NICE: AN ALGORITHM FOR NEAREST INSTANCE COUNTERFACTUAL EXPLANATIONS

A PREPRINT

Dieter Brughmans and David Martens

April 16, 2021

ABSTRACT

In this paper we suggest NICE: a new algorithm to generate counterfactual explanations for heterogeneous tabular data. The design of our algorithm specifically takes into account algorithmic requirements that often emerge in real-life deployments: the ability to provide an explanation for *all* predictions, being efficient in run-time, and being able to handle any classification model (also non-differentiable ones). More specifically, our approach exploits information from a nearest instance to speed up the search process. We propose four versions of NICE, where three of them optimize the explanations for one of the following properties: sparsity, proximity or plausibility. An extensive empirical comparison on 10 datasets shows that our algorithm performs better on all properties than the current state-of-the-art. These analyses show a trade-off between on the one hand plausiblity and on the other hand proximity or sparsity, with our different optimization methods offering the choice to select the preferred trade-off. An open-source implementation of NICE can be found at https://github.com/ADMAntwerp/NICE.

1 Introduction

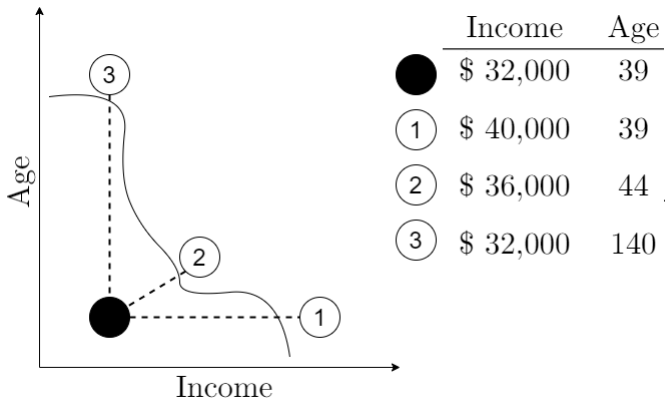
In the past decade, machine learning models have been succesfully deployed in many high-stakes decision making such as credit scoring [20], fraud detection [29, 2], and clinical healthcare [2]. However, due to the non-linearity of the models or high-dimensionality of the underlying data, for many models high performance has come at a cost of explainability [24, 41, 33]. The inability to explain automated decisions that impact individuals undermines the trust between data subject and data controllers [41]. Post-hoc explanation methods such as counterfactual explanations aim to reinstall this trust while keeping the performance of the decision mechanism [41]. Several laws have also pushed on providing explanations for algorithmic decision-making. One example is the Fair Credit Reporting Act in the United States [37]. It requires data controllers to provide specific reasons that negatively influence a data subject's credit score. Counterfactual explanations are a great fit here as they provide a set of minimum features required to change the predicted outcome. Another example comes from the General Data Protection Regulation (GDPR) in the European Union. Article 14 states that data subjects have the right to obtain meaningful information about the logic involved in automated decision making [9]. Current classification models have a high complexity and many parameters. Explaining the inner working of such a model will not be meaningful to a data subject. Counterfactual explanations on the other hand, highlight a set of input features that, when changed, alter the predicted decision [24]. These input features are much more understandable to humans as the form of counterfactual explanations has deep foundations in philosophy [18, 21, 34] and social sciences [26]; for it is similar to how a person thinks about a decision by asking the question: what could I have changed to achieve a different outcome? Additionally, counterfactual explanations allow data controllers to explain instances without disclosing any trade secrets or private data [1].

It is clear that in theory counterfactual explanations fit the legislative requirements and have the ability to make black-box machine learning models transparent and accountable. Spurred by these benefits, in recent years many counterfactual algorithms have been developed for tabular data (see e.g. the overviews by [39, 16]). However, most of them focus on generating counterfactuals without taking into account the algorithmic requirements in deployment. Consider for example custom fraud detection. In a country as Belgium, custom administration processes around 9.5 declarations every second [38]. Each of these cases have the potential for different forms of fraud such as illegal drug

traffic, importation of counterfeit goods, valuation fraud, smuggling, product misclassification and the manipulation of the origin of goods. Predictive algorithms are used in this context to identify high-risk targets which are further investigated by custom officers [38, 6]. Counterfactual explanations can be of great value here to improve collaboration. The set of features in this explanation can clarify which form of fraud might be committed, or what the main evidences for predicted fraud are (e.g. country risk, article code, weight and so on), thereby speeding up further investigations. Custom officers check many high-risk cases each minute, so explanation algorithms will have to match this speed to make them useful. This computational efficiency requirement also guarantees that these algorithms can be easily scaled without the need of excessive infrastructure. An additional requirement is that *all* observations can be explained in this domain. The absence of an explanation might give the (potentially wrong) impression that the predictive model is not certain about its prediction, undermining the trust in the predictive model and making it difficult for custom officers to act upon the output.

If we return to the example of credit scoring in the US, we notice the same requirements. In this case explanations for negative decisions are required by law, rendering counterfactual algorithms that cannot explain all of these credit rejection predictions, useless. Also computational inefficiency is costly here. In credit scoring, the data subjects are potential customers. Imagine a consumer applying for a loan at a bank. A number of variables are asked, such as income, profession, etc. to assess the credit risk. The classification model, which is efficient by design as well, as not to have the consumer wait for minutes or hours to get a decision, will provide a decision swiftly. In case the application is rejected, it is just as important to have an efficient explanation algorithm to come with a reasoning for the rejection. Having the consumer just sit there and wait is arguably unacceptable or at least bad business practise. A recent example in this domain is the Apple Card. Applicants, who are denied for this credit card, were enrolled in a "Path to Apple Card"[31]. This program suggests personalized actions that positively influence someones credit score such as decreasing debt or making payments on time. If the algorithmic requirements are met, counterfactual explanations could be a perfect fit for this application. One can imagine other domains such as clinical healthcare, where high impact decisions are made under time pressure, again requiring fast algorithms with perfect coverage.

Machine learning in general is a fast evolving field where models vary over applications and time. Current state-of-the-art classification models, might be outperformed in a few years. To ensure that counterfactual algorithms remain useful when classification models change, and to give data controllers full freedom over the choice of these models, there is a preference for model-agnostic explanation algorithms. This implies that the classification model is used simply as an output generating machine, based on provided input. We have thus identified three important algorithmic requirements. First, we have computational efficiency. Second, perfect coverage, which means we want the counterfactual algorithm to generate explanations for every prediction. And finally we have the required access, which preferably limits itself to the inputs and outputs of the models, making the algorithm model-agnostic. We can already note that there are situations where a trade-off occurs between these properties. For example, many of the current counterfactual algorithms specify a loss function that has to be optimized. At the moment this optimization is only computationally efficient for differentiable classification models. For this group of algorithms, model-agnosism comes at a cost of computational efficiency.



	Income	Age
●	$ 32,000	39
1	$ 40,000	39
2	$ 36,000	44
3	$ 32,000	140

Figure 1: Example of counterfactual explanations for loan approval.

In Figure 1 we show a simplified example of a counterfactual explanation in the domain of credit scoring. In this example a person's loan will be approved or denied based on their income and age. The graph shows a two-dimensional feature space. The black line represents a classification model that splits the feature space in two areas: if a person lands on the area on the left, his loan will be denied, if he lands on the other side, his loan will be approved. Now

a person represented by the black dot applies for a loan. She has an income of $32,000 and is 39 years old. The classification model denies the loan. If this person receives a raise, increasing her income by $8,000, she would land on white dot number 1 in the feature space where her loan will be approved. This $8,000 increase in income is an example of a counterfactual explanation. It is the change that has to be made to an observation in order to change its predicted class. This raise would result in the person being 39 years old and earning $40,000, which is called the counterfactual instance.

In this paper, we propose Nearest Instance Counterfactual Explanations (NICE), a new algorithm to find counterfactual explanations for tabular data with both numerical and categorical variables (summarized hereafter as heterogeneous). This type of data is widely used in machine learning applications where individuals are impacted, such as credit scoring, clinical healthcare, recruitment or fraud detection. Explanations are extremely important in this context as a wrong decision can have serious consequences [7].

