keep the number of feature differences and the overall distance score low, but does not in any way account for the relations between the features. This makes this method inappropriate for our dataset.

An improvement can be seen when the genetic version is used (DiCE genetic); the median realism score almost hits 0.6, and the distance to the training data is practically zero. We can attribute these better scores to the combination of profiles from the training data. However, this crossover still simply combines the feature values of two instances, which can create unlikely feature combinations. Mutation has a similar effect. It is interesting to see that this algorithm leads to significantly larger distances and more feature differences. It could be that by combining training instances, the newly formed amalgamation becomes more generalized for the target prediction and as such, moves further away from the input. One final aspect to note about both DiCE techniques is that they failed to generate a counterfactual for about 2% of the test inputs, thereby failing our desideratum for validity.

ReCo seems to score relatively well on all four metrics. As the method first finds the closest and most sparse training instance, this is an inherently realistic starting point. Because both sparsity and distance are optimized, in contrast to WhatIf, which only minimizes the distance, the obtained data points might already be sparser. Then by filtering, these two scores go down further whenever it is possible. The reason that we can filter so many differences without moving too far away from the training data and producing unlikely feature combinations, could be explained by a number of factors. First of all, the filter removes small or counterintuitive differences that are likely insignificant to the model. These limited differences will not cause the counterfactual to move too far away from the training data. Secondly, the features that are filtered could have little discriminatory power between the original and target output. This could be because their values are similar for instances of the original and target prediction in the training data. For example, if for both the original and the target NOC, the median of a feature in the training data is equal, it possibly has little discriminatory power between the two outcomes. As a final remark, we note that there are some outliers that score low on realism. When more feature differences are filtered, the likelihood increases that the values from those input features do not match with the leftover counterfactual feature values. Also note that the current realism metric is strict; it does not check if a feature value is close to known combinations in the training data, the values must match 100%. It might be interesting to see if adding a tolerance to this metric creates a more nuanced score, but we leave this for future work.

## Visualization

The visualization for the explanation of a single DNA profile prediction is depicted in Figure 3.

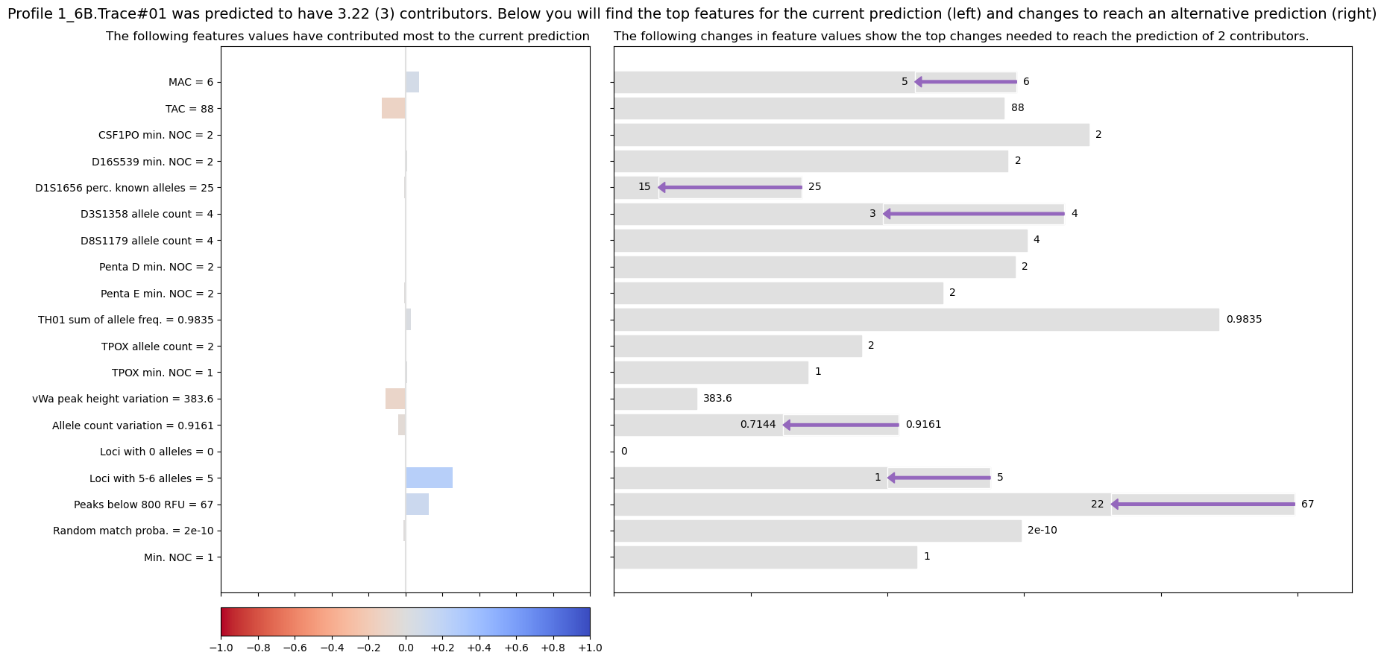
**

Figure 3: Visualization for the explanation of a profile with 3 contributors, that was correctly predicted to have 3 contributors (profile 1 in the user study). Its feature values are listed on the left and plotted on the right. SHAP values are depicted on the left with red and blue bars, and a counterfactual example generated by ReCo for a prediction of 2 contributors is shown on the right with arrows.

The top line informs the user about the current profile and what the model’s prediction is. The decimal output of the regression model can be used to give the user an impression about the certainty of the prediction; any value ending in .49 or lower will be rounded down; any value ending in .50 or higher will be rounded up. The top line further includes a summary of what information can be found in the figure. On the left-hand side, all 19 features and their values as defined by this profile are listed. These same feature values also appear in the right section as normalized grey bars, aligned with the feature values on the left.

The SHAP values are visible in the left section; red bars mean that the feature values pushed the prediction down, while blue bars represent feature values that pushed the prediction up. Note that SHAP starts from a base prediction of 3 contributors; this is the mean outcome value out of the 1-5 range. Starting from a prediction of 3, adding the SHAP values together forms the current prediction of 3.22. In this case, there are twelve feature values influencing the decision, though only about six or seven are clearly visible. We intentionally only added the SHAP value legend at the bottom as we do not want the users to focus on the exact values, but on the direction and relative size instead as it is possible that the exact values are underestimated due to the correlations between features. For this prediction, the model noticed this profile’s higher values of *MAC*, *loci with 5 and 6 alleles*, and *peaks below 800 RFU* (the stochastic threshold that applies to the data in this study) as indicators for more contributors. More alleles per loci indeed imply more donors, and lots of low peaks indicate a profile with less quantity of information which can be more prevalent with higher-order mixtures. In contrast, the TAC and peak height variation at locus vWA have lower values that typically occur in lower-order mixtures.

To generate a counterfactual explanation for this profile, we have set the target at 2 contributors. As often a minimum NOC is reported, it might be interesting to be able to rule out this option, and instead go with the prediction of 3 contributors. Within the application, the user can normally first explore the factual explanation consisting of the features and SHAP values before choosing a counterfactual target. The counterfactual that ReCo has found for this explanation has six lower feature values as denoted by the purple arrows. If any features would need to increase their value, the arrow would be olive-colored. The arrows demonstrate all the changes that are required to reach the target prediction. Three of the arrows relate to the three feature values that we discovered were pushing the prediction up (MAC, loci with 5 and 6 alleles, and peaks below 800 RFU). By adjusting these values, along with the other three feature values, a lower prediction can be achieved. It seems that to reach this target of 2 contributors, many features need to change, and by a large extent. This can provide an indication that the model is fairly certain that the NOC is not 2.

