**Introducing XAI to forensics: explanations for number of contributor predictions**

**Abstract**

Using machine learning to determine the number of contributors (NOC) in short tandem repeat (STR) profiles has been shown to obtain good accuracy. However, these predictions may not be fully understandable to the biologist user who is presented with these as a tool to help determine the number of donors. Therefore, we introduce the field of eXplainable artificial intelligence (XAI) to this problem, and to the audience in general. We apply XAI to NOC estimation by using a visual aid that incorporates explanations from SHAP values and counterfactual examples for each prediction. Existing methods for generating counterfactuals have not attempted to handle correlated features, causing those methods to find examples that are impossible given their feature combinations. Since the features from STR data are highly correlated, we have implemented a new method, ReCo, that generates realistic counterfactuals on highly correlated data. We show that ReCo outperforms state-of-the-art methods, with traditional metrics, as well as a novel realism metric. A final user evaluation demonstrates the opinions of end-users, a metric that we regard important in XAI studies.

1. **Introduction**
   1. *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 software that is used for DNA interpretation does require the NOC to be entered by the user [10, 11], which influences the calculation of the evidence [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 [18, 19]. However, machine learning algorithms are often considered to be “black-boxes” [20-27], as the predictions they output 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 [28]. However, using a simple model such as a decision tree leads to less accurate predictions; they reported a difference of over 10% as compared to a random forest. This method also relies heavily on filtering of artefacts, for which another decision tree is used. 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 models, since users want to know *why* a certain prediction is made [20-27]. The European Commission recently underlined the importance of explainability in a proposal for rules on AI systems in higher-risk settings such as law [29]. 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.

* 1. *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 [20, 22-27, 30]. With transparent models, one can derive the exact steps taken to arrive from input features to an output within reasonable time [23, 24]. 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 [23, 24, 31]. If all of these conditions are violated, a 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 [22, 23, 26, 27]. 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) [20, 22-27, 30]. 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 discovered, 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 each DNA-profile was predicted seems more fit. Since only a few studies have applied machine learning to this problem, there is no best suited model, so a model-agnostic approach is preferable. For generating local, model-agnostic explanations, there exist two generally accepted approaches; feature importance and counterfactuals.

Feature importance methods highlight the values of the input that have driven the model to make a certain prediction [20, 22-24, 26, 30]. 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 [32]. 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 the SHAP paper [32]. SHAP has been implemented in real-life cases such as predicting hypoxia based on clinical data [33], and predicting the most fitting eye-surgery type [34]. They seem to have obtained valuable information for what are important factors to ML models.

Counterfactual explanations are example data points which have a different prediction from the input data point [20, 22, 23, 26]. 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 [20, 21]. 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.

* 1. *Counterfactuals*

A counterfactual is an example instance that is similar to the instance we want to explain, but has a different prediction [35-50]. The differences in feature values between the input- and counterfactual instances can give the user an impression about the local decision space of the model. More formally, a counterfactual can be described as follows [37]:

“*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 [35-50]. To help the user relate this new prediction as a possibility for the original input, the counterfactual- and input instances must be similar. 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 [36-42, 44-47, 51]. Though some methods use or Euclidean distance [42, 46], 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 [36, 37, 39-41, 45, 47]. 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 is measured by the number of differences in feature values in comparison to the input [35, 36, 39, 41, 43-47, 49, 50], 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 [36, 38-41, 47], or providing counterfactuals that are actionable; meaning that the changes can be acted upon to reach that alternative outcome [36, 38, 39, 42, 43, 50]. 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 [44, 52], or can be artificially sampled [36-41, 45, 46, 48-50]. 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 quite 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 [37, 38, 40, 45], or take the input and perturb its feature values until a different outcome is reached [36, 39, 41, 46, 48-50]. While most tackle sampling by randomly changing feature values from the input instance or the training data [37-39, 45, 46, 48, 50], some take a more sophisticated route by using a genetic algorithm [36, 40, 41, 49]. 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 [36, 39, 45]. 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 [43-45]. 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 [39]. 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 [36]. 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 their counterfactuals [35, 48]. By only changing the features from the input instance that have negative SHAP values for target class B, a counterfactual could be found [48]. 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 [35].