Our key contribution is an algorithm that exploits information from a nearest instance in the training set to (1) generate explanations in a reasonable time, (2) for any classification model and (3) for any observation. We also show that these properties are not compromising the quality of our explanations. On the contrary, with an extensive comparison we found that NICE meets this functional requirements and delivers a higher quality than the current state-of-the-art.

2 Related work

Martens and Provost [24] were the first to introduce a counterfactual evidence method named SEDC[1] to explain document classification. Their method has been extended to behavioural data [33], images [39] and finally tabular data [10]. The optimization strategy of NICE is based on these works.

Wachter et al. [41] stated the problem as a loss function to be optimized. With this approach it is easy to impose desired properties on the explanations by adding extra terms to this loss function. Mothilal et al. [27] added a diversity parameter to generate multiple counterfactual explanations for each observation. Others implemented an autoencoder (AE) [22, 4] or prototype [22] loss, resulting in explanations closer to the data manifold. However, for heterogeneous tabular data, these loss functions face some challenges. First of all, they have difficulties handling nominal variables. Some solve this problem by one hot encoding the variables and adding an extra loss term to enforce a correct encoding [27, 14]. Others map the features into an ordinal vector space [5, 22]. A more substantial problem is that these loss functions can only be solved efficiently when the gradients are available, which is only the case for differentiable models. To date, the best performing models for tabular data often include tree-based ensembles [30, 20]. In this case, the gradients have to be calculated numerically which causes a bottleneck. Van Looveren and Klaise [22] have reduced this bottleneck by adding the distance to the training data to the loss function, causing the optimization to converge faster.

The concept of using nearest neighbours from the training set to create explanations has also been explored before. The most basic of this approaches is the What-if Tool (WIT) from Google [42]. This algorithm directly uses the nearest neighbour from a different class as explanation. This approach is very similar to one of the versions of NICE and as the experiments will show, these explanations already have some very desirable properties. Hence, these neighbours do tend to lie far from the instance to explain. Further optimization of this instance, which NICE does, is therefore often needed. Keane and Smyth [17] developed a Case-Based Reasoning (CBR) algorithm to find explanations from nearest neighbours. They start by finding so called native counterfactuals. These are pairs of differently classified points from the training data that differ by one or two features (called difference features). They use these native counterfactual as example cases. To explain an instance, the nearest native counterfactual pair is selected and only the difference features of this case are changed in the original instance. We differ from these methods with our optimization strategy. We apply a best-first heuristic approach that guarantees to find a counterfactual which is a hybrid between the nearest neighbour and the instance to explain. NICE also provides the choice between, sparse, proxy and plausible explanations. We also go further in our experiments by testing on a wide variety of datasets and metrics and by benchmarking ourselves against many existing counterfactual algorithms.

Previous research has pointed out several important properties of counterfactual explanations [16, 39]. **Proximity** refers to the distance between the input data and the counterfactual instance. Ideally, both instances are close to each other, making the explanations easier to act upon. For example, when providing explanations in a credit approval context, it is clear that an explanation that suggest a $8,000 raise is better than one that suggests a $10,000 one. Another closely related property is **sparsity** which refers to the number of features in an explanation. It is often claimed that sparser explanations are better as they are less complex. This statement stems from psychological research which finds that people can only process five to nine pieces of information at once [25]. Especially when working with high

[1] Pronounced as "Set See".

dimensional features spaces, a sparsity constraint is useful to ensure explanations remain comprehensible for humans. In Figure 1, counterfactual 1 is sparser than counterfactual 2. If a loan applicant only wants to change her income and does not want to wait until she gets older, counterfactual 1 is probably a better explanation for her. Pawelczyk et al. [32] pointed out that sparse counterfactual explanations may be vulnerable to classification model changes over time. The counterfactual instance possibly ends up in an area far from the data manifold where these models predict with high uncertainty. When a different classification model is trained on the same data, the previous explanation might no longer be valid. In our loan approval example this would correspond to a case where an applicant is told to raise her income by \$8,000. However, when she returns to the bank, another model has been put into production and her loan request is again rejected. Such occurrences would diminish confidence in counterfactual explanations. In the rest of this paper we will call this concept cross-model robustness. One property of counterfactual explanations that can avoid this problem is **plausibility**. It measures the closeness of the counterfactual instance to the data manifold. If an explanation in the loan approval example such as counterfactual 3 suggests that the person waits until she is 140 years old, this explanation is clearly not plausible as it lies far outside of the data manifold. Compared to the previous two, this property is more conceptual and cannot be measured directly. Proxies that have been used to measure plausibility are: the distance to the k-nearest neighbours [3] from the training data, the local outlier factor [15] and the reconstruction error from an AE trained on the training data [23, 22]. Some studies have shown that there is an inherent trade-off between sparsity and plausibility [3, 22]. Our experiments confirm this result: no single method succeeds in scoring the best on all properties. Different applications therefore require different explanations. NICE provides this flexibility to select the most appropriate explanation.

3 Methodology

In this section, we propose NICE: a nearest neighbour-based approach to generate counterfactual explanations. As we will show in the experiments, using these real instances from the training data substantially decreases runtime while also increasing desirable properties of the explanations such as proximity, plausibility and sparsity. We first explain the search process of our algorithm step by step. Afterwards, we provide more information about the different reward functions used to guide this process.

Assume an m-dimensional feature space $X \subset \mathbb{R}^m$ consisting of both categorical and numerical features, a feature vector $\boldsymbol{x} \in X$ has a corresponding label denoted as $y \in Y = \{-1, 1\}$ and a classification model f is trained which maps $\mathbb{R}^m$ in the class score vector such that $f(\boldsymbol{x}) \in [-1, 1]$ and leads to a predicted class $\hat{y}$. A counterfactual instance $\boldsymbol{x_c}$ for $\boldsymbol{x_0}$ minimizes the distance $d(\boldsymbol{x_0}, \boldsymbol{x_c})$ under the condition that $\hat{y_0} \neq \hat{y_c}$. Our algorithm is very flexible in its distance metric as it does not need the categorical variables to be mapped in an ordinal vector. In this paper, We choose the Heterogeneous Euclidean Overlap Method (HEOM) as a distance metric [43]. For each feature (F), the distance is calculated according to Formula (1), while the total distance is simply the L_1-norm of all feature distances. The L_1-norm is known to induce sparsity when minimized, which is a preferred property of counterfactual explanations. Furthermore, this metric guarantees that the contribution of each feature to the total distance is between 0 and 1. This makes it easy to add a cost multiplier to each feature, forcing the explanation to avoid certain features and prefer others.

$$d_F(a, b) = \begin{cases} 1 \text{ if } a \neq b \text{ for categorical F} \\ 0 \text{ if } a = b \text{ for categorical F} \\ \frac{|a-b|}{range(F)} \text{ for numerical F} \end{cases} \tag{1}$$

Figure 2 shows the process by which NICE searches for a counterfactual explanation. We start by selecting the nearest neighbour $\boldsymbol{x_{nn}}$ from the training set, for which holds: $\hat{y}_0 \neq \hat{y}_{nn}$ and $y_{nn} = \hat{y}_{nn}$. $\boldsymbol{x_{nn}}$ can already be used as a counterfactual instance and has some desirable properties. First, it is a real observation which makes it by definition plausible. In addition, the second condition implies that the observation is correctly classified by f. Therefore $\boldsymbol{x_{nn}}$ corresponds to an area in $\mathbb{R}^m$ where the predictions of f are arguably more justified. If classification model f would be replaced by a different one g, trained on the same data, there would be a higher probability that $\boldsymbol{x_{nn}}$ is also a counterfactual instance. We will refer to this version without optimization as NICE (`none`).