## User study results

The survey was quite extensive since it includes introductions for three types of explanations. As such, we expected that the response would be limited. In total, 7 useable answers were collected from DNA experts of several age groups from 18 to 54. One response had to be eliminated as they failed the qualification tests. Because this small group is not representative of the entire group of experts at the NFI, we treated the obtained results as a subjective collection of the participants’ opinions. The results of the first exercise about trust can be found in Figure 4. It presents if users gained or lost trust in the prediction after seeing the two explanations for profile 1 and 2.

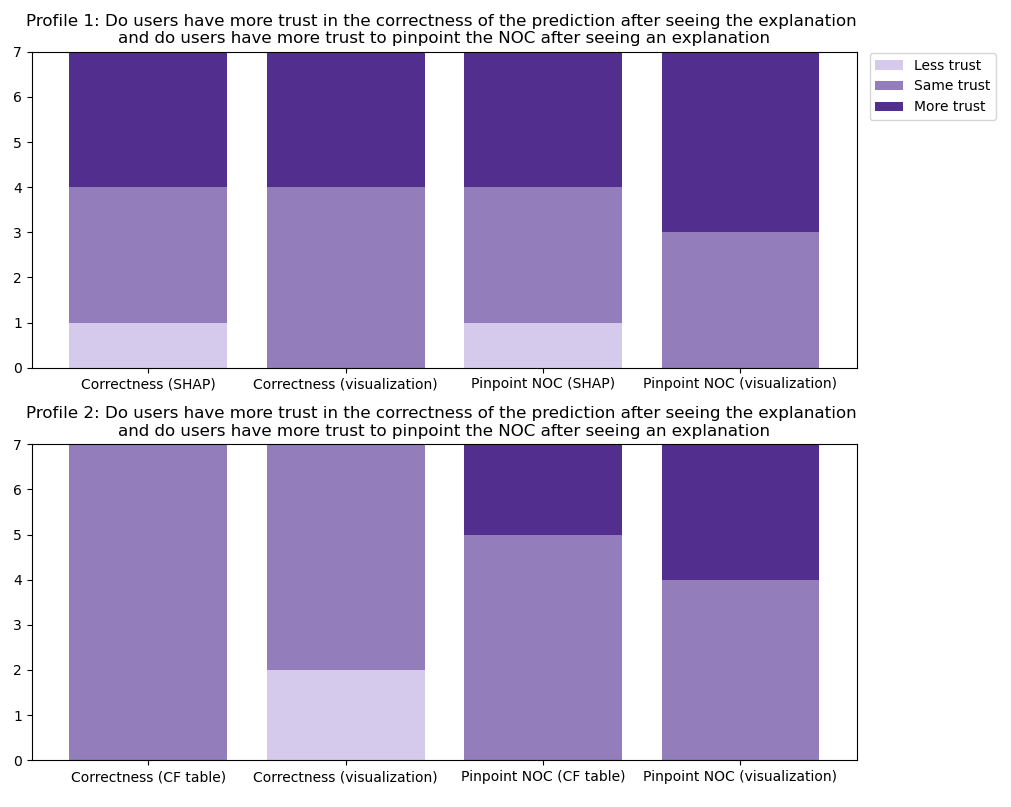


Figure 4: Results from the user study trust exercise. For profile 1 (top), it shows the influence of seeing a SHAP explanation in comparison to our visualization, on trust in the correctness of the model prediction, and on the users trust to pinpoint the NOC. For profile 2 (bottom), it shows the influence of seeing a CF table explanation in comparison to our visualization, on trust in the correctness of the model prediction, and on the users trust to pinpoint the NOC.

When the model is fairly certain about the prediction (profile 1, see Figure 3), seeing any explanation makes some users (3/7) both gain more trust in the prediction, and enables them to pinpoint the NOC better. This can be partly attributed to the fact that the feature values of the profile were first presented with the explanations. As such, a few participants (2/7) became more certain of a certain NOC because of the feature values, not because of what the SHAP force plot was trying to communicate. SHAP can induce some confusion seeing that 1 user gained less trust in the correctness of the prediction and considered a wider range of contributors. The way that the bars of the SHAP force plot work against each other, was not intuitive for some users (2/7) as they expressed difficulty with understanding it. For the visualization, most users (4/7) noted that a lot of change was required to reach the prediction of 2 contributors, and therefore dropped this outcome from consideration. One user thought that the visualization presented similar information to reach a prediction of 2 contributors as they would have thought, thereby increasing their trust in this explanation.

When the model is uncertain or incorrect (profile 2, see Figure 5), the CF table had no effect on how users perceived the correctness of the prediction, while the visualization made some users (2/7) trust the prediction less. From the additional textual input, a lot of users did mention that they started to doubt the prediction (5/7), but not all of them changed their answer. The remarks that participants made with the visualization related most frequently to the fact that only minor changes are required to change the prediction to 3; changing the TAC from 98 to 96. DNA experts would not make a different decision depending on such a small difference in TAC value, they always use ranges. As such, the experts began to doubt whether or not the model made a correct decision. One participant even noted that for a TAC of 98, there can be 2 artefact peaks and that therefore they thought the prediction was incorrect. The visualization in general made users more confident to pinpoint the NOC as they considered less options than with the SHAP or CF table explanations (more trust to pinpoint NOC in Figure 4).

In short, it seems that our visualization provides some insight into the model, which influences how users view and trust the prediction. The users might even feel more equipped to make a narrower estimation of the NOC. Our visualization seems to be less confusing than a SHAP force plot, and more informative than presenting a counterfactual in a table. Note that because the study was limited, these results are an indication of the participants opinions and might vary once repeated with a larger group of people. Since many of the features are quite difficult to understand at this point, we did not evaluate on how well the explanation can help experts in determining the NOC. Such an evaluation can be done in the future when the model and features have been developed further.

