PertCF: A Perturbation-Based Counterfactual Generation Approach

Betül Bayrak(✉) and Kerstin Bach

Norwegian University of Science and Technology, Høgskoleringen 1, 7034 Trondheim, Norway
{betul.bayrak,kerstin.bach}@ntnu.no

Abstract. Post-hoc explanation systems offer valuable insights to increase understanding of the predictions made by black-box models. Counterfactual explanations, an instance-based post-hoc explanation method, aim to demonstrate how a model's prediction can be changed with minimal effort by presenting a hypothetical example. In addition to counterfactual explanation methods, feature attribution techniques such as SHAP (SHapley Additive exPlanations) have also been shown to be effective in providing insights into black-box models. In this paper, we propose PertCF, a perturbation-based counterfactual generation method that benefits from the feature attributions. Our approach combines the strengths of perturbation-based counterfactual generation and feature attribution to generate high-quality, stable, and interpretable counterfactuals. We evaluate PertCF on two open datasets and show that it has promising results over state-of-the-art methods regarding various evaluation metrics like stability, proximity, and dissimilarity.

Keywords: Explainable artificial intelligence (XAI) · Counterfactual generation · Counterfactual explanations · Post-hoc explanation

1 Introduction

With the increasing use of artificial intelligence and machine learning models in our daily lives, understanding how these models make decisions has become increasingly important. Also, the increasing complexity of machine learning models has created understanding of how they make their predictions challenging. For example, two individuals with similar backgrounds submitted applications for a loan to purchase a home to the bank, which uses a black-box model to decide loan application assessments. But one applicant, Leo, was declined while the other, Maya, was approved as in Fig. 1. Leo wants to learn the reasons for the rejection of his loan application and what he needs to do to make an acceptable application. Counterfactual Explanations, a popular research field recently, can generate highly satisfactory explanations in such situations. [3,12]

Counterfactual explanations, a type of post-hoc explanation, provide valuable insights that help users understand the predictions made by black-box models, especially when the factors influencing the model's decision are not immediately

M. Bramer and F. Stahl (Eds.): SGAI 2023, LNAI 14381, pp. 174–187, 2023.
https://doi.org/10.1007/978-3-031-47994-6_13

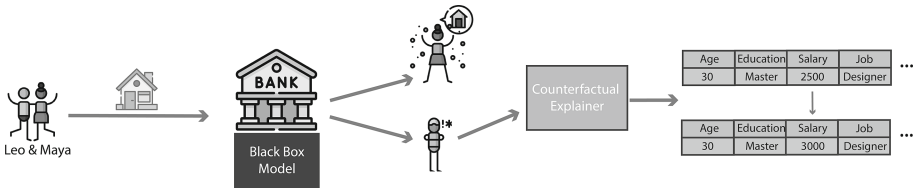

Fig. 1. Illustration of how a counterfactual explainer can provide insight into a loan application decision.

clear [4]. They also aim to demonstrate how a model's prediction can be changed with minimal effort by presenting hypothetical examples that answer the "what if?" question. For instance, "What if Leo earns $500 more, would his application be accepted?" is an example of a counterfactual explanation. Section 2.2 provides a more technical definition of counterfactual explanations.

Feature attribution based explanations are another method for post-hoc explanations. Those methods identify the contribution of each feature to a model's prediction and help to explain how changes in the input data can affect the output. One popular feature attribution technique is SHAP (SHapley Additive exPlanations), proposed by Ludberg and Lee [9] and uses game theory to assign values to each feature based on its contribution to the model's output. Another technique is LIME (Local Interpretable Model-Agnostic Explanations) [11], proposed by Ribeiro et al., and generates explanations by approximating the black-box model with a local linear model.

In this research paper, we introduce PertCF, an innovative approach to generating counterfactual explanations using perturbations, leveraging the feature attributions. Our method combines the advantages of perturbation-based counterfactual generation and feature attributions to produce counterfactuals that are of high quality, reliable, and easy to interpret.