In the next steps, we will optimize certain properties of our explanations by using $\boldsymbol{x_0}$ and $\boldsymbol{x_{nn}}$. The resulting counterfactual instance will always be a combination of these two instances. This significantly reduces our search space and consequently the runtime of our algorithm. The top two rows of Figure 2 show two data instances of $\boldsymbol{x_0}$ and $\boldsymbol{x_{nn}}$, with six features. The black squares represent the feature values for which both instances overlap. The white and gray squares respectively represent the feature values of $\boldsymbol{x_0}$ and $\boldsymbol{x_{nn}}$ for the remaining features. In a first iteration we start from $\boldsymbol{x_0}$ and create all possible combinations in which one non-overlapping feature is replaced with the value of $\boldsymbol{x_{nn}}$: $\boldsymbol{x_{1,1}}$ uses the value of the second feature from $\boldsymbol{x_{nn}}$, $\boldsymbol{x_{1,2}}$ uses the value of the third feature, and $\boldsymbol{x_{1,3}}$ uses

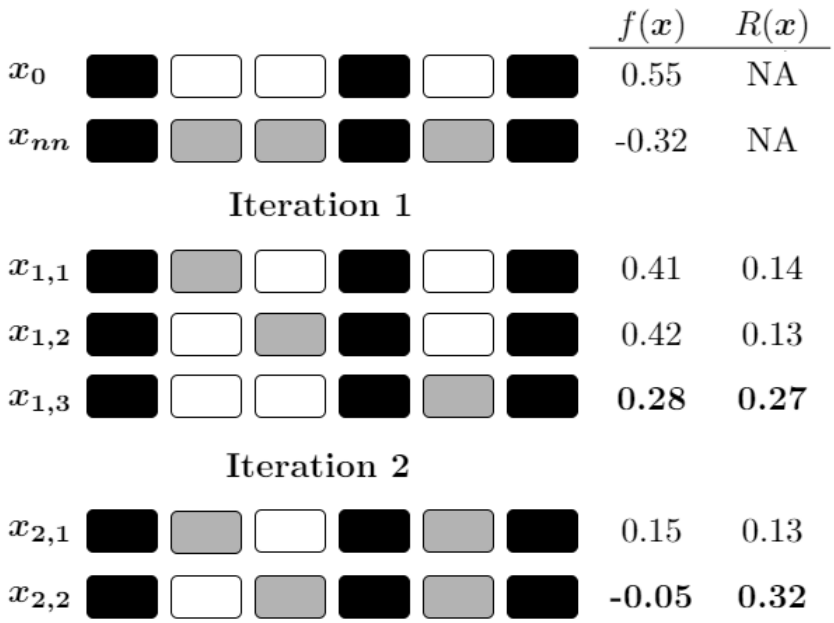


Figure 2: Optimization steps of NICE with the sparsity reward function

the value of the fifth feature. For each of these new hybrid instances we calculate the outcome of a reward function $R(\boldsymbol{x})$, which will be discussed in Section 3.1. The instance with the highest value for $R(\boldsymbol{x})$ has the most desirable properties. We first check if this instance is predicted as the opposite class of $\boldsymbol{x_0}$. If so, we have our counterfactual explanation and stop the search. In our example, this is not the case and we continue our search with $\boldsymbol{x_{1,3}}$. In the next iteration, we check the non-overlapping features of $\boldsymbol{x_{1,3}}$ and $\boldsymbol{x_{nn}}$. Again, we create all possible new combinations where one feature of $\boldsymbol{x_{1,3}}$ is replaced by the feature value of $\boldsymbol{x_{nn}}$. At this point, the candidate with the highest reward function ($\boldsymbol{x_{2,2}}$) is predicted as a different class, so now we have found a counterfactual explanation. If this had not been the case, NICE will continue in the same way until an explanation is found. The beauty of this design is that we will always end up with an explanation, as after the last iteration there is only one candidate left which is $\boldsymbol{x_{nn}}$, for which we know that it is a counterfactual instance.

3.1 Reward Functions

We suggest three reward functions. Each one will measure the effect of a perturbation on the score per unit of sparsity, proximity or plausibility. In the remainder of the paper, we will call these three versions: NICE (`spars`), NICE (`prox`) and NICE (`plaus`). Our reward function assumes a linear relationship between $f(\boldsymbol{x})$ and the concerned property. Despite the fact that this relationship is non-linear for most classification models, the experiments show that our best-first heuristic approach with these reward functions perform very well.

3.1.1 Sparsity

Sparsity happens to be the most straightforward property to optimize with our approach. By simply selecting the perturbation which has the highest prediction score at each iteration, we are effectively optimizing for sparsity as shown in Reward function (2).

$$R(\boldsymbol{x}) = \hat{y} \cdot \frac{f(\boldsymbol{x_{i-1,R_{max}}}) - f(\boldsymbol{x})}{sparsity(\boldsymbol{x_{i-1,R_{max}}}, \boldsymbol{x})} = \hat{y} \cdot (f(\boldsymbol{x_{i-1,R_{max}}}) - f(\boldsymbol{x})) \tag{2}$$

This function compares the score and sparsity of each candidate $\boldsymbol{x}$ with that of the best candidate $\boldsymbol{x_{i-1,R_{max}}}$ from the previous iteration. The sparsity difference between these instances is by definition one because we exactly add one extra feature to the explanation candidate each iteration. This allows us to remove the denominator from the formula. We then end up with a reward function that is exactly the same as the one used by SEDC [10]. The difference is that we replace the feature values with those of $\boldsymbol{x_{nn}}$, while SEDC uses the mean or mode of these features. The factor $\hat{y}$ ensures that the sign of our reward function is correct for both classes.

3.1.2 Proximity

Proximity refers to the distance from the original data point $\boldsymbol{x_0}$ to $\boldsymbol{x_c}$. In the reward function below we have replaced the sparsity measure of function (2) with a proximity measure.

$$R(x) = \hat{y} \cdot \frac{f(\boldsymbol{x_{i-1,R_{max}}}) - f(\boldsymbol{x})}{d(\boldsymbol{x_0}, \boldsymbol{x}) - d(\boldsymbol{x_0}, \boldsymbol{x_{i-1,R_{max}}})} \tag{3}$$

This function effectively calculates the decrease in prediction score per unit of distance. Sparsity and proximity often go hand in hand, and (as our results will show) both optimization methods often lead to the same explanation.

3.1.3 Plausibility

We use the AE reconstruction error as a proxy for plausibility. An AE uses a neural network to project an instance onto a latent space and then tries to reconstruct this instance [19]. The error represents how successful the instance is reconstructed. When we train an AE on our training data, we can use the reconstruction error of an instance to measure how similar it is to this data. A higher error represents a data point farther from the data manifold.

$$R(x) = \hat{y} \cdot \frac{f(\boldsymbol{x_{i-1,R_{max}}}) - f(\boldsymbol{x})}{(AE_{error}(\boldsymbol{x_{i-1,R_{max}}}) - AE_{error}(\boldsymbol{x}))^{-1}} \tag{4}$$

This reward function behaves differently from the two previous ones. First, it is possible that the hybrid instance has an AE error that is larger than the AE error of any of the two real observations. This is not the case with proximity and sparsity where the minimum is bounded by $\boldsymbol{x_0}$ and the maximum by $\boldsymbol{x_{nn}}$. It is even most likely that the AE error is larger for both those instances because it is not a real observation from the dataset. Second, the relationship between the score and the AE error is the most non-linear of all three, which is a challenge for our linear optimization strategy. Despite these downsides, the results show that it still is a valid optimization strategy. Also note that Formula (2) is part of the plausibility reward function (4). Therefore we are also optimizing for sparsity and the resulting explanation will be a balance between these two properties.

4 Experiments

We test NICE on eight datasets from the UCI repository [8] and two larger datasets retrieved from kaggle [13, 36]. Most datasets contain heterogeneous features and many relate to decisions with impact on individuals, such as from the credit scoring (credit_a, german and HCDR), clinical healthcare (cmc, hypothyroid and ICU) and marketing (churn) domains. We train two classification models on each dataset. Namely, A Random Forest classifier (RF) and an Artificial Neural Network (ANN). The data is split in 80% training data and 20% test data. The hyperparameters of each model are trained using a five-fold cross-validation. Finally, explanations are generated for all test instances (with a maximum of 1000 instances). Performance metrics and details about the datasets are shown in Table 1, where all our experiments are run on a Dell Lattitude 5501 notebook with an Intel i7-8665u CPU and 16 GB of working memory.