An aspect of generating counterfactuals with sampling-based methods that is largely overlooked or handled quite poorly is realism. As these samples are often generated by randomly changing feature values, or by combining instances, they might be quite 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. That means they would have started their career at age 5. 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 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 [36]. 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 [42]. This ensures that the query instance can be transformed to take on another the target output, which is relevant for actionable settings. Similar to these approaches, the range of feature values can be limited [38, 40]. Either based on the training data, or inputted by the user. When considering our 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. Neither 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 [39]. 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. A Julia implementation 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 [41].

One method derives counterfactuals from training instances, which relies on the assumption that there are inherently sparse counterfactuals in the training set [44]. 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 [38, 39, 53, 54], no method has 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, the presentation of counterfactuals presented is often not user-friendly. Most counterfactual methods for tabular data present the comparison of the input and counterfactual in a table [36, 39-41, 43, 44, 46, 52, 55]. 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 [47, 56, 57]. With a visual approach, these magnitudes can be communicated better. [58]. Though some previous visualizations were developed for counterfactuals [45, 50], 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 [50, 58].

* 1. *Contribution*

With this paper, we introduce the concept of eXplainable Artificial Intelligence (XAI) to

the field of forensic science by demonstrating its value on a practical issue. We generate explanations for individual predictions of the Number of Contributors (NOC) to a DNA profile which can be applied for any machine learning model. These explanations consist of SHAP values and counterfactual examples in a compound visualization which is the first to combine these techniques into one explanation. We implement a new method for finding realistic counterfactuals (ReCo) by deriving them from the training data. This produces examples that have fewer feature differences than using training examples, but are still plausible data points. To the best of our knowledge, this is the first method that handles correlated features automatically. Lastly, we have created a new realism metric that scores how plausible counterfactuals are in terms of their feature combinations, which is important with highly correlated features. The counterfactuals are assessed with objective metrics, while the full visual explanation is assessed as by a small group of users of the application.

1. **Materials and methods**
   1. *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 a mixture up to 5 donors. The mixtures were formed from 1174 different single donors and degraded 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 more samples to create a less sparse feature space. In a development version of the statistical library DNAStatistX [13], realistic raw profiles can be generated by using the same model that is used for calculating evidence (EuroForMix model [12]). This program was used to generate factors such as peak height, degradation, and mixture proportions within ranges derived from the original dataset. In Supplementary Table 2, the exact parameters can be found. Then, the genotypes are generated randomly based on population frequency using the EuroFixMix model. To ensure that all donors are visible in the generated profile, each donor must have an LR of at least 1000.

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

Once the features were derived from the sampled data, it appeared that about half of them seemed to have been drawn from a different distribution as compared to the original dataset of 590 instances (see 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 new data in a benchmarking study, which demonstrated that the model actually performs better once trained on the combined dataset of the original 590-, and sampled 5000 instances together (see Supplementary Table 5).

* 1. *Machine learning model*

Originally, the estimation of the NOC was treated as a classification problem by the NFI with their 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 and 8), we concluded that a regression model has the potential to achieve more accurate predictions. The explanations also benefit from using regression, as the outputs are not independent boxes, but they lie on a relational scale. This scale is apparent to the user. The model can then be defined to map the input profile to an output where .

Because determining the NOC with machine learning is still a novel approach, there is no consensus about which type of model is most fit. The Netherlands Forensics Institute (NFI) is looking to improve the model and used features in the future.

* 1. *Explanation goals*

The DNA-experts at the NFI require explanations for being able to use the machine

learning model in their decision-making process. We had to clearly identify the users’ needs by engaging with them, so that the best product could be made [59]. A complex machine learning model such as a random forest is considered a black-box which means that it is not clear to the user how the input leads to a certain prediction. However, the DNA-experts at the NFI consult the output of this model as a support system in their analysis of the NOC. In cases where the expert comes to a different conclusion than the model, the reasons why this discrepancy exists, are unclear. This means that there is no way to tell if the model is correct, or if the expert is. The users could have missed some information that the model has based its decision on, or the model could have made a decision based on the wrong factors. For such use cases, explanations are most valuable. Explanations can also be informative in general, as a confirmation to the user’s own estimation of the NOC. In short, the explanations should communicate two pieces of information; a general sense of why the model predicted the current NOC, and with which specific changes a different NOC could have been predicted. The general information gives an impression of the model’s focus, while the more detailed counterfactual gives a sense of thresholding values and how close the decision is.