This work presents several contributions, which are summarized as follows:

- PertCF combines the strengths of counterfactual explanation and feature attribution explanation methods.
- PertCF employs custom distance metrics tailored to the specific problem, offering two key benefits: (I) It utilizes SHAP values calculated individually for each class, enabling distinct class-based feature attribution. (II) It facilitates the incorporation of domain expertise and semantic data representation.
- Provided reproducible benchmarking experiments using open datasets and open-source implementation of the PertCF method and compared its performance with state-of-the-art methods (https://github.com/b-bayrak/PertCF-Explainer).

The structure of this paper is as follows. Section 2 provides an overview of the theoretical foundations of SHAP and counterfactual generation for counterfactual explanation with the state-of-the-art method. Section 3 presents the details of our proposed PertCF method. Section 4 reports the details and results of our

experiments, including an analysis of PertCF's performance on open datasets and its comparison with existing state-of-the-art methods. Furthermore, in Sect. 4.5, we discuss the implications and limitations of PertCF. Finally, Sect. 5 concludes the paper and suggests directions for future research.

2 Background and Related Work

This work focuses on counterfactual-based explanation systems and in this section, we provide fundamental information about SHAP, which is used as a feature attribution method and counterfactual explanation systems and how popular methods work.

2.1 SHAP

SHAP (SHapley Additive exPlanations) is introduced by Ludberg and Lee [9] and it is a popular feature attribution explanation method. SHAP is based on Shapley values which is a game theory concept and aims to assign a value to each feature in a prediction based on how much it contributed to the prediction compared to all other possible combinations of features.

Basically, Shapley values simulate the absence of a feature using the marginal expectation over a background distribution and SHAP values apply this concept to machine learning models by determining the contribution of each feature in the prediction of the model, while accounting for interactions between features.

2.2 Counterfactual Generation Methods

In philosophy, a counterfactual is a conditional statement that expresses what would have happened if circumstances were different from what actually occurred. In machine learning, counterfactuals are used to explain the decision-making process of a model by providing alternative hypothetical scenarios for a given prediction [13,14]. These systems are named counterfactual explainers and a simple example of counterfactual explainer usage can be seen in Fig. 1.

S is the set of samples and $x \in S$ and given an observed sample $x =< x_1, x_2, ..., x_m >$ with corresponding class label A, m is the number of features. x' represents a counterfactual of x (i.e. Fig. 2b), $x' =< x'_1, x'_2, ..., x'_m >$. x' belongs to class label B, where $B \neq A$.

NLN (Nearest Like Neighbour) refers to the nearest neighbor of a given sample that belongs to the same class. On the other hand, NUN (Nearest Unlike Neighbour) refers to the nearest neighbor of a given sample that belongs to a different class (See Fig. 2a). In other words NUN is the closest dissimilar observed point to sample x. Basically, NUN is one of the counterfactuals of sample x. However, the important thing is generating a good/feasible counterfactual to give hypothetical examples and generate higher quality explanations.

To generate high-quality explanations, several requirements need to be fulfilled. While some of these requirements highly dependent on the problem and

domain, others are common across various applications. Ideally, a counterfactual should be realistic, relevant, insightful, and trustworthy. In other words, it should allow for the interpretation of the explanation and the implementation of the changes required to achieve it in the real world. Another crucial requirement is diversity. Generating diverse counterfactuals does not mean generating more than one counterfactual but generating the counterfactuals by considering different characteristics of the data, covering a wide range of possibilities, providing a comprehensive understanding of the decision-making process of the model, and enhancing the explanatory power. While some requirements may not be objectively measurable without user reviews, others can be assessed using qualitative metrics to determine whether they have been met by the generated counterfactuals (See Sect. 4.3).

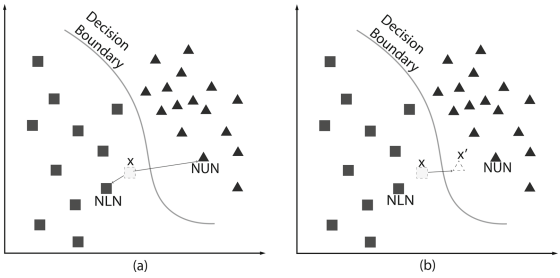

Fig. 2. (a) NLN and NUN of x, (b) NLN and NUN of x together with x': one of the possible counterfactuals of x.

