Actionable feature discovery in counterfactuals using feature relevance explainers.

WIRATUNGA, N., WIJEKOON, A., NKISI-ORJI, I., MARTIN, K., PALIHAWADANA, C. and CORSAR, D.

2021

Actionable Feature Discovery in Counterfactuals using Feature Relevance Explainers*

Nirmalie Wiratunga, Anjana Wijekoon, Ikechukwu Nkisi-Orji, Kyle Martin, Chamath Palihawadana, and David Corsar

School of Computing, Robert Gordon University, Aberdeen AB10 7GJ, Scotland, UK
{n.wiratunga, a.wijekoon1, i.nkisi-orji, k.martin3, c.palihawadana, d.corsar1}@rgu.ac.uk

Abstract. Counterfactual explanations focus on "actionable knowledge" to help end-users understand how a Machine Learning model outcome could be changed to a more desirable outcome. For this purpose a counterfactual explainer needs to be able to reason with similarity knowledge in order to discover input dependencies that relate to outcome changes. Identifying the minimum subset of feature changes to action a change in the decision is an interesting challenge for counterfactual explainers. In this paper we show how feature relevance based explainers (i.e. LIME, SHAP), can inform a counterfactual explainer to identify the minimum subset of "actionable features". We demonstrate our DisCERN (Discovering Counterfactual Explanations using Relevance Features from Neighbourhoods) algorithm on three datasets and compare against the widely used counterfactual approach DiCE. Our preliminary results show that DisCERN to be a viable strategy that should be adopted to minimise the actionable changes.

Keywords: Explainable AI, Counterfactual, Feature Relevance, Actionable Features

1 Introduction

Understanding a user's explanation need is central to a system's capability of provisioning an explanation which satisfies that need [7]. Typically an explanation generated by an AI system is considered to convey the internal state or workings of an algorithm that resulted in the system's decision [14]. In Machine Learning (ML) the decision tends to be a discrete label or class (or in the case of regression tasks a numeric value). Although explanations focused on the internal state or logic of the algorithm are helpful to ML researchers they are arguably less useful to an end-user who may be more interested in how their current circumstances could be changed to receive a desired (better) outcome in the future.

* This research is funded by the iSee project (https://isee4xai.com/) which received funding from EPSRC under the grant number EP/V061755/1. iSee is part of the CHIST-ERA pathfinder programme for European coordinated research on future and emerging information and communication technologies.

This calls for explanations that focus on discovering relationships between the input dependencies that led to the system's decision.

Nearest-Like Neighbour (NLN) based explainer, could focus on input dependencies by identifying similarity relationships between the current problem and the retrieved nearest neighbour [5]. Research has shown that when similarity computation is focused on feature selection and weighting, it can significantly improve retrieval of NLN [16, 15]. Accordingly it would be reasonable to expect that NLN-based explanation generation would also benefit from the knowledge of feature importance. Certainly having to focus on a few key features in domains with large numbers of features is likely to improve the cognitive burden of understanding NLN-based explanations.

Unlike NLN based explanations, a counterfactual explanation focuses on identifying "actionable knowledge"; which is knowledge about important causal dependencies between the input and the ML decision. Such knowledge helps to understand what could be changed in the input to achieve a preferred (desired) decision outcome. Typically a Nearest-Unlike-Neighbour (NUN) is used to identify the number of differences between the input and its neighbour, that when changed can lead to a change in the system's decision [4]. A key challenge that we address in this paper is to identify the minimum subset of feature value changes to achieve a change in the decision - the "actionable features". In this paper, we discover the minimum actionable feature changes using feature relevance-based explainer methods like LIME [10] or SHAP [6] which we introduce as the DisCERN algorithm.

The rest of the paper is organised as follows. Section 2 investigates the importance of counterfactual XAI and discusses key counterfactual methods and evaluation methodologies. Section 3 presents the DisCERN algorithm to improve the discovery of actionable features in counterfactuals. Section 4 presents the evaluation methodologies, datasets and performance metrics. Section 5 presents the evaluation results. Finally we draw conclusions and discuss future work in Section 6.