It was interesting to note that this lines up nicely with the two questions that a good explanation of a single prediction should answer [20, 37, 38]:

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 [60]. From their definitions, our 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 [61]. 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 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.

* 1. *Desiderata counterfactual explanations*

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

we must accommodate.

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

As the NFI is looking to continue development on their machine learning model, a model-agnostic explanation method is preferable. In this way, the same explanations can be generated regardless of the underlying algorithm. We do 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, 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.

It should be possible to generate a counterfactual for any input. If the closest counterfactual example is still quite different 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.

Sparsity is encouraged to prevent users from experiencing cognitive overload. We know that humans pick explanations in a biased way [21], meaning that if many options are available, only a few will be selected. The number of difference feature values between the input- and counterfactual instances 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 instances, we first had to analyse the underlying data. The choice of distance should be catered towards the problem [37]. As our dataset has outliers, and most features are not normally distributed (see Supplementary Figures 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) [37, 39, 44, 45, 47], 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 [36, 38]. This is quite robust even for unscaled and unnormalized features with lots of outliers, which is the case in this dataset. The equation for distance 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; to provide a user with multiple routes to reach the different outcome [37, 41, 47]. Moreover, presenting multiple, possible contradicting examples does not seem like user friendly introduction to counterfactual explanations.

The desiderata discussed so far have mostly been covered quite well in the literature. For realism, there is not such a nice definition. For the problem of NOC estimation, it is essential to present the user with data that is somewhat plausible. None of the methods discussed in section 1.x, 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, even though these are both normal feature values when looking at the feature distributions. Other approaches place constraints on the sampled data. Though this is certainly interesting and helpful, it might not be the best solution currently. Since the features used will change in the near future and they are highly correlated with multiple other features, and how these relationships can be modelled is not trivial and therefore becomes time-consuming. The training data consists of the most realistic instances that we could use. It is therefore a good starting point for our explanations.

* 1. *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 trainings 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 [36]. 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 [62, 63].

The counterfactual instance is part of 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 .
2. Compute the SHAP values for both the input- and the counterfactual instance, per feature in . Then subtract these SHAP values of the input instance from the SHAP values of the counterfactual instance. This set is sorted by the magnitude of each value. This gives us an impression of which changes in feature values from the input- 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.
3. 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 instance, 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. These feature differences are most likely not relevant to help reach the counterfactual prediction, and could therefore possibly be filtered from the counterfactual instance.
4. 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 .
5. 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:

Table 1: Example of how a counterfactual instance is filtered. The input instance has a prediction of 4, and the counterfactual instance has a prediction of 3. Therefore, the direction of the change in prediction is negative. The SHAP values of all three features are calculated for both the input- and counterfactual instances. The change in SHAP value from the input- to the counterfactual instance gives an impression whether or not the feature difference is relevant. For Feature 1, the SHAP change is negative which matches the direction of the change in prediction. On the other hand, the SHAP change in Feature 2 is positive, and the SHAP change in Feature 3 is small. These features differences are therefore likely not relevant to the counterfactual, and thus can be filtered. The counterfactual will take the values of the input instance for Feature 2 and 3 as long as the target prediction is kept.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **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 |

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.

* 1. *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. 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 next most correlated feature with to be . In this way, the score is grounded in the values of a real instance.

Please refer to Figure x 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 TAC = 30 (from the counterfactual) in combination 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.