In the past few years, several methods for generating counterfactuals have been proposed. We describe two of these methods and their details, which we use for comparison with our proposed approach.

The DiCE (Diverse Counterfactual Explanations) method [10], emphasizes the significance of diversity in producing actionable counterfactual explanations. It presents a comprehensive optimization framework that highlights the need to balance trade-offs, considers causal implications, and addresses optimization challenges when generating counterfactual explanations. In the publication, the authors provide a quantitative evaluation framework for counterfactuals that evaluates validity, proximity, and diversity. The iterative nature of DiCE allows for generating more than one counterfactual for an instance. Instead of using customized distance measures, it uses the mean of feature-wise l_1 distances between the counterfactual and the sample for continuous features. For categorical features, it uses a simple metric that assigns one if the counterfactual's value for any categorical feature differs from the sample input; otherwise, it assigns zero.

CF-SHAP[1], which is a feature attribution based counterfactual generation method, calculates Shapley values of each feature, and for each individual prediction, the method generates a set of counterfactual examples that show how

changing the input features would affect the predicted outcome. The counterfactual examples are generated by iterative adjusting the input features using a greedy optimization algorithm until a desired outcome is achieved. In the publication, the authors provide a quantitative evaluation framework for counterfactuals that evaluates plausibility and counterfactual-ability. Instead of using customized distance measures, it uses the Manhattan distance over the quantile space.

Both, the DiCE and CF-SHAP approaches may not fully capture the relevance between the categories as they do not use customized distance measures. Also, they only work with binary classification models which decreases the compatibility of the methods.

3 PertCF Method

PertCF is a perturbation-based counterfactual generation method that proposes a recursive approach to generate the best-performing counterfactual. The approach is based on generating a new sample, which is a counterfactual candidate, by perturbing the source sample with respect to the target sample. Section 3.2 provides technical details of the counterfactual generation procedure, while Sect. 3.1 provides insights into how PertCF uses feature attribution, and Sect. 3.3 gives insights into how domain knowledge is incorporated into PertCF.

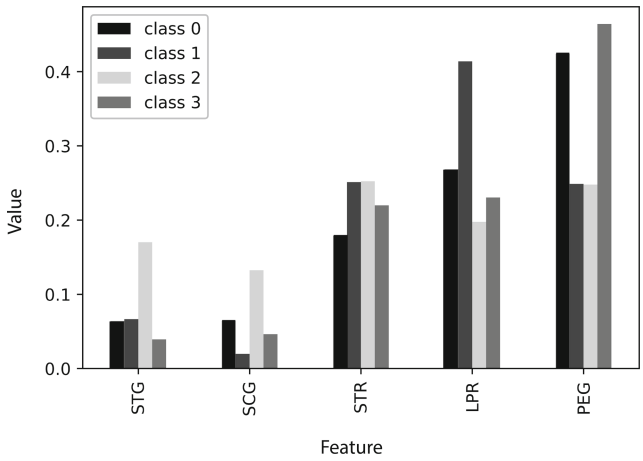

Fig. 3. Average SHAP values calculated for User Knowledge Modeling Dataset.

3.1 Feature Attribution

The PertCF method benefits from feature attribution for similarity and distance functions. For each class, the average SHAP values are calculated, and they are used for setting similarity and distance functions. In this way, we project the

different attribution levels of the features as characteristics of the classes. For example in Fig. 3, there are 4 classes and 5 features and for $class0$ the most important feature is 'PEG' but for $class1$ it is 'LPR'.

In the counterfactual generation process, the counterfactual candidates are generated by perturbation (details in Sect. 3.2), and the amount of perturbation is calculated using *shap_target* which is the average SHAP values of the target class for the counterfactual to be generated. Also, the average SHAP values are used as weights of the similarity functions for each class. In this way, when the distance between two instances is measured, a feature with higher attribution will affect the result more than others. In PertCF, similarity measures are used in 3 different aims: (I) detecting the NUNs and (II) measuring the distance between the last two generated candidates (Sect. 3.2), and (III) evaluation metrics to measure the quality of the generated counterfactuals (Sect. 4.3).