2 Related Work

Like many explanation methods, counterfactual explanations are rooted within the study of human psychology. Counterfactual thinking is a mental exercise where we attempt to isolate the specific actions which contributed to a (usually undesirable) outcome, with the goal of identifying how we could alter these facts to reach an alternative (and often more desirable) outcome [11]. In this manner we derive a form of causal explanation for the outcome that was actually achieved, allowing us to reason about how a different outcome could be achieved in future [3]. To clarify, consider the following fictitious example of a runner who was placed third in a race: "I won the bronze medal for my race (actual outcome), but I would have won the gold medal (desired outcome) if I hadn't tripped (causal action which changed outcome)". Through this thought process, the runner has derived what they believe to be a causal action that led to receiving the bronze medal. With that knowledge inferred, the runner can then reason that in order to achieve a better outcome, they should run a similar race again, but avoid tripping. Likewise with a counterfactual explanation we aim to explain why the system generated a specific outcome instead of another, by inferring causal relationships between input features [14].

Case-based Reasoning (CBR) [4] and optimisation techniques [9, 13] have been the pillars of discovering counterfactuals from data. Recent work in CBR has shown how counterfactual case generation can be conveniently supported through the case adaptation stage, where query-retrieval pairs of successful counterfactual explanation experiences are used to create an explanation casebase [4]. NICE [1] is another CBR and NUN based approach which performs an exhaustive search to discover the counterfactual by incrementally copying the features of the NUN to the query and checking for class change. Unlike the CBR approach to counterfactual generation, DiCE [9] trains a generative model using gradient descent optimisation to output multiple perturbed diverse counterfactuals. With the CBR approach additional counterfactuals can be identified by increasing the neighbourhood. This ability to provide multiple counterfactuals has been found to improve the end-user's mental model. In our work we also adopt CBR's NUN method to find counterfactuals but instead of the adaptation CBR step or exhaustive search in NICE we opt for feature relevance explainers to inform us on actionable feature discovery. In doing so we avoid the need to create similarity based explanation case bases [4] or perform exhaustive search [1] yet maintain the advantage of locality-based explanations which ensure valid counterfactuals that are often harder to guarantee with optimisation methods. Notably, our approach is contrasting to the feature relevance explainer proposed in [8] where counterfactuals generated from optimisation based methods are used to determine feature weights.

Quantitative evaluation of counterfactual explanations focus on measures that can ascertain properties of good counterfactuals. Here explanatory competency is defined as the competency of the explainer CBR system to solve any future query. It is measured by the fraction of queries that are currently explained by the explainer CBR system - coverage of the casebase. In contrast, authors of the DiCE algorithm proposed two performance measures to evaluate a counterfactual as proximity and sparsity [9]. Sparsity and proximity refers to the number of feature changes and amount of change needed to achieve class change. They also proposed two additional measures to evaluate multiple counterfactuals as validity and diversity. Validity measures if the counterfactuals presented by the method actually belong to the desirable class (i.e. not the same class as the query). Diversity measures the heterogeneity between multiple counterfactuals. Plausibility is another quantitative measure which checks the validity of a generated counterfactual with respect to the boundaries of real features [1]. We find validity, diversity and plausibility are inapt for our work because 1) by selecting a NUN we ensure 100% validity; 2) we are only selecting a single counterfactual which invalidates a measure for diversity; and 3) a DisCERN counterfactual is derived from a real data instance (NUN) to ensure 100% plausibility. However,

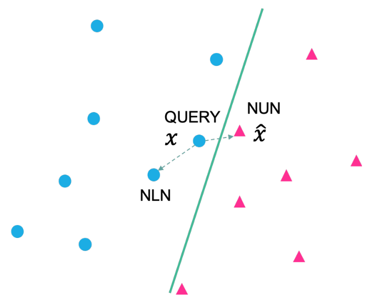


Fig. 1: Nearest Like and Unlike neighbours in a 2D features space ($m = 2$)

we find proximity and sparsity can be adopted to compare different counterfactual methods and measure the efficiency of actionable feature discovery.

3 Methods

The DisCERN algorithm uses feature relevance to identify the minimum subset of changes needed to form a counterfactual explanation from a retrieved NUN. Here we formalise the NUN counterfactual approach and thereafter discuss how weights from Feature Relevance Explainers can be used in DisCERN to discover actionable features.