	AUC (ANN)	AUC (RF)	Instances	Explained Instances	Features	Cat. Features	Num. Features
credit_a	0.913	0.925	690	138	15	10	5
cmc	0.637	0.680	844	169	9	7	2
german	0.744	0.738	1,000	200	20	17	3
hypothyroid	0.972	0.996	3,163	633	25	18	7
churn	0.923	0.923	5,000	1,000	18	3	15
clean2	0.999	0.999	6,598	1,000	166	0	166
magic	0.877	0.991	19,020	1,000	10	0	10
adult	0.898	0.917	48,842	1,000	12	8	4
ICU	0.868	0.882	91,713	1,000	184	8	176
HCDR	0.740	0.731	307,511	1,000	120	16	104

Table 1: Descriptive statistics and performance metrics of all datasets.

We compare our counterfactual algorithm to four existing ones. Each of which has similarities with NICE. The first algorithm is the What-if Tool (WIT) [42]. This interactive tool selects the nearest instance from the training set that is classified in a different class. Besides two small differences, this method is exactly the same as NICE

(none). First, WIT selects a counterfactual instance from the complete trainingset and not only the correctly classified ones. Second, the distance metric is slightly different: for numerical features it standardizes the differences with the standard deviation and not the range of the feature values in the trainingset. The second algorithm is the Case-Based Reasoning system (CBR) for counterfactual explanations [17]. Just like NICE and WIT it also uses nearest instances to find counterfactual. The difference is the optimization method, which limits the search to explanations that have a maximum sparsity of two features. If an explanation is found, it is therefore always very sparse. Also different from NICE, this method does not guarantee that an explanation will be found. The third algorithm in our comparison is SEDC for tabular data [10]. The optimization method is exactly the same as NICE (spars). The difference between both algorithms is their search space. Whereas NICE replaces feature values with those of the nearest instance, SEDC replaces them with the respective mean or mode of each feature. The last algorithm is CFproto [22]. The process of this algorithm is quite different from NICE. CFproto defines a loss function, which it optimizes. What makes this method better than its peers is that it has found a way to work with categorical data in the loss function and is also faster in optimizing it, compared to its gradient-based peers. The purpose of this algorithm is very similar to that of NICE (plaus). It tries to find a balance between close and plausible explanations and uses the training data to achieve this. In our experiments each algorithm is given access to the training data and the class prediction score of the classification model. The speed of CFproto would surely improve when given access to the gradients of the ANN. But to level the playing field we used the model-agnostic version of CFproto in all our experiments. In the next sections we compare all four versions of NICE with WIT, CBR, SEDC and CFproto. We distinguish between the properties of the algorithm and the properties of the explanations. Finally, we also look at the differences between the versions of NICE and provide guidance on their use. A complete overview of all results can be found in Appendix 6.

4.1 Algorithmic requirements

First, we check if all reviewed algorithms meet the algorithmic requirements [35]. These properties are often overlooked but are nevertheless important, as they determine whether these algorithms can be implemented in real-world applications. The three main properties are access, time and coverage [16]. The required access of all these algorithms is the same. Each one needs access to the training data and the scoring output of the classification model. Coverage refers to the percentage of instances for which a counterfactual explanation is found. Finally, time measures the duration it takes to come up with this explanation.

Table 2 panel A shows the average coverage and time over all data sets with the best performing algorithm in bold face. It is immediately noticeable that CFproto needs much more time to generate explanations than all other algorithms. To generate explanations for a RF, CFproto takes on average more than 3 minutes. All others are able to generate explanations in around 1 second or less. When we compare the different versions of NICE, we notice that NICE (none) is the fastest. This makes sense as it does not optimize the explanation after finding a nearest neighbour. Of the optimization versions of NICE, the sparsity version is the fastest followed by the proximity and plausibility versions respectively. Again this makes sense, as calculating the distance or AE error slows down the algorithm at each iteration. The AE error is clearly the most computationally expensive which makes NICE (plaus) the slowest of our three implementations. Yet, even this version continues to generate explanations in around 1 second on average. In addition, these experiments were performed on a basic notebook. The nearest neighbor search and part of NICE's optimization could easily be parallelized, which would reduce the required time even further. As a result, we conclude that except for CFproto all these algorithms are fast enough to be used in real-time applications.

The reported coverage in Table 2 truly shows the strength of NICE. By construction, all versions have a 100% coverage. In reality, anything less than perfect coverage is often not accepted. However, no other algorithm besides WIT complies with this requirement. For SEDC and CBR, this is the main drawback: with respective coverages between 69.0-76.3% or 42.2-59.2%, these algorithms are not useful in many real-world applications where the algorithm should be able to generate an explanation for all predictions. SEDC did always have a perfect coverage on one of both classes. SEDC replaces feature values with their mean or mode. After the maximum number of replacements, the algorithm ends with an instance consisting of only means or modes. If we are looking for a counterfactual instance that belongs to the predicted class of this instance, it will therefore always be found. In some applications such as fraud detection, credit scoring and clinical healthcare, we are mostly interested in explanations for one class. If this matches the class with perfect coverage for SEDC, it is a valid option. CBR limits its search to so called 'good counterfactuals' [17] which have a maximum sparsity of two features. If it would allow more features in its explanations, the coverage could increase. This would however also have an effect on the sparsity of all explanations because the algorithm has no mechanism to search for the sparsest one as it just takes the closest case. CFproto performs better than these two algorithms. With a 97% coverage for the ANN, it is a valid option to explain this model. However for the RF, the coverage drops to 73%. We suspect that the reason for this is that $f(\boldsymbol{x})$ is smoother for a ANN than an RF with small changes in x. For a RF, small changes in the input vector often have no effect on the predicted score, which may be challenging for the optimizer.

	Random Forest (RF)								Artificial Neural Network (ANN)							
	NICE (none)	**NICE** (prox)	**NICE** (spars)	**NICE** (plaus)	**WIT**	**CBR**	**SEDC**	**CFproto**	**NICE** (none)	**NICE** (prox)	**NICE** (spars)	**NICE** (plaus)	**WIT**	**CBR**	**SEDC**	**CFproto**
Panel A: Properties of the counterfactual algorithims. The results are averaged over all datasets for each classification model.																
Coverage (%)	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	42.2	69.0	73.5	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	59.2	76.3	97.2
Time (ms)	**98.6**	418.2	189.2	1314.6	145.7	943.3	159.5	234222.1	84.6	162.9	96.9	463.4	142.7	643.5	**39.0**	25577.3
CM robustness (%)	**68.8**	34.4	38.7	52.7	62.7	26.6	34.8	23.2	**86.4**	56.3	54.5	67.0	75.0	35.2	29.2	38.6
Panel B: Average ranks for each counterfactual property. On all properties, the null-hypothesis of indifferent ranks is rejected at a significance level of 5 percent with a Friedman Rank test. The Nemenyi critical difference for all metrics is 0.12.																
Time	**1.51**	5.16	3.31	6.30	2.25	4.97	4.82	7.69	**1.95**	5.17	3.36	6.21	2.69	4.65	4.21	7.76
Proximity	5.78	**2.09**	2.32	4.04	5.53	6.52	5.26	4.45	6.35	**2.28**	2.59	4.31	5.98	5.92	4.70	3.88
Sparsity	6.01	2.88	**2.07**	4.10	5.94	5.82	4.12	5.07	6.51	2.89	**2.24**	4.25	6.43	4.99	3.84	4.84
AE-Error	3.79	4.56	4.49	**3.53**	3.72	6.40	4.14	5.36	**3.93**	4.68	4.66	**3.92**	**3.92**	6.05	4.09	4.75
5NN distance	**2.20**	4.72	4.77	3.46	**2.18**	6.74	6.15	5.77	**2.21**	4.90	4.87	3.63	**2.23**	6.49	6.12	5.56
Panel C: The percentage of explanations for which each model scored best on a metric.																
Time	**58.8**	0.0	0.0	0.0	28.0	2.0	11.7	0.0	**44.3**	1.2	6.3	0.0	20.0	17.8	15.3	0.0
Proximity	1.7	**55.4**	34.7	16.0	2.0	5.5	7.8	23.6	0.8	**49.0**	30.1	14.4	1.5	8.2	13.6	27.4
Sparsity	2.2	39.8	**77.1**	29.7	2.3	16.2	39.6	12.6	1.8	47.8	**68.7**	31.7	2.4	29.8	45.6	9.5
AE Error	24.2	7.5	7.0	21.7	25.4	8.4	**32.6**	11.2	26.8	6.8	6.8	14.9	26.3	10.3	**32.5**	14.6
5NN distance	69.5	8.1	5.5	27.5	**71.0**	4.2	5.6	4.4	**69.8**	6.5	5.8	25.0	68.9	6.1	4.2	4.4

Table 2: Summarized results of all experiments.