3.2 Counterfactual Generation Procedure

To generate counterfactuals for given instances, the input is x and its corresponding class label. The output will be the generated counterfactual x' of instance x.

Initially, we detect the NUN of x which is the nearest observed counterfactual (Fig. 4a), and assign *target_label* as the class of NUN. To generate x', we generate counterfactual candidates c_i by perturbing s (*source*) with respect to t (*target*). For the first iteration, x is assigned as s and *nun* is assigned as t, and the first counterfactual candidate (c_1) is generated by perturbing x with respect to NUN (Fig. 4b). This process is repeated by selecting new s and t to generate better candidates until the termination criteria are met. There are two termination criteria, (1) the number of iterations *num_iter* for generating c_i and (2) the distance between the last two generated candidates. In each iteration, before setting s and t, we check if the current situation satisfies the termination criteria. If one of the criteria is met, the last generated candidate is selected and assigned as x', and then the process ends.

If the iteration limit is not reached and c_i does not belong to *target_label*, we need to approach t in the next iteration and we perturb c_i with respect to s to generate c_{i+1}. For example, in Fig. 4b, after generating c_1 that does not satisfy termination criteria and does not belong to *target_label*, we perturb c_1 and c_2 is generated as shown in Fig. 4c.

If the iteration limit is not reached and c_i belongs to *target_label*, we add c_i to the candidate list. Then, we check the distance between c_i and c_{i-1}. If the distance is smaller than the threshold μ, which is calculated based on the distance between s and t with a provided coefficient, it means that the generated candidates are getting closer to each other, and we need to stop at an optimal point to improve efficiency. Thus, the process ends, and c_i is selected as x'. Otherwise, in the next iteration, we perturb c_i with respect to t to generate c_{i+1}. For example, in Fig. 4c, after generating c_2, which belongs to *target_label* but does not satisfy the termination criteria, we perturb c_2 with respect to c_1, and c_3 is generated as shown in Fig. 4d.

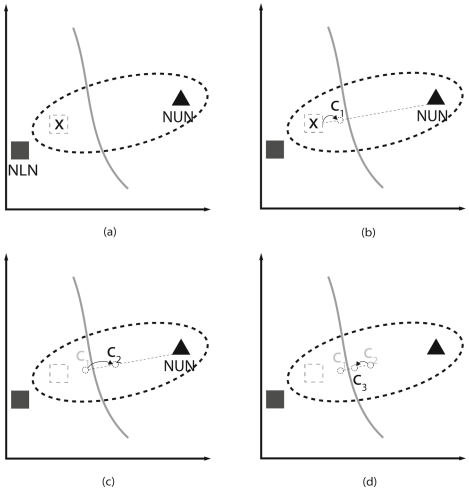

Fig. 4. Steps for generating counterfactual examples using PertCF. (a) Starting scenario that shows instance x and its NUN. (b) Generation of the first candidate c_1 by perturbing x with respect to NUN. (c) Generation of the second candidate c_2 by perturbing c_1 with respect to NUN. (d) Generation of the next candidate c_3 by perturbing c_2 with respect to c_1.

If the iteration limit is reached, the last generated candidate is selected and assigned as x', and the process ends. However, if the candidate list is empty, it means no candidates from the expected class could be generated during the iterations, and the process starts over with the second closest NUN of x.

The perturbation process is designed differently for numeric and categoric features. For numeric features, a calculated perturbation value is added to the feature value. The same procedure applies to ordinal features. For nominal features, the feature value changes if the similarity value (w) of the categories is lower than threshold α.