3.1 Nearest-Unlike-Neighbour Counterfactual

The goal of a counterfactual explanation is to guide the end-user to achieve class change (i.e. actionable), with a focus on minimising the number of changes needed to flip the decision to a more desirable outcome for the end-user. Given a query instance, a data instance qualifies as a NUN only if the instance is the nearest neighbour of the query and belongs to a different class from the query.

Formally, given a query instance, $x = \{f_1, f_2, ..., f_m\}$, its counterfactual, $\hat{x} = \{\hat{f}_1, \hat{f}_2, ..., \hat{f}_m\}$, is identified as the NUN in the feature space [4] (Figure 1). Here f_i is the feature value at index i and m is the number of features used to represent instances.

Usually the distance between instances is calculated using Euclidean distance. Other distance measures include Manhattan distance and inverse Cosine similarity. Discovering the minimal number of actionable features (from a maximum of m potential feature changes) is one of the main challenges for counterfactual explainers. With DisCERN, we address this challenge by exploiting Feature Relevance Explainers.

3.2 Feature Relevance Explainers

LIME [10] is a model-agnostic feature relevance explainer which creates an interpretable model around a given data instance, p to estimate how each feature contributed to the black-box model outcome. LIME creates a set of perturbations within the neighbourhood of p and they are labelled using the black-box model. This new labelled dataset is used to create a linear interpretable model. The resulting surrogate model is interpretable and only locally faithful to the black-box model (i.e. correctly classifies the data instance p but not all data instances). The new interpretable model is used to predict the classification outcome of p that needs to be explained and obtain the weights that indicate how each feature contributed to the outcome.

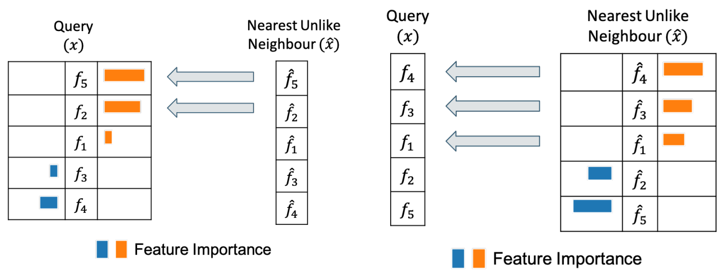
SHAP [6] is a model-agnostic feature relevance explainer which demonstrated improved computational performance and better consistency compared to LIME. SHAP is based on the shapley regression values introduced in game theory [12]. Shapley values are calculated by creating linear models using subsets of features present in p. More specifically, a model is trained with a subset of features of size m' and another model is trained with a subset of features of size $m' + \hat{m}$. Here $m' + \hat{m} <= m$ and the second model additionally includes a set of features $\hat{m}$ selected from the set of features that were left out in the first model. A set of such model pairs are created for all possible feature combinations. For a given data instance p, that needs to be explained, the prediction differences of these model pairs are averaged to find the explainable feature relevance weights.

3.3 Feature weights from Feature Relevance Explainers

Feature Relevance Explainers provide a feature relevance weights vector for any given data instance. In DisCERN, we use the magnitude of the relevance weights as a method to order features. Formally, the output of a feature relevance explainer is a vector of weights, $w = (w_1, w_2, \ldots, w_m)$, where w_i is a real-valued weight assigned to feature, f_i. A positive weight ($w_i >= 0$) indicates that the corresponding feature contributes positively and a negative weight ($w_i < 0$) corresponds negatively towards the predicted outcome. These relevance weights are used to define an ordering on features. Given a weights vector, w, the overall value, $w(.)$, is the weight lookup of a feature included in x. This is used to define the ordering $\preceq$ for features:

$$f_i \preceq_w f_j \text{ if and only if } w(f_i) \geq w(f_j) \tag{1}$$

According to [6], there are three desirable properties of a feature relevance explainer: local accuracy, missingness and consistency. Local accuracy guarantees that the surrogate model and black-box model predicts the same outcome for a given data instance. Missingness ensures that there there is no weight contribution from a missing feature (i.e. $w_i = 0$ if $f_i = 0$). Consistency refers to the quality whereby an explanation is reproducible (i.e the same data instance results in the same explanation). While LIME satisfies local accuracy and missingness, SHAP satisfied all three properties. Both methods can be used to provide the weights vector, w, for a query, or its NUN.