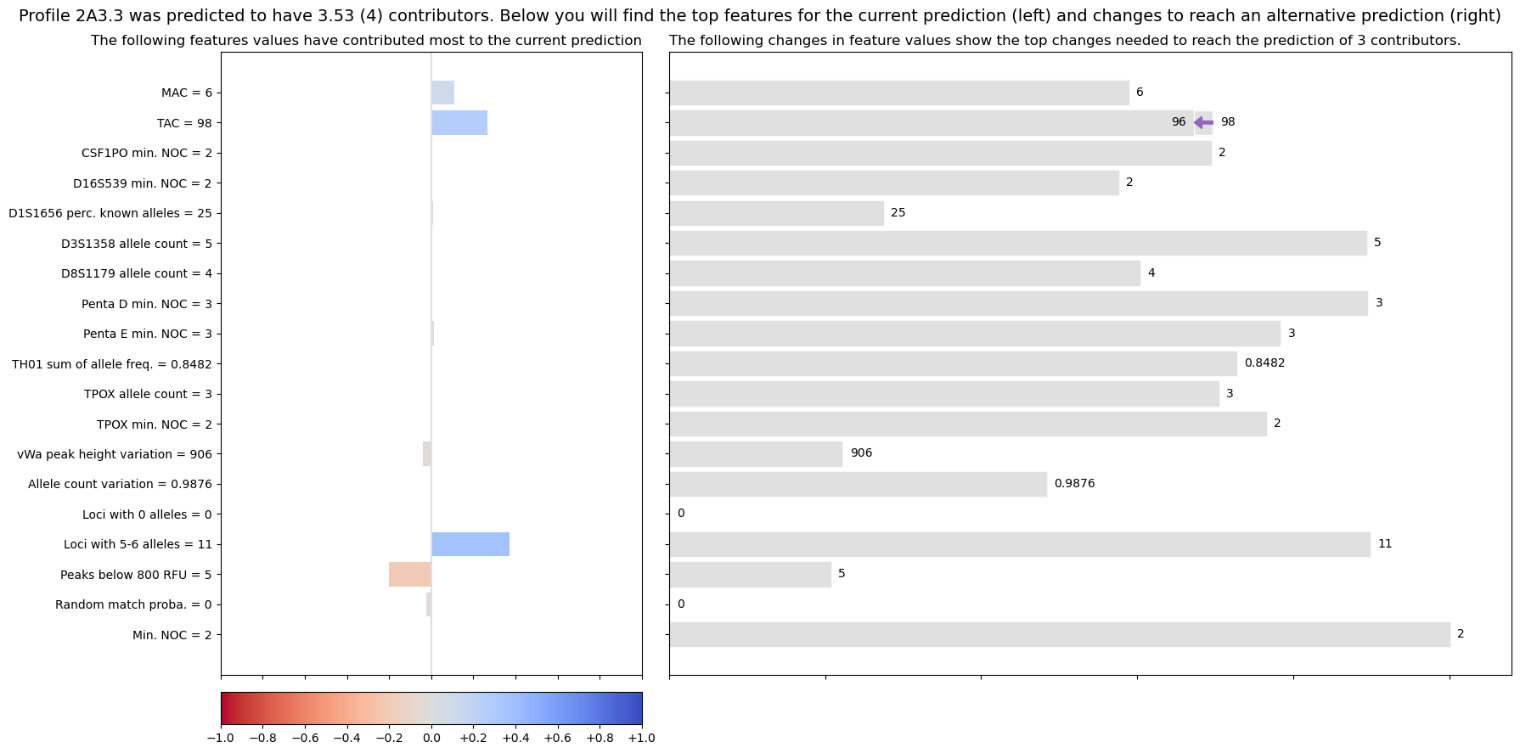


Figure 5: Visualization for the explanation of a profile with 3 contributors, that was incorrectly predicted to have 4 contributors (profile 2 in the user study). Its feature values are listed on the left and plotted on the right. SHAP values are depicted on the left with red and blue bars, and a counterfactual example generated by ReCo for a prediction of 3 contributors is shown on the right with arrows.

The results of the second task about user preferences can be found in Table 2. Our compound visualization scored the best out of the three options, though some users had a preference for SHAP for its ease of use. The experts who preferred our visualization, mostly chose it because of its visual representation, the amount of available information and because the information was easy to find.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ease of use | Appeal | Completeness |
| SHAP force plot | 2 | 2 | 1 |
| Counterfactual table | 0 | 0 | 1 |
| Compound visualization | 5 | 5 | 5 |

Table 2: Results of user preferences. The numbers represent how many users selected each type of explanation they preferred in terms of ease of use, appeal and completeness.

## Future work

As the DNA experts we consulted at the NFI have indicated, the features on which the explanation are based are still difficult to comprehend. Some of the variables also encode redundant information such as *[locus] min. NOC* which is the same as *[locus] allele count* divided by 2 and rounded down. It seems that the features can be further investigated on redundancy, perhaps re-designed and expanded upon. For one, to ensure that they are understandable to users on how they relate to the NOC estimation task, and secondly that they are as informative to the machine learning models as possible.

It might benefit the NOC estimation problem to develop multiple binary models that differentiate between just two options; one for 1 or 2 contributors; one for 2 or 3; etc. This could create more specialized models, and thus more specific explanations. We refer to an implementation of such a structure for selecting the most suitable eye-surgery option for a patient [35].

Another direction of interest is to further develop the proposed realism metric. For example, by introducing some matching tolerance with values from the training data, or by comparing more feature combinations than with the top correlated variable. It could also be incorporated into the fitness function of a genetic sampling algorithm. In this way, the algorithm can optimize on generating counterfactuals with realistic feature combinations as well.

# Conclusion

This study describes an implementation of XAI for predictions of the number of contributors of DNA profiles which can be applied to any type of machine learning model. The explanation consists of SHAP values and a counterfactual example incorporated into a compound visualization, which has been evaluated by a small group of DNA experts. From their observations, it seems that the visualization provides some insight into the predictions of the model. We further present a method for finding realistic counterfactuals, called ReCo. ReCo creates a counterfactual by first obtaining the most suitable training instance, and then filtering the irrelevant feature value differences between this instance and the input. This produces examples that have fewer feature differences than by using training examples, and are more plausible than counterfactuals generated by sampling-based approaches. To the best of our knowledge, ReCo is the first method that can handle correlated data automatically. Additionally, a realism metric was defined that scores how plausible counterfactuals are in terms of their feature combinations.

Finally, we hope that this study encourages other implementations of machine learning to incorporate an XAI-component, especially when the users of such models are not familiar with the underlying concepts of machine learning.

# Acknowledgements

We are thankful to Jerry Hoogenboom for generating the 5000 samples, Jennifer van der Linden for getting us up to speed with the data and model, the NFI group BiS for participating in the user studies, and Jason van Breukelen for providing general feedback.

# References

[1] M.D. Coble, J.A. Bright, J.S. Buckleton, J.M. Curran, Uncertainty in the number of contributors in the proposed new CODIS set, Forensic Science International: Genetics 19 (2015) 207-211.

[2] C.C.G. Benschop, H. Haned, L. Jeurissen, P.D. Gill, T. Sijen, The effect of varying the number of contributors on likelihood ratios for complex DNA mixtures, Forensic Science International: Genetics 19 (2015) 92-99.

[3] H. Haned, L. Pène, J.R. Lobry, A.B. Dufour, D. Pontier, Estimating the Number of Contributors to Forensic DNA Mixtures: Does Maximum Likelihood Perform Better Than Maximum Allele Count?, Journal of Forensic Sciences 56(1) (2011) 23-28.

[4] A. Biedermann, S. Bozza, K. Konis, F. Taroni, Inference about the number of contributors to a DNA mixture: Comparative analyses of a Bayesian network approach and the maximum allele count method, Forensic Science International: Genetics 6(6) (2012) 689-696.

[5] D.R. Paoletti, D.E. Krane, T.E. Doom, M. Raymer, Inferring the Number of Contributors to Mixed DNA Profiles, IEEE/ACM Transactions on Computational Biology and Bioinformatics 9(1) (2012) 113-122.

[6] B.A. Young, K.B. Gettings, B. McCord, P.M. Vallone, Estimating number of contributors in massively parallel sequencing data of STR loci, Forensic Science International: Genetics 38 (2019) 15-22.

[7] C.M. Grgicak, S. Karkar, X. Yearwood-Garcia, L.E. Alfonse, K.R. Duffy, D.S. Lun, A large-scale validation of NOCIt's a posteriori probability of the number of contributors and its integration into forensic interpretation pipelines, Forensic Science International: Genetics 47 (2020).

[8] H. Swaminathan, C.M. Grgicak, M. Medard, D.S. Lun, NOCIt: A computational method to infer the number of contributors to DNA samples analyzed by STR genotyping, Forensic Science International: Genetics 16 (2015) 172-180.

[9] C. Benschop, A. Backx, T. Sijen, Automated estimation of the number of contributors in autosomal STR profiles, Forensic Science International: Genetics Supplement Series 7 (2019).