To perturb s with respect to t and generate the perturbed instance p, for each feature f if f is numeric,

$$p_f = s_f + shap\_target_f * (t_f - s_f) \tag{1}$$

however, if f is nominal,

$$p_f = \begin{cases} t_f & \text{if } w_f < \alpha \\ s_f & \text{otherwise} \end{cases} \tag{2}$$

3.3 Incorporation of Domain Knowledge

In explanation systems, incorporating domain knowledge is crucial for improving the accuracy and interpretability of machine learning models. Domain knowledge

helps to generate meaningful explanations and mitigate the risk of unintended consequences or bias. The PertCF method facilitates the incorporation of expert knowledge by modeling the distance and similarity measures. For instance, it allows projecting the relationship among values of nominal features. By utilizing these measures, the challenges can be addressed in the field and ensure compliance with GDPR regulations. This approach streamlines the process of incorporating domain knowledge, enabling to the generation of more accurate and interpretable results.

4 Experiments

To understand how PertCF performs on different datasets and setups. We compare it with state-of-the-art methods and conduct a series of experiments. This section provides details on the experimental setup, results, and discussion.

4.1 Experimental Setup/Design

Instance-based post-hoc explanation systems use the instance and its corresponding predicted class label. In our experiments, we required well-performed decision-making models to obtain accurate predictions. Therefore, we utilized the Gradient Boosting Classifier for the User Knowledge Modeling and South German Credit datasets, achieving accuracy scores of approximately 0.98 and 0.81, respectively.

To model distance and similarity measures and to retrieve most similar samples using the customized similarity measures, we use myCBR [2], an open-source tool providing a framework for similarity-based retrieval.

4.2 Datasets

In the experiments, the User Knowledge Modeling Dataset [7] and the South German Credit Dataset [6] are used (See Table 1). The User Knowledge Modeling dataset pertains to students' knowledge levels regarding Electrical DC Machines. It comprises five numeric features and one categorical (label) feature, constituting a multi-class classification task. The South German Credit Dataset encompasses 21 columns detailing attributes of credit applicants and their creditworthiness categorized as either good or bad. It encompasses three numeric and 18 categorical features, suitable for binary-class classification.

Our motivation is testing PertCF on open datasets to ensure that our work is reproducible and does not depend on a certain kind of data collection. Further, by providing our source code we aim at increasing the transparency of the experiments, providing a benchmark for evaluating the performance of future methods, and covering different domains and problems, addressing a diverse set of challenges.

Table 1. Characteristics of the Datasets

	Size	Feature	Numeric	Categoric	Class
Credit[a]	1000	21	3	18	2
Knowledge[b]	403	6	5	1	4

[a]South German Credit Dataset
[b]User Knowledge Modeling Dataset

4.3 Evaluation Metrics

There are various methods in the literature to measure the quality of generated counterfactuals 2. In our experiments for PertCF we selected applicable metrics, namely dissimilarity, sparsity, instability, and run-timeto compare its results with state-of-the-art methods[1,10]. Additionally, we discuss important considerations in Sect. 4.5.

– **Dissimilarity:** Measures how dissimilar x and x' and it is calculated as mean of the distances between x and each elements of C. The lower, the better.

$$dissimilarity = \frac{1}{\|C\|} \sum_{x' \in C} dist(x, x') \tag{3}$$

– **Sparsity:** Measures how many features of x should be changed to achieve x'.

$$f(x_i, x'_i) = \begin{cases} 1 & \text{if } x_i = x'_i \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$sparsity = \frac{1}{\|C\|} \sum_{x' \in C} \frac{1}{m} \sum_{i=1}^{m} f(x_i, x'_i) \tag{5}$$

– **Instability:** Measures the stability of generated counterfactuals. If x and y are very similar samples, a stable counterfactual generation system should generate very similar counterfactuals x' and y'. To measure the stability, x is perturbed to generate a very close sample (y) to x and measure the distance between x' and y'. The lower, the better.

$$instability = dist(x', y') \tag{6}$$

– **Runtime:** Refers to the time taken to generate a counterfactual for a given input instance. The lower, the better.