Query (x)
Nearest Unlike Neighbour ($\hat{x}$)
f_5 f_2 f_1 f_3 f_4
$\hat{f}_5$ $\hat{f}_2$ $\hat{f}_1$ $\hat{f}_3$ $\hat{f}_4$
Feature Importance
Query (x)
Nearest Unlike Neighbour ($\hat{x}$)
f_4 f_3 f_1 f_2 f_5
$\hat{f}_4$ $\hat{f}_3$ $\hat{f}_1$ $\hat{f}_2$ $\hat{f}_5$
Feature Importance

(a) Query Importance (b) NUN Importance

Fig. 2: Actionable Feature Discovery with QI and NI

3.4 Actionable Feature Discovery with Feature Relevance Explainers

The number of feature changes, n, required to achieve a class change, can range from 1 to m ($1 <= n <= m$). We propose two methods to discover actionable features, with the goal of minimising the number of feature changes (n) needed for a succinct yet actionable explanation. The first method is to replace the values of the most important features in the query with the corresponding feature values from its NUN; or a second alternative is to identify the most relevant features of the NUN and reuse those feature values in the modified query instance.

We illustrate these two methods in Figure 2. Here, we depict an example with 5 features where we replace features in the query until a class change is observed. In the left, the query features are ordered by their feature weights and the most important features are replaced by the respective NUN feature values (QI). In the right, the NUN features are ordered by their feature weights and the most important features are reused by the query (NI). QI and NI achieve class change with 2 and 3 feature replacements respectively.

3.5 DisCERN: Discovering Counterfactual Explanations with Relevance Features from NUNs

Algorithm 1 brings together methods from Sections 3.1- 3.4 to recognise actionable features to generate NUN counterfactuals. Here $y = \mathcal{F}(x)$ is the classifier prediction for the query and, $\hat{y} = \mathcal{F}(\hat{x})$ is the class prediction for the NUN. Relevance weights from Section 3.3 are identified in reference to either the query or the NUN, which we have denoted as p; and the *Order* method provides a list of feature indices ranked by the relevance weights. Feature values are iteratively replaced until the actionable change condition is met (i.e., $y \neq y'|y' = \mathcal{F}(x')$) or the query is completely changed to the NUN (which is guaranteed to result in class change). Thus DisCERN returns x' as the counterfactual. Clearly the fewer replacement iterations needed the better the actionable features being discovered.

Algorithm 1 DisCERN [$RelExp$, p]

```
Require: $(x, y)$: query and label pair
Require: $(\hat{x}, \hat{y})$: NUN and label pair
Require: $\mathcal{F}$: classification model
Require: $p$: either $x$ or $\hat{x}$                          ▷ as described in Section 3.4
Require: RelExp: Feature Relevance Explainer
 1: $W = RelExp(p)$                                            ▷ see Equation 1
 2: $\hat{W} = Order(W)$
 3: Initialise $y' = y$; $x' = x$                              ▷ init counterfactual as query
 4: for $w_i \in \hat{W}$ do
 5:     if $f'_i \neq \hat{f}_i$ then                          ▷ i is the index of $w_i$
 6:         $f'_i = \hat{f}_i$                                 ▷ copy the NUN feature value to the query
 7:         $y' = \mathcal{F}(x')$                             ▷ class prediction for perturbed query
 8:         if $y' \neq y$ then                                ▷ check for class change
 9:             Break
10:         end if
11:     end if
12: end for
13: return $x'$                                                ▷ returns the counterfactual
```

4 Evaluation

The goal of this evaluation is twofold. First a comparable study investigates different feature ordering strategies to find the most effective for actionable feature discovery. Second a comparative evaluation determines the effectiveness of the actionable feature discovery for counterfactual creation. We compare the counterfactual creation methods QI and NI with DiCE [9] and to a random feature ordering baseline.