[10] M.D. Coble, J.-A. Bright, Probabilistic genotyping software: An overview, Forensic Science International: Genetics 38 (2019) 219-224.

[11] D. Taylor, J.-A. Bright, J. Buckleton, Interpreting forensic DNA profiling evidence without specifying the number of contributors, Forensic Science International: Genetics 13 (2014) 269-280.

[12] Ø. Bleka, G. Storvik, P. Gill, EuroForMix: An open source software based on a continuous model to evaluate STR DNA profiles from a mixture of contributors with artefacts, Forensic Science International: Genetics 21 (2016) 35-44.

[13] C.C.G. Benschop, J. Hoogenboom, F. Bargeman, P. Hovers, M. Slagter, J. van der Linden, R. Parag, D. Kruise, K. Drobnic, G. Klucevsek, W. Parson, B. Berger, F.X. Laurent, M. Faivre, A. Ulus, P. Schneider, M. Bogus, A.L.J. Kneppers, T. Sijen, Multi-laboratory validation of DNAxs including the statistical library DNAStatistX, Forensic Science International: Genetics 49 (2020) 102390.

[14] C.C.G. Benschop, A. Nijveld, F.E. Duijs, T. Sijen, An assessment of the performance of the probabilistic genotyping software EuroForMix: Trends in likelihood ratios and analysis of Type I & II errors, Forensic Science International: Genetics 42 (2019) 31-38.

[15] T. Bille, S. Weitz, J.S. Buckleton, J.-A. Bright, Interpreting a major component from a mixed DNA profile with an unknown number of minor contributors, Forensic Science International: Genetics 40 (2019) 150-159.

[16] J.S. Buckleton, J.-A. Bright, K. Cheng, H. Kelly, D.A. Taylor, The effect of varying the number of contributors in the prosecution and alternate propositions, Forensic Science International: Genetics 38 (2019) 225-231.

[17] T.M. Clayton, J.P. Whitaker, R. Sparkes, P. Gill, Analysis and interpretation of mixed forensic stains using DNA STR profiling, Forensic Science International 91(1) (1998) 55-70.

[18] 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 Science International: Genetics 43 (2019) 102150.

[19] M.A. Marciano, J.D. Adelman, Developmental validation of PACE™: Automated artifact identification and contributor estimation for use with GlobalFiler™ and PowerPlex® fusion 6c generated data, Forensic Science International: Genetics 43 (2019).

[20] M. Kruijver, H. Kelly, K. Cheng, M.-H. Lin, J. Morawitz, L. Russell, J. Buckleton, J.-A. Bright, Estimating the number of contributors to a DNA profile using decision trees, Forensic Science International: Genetics 50 (2021) 102407.

[21] B. Mittelstadt, C. Russell, S. Wachter, Explaining Explanations in AI, 2018.

[22] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artificial Intelligence 267 (2019) 1-38.

[23] D.V. Carvalho, E.M. Pereira, J.S. Cardoso, Machine learning interpretability: A survey on methods and metrics, Electronics (Switzerland) 8(8) (2019).

[24] A. Barredo Arrieta, N. Diaz-Rodriguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, F. Herrera, Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, Information Fusion 58 (2020) 82-115.

[25] Z.C. Lipton, The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery, Queue 16(3) (2018).

[26] L.H. Gilpin, D. Bau, B.Z. Yuan, A. Bajwa, M. Specter, L. Kagal, Explaining Explanations: An Overview of Interpretability of Machine Learning, 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), 2018, pp. 80-89.

[27] A. Adadi, M. Berrada, Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI), IEEE Access 6 (2018) 52138-52160.

[28] M. Du, N. Liu, X. Hu, Techniques for interpretable machine learning, Communications of the ACM 63(1) (2020) 68-77.

[29] M. Kruijver, H. Kelly, K. Cheng, M.-H. Lin, J. Morawitz, L. Russell, J. Buckleton, J.-A. Bright, Estimating the number of contributors to a DNA profile using decision trees, Forensic Science International: Genetics.

[30] E. Commision, Fostering a European approach to Artificial Intelligence, 2021.

[31] W.J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, B. Yu, Definitions, methods, and applications in interpretable machine learning, Proceedings of the National Academy of Sciences of the United States of America 116(44) (2019) 22071-22080.

[32] R.R. Fernández, I. Martín de Diego, V. Aceña, A. Fernández-Isabel, J.M. Moguerza, Random forest explainability using counterfactual sets, Information Fusion 63 (2020) 196-207.

[33] S. Lundberg, S.-I. Lee, A Unified Approach to Interpreting Model Predictions, 2017.

[34] S.M. Lundberg, B. Nair, M.S. Vavilala, M. Horibe, M.J. Eisses, T. Adams, D.E. Liston, D.K. Low, S.F. Newman, J. Kim, S.I. Lee, Explainable machine-learning predictions for the prevention of hypoxaemia during surgery, Nat Biomed Eng 2(10) (2018) 749-760.

[35] T.K. Yoo, I.H. Ryu, H. Choi, J.K. Kim, I.S. Lee, J.S. Kim, G. Lee, T.H. Rim, Explainable Machine Learning Approach as a Tool to Understand Factors Used to Select the Refractive Surgery Technique on the Expert Level, Transl Vis Sci Technol 9(2) (2020) 8.

[36] Y. Ramon, D. Martens, F. Provost, T. Evgeniou, A comparison of instance-level counterfactual explanation algorithms for behavioral and textual data: SEDC, LIME-C and SHAP-C, Advances in Data Analysis and Classification 14(4) (2020) 801-819.

[37] S. Dandl, C. Molnar, M. Binder, B. Bischl, Multi-Objective Counterfactual Explanations, in: T. Bäck, M. Preuss, A. Deutz, H. Wang, C. Doerr, M. Emmerich, H. Trautmann (Eds.) Parallel Problem Solving from Nature – PPSN XVI, Springer International Publishing, Cham, 2020, pp. 448-469.

[38] S. Wachter, B. Mittelstadt, C. Russell, Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR, Harvard journal of law & technology 31 (2018) 841-887.

[39] A.-H. Karimi, G. Barthe, B. Balle, I. Valera, Model-agnostic counterfactual explanations for consequential decisions, International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 895-905.

[40] R.K. Mothilal, A. Sharma, C. Tan, Explaining machine learning classifiers through diverse counterfactual explanations, 2020, pp. 607-617.

[41] S. Sharma, J. Henderson, J. Ghosh, CERTIFAI: A common framework to provide explanations and analyse the fairness and robustness of black-box models, 2020, pp. 166-172.

[42] M. Schleich, Z. Geng, Y. Zhang, D. Suciu, GeCo: Quality Counterfactual Explanations in Real Time, 2021.

[43] R. Poyiadzi, K. Sokol, R. Santos-Rodriguez, T. Bie, P. Flach, FACE: Feasible and Actionable Counterfactual Explanations, 2020.

[44] J. Moore, N. Hammerla, C. Watkins, Explaining deep learning models with constrained adversarial examples, 2019, pp. 43-56.

[45] M. Keane, B. Smyth, Good Counterfactuals and Where to Find Them: A Case-Based Technique for Generating Counterfactuals for Explainable AI (XAI), 2020.

[46] R.M. Grath, L. Costabello, C.L. Van, P. Sweeney, F. Kamiab, Z. Shen, F. Lécué, Interpretable Credit Application Predictions With Counterfactual Explanations, ArXiv abs/1811.05245 (2018).

[47] A. White, A. Garcez, Measurable Counterfactual Local Explanations for Any Classifier, ECAI, 2020.