4.4 Experimental Results

As stated in Sect. 3, PertCF employs two termination criteria, both of which rely on pre-defined variables. Therefore, in the experiments, we demonstrate the performance of the PertCF method using different parameters and datasets. Additionally, we compared PertCF with state-of-the-art methods using the selected parameters.

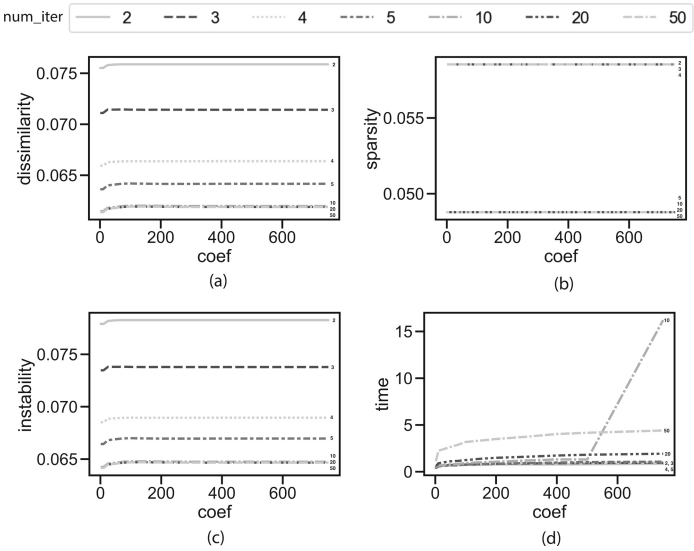

Fig. 5. The results of the parameter experiments for the User Knowledge Modeling dataset.

Performance of the PertCF. The first termination criterion is the maximum number of iterations to generate a counterfactual, and it is represented as num_iter. The other criterion is the distance between the last two generated candidates d, and it relies on the $coef$ variable, $d = dist(x, NUN)/coef$. Therefore, num_iter and $coef$ parameters affect the performance of PertCF and the outperforming parameters differ according to the dataset characteristics. Thereby, we ran a series of experiments with various different values num_iter and $coef$.

Figure 5 illustrates the results of the User Knowledge Modeling dataset. The performance of higher num_iter is better in terms of dissimilarity, instability, and sparsity, and in general, when $coef$ is higher than 10, the results are almost stabilized. Therefore, choosing $coef$ between 1 and 10 and num_iter as 5 or 10 might be optimal when considering runtime complexity. Figure 6 illustrates the results of the South German Credit dataset. When num_iter is set to 3 or 5, there is a boost in performance in terms of dissimilarity, instability, and sparsity. However, the effect of $coef$ is not clearly observable in the experiments. Therefore, considering the runtime complexity, which is directly proportional to num_iter and $coef$, it might be optimal to choose $coef$ between 5 and 10 and num_iter as 3 or 5.

The choice of parameters mainly depends on the characteristics of the datasets, but we can observe that there is a trade-off between computation time and the value of $coef$.

Comparison with the State-of-the-Art Methods. To evaluate the performance of our proposed method, we compared it with several state-of-the-

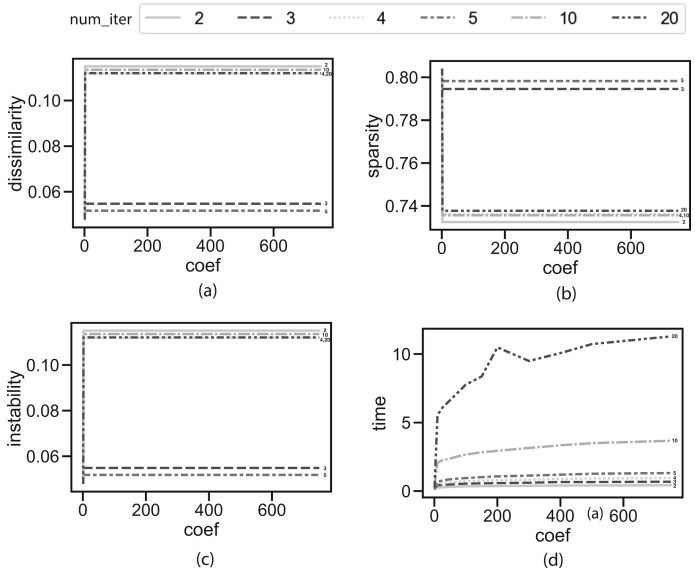

Fig. 6. The results of the parameter experiments for the South German Credit dataset.