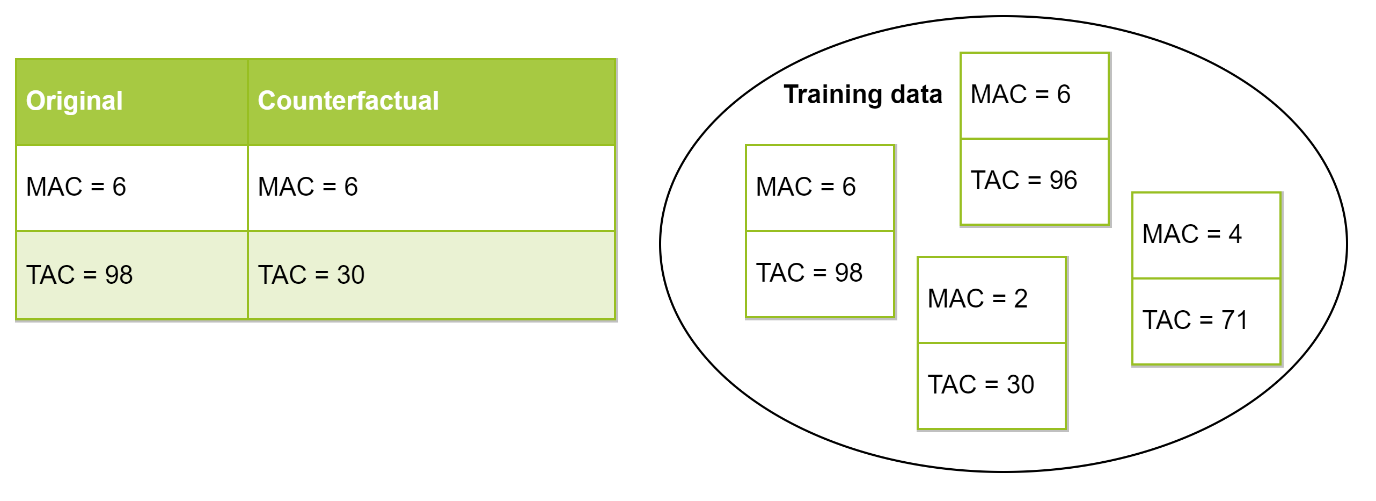


Figure 1: Toy example of a counterfactual that gets a realism score of 0; the proposed counterfactual contains a feature combination that does not occur in the training data.

* 1. *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 decision, along with how a different prediction could have been achieved. We believe this is the first visualization to unify counterfactuals with SHAP values. The following requirements were derived from conferring with the DNA-experts from the NFI in addition to some desiderata already expressed in the literature [64, 65].

First of all, the visualization is *consistent*. Each profile is presented in the same format; all features will be presented on the same location, and feature values will be plotted on a normalized scale. For normalization, we used a quantile transformer as this maps all feature values between 0-1 while spreading out the most frequent values [66]. In this way, the size of differences between feature values can be perceived more clearly. This consistency helps users reach some level of familiarity with the visualization over time as it allows for comparison between profiles.

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 [67].

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

The visualization was developed using matplotlib.

1. **Results and discussion**

This work presents two distinct products; a counterfactual method, and a compound visualization which lend itself to objective and subjective evaluation respectively. Both of these aspects were developed specifically to the domain of NOC estimation and should therefore be evaluated within the context of this specific problem [25, 30]. That is why only the dataset described in section 2.1 was used for the evaluation. It is important to compare against past approaches to understand how our results relate to them [39].

* 1. *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 [35, 36, 38, 39, 41, 43-45], 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 metric proposed in section 2.6.

These metrics are used in comparison with other counterfactual methods. 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 [52]. DiCE is a sampling approach that generates counterfactuals from the input by randomly sampling its feature values [39]. It starts from the input instance, and randomly picks a feature to be given a random value until the target prediction is reached. The implementation of DiCE genetic similar GeCo [41], 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.