Based on these findings, we can conclude that NICE and WIT have the most desirable algorithmic properties. The perfect coverage ensures that these algorithms can be used in applications where explanations are required by law such as under Fair Credit Reporting Act [37]. All versions of NICE also have an efficient runtime. This is a must for high-stakes decision making under time pressure like fraud detection, credit scoring and clinical healthcare. Appendix 6 shows that all versions of NICE scale well with the number of features. Even for the largest datasets with over 180 features, it has an average runtime of less than 3 seconds and a maximum of 10 seconds. This makes NICE useful in Fraud detection and other domains were many predictions are made each second and scalability is a priority. Taking in account all algorithmic requirements, we conclude that NICE and WIT are the best option for any generic classification model. The other algorithms are useful in specific situations.

4.2 Explanation Requirements

We compare all algorithms on three main requirements for the explanations: sparsity, proximity and plausibility. The first two are well defined as discussed before. Plausibility is less straightforward to measure. The AE error is a good proxy but will bias the comparison in favor of NICE (`plaus`), as it is the only algorithm that optimizes for this metric. Therefore we added two more metrics to measure plausibility: the average distance to the 5 nearest neighbours (5NN) and a measure taken from [32] which we call cross-model robustness. It is argued that if counterfactual instances respect the data-manifold, they are less vulnerable to classification model uncertainty or changes over time [32]. We can measure this by checking the percentage of instances for which an explanation is also valid for another classification model trained on the same data. We use two classification models in our experiments, so we can easily check whether an explanation for one model is also an explanation for the other.

The imperfect coverage of CBR, SEDC and CFproto makes the comparison challenging. Uncovered instances are typically harder to explain, which result in explanations with inferior properties for the algorithms that are able to explain them. For this reason, simple averages over all covered instances for each respective algorithm (see Appendix 6) can give a biased view. Furthermore, the overlapping sample of instances that are covered by all algorithms is very small and biased to the algorithms with lower coverage. On top of that, the values for different datasets are often not comparable. This is the case for sparsity, proximity, AE error and 5NN distance. For these metrics, we rank the data for each observation, giving a rank of 1 to the best performing explanation algorithm and a rank of 8 to the worst. If no counterfactual is available, we give this algorithm the worst rank for this observation. If there is a tie, we use the average rank. We report the averages over all datasets of these ranks in panel B of Table 2 and submit them to the Friedman test [11, 12] with a significance of 5%. If this test reject the null hypothesis of indifferent rank-means, we calculate the critical difference using a Nemenyi test [28]. When a pair-wise difference between average ranks exceeds this critical value, we conclude that their difference is statistically significant. The lowest ranks for each metric in Table 2 are in bold face and underlined. Ranks that are not significant worse than the lowest one are just in bold face. Of course, low coverage has a major impact on the ranks, which mainly explains the poor performance of SEDC and CBR on these tests. To illustrate the influence of the coverage, we show another metric. In panel C of Table 2 the percentage of explanations, for which each algorithm has the best performance on a metric, is shown.[2] Because these results are also expressed in percentages, they can be easily related with coverage. Finally, for cross-model robustness we just show the percentage of observations which are resistant to classification model changes.

For all means, the Friedman test rejects the null hypothesis of indifference. All metrics are compared over the same number of observations and algorithms, causing the critical difference to be always 0.12. Most differences are statistically significant according to he post-hoc Nemeyi test. Two notable exceptions, which we will return to later, are the mutual differences between firstly NICE (`none`) and WIT and secondly NICE (`spars`) and NICE (`prox`).

We first look at the proximity of all explanations. All algorithms have some sort of proximity constraint. Being in direct form or induced by a sparsity constraint. The best performing algorithm in terms of proximity is NICE (`prox`) for both classification models with a significant difference. A close second is NICE (`spars`) followed by NICE (`plaus`). NICE (`none`) is among the worst performing algorithms based on proximity. Despite the perfect coverage, it is the second-last-ranked algorithm for an ANN . This shows how challenging it is to find a close explanation in the observations of the training set. Next, we take a look at sparsity. The main aim of both SEDC, CBR and NICE (`spars`) is to generate the most sparse counterfactual possible. The latter clearly appears to be the winner here. For both classification models it has by far the lowest average rank (2.07 and 2.24). Again SEDC and CBR are amongst the worst performing algorithms based on their rank. The percentage best in panel C shows that this is mainly because of their lower coverage. If an explanation is found, it is the sparsest of *all* explanations around half of the time for both algorithms. For example, SEDC finds counterfactual explanations in 69.0% of the cases for a RF, and of those it has the sparsest solution of all algorithms in 57.3% (=39.6/69.0) of the time. But also based on this metric, NICE (`spars`) has the best performance by providing the sparsest explanation in 77.1% of all observations.

[2]Note that the sum of each metric is not always equal to 100 percent. This is due to draws between the best scoring algorithms.

Finally, we take a look at the plausibility of all counterfactual explanations. Both NICE (`plaus`) and CFproto's main goal is to generate explanations which lie close to the data manifold. The average ranks for the AE error lie closer to each other than the proximity and sparsity ranks. For a RF, NICE (`plaus`) scores the best on this metric as expected. For an ANN, NICE (`plaus`), NICE (`none`) and WIT are better than the other algorithms, but there is no significant difference between them. In the ranks for the average 5NN distance, we see the same three algorithms appearing at the top. NICE (`none`) and WIT are tied for the first place followed by NICE (`plaus`). Finally, we also see this pattern in the cross-model robustness. NICE (`none`) is the best performing algorithm with 68.8% and 86.4% of cross-model stability for the RF and ANN respectively. This is higher than the 62.7% and 75.0% for WIT. NICE (`none`) only looks for counterfactual instances among the correctly classified observations from the training set, which avoids areas of uncertainty in the feature space. As we have seen before, this comes with a cost of proximity and a benefit in computing time. In third place, we have again NICE (`plaus`) were 52.7% and 67% of the explanations are robust to changes in the classification model. CFproto underperforms in terms of plausibility. If we zoom in on the results of the ANN, where coverage is no problem for CFproto, we see that NICE (`plaus`) scores significantly better than CFproto for all metrics but proximity. NICE's other two optimization techniques are even significantly better for every single metric. Based on these results, we would not recommend to use CFproto to generate counterfactual explanations. The same goes for SEDC and CBR, which are less useful in a general setup due to their low coverage. In specific situations where a perfect coverage can be guaranteed, such as for one of the classes with SEDC, these algorithms may still be optimal as they have proven to generate high quality explanations. For WIT and all implementations of NICE we do see the potential general value. Also note that both SEDC and CFproto are not limited to tabular data. For other applications such as explaining images, these algorithms can also be useful [40, 22].