art methods on the same datasets (See Table 2). The results indicate that our method outperforms the others in terms of dissimilarity and instability, which means that the PertCF method generates more stable and consistent counterfactuals than the compared methods. However, when considering the sparsity measure, the DICE method outperforms our method. It is important to note that in the South German Credit dataset, the majority of the features are nominal, which makes the comparison between DICE and PertCF results closer than in an all-numeric dataset.

Our findings suggest that PertCF can effectively address the challenges of the counterfactual generation process. However, there are still several limitations and open questions that need to be addressed to improve its performance further. Some of these limitations and open points are discussed in detail in Sect. 4.5.

4.5 Discussion

The experiments conducted in this work aimed to evaluate the effectiveness of the proposed PertCF method compared to state-of-the-art methods for generating counterfactual explanations using two open datasets to demonstrate the performance of PertCF under different conditions. The results showed that PertCF outperformed other state-of-the-art methods in terms of dissimilarity and instability. However, if we consider the higher sparsity is better, our method does not outperform others. In the literature, sparsity is implemented as L_0 norm between x and x' [5] or as a pre-defined threshold [8]. We implemented it as in

Table 2. Performance of Counterfactual Generation Methods

	Dissim.	Sparsity	Instability	Time
South German Credit				
DICE	0.0557	**0.9111**	0.0560	0.0736
CFshap	0.2555	0.5842	0.2555	**0.0058**
$PertCF^c$	**0.0517**	0.7983	**0.0518**	0.4069
User Knowledge Modeling				
DICE	0.1727	**0.6423**	0.1769	0.1180
CFshap	0.1792	0.0293	0.1806	**0.0011**
$PertCF^d$	**0.0636**	0.0585	**0.0664**	0.2827

[c] $num\_iter = 5$ and $coef = 5$
[d] $num\_iter = 5$ and $coef = 3$

the first definition, which basically measures how many features were changed to go from the original data point (x) to the counterfactual (x'). However, sparsity strongly depends on the topic or field being studied. For example, in a medical diagnosis explanation that mainly consists of numeric features, the possibility of having dependencies between features is high, and the target to change the diagnosis might include a set of changes in the features, and this explanation has low sparsity. In contrast, in the bank loan example, if the application owner has many changes that could result in changing the application result in the explanation, having a higher sparsity makes the explanation more reasonable and applicable. Therefore, the discriminative power of the sparsity metric is related to the concept being studied.

While our proposed method may not outperform others in terms of run-time, there are several underlying reasons like the used tools, implementation, and the number of generated candidates to reach the final counterfactual. The PertCF method generates multiple candidate counterfactuals before selecting the final one, which requires additional computational resources. This process ensures that we consider various possibilities and choose the most effective counterfactual. Nevertheless, the increased computational cost of generating multiple candidate counterfactuals may affect the run-time of our method compared to other methods. However, we believe that the advantages of generating these candidates outweigh the potential cost in terms of run-time, especially for complex datasets or critical applications where dissimilarity and stability are crucial for effective counterfactual generation. With further research and improvements in the algorithms and tools used, it may be possible to optimize the run-time of our method while maintaining its advantages.

Another issue that should be taken into consideration is whether the number of generated counterfactuals can be used as a metric to evaluate the performance of the method. Some methods generate multiple counterfactuals, and depending on the application field, it might be useful to provide multiple counterfactuals. However, the important thing is not the number of counterfactuals but their

diversity of them. However, the PertCF method provides only one counterfactual for an instance, and we believe that it can be applied as a multiple counterfactual generator. With the current setup, this can be done in two ways. First, several of the generated candidates with high diversity can be selected and provided as counterfactuals. Second, in multi-class classification, a counterfactual can be generated to provide insights into how to switch the model's prediction from the current situation to all other classes.