[48] C. Russell, Efficient search for diverse coherent explanations, 2019, pp. 20-28.

[49] S. Rathi, Generating Counterfactual and Contrastive Explanations using SHAP, 2019.

[50] R. Guidotti, A. Monreale, F. Giannotti, D. Pedreschi, S. Ruggieri, F. Turini, Factual and Counterfactual Explanations for Black Box Decision Making, IEEE Intelligent Systems 34(6) (2019) 14-23.

[51] O. Gomez, S. Holter, J. Yuan, E. Bertini, ViCE, 2020, pp. 531-535.

[52] K. Sokol, P. Flach, Desiderata for interpretability: Explaining decision tree predictions with counterfactuals, 2019, pp. 10035-10036.

[53] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viégas, J. Wilson, The What-If Tool: Interactive Probing of Machine Learning Models, IEEE Transactions on Visualization and Computer Graphics 26(1) (2020) 56-65.

[54] S. Barocas, A.D. Selbst, M. Raghavan, The hidden assumptions behind counterfactual explanations and principal reasons, 2020, pp. 80-89.

[55] L. Bertossi, Score-Based Explanations in Data Management and Machine Learning, 2020, pp. 17-31.

[56] A. Adhikari, D.M.J. Tax, R. Satta, M. Faeth, LEAFAGE: Example-based and Feature importance-based Explanations for Black-box ML models, IEEE International Conference on Fuzzy Systems, 2019.

[57] K. Sokol, P. Flach, Conversational Explanations of Machine Learning Predictions Through Class-contrastive Counterfactual Statements, 2018, pp. 5785-5786.

[58] K. Sokol, P. Flach, One Explanation Does Not Fit All: The Promise of Interactive Explanations for Machine Learning Transparency, KI - Kunstliche Intelligenz 34(2) (2020) 235-250.

[59] S. Verma, J.P. Dickerson, K. Hines, Counterfactual Explanations for Machine Learning: A Review, ArXiv abs/2010.10596 (2020).

[60] C.C.G. Benschop, J. Hoogenboom, P. Hovers, M. Slagter, D. Kruise, R. Parag, K. Steensma, K. Slooten, J.H.A. Nagel, P. Dieltjes, V. van Marion, H. van Paassen, J. de Jong, C. Creeten, T. Sijen, A.L.J. Kneppers, DNAxs/DNAStatistX: Development and validation of a software suite for the data management and probabilistic interpretation of DNA profiles, Forensic Sci Int Genet 42 (2019) 81-89.

[61] 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.

[62] A.R. Akula, S. Todorovic, J.Y. Chai, S. Zhu, Natural Language Interaction with Explainable AI Models, CVPR Workshops, 2019.

[63] C. Molnar, G. Konig, J. Herbinger, T. Freiesleben, S. Dandl, C.A. Scholbeck, G. Casalicchio, M. Grosse-Wentrup, B. Bischl, Pitfalls to Avoid when Interpreting Machine Learning Models, ArXiv abs/2007.04131 (2020).

[64] G. Chiandussi, M. Codegone, S. Ferrero, F.E. Varesio, Comparison of multi-objective optimization methodologies for engineering applications, Computers & Mathematics with Applications 63(5) (2012) 912-942.

[65] N. Gunantara, A review of multi-objective optimization: Methods and its applications, Cogent Engineering 5(1) (2018) 1502242.

[66] K. Sokol, P. Flach, Explainability fact sheets: A framework for systematic assessment of explainable approaches, FAT\* 2020 - Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, 2020, pp. 56-67.

[67] K. Sokol, P. Flach, Counterfactual explanations of machine learning predictions: Opportunities and challenges for AI safety, 2019.

[68] s.-l. developers, sklearn.preprocessing.QuantileTransformer. <<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html>>, 2020 (accessed 25-05-2021.).

[69] P. Kovesi, Good Colour Maps: How to Design Them, ArXiv abs/1509.03700 (2015).

[70] S.M. Lundberg, B. Nair, M.S. Vavilala, M. Horibe, M.J. Eisses, T. Adams, D.E. Liston, D.K.-W. Low, S.-F. Newman, J. Kim, S.-I. Lee, Explainable machine-learning predictions for the prevention of hypoxaemia during surgery, Nature Biomedical Engineering 2(10) (2018) 749-760.

[71] M.G. KENDALL, A NEW MEASURE OF RANK CORRELATION, Biometrika 30(1-2) (1938) 81-93.

**Supplementary Material 1: data analysis and sampling**

The original features and their descriptions can be found in Supplementary Table 1 [18]. The feature names were edited to be more descriptive and consistent as they are presented to users.

|  |  |  |
| --- | --- | --- |
| **Original feature name** | **New feature name** | **Description** |
| MAC | MAC | Maximum Allele Count |
| TAC | TAC | Total Allele Count |
| MinNOC\_CSF1PO | CSF1PO min. NOC | Minimal NOC based on locus CSF1PO (allele count at locus CSF1PO / 2, rounded up) |
| MinNOC\_D16S539 | D16S539 min. NOC | Minimal NOC based on locus D16S539 (allele count at locus D16S539 / 2, rounded up) |
| PercAF\_D1S1656 | D1S1656 perc. known alleles | Number of alleles at locus D1S1656 as a percentage of all known alleles at D1S1656 in the allele frequency file |
| AlleleCount\_D3S1358 | D3S1358 allele count | Allele count at locus D3S135 |
| AlleleCount\_D8S1179 | D8S1179 allele count | Allele count at locus D8S1179 |
| MinNOC\_Penta D | Penta D min. NOC | Minimal NOC based on locus Penta D (allele count at locus Penta D / 2, rounded up) |
| MinNOC\_Penta E | Penta E min. NOC | Minimal NOC based on locus Penta E (allele count at locus Penta E / 2, rounded up) |
| SumAF\_TH01 | TH01 sum of allele freq. | Sum of frequencies of the alleles at TH01 defined in the allele frequency file |
| AlleleCount\_TPOX | TPOX allele count | Allele count at locus TPOX |
| MinNOC\_TPOX | TPOX min. NOC | Minimal NOC based on locus TPOX (allele count at locus TPOX / 2, rounded up) |
| stdHeight\_vWA | vWA peak height variation | Standard deviation of peak heights at locus vWA(average variation from the mean peak height at locus vWa) |
| stdAllele | Allele count variation | Standard deviation of the number of alleles per locus (average variation from the mean number of alleles per locus) |
| MAC0 | Loci with 0 alleles | Number of loci with 0 alleles |
| MAC5-6 | Loci with 5-6 alleles | Number of loci with 5 or 6 alleles |
| peaksBelowRFU | Peaks below 800 RFU | Number of peaks below the stochastic threshold of 800 RFU |
| MatchProbability | Random match proba. | Probability of a random Dutch person matching to this DNA profile |
| MinNOC | Min. NOC | Minimal NOC (locus with lowest allele count / 2, rounded up) |

Supplementary Table 1: Overview of the 19 features and their descriptions from the original RFC19 model [18]. New feature names were created to be more consistent with their definitions and each other.