4.1 Evaluation Methodology

We compare feature ordering heuristics using feature relevance knowledge from:

1. LIME and SHAP: feature weights from feature relevance explainers discussed in Section 3.2.
2. LIME$_C$ and SHAP$_C$: these are two class level feature relevance explainer versions of LIME and SHAP weights respectively, where for each class, the aggregated feature relevance is the mean feature relevance weights over all training data instances for that class.
3. Chi2: Chi-Squared feature selection method applied on the dataset and features ordered by p-value.

Experiments for actionable feature discovery compare the following methods:

1. DisCERN [RND,Null]: A randomly ordered set of feature indices are used in Algorithm DisCERN. Note that p is unspecified (i.e. null) with random feature ordering.

2. DiCE: Optimisation based counterfactual explainer method [9]
3. DisCERN [SHAP,QI]: Query Feature Importance (QI) using SHAP feature ordering
4. DisCERN [SHAP,NI]: NUN Feature Importance (NI) using SHAP feature ordering

4.2 Datasets

Experiments are carried out on 3 datasets as follows: feature ordering comparative study is conducted using the Moodle dataset and results presented in Section 5.1; and the actionable feature discovery comparative study is conducted using the Moodle dataset as well as the Loan-2015 and Alcohol datasets, with results presented in Section 5.2.

Moodle Dataset is constructed from records of student footprints on the Moodle Virtual Learning Environment (VLE) for a single class delivered within Robert Gordon University. VLE interactions help to capture vital touchpoints that can be used as proxy measures of student engagement. The dataset consists of 74 students who were enrolled for a Computer Science class during Semester 1 of 2020/2021 at RGU. The dataset contains 95 features, where each feature is a learning resource stored on the Moodle VLE and the feature value is the number of times it was accessed by a student.

The ML task is to predict if a student gets a higher or a lower grade based on their Moodle footprint. This task is based on the assumption that there is a causal relationship between the Moodle access and the final grade of a student. We consider grades *A* and *B* as *Higher* grades and *C*, *D*, *E* and *F* as *Lower* grades. Grades were consolidated as *Higher* and *Lower* to mitigate the comparably lower number of data instances and class imbalance. This formed a dataset of 74 instances for a binary classification task. A RandomForest classifier of 500 trees was used to predict the grade based on the Moodle footprint. The classifier achieved 83% accuracy over three stratified folds. Note that when explaining an outcome, we assume that the classifier has correctly predicted the grade.

The explanation intent explored is of type *Why* student A did *not* receive a grade X? instead of *Why* did student A receive grade Y? The latter can be explained using a feature relevance explainer presenting the contribution of the most important features for the predicted grade; and the former *Why not* type question explained through a counterfactual explanation to guide the student to achieve a more desirable outcome in the future.

Loan-2015 dataset is the subset of 2015 records from the Lending Club loan dataset on kaggle [1]. We limit the dataset to records from 2015 to create the loan-2015 dataset of 421,095 data instances with 151 features. In this paper we consider all features as actionable. However, this dataset include features such

[1] https://www.kaggle.com/wordsforthewise/lending-club

as income, home ownership and length of employment that are considered non-actionable in the real-world. The ML task is to predict if a loan will be fully paid or not and this outcome is used to accept or reject future loan requests. We apply data pre-processing steps recommended by the data providers to obtain a dataset with 342,865 instances and 115 features to perform binary classification. A RandomForest classifier with 500 trees achieved a 97% accuracy over three stratified folds. The desirable outcome for an end-user is accepting a loan request (i.e. similar users successfully re-paid their loans). For example an explanation request can take the form of *Why* person A did *not* receive the loan? and a counterfactual can guide the end-user to make necessary adjustments to receive a desirable outcome in future.

Alcohol Dataset is the Blood Alcohol Concentration (BAC) dataset which consists of 127,800 data instances with 5 features [2]. It includes features such as gender, if a meal was taken, the duration between the meal and BAC test. For the experiments in this paper we consider all features as actionable. The ML task for this dataset is to predict if the BAC is over a regulatory limit. Accordingly, in a pre-processing step the dataset is converted in to a binary classification task by recognising the two classes with the BAC regulatory limit as the decision threshold. A RandomForest classifier achieved 99% accuracy with the resulting dataset over three stratified folds. Similar to previous two dataset this use case also presents a desirable outcome for the end-user which is to maintain the BAC below the threshold. Accordingly, counterfactual explanations are sought by individuals who have a BAC above the threshold and are looking to understand how they might keep their BAC below the threshold by better managing one or more actionable features.