One of the key limitations of the proposed method is that it is only applicable to tabular data. Counterfactual generation methods are highly compatible with tabular data because they rely on manipulating individual feature values to generate counterfactual instances that are close to the original instance but result in a different predicted outcome. However, recent research has shown that counterfactual explanation methods can also be applied to other types of data, such as text and image data. Another limitation of counterfactual generation methods is that it is typically used with classification models, as the method requires discrete prediction output. However, researchers are actively exploring ways to extend counterfactual explanation methods to other types of models, such as regression models, to make them more widely applicable.

5 Conclusion and Future Work

The experimental results demonstrate that the proposed PertCF method is a promising approach as a perturbation-based counterfactual generator for counterfactual explanations. Using SHAP values that are calculated for each class helps to project the different levels of feature attributions for every class. By combining the strengths of perturbation-based counterfactual generation and feature attributions, PertCF outperforms existing state-of-the-art methods on evaluation metrics such as proximity and dissimilarity. Moreover, it offers valuable insights into black-box models and helps improve their interpretability. We believe that PertCF can be applied in various domains, including healthcare, finance, and e-commerce, where interpretability and transparency are essential. Future research can explore the application of PertCF in these domains and further evaluate its effectiveness.

Acknowledgements. This work has been supported by the Research Council of Norway through the EXAIGON project (ID 304843).

References

1. Albini, E., Long, J., Dervovic, D., Magazzeni, D.: Counterfactual shapley additive explanations. In: 2022 ACM Conference on Fairness, Accountability, and Transparency, pp. 1054–1070 (2022)
2. Bach, K., Althoff, K.-D.: Developing case-based reasoning applications using myCBR 3. In: Agudo, B.D., Watson, I. (eds.) ICCBR 2012. LNCS (LNAI), vol. 7466, pp. 17–31. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32986-9_4

3. Celar, L., Byrne, R.M.: How people reason with counterfactual and causal explanations for artificial intelligence decisions in familiar and unfamiliar domains. Mem. Cogn., 1–16 (2023)
4. Dai, X., Keane, M.T., Shalloo, L., Ruelle, E., Byrne, R.M.: Counterfactual explanations for prediction and diagnosis in XAI. In: Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society, pp. 215–226 (2022)
5. Dandl, S., Molnar, C., Binder, M., Bischl, B.: Multi-objective counterfactual explanations. In: Bäck, T., et al. (eds.) PPSN 2020. LNCS, vol. 12269, pp. 448–469. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58112-1_31
6. Groemping, U.: South German credit data: correcting a widely used data set. Rep. Math. Phys. Chem. Berlin, Germany, Tech. Rep. **4**, 2019 (2019)
7. Kahraman, H., Colak, I., Sagiroglu, S.: Developing intuitive knowledge classifier and modeling of users' domain dependent data in web, knowledge based systems (2013)
8. Keane, M.T., Smyth, B.: Good counterfactuals and where to find them: a case-based technique for generating counterfactuals for explainable AI (XAI). In: Watson, I., Weber, R. (eds.) ICCBR 2020. LNCS (LNAI), vol. 12311, pp. 163–178. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58342-2_11
9. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
10. 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)
11. 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)
12. Shang, R., Feng, K.J.K., Shah, C.: Why am I not seeing it? understanding users' needs for counterfactual explanations in everyday recommendations. In: Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, FAccT 2022, pp. 1330–1340. Association for Computing Machinery, New York (2022). https://doi.org/10.1145/3531146.3533189
13. Wachter, S., Mittelstadt, B., Russell, C.: Counterfactual explanations without opening the black box: Automated decisions and the GDPR. Harv. JL Tech. **31**, 841 (2017)
14. Yacoby, Y., Green, B., Griffin Jr, C.L., Doshi-Velez, F.: "If it didn't happen, why would I change my decision?": how judges respond to counterfactual explanations for the public safety assessment. In: Proceedings of the AAAI Conference on Human Computation and Crowdsourcing, vol. 10, pp. 219–230 (2022)