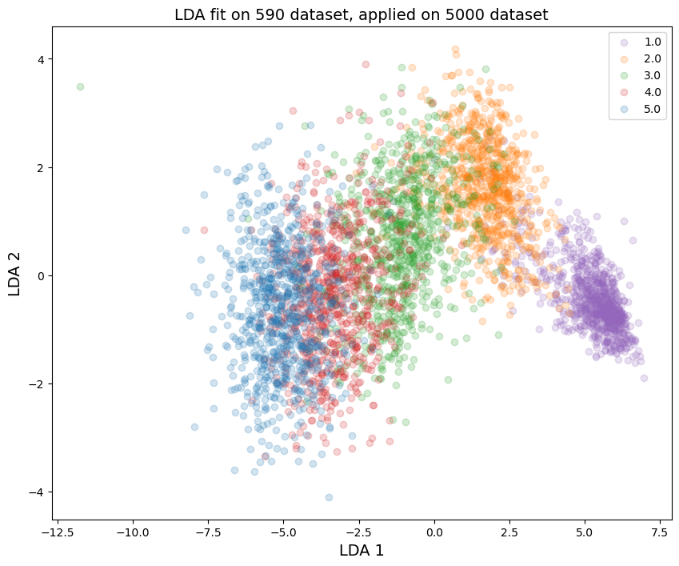
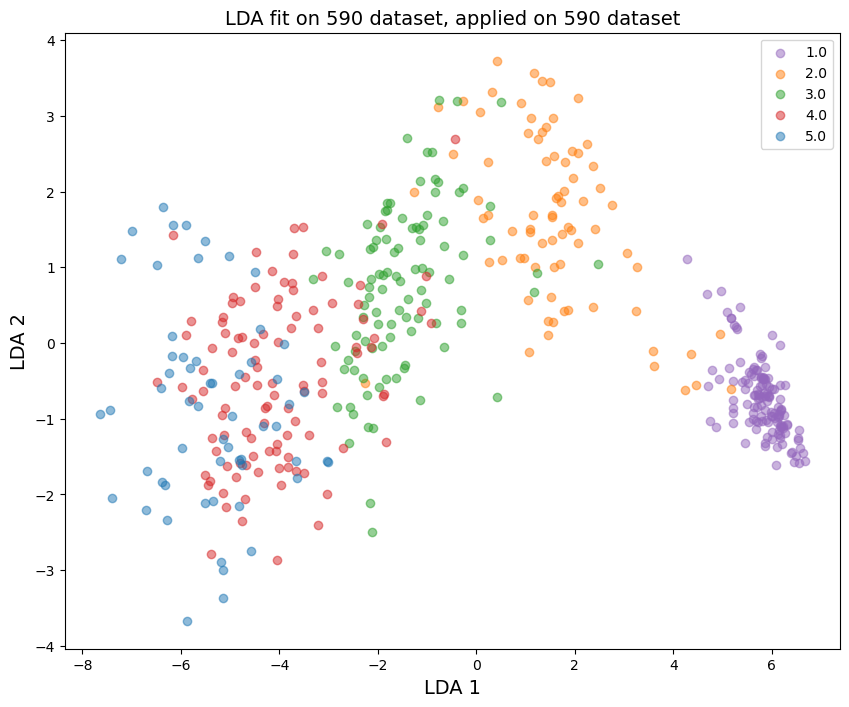
Additional data was sampled to handle the sparsity of the original dataset. Supplementary Table 2 details the parameters used to generate 5000 mixture profiles using a development version of DNAStatistX. For each number of contributors (1-5), 1000 profiles were generated. The used STR kit is PowerPlex Fusion 6CTM (PPF6C, Promega) with dye-specific detection thresholds as used by default at the NFI. Dutch population frequency data was used [61]. After generating a profile, LRs were calculated using each donor in a mixture as the person of interest under H1. Only mixtures for which all donors reached a minimum LR of 1000 were included in the dataset. LR calculations were performed using the true NOC under the propositions, using theta correction of 0.01 and using the kit settings for PPF6C as implemented in DNAStatistX.

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| Drop-in prC | 0.05 |
| Drop-in lambda | 0.01 |
| Average peak heights | (100, 20000) |
| Variation coefficient peak heights | (0.1, 1.0) |
| Degradation | (0.4, 1.1) |

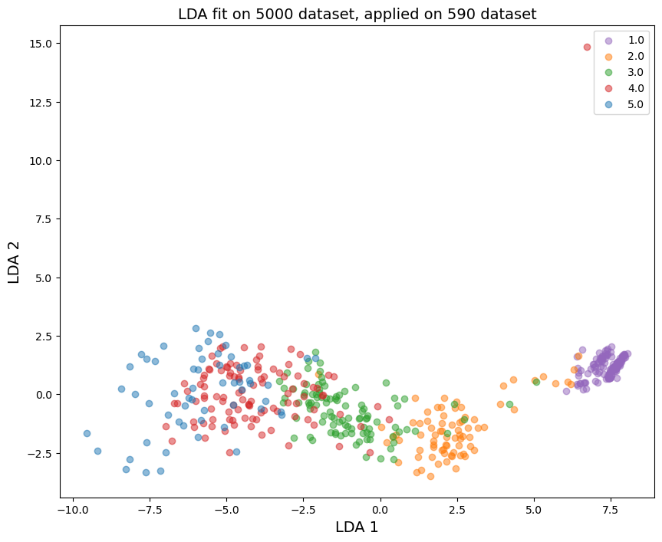
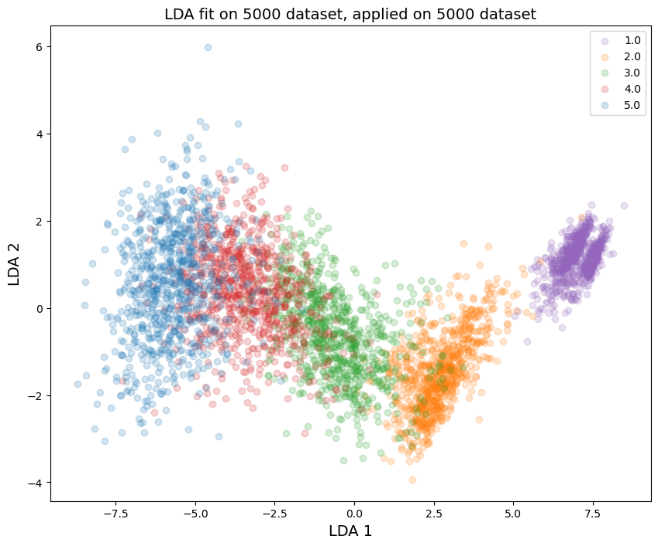
Supplementary Table 2: Sampling parameters as used in the development version of DNAStatistX for simulation of the 5000 mixture profiles.

The following section shows a comparative data analysis of the original dataset and the sampled dataset.

By fitting LDA on the original dataset of 590 samples (from here on referred to as “590-dataset”) and applying it to the dataset of the 5000 sampled instances (from here on referred to as “5000-dataset”) as shown in Supplementary Figure 1, it appears that the 590-dataset captures a lot of the variance that is also present in the 5000-dataset. The spread of the 5000 samples is broad over the two LDA dimensions. However, looking at Supplementary Figure 2, it seems that by fitting LDA on the 5000-dataset, only LDA 1 captures a good spread of the 590-dataset. LDA 2 does not contain much differential information for the 590-dataset. This is to be expected as the 5000-dataset is artificially created and therefore might contain less unexpected variation which is present in the 590-dataset. The one outlier is a 4-person mixture that has a high TAC of 138 which is the highest TAC value in the dataset. In the LDA for the 590-dataset it is in the middle of the red cluster.