4.3 Performance Measures

In this paper we present two performance metrics to perform a quantitative evaluation of the DisCERN algorithm. We note that these measures correspond to sparsity and proximity in [9], but are not identical.

Mean number of feature changes ($\#F$) required to achieve class change is calculated as follows:

$$\#F = \frac{1}{N \times m} \sum_{j=1}^{N} \sum_{i=1}^{m} 1_{[\hat{f}_i \neq f_i]} \tag{2}$$

Here the number of features with different values between the counterfactual ($\hat{x}$) and the query (x) are calculated and averaged; where N refers to the number of query instances, and m is the number of features.

Table 1: Comparison of feature ordering strategies

	#F		$F	
DisCERN [,]	QI	NI	QI	NI
LIME	8.14	8.61	0.2642	0.2726
LIME$_C$	11.41	10.38	0.2308	0.2524
SHAP	**7.69**	**8.32**	0.2660	0.2454
SHAP$_C$	10.28	10.18	**0.2085**	**0.2068**
Chi2	12.27		0.2700	

Mean amount of feature changes ($\$F$) required to achieve class change is calculated as follows:

$$\$F = \frac{1}{N \times \#F} \sum_{j=1}^{N} \sum_{i=1}^{m} (|\hat{f}_i - f_i|) \tag{3}$$

Here the sum of feature differences are averaged over $\#F$ and the number of query instances (N). All continuous features are min/max normalised and therefore, continuous feature differences are between 0 and 1 whereas categorical feature differences are always 1 (using the overlap distance). Accordingly, datasets with more categorical features will have higher $\$F$ value, which means that the $\$F$ measure is not comparable across datasets.

5 Results

5.1 Comparison of Feature Ordering Strategies

A comparison of DisCERN settings with 5 alternative options for *RelExp*; and 2 alternatives for p appear in Table 1 using the Moodle dataset. Each alternative's performance is compared on $\#F$ and $\$F$. Note that there is no difference between QI and NI when using Chi2 because the feature ordering applies to the entire dataset. The comparison of feature ordering strategies show that SHAP achieves the best performance over LIME and Chi2 with both QI and NI methods (see bold font). SHAP using the QI feature ordering method has achieved lowest $\#F$, whilst SHAP$_C$ has lowest $\$F$. However, since SHAP$_C$ requires additional feature changes to achieve class change, we consider SHAP to be a preferable strategy. Moreover, we observe that LIME also achieves comparable performances for both $\#F$ and $\$F$. Notably, Chi2 failed to outperform both LIME and SHAP strategies in both minimising number of features and amount of change. Overall, these results emphasise the importance of feature relevance explainers as a proxy to identifying features important to achieve class change.

5.2 Evaluation of Actionable Feature Discovery

Table 2 provides a comparison of our DisCERN counterfactual algorithm with Random and DiCE. Note that in DisCERN, QI and NI are using SHAP as the

Table 2: Comparison of counterfactual methods on $\#F$

Dataset(total no of features)	DisCERN [RND,Null]	DiCE	DisCERN [SHAP,] QI	NI
Moodle(95)	21.62	10.21	**7.69**	8.32
Loan-2015(115)	21.911	**2.59**	6.86	5.51
Alcohol(5)	2.16	2.53	**2.11**	2.12

Table 3: Comparison of counterfactual methods on $\$F$

Dataset	DisCERN [RND,Null]	DiCE	DisCERN [SHAP,] QI	NI
Moodle	0.2924	0.6344	0.2660	**0.2454**
Loan-2015	**0.0569**	0.7763	0.0760	0.0711
Alcohol	**0.0909**	0.6707	0.0929	0.0925

feature ordering strategy. Results suggests that DisCERN with QI achieves the best performance on the Moodle and Alcohol datasets and DiCE achieves best performance with the Loan-2015 dataset (see bold font). DisCERN with QI and NI achieve comparable performances across all three datasets which resembles findings in Table 1. Interestingly, DiCE failed to outperform Random feature ordering on the Alcohol dataset which could be due to the limited amount of features available.