Again, All these show that a trade-off must always be made between plausibility on the one hand and proximity or sparsity on the other hand. None of the algorithms perform best on all metrics, but all implementations of NICE represent a good trade-off. When plausibility is desired, potentially at the expense of proximity, NICE (`none`) is the best option. Imagine, for example, a situation where the classification model often changes drastically. The high cross-model robustness then ensures that the explanations usually remain usable when adjusting the classification model. When we take a small step to the proximity side of the spectrum, we have WIT. This algorithm generates slightly closer explanations at the expense of cross-model robustness and computation time. In the middle of the spectrum we have NICE (`plaus`) this method scores well on all metrics and provides a good trade-off. When proximity or sparsity may come at a cost of plausibility, we advice to select NICE with their respective optimization method. And note again that in these situations the plausibility is still not that bad. With NICE, the counterfactual instance is always a combination of two existing observations, which is favorable for plausibility.

5 Conclusion

In this paper, we introduced NICE, an algorithm to generate counterfactual explanations for heterogeneous tabular data. NICE is able to provide fast explanations for different types of classification models with a perfect coverage, thereby making it suitable for real-world applications where these algorithmic properties are expected. In the extensive experiments, we have shown that the four versions of NICE provided results with favorable properties compared to existing algorithms. Once again we notice a trade-off between on the one hand plausiblity and on the other hand proximity or sparsity. Our different optimization methods offer the choice to select the preferred trade-off. NICE exploits information from the training data to decrease the search time, guarantee perfect coverage and increase the plausibility of counterfactual explanations. A downside of this approach is that the training data is required, which is not always the case. Think for example of settings where third party APIs are used to make predictions. Future research should test these algorithms in real-life settings and use feedback from all stakeholders in the predictive decision making process. A multidisciplinary approach with data scientists, domain experts and end users is needed to further improve the properties of counterfactual explanations and see for which applications they are most valuable. Furthermore, it would be interesting to test if NICE's approach could be extended to different datatypes such as behavioural, image or time-series data.

References

[1] Solon Barocas, Andrew D Selbst, and Manish Raghavan. "The hidden assumptions behind counterfactual explanations and principal reasons". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 2020, pp. 80–89.

[2] Alison Callahan and Nigam H Shah. "Machine learning in healthcare". In: *Key Advances in Clinical Informatics*. Elsevier, 2017, pp. 279–291.

[3] Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. "Multi-objective counterfactual explanations". In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2020, pp. 448–469.

[4] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. "Explanations based on the missing: Towards contrastive explanations with pertinent negatives". In: *Advances in neural information processing systems* 31 (2018), pp. 592–603.

[5] Amit Dhurandhar, Tejaswini Pedapati, Avinash Balakrishnan, Pin-Yu Chen, Karthikeyan Shanmugam, and Ruchir Puri. "Model agnostic contrastive explanations for structured data". In: *arXiv preprint arXiv:1906.00117* (2019).

[6] Luciano A Digiampietri, Norton Trevisan Roman, Luis AA Meira, Jorge Jambeiro Filho, Cristiano D Ferreira, Andreia A Kondo, Everton R Constantino, Rodrigo C Rezende, Bruno C Brandao, Helder S Ribeiro, et al. "Uses of artificial intelligence in the Brazilian customs fraud detection system". In: *Proceedings of the 2008 international conference on digital government research*. 2008, pp. 181–187.

[7] Finale Doshi-Velez and Been Kim. "Towards a rigorous science of interpretable machine learning". In: *arXiv preprint arXiv:1702.08608* (2017).

[8] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: `http://archive.ics.uci.edu/ml`.

[9] European Parliament. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. 2016.

[10] Carlos Fernández-Loría, Foster Provost, and Xintian Han. *Explaining Data-Driven Decisions made by AI Systems: The Counterfactual Approach*. 2020. arXiv: `2001.07417 [cs.LG]`.

[11] Milton Friedman. "The use of ranks to avoid the assumption of normality implicit in the analysis of variance". In: *Journal of the american statistical association* 32.200 (1937), pp. 675–701.

[12] Milton Friedman. "A comparison of alternative tests of significance for the problem of m rankings". In: *The Annals of Mathematical Statistics* 11.1 (1940), pp. 86–92.

[13] Home Credit Group. *Home Credit Default Risk*. `https://www.kaggle.com/c/home-credit-default-risk/data`. Accessed: 2020-09-15. 2018.

[14] Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. "Towards realistic individual recourse and actionable explanations in black-box decision making systems". In: *arXiv preprint arXiv:1907.09615* (2019).

[15] Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, and Hiroki Arimura. "DACE: Distribution-Aware Counterfactual Explanation by Mixed-Integer Linear Optimization". In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, Christian Bessiere (Ed.). International Joint Conferences on Artificial Intelligence Organization*. 2020, pp. 2855–2862.

[16] Amir-Hossein Karimi, Gilles Barthe, Bernhard Schölkopf, and Isabel Valera. "A survey of algorithmic recourse: definitions, formulations, solutions, and prospects". In: *arXiv preprint arXiv:2010.04050* (2020).

[17] Mark T Keane and Barry Smyth. "Good Counterfactuals and Where to Find Them: A Case-Based Technique for Generating Counterfactuals for Explainable AI (XAI)". In: *arXiv preprint arXiv:2005.13997* (2020).

[18] Boris Kment. "Counterfactuals and explanation". In: *Mind* 115.458 (2006), pp. 261–310.

[19] Mark A Kramer. "Nonlinear principal component analysis using autoassociative neural networks". In: *AIChE journal* 37.2 (1991), pp. 233–243.

[20] Stefan Lessmann, Bart Baesens, Hsin-Vonn Seow, and Lyn C Thomas. "Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research". In: *European Journal of Operational Research* 247.1 (2015), pp. 124–136.

[21] David Lewis. *Counterfactuals*. John Wiley & Sons, 2013.

[22] Arnaud Van Looveren and Janis Klaise. "Interpretable Counterfactual Explanations Guided by Prototypes". In: *CoRR* abs/1907.02584 (2019). arXiv: 1907.02584. URL: http://arxiv.org/abs/1907.02584.

[23] Divyat Mahajan, Chenhao Tan, and Amit Sharma. "Preserving causal constraints in counterfactual explanations for machine learning classifiers". In: *arXiv preprint arXiv:1912.03277* (2019).

[24] David Martens and Foster Provost. "Explaining Data-Driven Document Classifications". In: *MIS Quarterly* 38.1 (2014), pp. 73–100. ISSN: 02767783, 21629730. URL: https://www.jstor.org/stable/26554869.

[25] George A Miller. "The magical number seven, plus or minus two: Some limits on our capacity for processing information." In: *Psychological review* 63.2 (1956), p. 81.

[26] Tim Miller. "Explanation in artificial intelligence: Insights from the social sciences". In: *Artificial Intelligence* 267 (2019), pp. 1–38. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2018.07.007. URL: http://www.sciencedirect.com/science/article/pii/S0004370218305988.

[27] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. "Explaining machine learning classifiers through diverse counterfactual explanations". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 2020, pp. 607–617.

[28] Peter Nemenyi. "Distribution-free multiple comparisons". In: *Biometrics*. Vol. 18. 2. International Biometric Soc 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210. 1962, p. 263.

[29] Eric WT Ngai, Yong Hu, Yiu Hing Wong, Yijun Chen, and Xin Sun. "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature". In: *Decision support systems* 50.3 (2011), pp. 559–569.

[30] Randal S Olson, William La Cava, Patryk Orzechowski, Ryan J Urbanowicz, and Jason H Moore. "PMLB: a large benchmark suite for machine learning evaluation and comparison". In: *BioData mining* 10.1 (2017), pp. 1–13.

[31] *Path to Apple Card*. https://support.apple.com/en-us/HT211030. Accessed: 2021-04-09.

[32] Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. "On Counterfactual Explanations under Predictive Multiplicity". In: *Conference on Uncertainty in Artificial Intelligence*. PMLR. 2020, pp. 809–818.

[33] Yanou Ramon, David Martens, Foster Provost, and Theodoros Evgeniou. "A comparison of instance-level counterfactual explanation algorithms for behavioral and textual data: SEDC, LIME-C and SHAP-C". In: *Advances in Data Analysis and Classification* (2020), pp. 1–19.