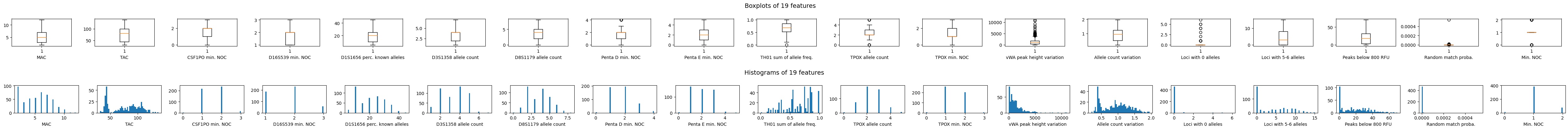


Supplementary Figure 1: LDA fit on the 590-dataset, applied to both that same dataset and to the 5000-dataset. DNA profiles consisting of one, two, three, four, or five donors are presented as purple, orange, green, red and blue circles, respectively.

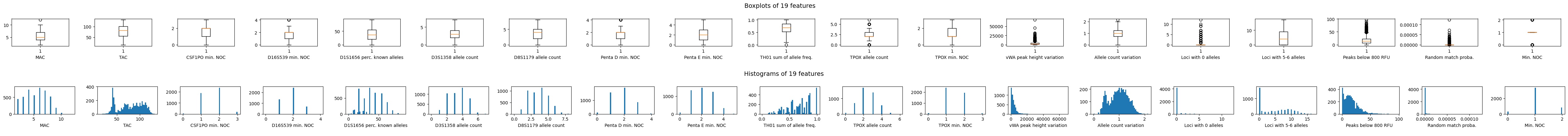


Supplementary Figure 2: LDA fit on the 5000-dataset, applied to both that same dataset and to the 590-dataset. DNA profiles consisting of one, two, three, four, or five donors are presented as purple, orange, green, red and blue circles, respectively.

Plotting the 19 features in boxplots and histograms show that they are mainly not normally distributed and contain many outliers. This is true for both datasets as can be seen in Supplementary Figure 3 and 4.



Supplementary Figure 3: Boxplots and histograms for each of the 19 features in the training set of the 590-dataset.



Supplementary Figure 4: Boxplots and histograms for each of the 19 features in the training set of the 5000-dataset.

Ideally all 19 features should follow comparable distributions for both datasets. From visual inspection, it is clear that this is not the case for all these features. By using the two-sided Kolomogorov\_Smirnov (KS) statistic[[1]](#footnote-1), it was determined that 8 features do not appear to be drawn from the same distribution. The results of this statistic are listed in Supplementary Table 3, where a large KS statistic or small p-value (less than 1.0e-2) corresponds to rejecting the null hypothesis assuming the samples were drawn from the same distribution. For discrete variables, a different KS statistic implementation was used[[2]](#footnote-2).

|  |  |  |
| --- | --- | --- |
|  | **KS statistic** | **p-value** |
| **MAC** | **0.12** | **3.0e-7** |
| **TAC** | **0.09** | **1.2e-4** |
| CSF1PO min. NOC | 0.04 | 3.3e-1 |
| **D16S539 min. NOC** | **0.09** | **3.3e-4** |
| **D1S1656 perc. known alleles** | **0.63** | **4.7e-200** |
| D3S1358 allele count | 0.06 | 4.8e-2 |
| D8S1179 allele count | 0.06 | 2.5e-2 |
| Penta D min. NOC | 0.04 | 3.4e-1 |
| Penta E min. NOC | 0.06 | 3.3e-2 |
| TH01 sum of allele freq. | 0.07 | 1.9e-2 |
| TPOX allele count | 0.05 | 9.6e-2 |
| TPOX min. NOC | 0.02 | 9.9e-1 |
| **vWA peak height variation** | **0.42** | **5.7e-83** |
| **Allele count variation** | **0.13** | **7.2e-8** |
| Loci with 0 alleles | 0.05 | 1.6e-1 |
| Loci with 5-6 alleles | 0.05 | 9.0e-2 |
| **Peaks below 800 RFU** | **0.19** | **5.7e-18** |
| **Random match proba.** | **0.10** | **9.4e-5** |
| Min. NOC | 0.05 | 1.6e-1 |

Supplementary Table 3: KS statistic results for the 19 features comparing the 590- and 5000-datasets. Bold features have a large statistic value or small p-value, and therefore the null hypothesis is rejected. This means that these features appear to be drawn from different distributions.

For analyzing feature correlations, we applied Kendall rank correlation coefficient. It is suitable for features that are not normally distributed (as is assumed for Pearson), and is more robust to outliers [63, 71]. Most features are highly correlated as can be seen in Supplementary Figure 5. A slight decrease in correlation can be observed between the 590- and 5000-dataset, though most values still lie above 0.4.



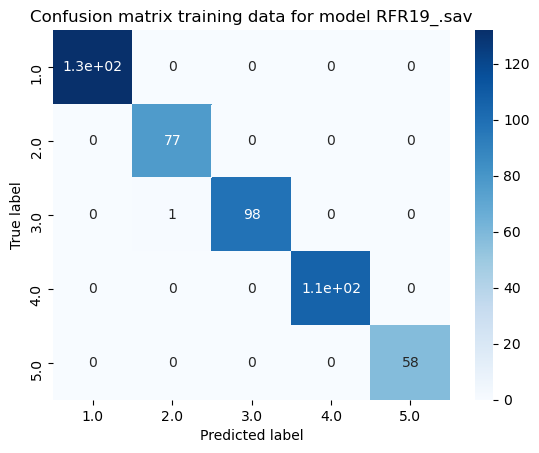
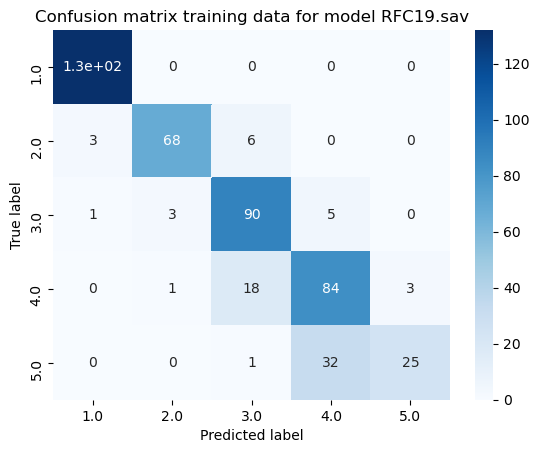
Supplementary Figure 5: Kendall feature correlations of the 19 features in the training set of the 590-dataset (left) and 5000-dataset (right).

In a short benchmarking study, we explored if A) regression can possibly outperform classification for NOC estimation and B) training on the new merged 5590-dataset has benefits for performance for NOC estimation.