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

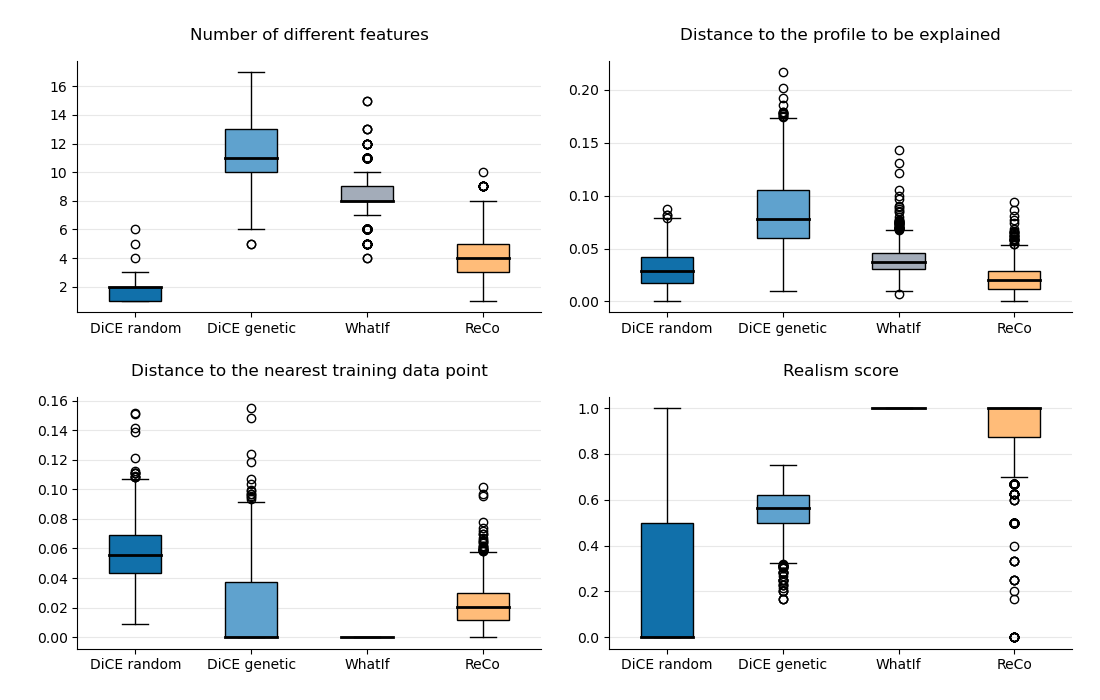


Figure : Quantitative evaluation of ReCo in comparison to WhatIf, DiCE random and DiCE genetic on four different metrics; the number of feature differences, the distance to the input and closest training data, and realism.

While DiCE random performs best in terms of the number of 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 infeasible for our dataset.

An improvement can be seen when the genetic version is used; the median realism score almost hits a sufficient 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.

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.

ReCo seems to score quite well on all four scores. 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 with WhatIf, which only minimizes the distance, the obtained data points might already be sparser. Then by filtering, these two scores go down even 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 tiny or counterintuitive differences that are likely insignificant to the model. These small differences will not cause the counterfactual to move too far away from the training data. Moreover, the features that are filtered must have little discriminatory power between the original- and target output. This is likely because its values are similar for instances of the original- and target prediction in the training data. For example, a certain test profile with an allele count of 3 at locus D8S1179 was predicted to have four contributors, while the counterfactual with an allele count of 4 at this locus had a prediction of three contributors. For both the original- and the target NOC, the median of this feature in the training data is at 4. As such, it has little discriminatory power between the two outcomes. As a final remark, we note that there are expectedly quite some outliers that score low on realism. When more feature differences are filtered away, the likelihood increases that the value 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.

* 1. *Qualitative evaluation ReCo*

Some approaches pick one profile to generate an explanation for, and analyze if the explanation is logical based on domain knowledge [31, 36, 38-40, 43, 47].

* 1. *Subjective evaluation compound visualization*

The XAI domain, especially within counterfactuals, is still very focused on objective evaluation. For any application of XAI however, a user evaluation is most important as explanations are designed to help people [39, 47, 55, 57, 58]. The visualization was specifically designed for the DNA-experts of the NFI, within the context of NOC estimation. It is therefore important to assess it in this real-life environment [68]. The explanation consists of two parts, SHAP values and a counterfactual example. To the user these are presented in a single visualization of a prediction as shown in Figure x. The explanation should therefore be evaluated as one complete unit, not by its separate components. It is not relevant to see how well the individual pieces explain the prediction as they will not be used as such.