[34] David-Hillel Ruben. *Explaining explanation*. Routledge, 2015.

[35] Kacper Sokol and Peter Flach. "Explainability fact sheets: a framework for systematic assessment of explainable approaches". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 2020, pp. 56–67.

[36] Global Women in Data Science Conference The Global Open Source Severity of Illness Score Consortium. *Intesnive Care Unit*. https://www.kaggle.com/c/widsdatathon2020/data. Accessed: 2020-09-15. 2020.

[37] United States Congress. *An Act to amend the Federal Deposit Insurance Act to require insured banks to maintain certain records, to require that certain transactions in U.S. currency be reported to the Department of the Treasury, and for other purposes.* 1970.

[38] Jellis Vanhoeyveld, David Martens, and Bruno Peeters. "Value-added tax fraud detection with scalable anomaly detection techniques". In: *Applied Soft Computing* 86 (2020), p. 105895.

[39] Sahil Verma, John Dickerson, and Keegan Hines. *Counterfactual Explanations for Machine Learning: A Review*. 2020. arXiv: 2010.10596 [cs.LG].

[40] Tom Vermeire and David Martens. "Explainable Image Classification with Evidence Counterfactual". In: *arXiv preprint arXiv:2004.07511* (2020).

[41] Sandra Wachter, Brent Mittelstadt, and Chris Russell. "Counterfactual explanations without opening the black box: Automated decisions and the GDPR". In: *Harv. JL & Tech.* 31 (2017), p. 841.

[42] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. "The what-if tool: Interactive probing of machine learning models". In: *IEEE transactions on visualization and computer graphics* 26.1 (2019), pp. 56–65.

[43] D Randall Wilson and Tony R Martinez. "Improved heterogeneous distance functions". In: *Journal of artificial intelligence research* 6 (1997), pp. 1–34.

6 Appendix

	NICE (none)	**NICE** (prox)	**NICE** (spars)	**NICE** (plaus)	**WIT**	**CBR**	**SEDC**	**CFproto**
				Time (ms)				
credit_a	**4.1**	35.5	11.1	94.4	5.2	11.4	26.2	55,839.9
cmc	**4.4**	35.2	11.5	83.5	7.0	28.1	18.2	49,537.2
german	7.5	53.6	22.7	147.0	**5.8**	45.4	17.4	49,490.7
hypothyroid	7.2	68.7	27.1	178.2	**5.7**	42.0	35.7	55,958.0
churn	**4.7**	45.4	13.7	95.2	5.0	37.4	15.9	52,275.2
clean2	**9.5**	259.6	83.9	1977.9	11.4	31.6	153.6	58,903.6
magic	**82.2**	1301.8	433.1	2297.1	86.1	1495.0	369.8	774,119.0
adult	**49.6**	271.7	87.0	387.4	52.1	589.3	156.3	468,297.2
ICU	**114.4**	463.5	209.3	3006.4	188.9	1135.4	227.8	132,912.9
HCDR	436.0	579.0	498.7	1453.2	690.3	3404.2	**91.9**	124,440.4
				Coverage (%)				
credit_a	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	88.4	74.6	12.3
cmc	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	87.0	90.5	88.8
german	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	33.5	59.5	74.5
hypothyroid	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	34.0	99.7	70.6
churn	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	55.7	21.5	71.0
clean2	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	0.5	61.8	31.6
magic	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	79.3	96.0	100.0
adult	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	81.9	65.1	64.5
ICU	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	3.1	48.0	89.5
HCDR	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	26.0	100.0	91.7
				Sparsity				
credit_a	6.85	2.46	**1.56**	4.16	6.73	1.95	2.07	3.71
cmc	2.17	1.63	**1.48**	1.80	2.14	1.67	1.67	2.21
german	6.81	2.48	1.99	4.40	6.73	1.81	**1.39**	2.19
hypothyroid	6.36	3.63	3.24	4.91	6.36	**1.99**	3.85	5.13
churn	13.55	2.42	2.12	6.39	13.55	1.82	**1.41**	7.79
clean2	138.88	19.48	17.00	98.86	138.88	**2.00**	12.29	62.25
magic	9.99	5.55	4.64	7.99	9.99	1.95	**1.82**	3.56
adult	3.15	1.41	**1.37**	1.59	3.14	1.69	2.50	1.75
ICU	101.12	13.91	10.40	76.03	100.56	**1.87**	8.52	79.56
HCDR	36.04	5.91	3.25	21.61	35.41	**1.88**	3.74	21.31
				Proximity				
credit_a	2.87	**1.18**	1.24	2.02	2.81	1.19	1.45	1.25
cmc	0.89	**0.73**	0.75	0.81	0.85	0.74	1.19	1.11
german	4.56	1.40	1.53	3.05	4.47	1.81	**1.21**	1.31
hypothyroid	1.85	1.15	1.25	1.53	1.85	**0.77**	1.92	1.36
churn	1.35	**0.41**	0.42	0.75	1.34	0.41	0.50	0.86
clean2	10.18	1.86	2.56	7.35	10.18	**0.81**	3.11	5.17
magic	0.31	0.19	0.20	0.27	0.31	0.45	0.32	**0.14**
adult	1.12	**0.39**	0.41	0.46	1.11	0.81	1.97	0.57
ICU	14.32	1.65	2.10	10.76	13.78	**0.86**	1.80	6.94
HCDR	5.71	**0.76**	0.99	3.58	5.26	1.26	1.46	3.06

Table 3: Average Time,Coverage,Sparsity and Proximity for each dataset under with a Random Forest classification model

	NICE (none)	**NICE** (prox)	**NICE** (spars)	**NICE** (plaus)	**WIT**	**CBR**	**SEDC**	**CFproto**
	AE error							
credit_a	0.1085	0.1117	0.1107	0.1075	0.1079	0.1073	**0.1058**	0.1155
cmc	0.1324	0.1355	0.1347	0.1325	0.1292	0.1342	**0.1096**	0.1398
german	0.1341	0.1437	0.1429	0.1356	**0.1331**	0.1522	0.1405	0.1458
hypothyroid	0.0701	0.0840	0.0823	0.0749	0.0701	0.0853	**0.0495**	0.0765
churn	0.1510	0.1605	0.1600	0.1552	**0.1500**	0.1532	0.1595	0.1683
clean2	0.2985	0.3072	0.3065	0.2918	0.2985	0.3254	0.2921	**0.2833**
magic	0.5524	0.5544	0.5543	0.5491	0.5525	0.5339	**0.5148**	0.5608
adult	0.0576	0.0604	0.0605	0.0601	0.0577	0.0594	**0.0548**	0.0626
ICU	0.2503	0.2558	0.2563	0.2503	0.2490	0.2628	0.2566	**0.2438**
HCDR	0.3274	0.3294	0.3290	0.3275	0.3273	0.3333	**0.3229**	0.3350
	5NN Distance							
credit_a	**1.34**	2.04	2.03	1.71	1.34	2.26	1.83	2.12
cmc	0.35	0.42	0.40	0.38	0.34	0.45	**0.33**	0.50
german	3.01	4.07	4.01	3.47	**2.97**	4.66	4.23	4.45
hypothyroid	0.48	0.87	0.79	0.66	**0.48**	0.74	0.93	0.74
churn	0.81	1.09	1.08	0.99	**0.81**	1.10	1.10	1.09
clean2	**3.57**	5.79	6.04	4.90	**3.57**	5.90	6.69	5.88
magic	0.17	0.21	0.21	0.18	**0.17**	0.25	0.28	0.25
adult	**0.25**	0.44	0.44	0.41	0.26	0.44	0.50	0.50
ICU	7.93	11.76	11.83	9.28	**7.86**	13.10	13.75	10.92
HCDR	3.28	4.89	4.84	3.95	**3.26**	4.92	5.05	5.85
	Cross-Model Robustness (%)							
credit_a	**84.1**	73.9	76.1	79.7	81.9	71.0	48.6	6.5
cmc	52.1	55.6	**55.6**	53.3	52.7	52.1	46.7	38.5
german	**80.5**	44.5	48.0	62.0	73.5	24.5	34.5	43.5
hypothyroid	**42.7**	24.0	25.6	28.8	42.7	11.8	27.3	19.1
churn	**80.1**	56.5	57.0	64.1	76.6	37.7	16.6	19.3
clean2	**99.8**	5.7	20.5	68.3	99.8	0.2	24.7	18.0
magic	22.7	22.1	22.3	22.4	22.5	**54.3**	54.5	20.0
adult	**53.5**	38.2	38.2	38.8	53.1	44.8	46.8	13.6
ICU	**89.0**	19.6	23.0	63.1	55.5	1.6	16.3	23.1
HCDR	**82.4**	59.8	69.5	69.3	78.6	20.0	50.6	43.4