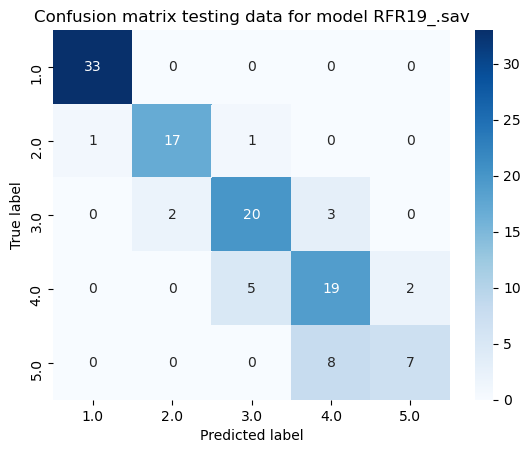
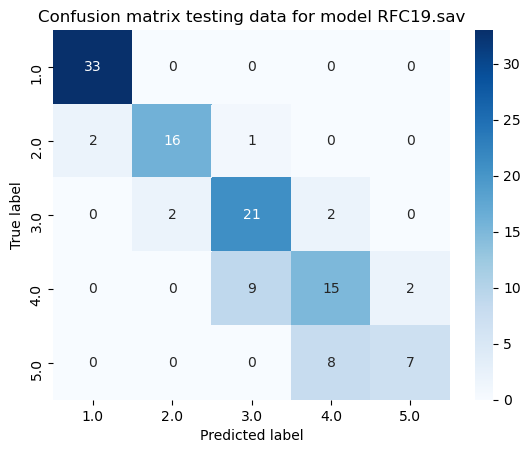
Supplementary Table 4 shows which models were used to compare regression and classification, while Supplementary Figure 6 and 7 list the results on the training- and test data respectively.

|  |  |  |
| --- | --- | --- |
|  | **RFC19 model** | **RFR19 model** |
| Model type | Random forest classifier[[3]](#footnote-3) | Random forest regressor[[4]](#footnote-4) |
| Model parameters | As described in [18] | Defaults |
| Dataset used | 590-dataset | 590-dataset |

Supplementary Table 4: Models and model parameters used for a short benchmarking to comparing the original model with a default regressor, on the original 590-dataset.



Supplementary Figure 6: Confusion matrices of the RFC19 (left) and RFR19 (right) models applied on the training data from the 590-dataset. The regression model performs almost perfectly (99% accuracy), where the classifier performs a bit worse (85%), and predicts some instances more than one class off.



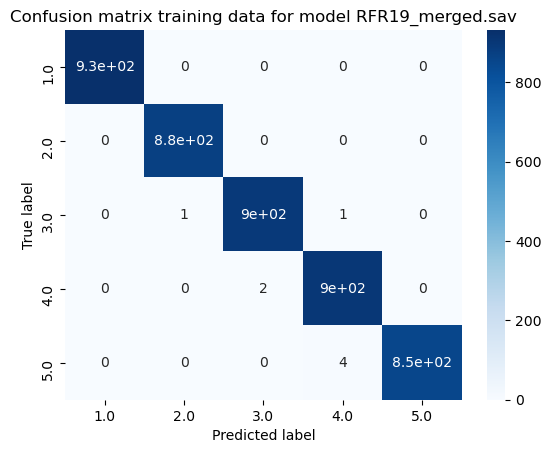
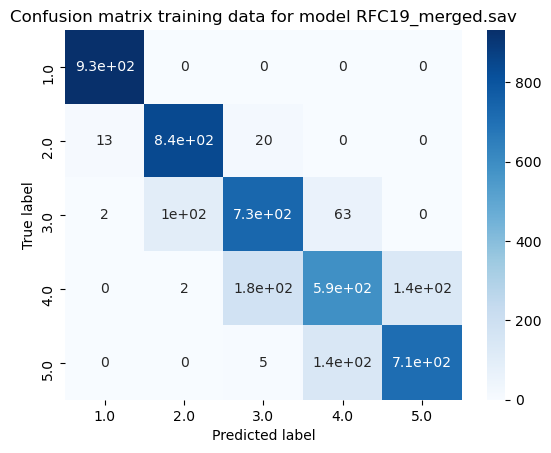
Supplementary Figure 7: Confusion matrices of the RFC19 (left) and RFR19 (right) models applied on the test data from the 590-dataset. The regressor obtains similar or better results than the classifier (81% accuracy versus 78%). Especially for profiles of 4 donors, there is consistent better performance from the regression model. For profiles of 2, 3, and 5 donors, the number of correct predictions is on average the same for the classifier and the regressor.

The larger discrepancy between the train and test performance for the regressor is due to overfitting. The default parameters of the regressor in comparison to the optimized classification parameters are more tuned towards larger datasets. These results show promise that a regression model could outperform a classification model once put through more rigorous training.

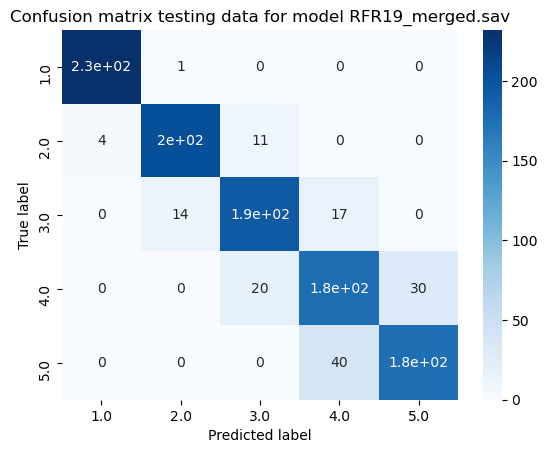
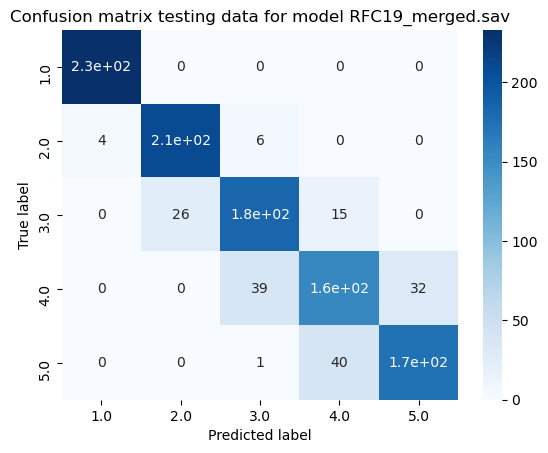
Supplementary Table 5 shows which models and datasets were used to test the merged 5590-dataset, while Supplementary Figure 8 and Supplementary Figure 9 list the results on the training- and test data respectively.

|  |  |  |
| --- | --- | --- |
|  | RFC19\_merged model | RFR19\_merged model |
| Model type | Random forest classifier | Random forest regressor |
| Model parameters | As described in [18] | Defaults |
| Dataset used | 5590-dataset (merged 590- and 5000-datasets) | 5590-dataset (merged 590- and 5000-datasets) |

Supplementary Table 5: Models and model parameters used for a short benchmarking study to explore if the new merged 5590-dataset has benefits for performance on the original 590-dataset.



Supplementary Figure 8: Confusion matrices of the RFC19\_merged (left) and RFR19\_merged (right) models on the training data from the 5590-dataset.



Supplementary Figure 9: Confusion matrices of the RFC19\_merged (left) and RFR19\_merged (right) models on the test data from the 5590-dataset. Accuracy now lies at about 85% for classification, and 88% for regression. Improvement lies mainly for predicting 3 and 4 contributors.

Additional analysis summarized in Supplementary Table 6 shows that the regression model trained on the 5590-dataset performs about equally well on samples originating from the 590- and 5000-dataset, while the classification model performs slightly worse on samples from the 590-dataset. The overall performance of the regression model is also slightly better, showing its potential for future applications.

|  |  |  |
| --- | --- | --- |
|  | RFC19\_merged | RFR19\_merged |
| Total test accuracy | 85% | 88% |
| Test accuracy on samples from 590-dataset | 82% | 86% |
| Test accuracy on samples from 5000-dataset | 86% | 88% |

Supplementary Table 6: Performance of the RFC19\_merged and RFR19\_merged models on the test data from the 5590-dataset. We compare the accuracy on samples that originate from the original 590-dataset and the 5000-dataset separately.

1. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ks\_2samp.html [↑](#footnote-ref-1)
2. https://rdrr.io/cran/dgof/man/ks.test.html [↑](#footnote-ref-2)
3. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html [↑](#footnote-ref-3)
4. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.htmls [↑](#footnote-ref-4)