There were two main aspects that we wanted to evaluate. The first

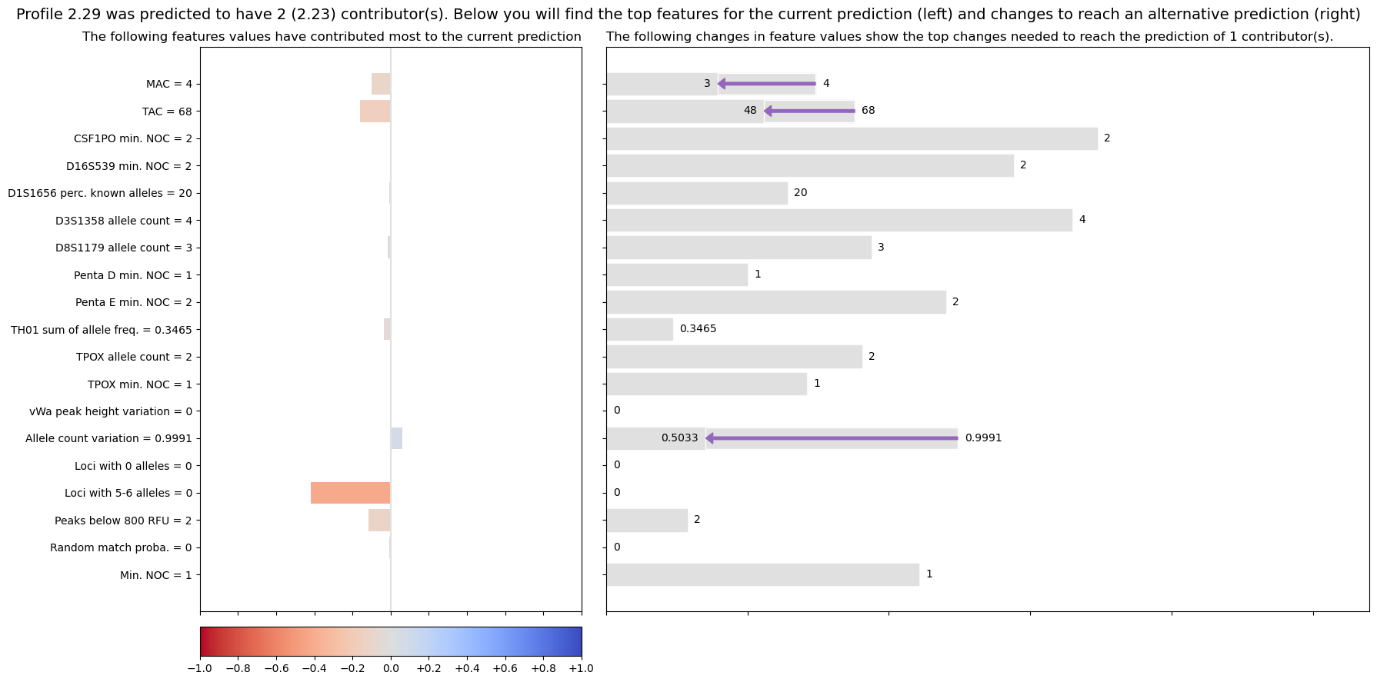
The visualization was compared against two frequently-used presentations of explanations. The first is a SHAP force plot [69], which seems to be the simplest visualization for single predictions. The second comparison will be made against the representation that is most common for counterfactuals, a table [36, 39-41, 43, 44, 46, 52, 55].

.

The goal of the user study is to find out if the users can gain some information from the explanation about certain decisions by the model, and if they find it nice to use. Opinions of the users are the most important aspect to evaluate, they should be comfortable using the explanation.

Few have actually performed a user study [55]. Some aspects that they consider such as information sufficiency, competence (explanation corresponds to how I make decisions), confidence (explanation made me more confident about my decision), can also be applied to the NOC domain.

Figure 3: Visualization for a single NOC prediction, including SHAP values on the left and a counterfactual example generated by ReCo on the right.



* 1. *Future work*

Although the current implementation of ReCo is used for regression, it only requires minor adaptations to fit classification as well.

One of the most exciting next steps would be to incorporate our realism score into a sampling algorithm. In this way, the advantages of speed and closer distance to the input might be obtained, while still generating instances with plausible feature combinations.

1. **Conclusion**

This study describes

**Acknowledgements**

We are thankful to Corina Benschop for insightful discussions, Jerry Hoogenboom for sampling 5000 DNA mixtures, and the NFI group BiS for participating in the user studies.

**References**

1. Coble, M.D., et al., *Uncertainty in the number of contributors in the proposed new CODIS set.* Forensic Science International: Genetics, 2015. **19**: p. 207-211.

2. Benschop, C.C.G., et al., *The effect of varying the number of contributors on likelihood ratios for complex DNA mixtures.* Forensic Science International: Genetics, 2015. **19**: p. 92-99.