Table 4: Average Plausibility metrics for each dataset under a Random Forest classification model

	NICE (none)	**NICE** (prox)	**NICE** (spars)	**NICE** (plaus)	**WIT**	**CBR**	**SEDC**	**CFproto**
				Time (ms)				
credit_a	**1.6**	15.8	3.6	63.4	2.1	5.9	7.8	16294.1
cmc	**2.1**	18.1	4.7	35.9	3.5	11.1	6.5	14023.1
german	3.3	27.7	10.6	89.1	**2.8**	29.6	9.4	18201.0
hypothyroid	3.2	38.9	14.6	90.6	**2.6**	21.6	15.0	16877.5
churn	**1.8**	28.1	5.1	52.1	2.0	19.5	6.4	13615.4
clean2	**8.0**	243.4	56.6	887.8	9.0	58.0	43.0	13817.4
magic	**2.4**	23.7	5.1	30.7	2.9	47.6	4.7	12110.2
adult	**4.8**	22.6	8.5	44.9	5.4	142.2	10.4	15087.6
ICU	**100.3**	335.5	127.0	1242.6	202.7	1392.3	122.4	61370.2
HCDR	483.2	474.7	477.1	960.8	794.0	2912.9	**34.4**	47679.0
				Coverage (%)				
credit_a	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	63.8	65.2	59.4
cmc	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	94.1	83.4	98.2
german	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	35.0	70.5	98.5
hypothyroid	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	21.2	61.0	98.7
churn	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	94.3	25.6	100.0
clean2	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	7.7	82.5	100.0
magic	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	94.3	95.6	100.0
adult	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	84.4	66.7	87.4
ICU	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	20.5	98.7	99.4
HCDR	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	76.5	100.0	100.0
				Sparsity				
credit_a	6.89	1.43	**1.28**	4.71	6.65	1.95	1.44	5.23
cmc	2.34	1.51	**1.41**	1.85	2.24	1.72	1.69	2.39
german	6.90	2.54	2.08	4.21	6.87	**1.64**	1.95	3.12
hypothyroid	6.43	3.72	3.53	4.61	6.43	**1.98**	3.09	5.61
churn	13.56	2.43	2.21	7.05	13.58	1.99	**1.96**	6.90
clean2	138.67	17.51	13.34	70.46	138.60	**1.99**	5.65	60.90
magic	9.99	3.09	2.58	5.96	9.99	**1.92**	2.05	5.14
adult	3.28	1.90	1.75	2.05	3.20	**1.71**	2.27	2.14
ICU	99.28	12.21	5.66	60.36	98.58	**1.90**	7.06	72.41
HCDR	35.55	2.53	**1.79**	17.04	34.73	1.85	2.84	15.96
				Proximity				
credit_a	2.93	**1.19**	1.22	2.21	2.79	1.31	1.37	2.40
cmc	1.14	**0.97**	0.98	1.05	1.07	1.10	1.34	1.24
german	4.66	1.85	1.95	3.01	4.62	**1.64**	1.78	1.83
hypothyroid	2.10	1.62	1.65	1.79	2.09	**1.00**	1.54	1.45
churn	1.39	**0.45**	0.47	0.84	1.38	0.62	0.63	0.90
clean2	10.18	2.77	3.00	5.70	10.16	**1.03**	2.18	4.66
magic	0.50	0.31	0.31	0.40	0.48	0.57	0.41	**0.28**
adult	1.26	0.80	0.81	0.87	1.21	0.97	1.72	**0.73**
ICU	13.29	1.78	2.30	8.49	12.96	**1.17**	2.54	160.42
HCDR	5.59	**0.77**	0.89	2.98	5.04	0.86	1.62	3.66

Table 5: Average Time, Coverage, Sparsity and Proximity for each dataset under an Artificial Neural Network classification model

	NICE (none)	**NICE** (prox)	**NICE** (spars)	**NICE** (plaus)	**WIT**	**CBR**	**SEDC**	**CFproto**
	AE error							
credit_a	0.1092	0.1122	0.1120	0.1108	**0.1091**	0.1100	0.1102	0.1187
cmc	0.1456	0.1475	0.1470	0.1474	0.1440	0.1482	**0.1211**	0.1434
german	0.1334	0.1390	0.1383	0.1337	0.1324	0.1482	**0.1317**	0.1499
hypothyroid	**0.0624**	0.0734	0.0719	0.0681	0.0631	0.0847	0.0702	0.0707
churn	0.1598	0.1677	0.1675	0.1642	0.1578	**0.1527**	0.1630	0.1692
clean2	0.2986	0.3026	0.3018	0.2965	0.2987	0.3210	0.2970	**0.2851**
magic	0.3115	0.3147	0.3150	0.3115	0.3120	0.3240	**0.2870**	0.3195
adult	0.0579	0.0593	0.0594	0.0589	0.0580	0.0592	**0.0550**	0.0602
ICU	0.2467	0.2539	0.2543	0.2497	0.2472	0.2593	**0.2453**	0.2483
HCDR	0.3255	0.3271	0.3269	0.3255	0.3249	0.3274	**0.3217**	0.3360
	5NN Distance							
credit_a	**1.38**	2.08	2.06	1.67	1.39	2.07	2.05	3.00
cmc	0.38	0.43	0.42	0.40	**0.37**	0.50	0.38	0.59
german	3.08	3.93	3.88	3.51	**3.02**	4.47	4.08	4.69
hypothyroid	**0.54**	0.74	0.73	0.68	0.55	0.79	1.26	0.83
churn	0.82	1.09	1.08	0.98	**0.81**	1.23	1.16	1.07
clean2	**3.58**	6.04	6.05	5.30	3.58	6.30	6.20	6.24
magic	**0.20**	0.26	0.26	0.22	0.20	0.24	0.33	0.27
adult	**0.25**	0.38	0.37	0.34	0.26	0.45	0.61	0.54
ICU	**7.44**	11.37	11.29	9.28	7.48	12.94	11.40	165.36
HCDR	**3.26**	4.84	4.80	4.07	3.28	4.81	4.73	6.39
	Cross-Model Robustness (%)							
credit_a	**91.3**	72.5	71.7	85.5	76.1	47.1	30.4	6.5
cmc	**84.6**	74.6	74.6	79.3	63.9	71.6	36.1	45.0
german	**93.0**	66.0	65.0	73.5	88.0	14.5	45.5	62.0
hypothyroid	**100.0**	69.5	66.8	71.9	88.3	10.9	24.3	54.8
churn	**98.9**	81.0	81.9	86.9	92.2	50.4	16.3	59.3
clean2	**100.0**	8.3	7.8	46.2	99.7	1.6	4.2	11.1
magic	**96.8**	74.1	72.4	86.4	70.6	66.0	65.8	48.8
adult	**93.8**	88.0	86.2	90.5	80.7	60.8	33.1	60.2
ICU	**38.4**	18.8	15.1	24.3	33.0	7.7	13.3	18.5
HCDR	**80.3**	52.2	47.9	58.8	64.2	36.2	40.9	22.3

Table 6: Average plausibility metrics for each dataset under an Artificial Neural Network classification model