Results in Table 3 indicate that with DisCERN (with either QI or NI) we also achieve class changes with lowest $\$F$ values (see bold font). It is unusual that DisCERN with Random ordering resulted in lowest $\$F$ on the Loan-2015 dataset. $\$F$ performance of DisCERN with QI and NI is better compared to DiCE on all three datasets. For instance, for a query in the Loan-2015 dataset, the total amount of change with the DiCE method is $2.01(2.59 \times 0.7763)$ and with QI is $0.46(6.93 \times 0.0660)$. In situations where actionable features are not "easy to change", it is more feasible to use DisCERN over DiCE.

6 Conclusion

In this paper, we presented a novel approach to finding actionable knowledge when constructing an explanation using a counterfactual. We used feature relevance explainers as a strategy to discover features that are most significant to a predicted class and then used that knowledge to discover the actionable features to achieve class change with minimal change. We demonstrated our approach DisCERN using three datasets one of which (Moodle Dataset) is an original contribution.

Our empirical results showed that SHAP is the most optimal feature relevance explainer for ordering actionable features. Comparison of the QI and NI counterfactual methods introduced in this paper have either outperformed or achieved comparable performance over DiCE. The results have also highlighted the need to find balance between the number of feature changes and amount of feature change based on the selected actionable features. However, we find there is conclusive evidence that feature relevance explainers are an important proxy to discovering actionable features and minimising the changes required. Future work will expand upon our evaluation to include additional real-world datasets and the use of qualitative evaluation through crowd-sourcing techniques.

References

1. Brughmans, D., Martens, D.: Nice: An algorithm for nearest instance counterfactual explanations. arXiv preprint arXiv:2104.07411 (2021)
2. Cunningham, P., Doyle, D., Loughrey, J.: An evaluation of the usefulness of case-based explanation. In: International conference on case-based reasoning. pp. 122–130. Springer (2003)
3. Harris, P.L., German, T., Mills, P.: Children's use of counterfactual thinking in causal reasoning. Cognition **61**(3), 233–259 (1996)
4. Keane, M.T., Smyth, B.: Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable ai (xai). In: International Conference on Case-Based Reasoning. pp. 163–178. Springer (2020)
5. Kenny, E.M., Keane, M.T.: Twin-systems to explain artificial neural networks using case-based reasoning: Comparative tests of feature-weighting methods in ann-cbr twins for xai. In: Proceedings of IJCAI-19. pp. 2708–2715 (2019)
6. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems **30**, 4765–4774 (2017)
7. Mohseni, S., Zarei, N., Ragan, E.D.: A survey of evaluation methods and measures for interpretable machine learning. arXiv preprint arXiv:1811.11839 (2018)
8. Mothilal, R.K., Mahajan, D., Tan, C., Sharma, A.: Towards unifying feature attribution and counterfactual explanations: Different means to the same end. arXiv preprint arXiv:2011.04917 (2020)
9. Mothilal, R.K., Sharma, A., Tan, C.: Explaining machine learning classifiers through diverse counterfactual explanations. In: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency. pp. 607–617 (2020)
10. Ribeiro, M.T., Singh, S., Guestrin, C.: " why should i trust you?" explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 1135–1144 (2016)
11. Roese, N.J.: Counterfactual thinking. Psychological bulletin **121**(1), 133 (1997)
12. Shapley, L.S.: A value for n-person games. In: Contributions to the Theory of Games. pp. 307–317 (1953)
13. Timmis, J., Edmonds, C.: A comment on opt-ainet: An immune network algorithm for optimisation. In: Genetic and Evolutionary Computation Conference. pp. 308–317. Springer (2004)
14. Wachter, S., Mittelstadt, B., Russell, C.: Counterfactual explanations without opening the black box: Automated decisions and the gdpr. Harv. JL & Tech. **31**, 841 (2017)
15. Wettschereck, D., Aha, D.W., Mohri, T.: A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. Artificial Intelligence Review **11**(1), 273–314 (1997)
16. Wiratunga, N., Koychev, I., Massie, S.: Feature selection and generalisation for retrieval of textual cases. In: European Conference on Case-Based Reasoning. pp. 806–820. Springer (2004)