3. Haned, H., et al., *Estimating the Number of Contributors to Forensic DNA Mixtures: Does Maximum Likelihood Perform Better Than Maximum Allele Count?* Journal of Forensic Sciences, 2011. **56**(1): p. 23-28.

4. Biedermann, A., et al., *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, 2012. **6**(6): p. 689-696.

5. Paoletti, D.R., et al., *Inferring the Number of Contributors to Mixed DNA Profiles.* IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2012. **9**(1): p. 113-122.

6. Young, B.A., et al., *Estimating number of contributors in massively parallel sequencing data of STR loci.* Forensic Science International: Genetics, 2019. **38**: p. 15-22.

7. Grgicak, C.M., et al., *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, 2020. **47**.

8. Swaminathan, H., et al., *NOCIt: A computational method to infer the number of contributors to DNA samples analyzed by STR genotyping.* Forensic Science International: Genetics, 2015. **16**: p. 172-180.

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

10. Coble, M.D. and J.-A. Bright, *Probabilistic genotyping software: An overview.* Forensic Science International: Genetics, 2019. **38**: p. 219-224.

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

12. Bleka, Ø., G. Storvik, and 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, 2016. **21**: p. 35-44.

13. Benschop, C.C.G., et al., *Multi-laboratory validation of DNAxs including the statistical library DNAStatistX.* Forensic Science International: Genetics, 2020. **49**: p. 102390.

14. Benschop, C.C.G., et al., *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, 2019. **42**: p. 31-38.

15. Bille, T., et al., *Interpreting a major component from a mixed DNA profile with an unknown number of minor contributors.* Forensic Science International: Genetics, 2019. **40**: p. 150-159.

16. Buckleton, J.S., et al., *The effect of varying the number of contributors in the prosecution and alternate propositions.* Forensic Science International: Genetics, 2019. **38**: p. 225-231.

17. Clayton, T.M., et al., *Analysis and interpretation of mixed forensic stains using DNA STR profiling.* Forensic Science International, 1998. **91**(1): p. 55-70.

18. Benschop, C.C.G., et al., *Automated estimation of the number of contributors in autosomal short tandem repeat profiles using a machine learning approach.* Forensic Science International: Genetics, 2019. **43**: p. 102150.

19. Marciano, M.A. and 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, 2019. **43**.

20. Mittelstadt, B., C. Russell, and S. Wachter, *Explaining Explanations in AI*. 2018.

21. Miller, T., *Explanation in artificial intelligence: Insights from the social sciences.* Artificial Intelligence, 2019. **267**: p. 1-38.

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

23. Barredo Arrieta, A., et al., *Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI.* Information Fusion, 2020. **58**: p. 82-115.

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

25. Gilpin, L.H., et al. *Explaining Explanations: An Overview of Interpretability of Machine Learning*. in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. 2018.

26. Adadi, A. and M. Berrada, *Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI).* IEEE Access, 2018. **6**: p. 52138-52160.

27. Du, M., N. Liu, and X. Hu, *Techniques for interpretable machine learning.* Communications of the ACM, 2020. **63**(1): p. 68-77.

28. Kruijver, M., et al., *Estimating the number of contributors to a DNA profile using decision trees.* Forensic Science International: Genetics.

29. Commision, E., *Fostering a European approach to Artificial Intelligence*. 2021.

30. Murdoch, W.J., et al., *Definitions, methods, and applications in interpretable machine learning.* Proceedings of the National Academy of Sciences of the United States of America, 2019. **116**(44): p. 22071-22080.

31. Fernández, R.R., et al., *Random forest explainability using counterfactual sets.* Information Fusion, 2020. **63**: p. 196-207.

32. Lundberg, S. and S.-I. Lee, *A Unified Approach to Interpreting Model Predictions*. 2017.

33. Lundberg, S.M., et al., *Explainable machine-learning predictions for the prevention of hypoxaemia during surgery.* Nat Biomed Eng, 2018. **2**(10): p. 749-760.

34. Yoo, T.K., et al., *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, 2020. **9**(2): p. 8.

35. Ramon, Y., et al., *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, 2020. **14**(4): p. 801-819.

36. Dandl, S., et al. *Multi-Objective Counterfactual Explanations*. in *Parallel Problem Solving from Nature – PPSN XVI*. 2020. Cham: Springer International Publishing.

37. Wachter, S., B. Mittelstadt, and C. Russell, *Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR.* Harvard journal of law & technology, 2018. **31**: p. 841-887.

38. Karimi, A.-H., et al. *Model-agnostic counterfactual explanations for consequential decisions*. in *International Conference on Artificial Intelligence and Statistics*. 2020. PMLR.

39. Mothilal, R.K., A. Sharma, and C. Tan. *Explaining machine learning classifiers through diverse counterfactual explanations*. 2020.

40. Sharma, S., J. Henderson, and J. Ghosh. *CERTIFAI: A common framework to provide explanations and analyse the fairness and robustness of black-box models*. 2020.

41. Schleich, M., et al., *GeCo: Quality Counterfactual Explanations in Real Time*. 2021.

42. Poyiadzi, R., et al., *FACE: Feasible and Actionable Counterfactual Explanations*. 2020. 344-350.

43. Moore, J., N. Hammerla, and C. Watkins, *Explaining deep learning models with constrained adversarial examples*. 2019. p. 43-56.

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

45. Grath, R.M., et al., *Interpretable Credit Application Predictions With Counterfactual Explanations.* ArXiv, 2018. **abs/1811.05245**.

46. White, A. and A. Garcez. *Measurable Counterfactual Local Explanations for Any Classifier*. in *ECAI*. 2020.

47. Russell, C. *Efficient search for diverse coherent explanations*. 2019.

48. Rathi, S., *Generating Counterfactual and Contrastive Explanations using SHAP*. 2019.

49. Guidotti, R., et al., *Factual and Counterfactual Explanations for Black Box Decision Making.* IEEE Intelligent Systems, 2019. **34**(6): p. 14-23.

50. Gomez, O., et al. *ViCE*. 2020.

51. Sokol, K. and P. Flach. *Desiderata for interpretability: Explaining decision tree predictions with counterfactuals*. 2019.

52. Wexler, J., et al., *The What-If Tool: Interactive Probing of Machine Learning Models.* IEEE Transactions on Visualization and Computer Graphics, 2020. **26**(1): p. 56-65.

53. Barocas, S., A.D. Selbst, and M. Raghavan. *The hidden assumptions behind counterfactual explanations and principal reasons*. 2020.

54. Bertossi, L., *Score-Based Explanations in Data Management and Machine Learning*. 2020. p. 17-31.

55. Adhikari, A., et al. *LEAFAGE: Example-based and Feature importance-based Explanations for Black-box ML models*. in *IEEE International Conference on Fuzzy Systems*. 2019.

56. Sokol, K. and P. Flach. *Conversational Explanations of Machine Learning Predictions Through Class-contrastive Counterfactual Statements*. 2018.

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

58. Verma, S., J.P. Dickerson, and K. Hines, *Counterfactual Explanations for Machine Learning: A Review.* ArXiv, 2020. **abs/2010.10596**.

59. Bhatt, U., et al. *Explainable machine learning in deployment*. 2020.

60. Akula, A.R., et al. *Natural Language Interaction with Explainable AI Models*. in *CVPR Workshops*. 2019.

61. Molnar, C., et al., *Pitfalls to Avoid when Interpreting Machine Learning Models.* ArXiv, 2020. **abs/2007.04131**.

62. Chiandussi, G., et al., *Comparison of multi-objective optimization methodologies for engineering applications.* Computers & Mathematics with Applications, 2012. **63**(5): p. 912-942.

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

64. Sokol, K. and P. Flach. *Explainability fact sheets: A framework for systematic assessment of explainable approaches*. in *FAT\* 2020 - Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 2020.

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

66. developers, s.-l. *sklearn.preprocessing.QuantileTransformer*. 2020 25-05-2021]; 0.24.2:[Available from: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html>.

67. Kovesi, P., *Good Colour Maps: How to Design Them.* ArXiv, 2015. **abs/1509.03700**.

68. Doshi-Velez, F. and B. Kim, *Towards A Rigorous Science of Interpretable Machine Learning.* arXiv: Machine Learning, 2017.

69. Lundberg, S.M., et al., *Explainable machine-learning predictions for the prevention of hypoxaemia during surgery.* Nature Biomedical Engineering, 2018. **2**(10): p. 749-760.

**Supplementary Material**