IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 7 July 2024, accepted 1 August 2024, date of publication 9 August 2024, date of current version 20 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3441037

RESEARCH ARTICLE

Counterfactual Explanations With Multiple Properties in Credit Scoring

XOLANI DASTILE[1] **AND TURGAY CELIK**[1,2]
[1]School of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg 2000, South Africa
[2]Faculty of Engineering and Science, University of Agder, 4630 Kristiansand, Norway

Corresponding author: Turgay Celik (celikturgay@gmail.com)

ABSTRACT EXplainable Artificial Intelligence (XAI) aims to reveal the reasons behind predictions from non-transparent classifiers. Explanations of automated decisions are important in critical domains such as finance, legal, and health. As a result, researchers and practitioners in recent years have actively worked on developing techniques that explain decisions from machine learning algorithms. For instance, an explanation technique called counterfactual explanation has recently been gaining traction in XAI. The interest in counterfactual explanations stems from the ability of the explanations to reveal what could have been different to achieve a desired outcome, as opposed to only highlighting important features. For instance, if a customer's loan application is denied by the bank, a counterfactual will indicate the changes required for the customer to qualify for the loan in the future. For a counterfactual to be considered effective, several counterfactual properties must hold. This paper proposes a novel optimization formulation designed to generate counterfactual explanations that possess multiple properties concurrently. The efficacy of the proposed method is assessed on a publicly available credit dataset. The results showed a trade-off between validity and sparsity, which are both parts of a suite of counterfactual properties. Furthermore, the results showed that our proposed approach compromises validity to some degree but strikes a good balance between validity and sparsity.

INDEX TERMS Counterfactual explanations, credit scoring, optimization, eXplainable Artificial Intelligence (XAI).

I. INTRODUCTION

Automated decisions that are generated by some of the machine learning techniques may have unfavorable consequences. For example, in a banking setting, if a loan application decision relies on a machine learning technique, the loan application may be denied due to unintended bias that may reside in data. As a consequence, the bank may be subjected to a regulatory fine. The consequence for the applicant may be that he/she is unable to send his/her child to university if the primary purpose of the loan was to pay university tuition.

An emerging field known as eXplainable Artificial Intelligence (XAI) seeks to remediate the lack of explanation of machine learning techniques. Several explanation techniques can be found in the literature, and these include global and local explanation techniques [1]. A global explanation seeks to explain the model as a whole. A local explanation seeks to explain only a single prediction at a time. In addition, the explanations are either referred to as ante-hoc or post-hoc [1]. An ante-hoc explanation is intrinsically explainable. A post-hoc explanation seeks to explain the behavior of the already trained model. An example of an ante-hoc explanation is a decision tree, and that of the post-hoc is a counterfactual explanation.

The focus of this paper is on counterfactual explanations. Counterfactual explanations are not new and have a long history in fields such as psychology, philosophy, and social sciences [2], and are now being used in machine learning. At its core, a counterfactual is a hypothetical scenario that describes what would have happened if circumstances had been different from what they actually were [1].

The associate editor coordinating the review of this manuscript and approving it for publication was Genoveffa Tortora.

A typical example of a counterfactual explanation is a loan application [3]: *"Imagine you applied for a credit application at a bank. Unfortunately, the bank rejects your application. Now, you would like to know why. In particular, you would like to know what would have to be different so that your application would have been accepted. A possible explanation might be that you would have been accepted if you earned 500$ more per month and if you did not have a second credit card."*

Despite the prevalence of counterfactual explanations, there still remain limitations. The first limitation is that a counterfactual explanation assumes that a non-transparent classifier will not change over time, whereas in reality, it is possible for a classifier to change due to data drift [2]. The second limitation is that the counterfactual explanations are unable to handle missing data [2]. Despite these limitations, there is an interest in using counterfactual explanations within the machine learning community. However, there is no consensus within the machine learning community on how to assess the performance of counterfactual explanations [1]. The simplest way to measure the performance of counterfactual explanations is to look at several properties [2]. These properties include, *validity*, *sparsity*, *similarity*, *actionability*, *plausibilty* to mention a few. Please note that this list of properties is not exhaustive.

This research is driven by the need to enhance transparency and fairness in credit scoring processes. By investigating the use of counterfactual explanations, this paper aims to provide deeper insights into decision-making processes and offer a pathway towards more equitable and understandable credit assessments. This study extends our previous papers [4], [5], [6], where the aim was to improve the way creditworthiness is evaluated, making it more accessible and comprehensible to both lenders and borrowers. It is important to draw the reader's attention to differences between Dastile et al. [6] and the current study. The stark differences are in optimisation formulation, strong emphasis on achieving specific counterfactual properties, and the use of diverse optimisation methods for counterfactual generation which were not covered in [6]. Our key contribution lies in introducing a novel optimization problem formulation that generates counterfactual explanations, uniquely underpinned by multiple desirable properties such as validity, sparsity, actionability, plausibility, and similarity. The approach addresses a significant gap in current explainable AI practices that often yield counterfactuals which, while theoretically valid, fall short in practical applicability. Our innovation lies in a new approach that balances validity with sparsity in counterfactual explanations. While it moderately adjusts validity, it substantially improves sparsity, achieving a practical balance when using counterfactual explanations. Hence, the balanced trade-off ensures our counterfactuals are effective and user-friendly, marking a notable advancement in their real-world applicability. Furthermore, according to the best of our knowledge, our study is distinctive as it is the first to employ statistical methods to rigorously test and compare the effectiveness of counterfactual explanations produced by various approaches.

The organization of this paper is as follows. In Section II, related work is presented. In Section III, the methodology is discussed. The experiments are covered in Section IV. In Section V, the results are presented. The conclusion is in Section VII.

II. RELATED WORK

Machine learning classifiers that are unable to explain their predictions are not suitable for use in highly regulated industries such as finance, legal, and healthcare. Several explanation techniques can be found in the literature, and these techniques can mitigate the lack of explainability of the machine learning classifiers. One of the techniques is the counterfactual explanation. The counterfactuals are a class of explanations that not only inform the developers and users of machine learning classifiers but also individuals who are affected by the predictions from machine learning classifiers. In another study by Molnar [7], counterfactual explanations are described as changes to feature values that would alter a prediction outcome, making them an intuitive method for model interpretability. Molnar [7] provides a framework for understanding how small modifications can impact credit scoring decisions, which is crucial for ensuring fairness and transparency in financial models. Sokol and Flach [8] investigated the challenges and opportunities of counterfactual explanations. The opportunities or benefits of using counterfactuals include interpretability, fairness, and model debugging. The security challenges of using counterfactuals involve data stealing. Hence, the empirical results in [8] showed that when improving the safety of an AI system through transparency, the AI system can be exposed to security risks and breaches of data privacy. The privacy implications of counterfactual explanations were explored by Goethals et al. [9], who highlighted the risk of privacy attacks when real instances are used as counterfactuals. They proposed enhancing privacy through *k-anonymity*, ensuring that counterfactual instances cannot be easily linked back to individuals in the dataset.

It is imperative to generate counterfactual explanations that are of high quality, and the quality is measured by counterfactual properties. For example, Grath et al. [10] proposed two weighting strategies for generating sparse counterfactuals using ANOVA F-values and K-nearest neighbor. The empirical results showed that the proposed approach leads to more interpretable results than baseline counterfactual generating methods. Laugel et al. [11] proposed a technique that identifies a close neighbor classified differently to the datapoint that needs to be explained, where the closeness definition includes a sparsity constraint. The empirical results showed that the proposed approach can be used to attain knowledge about the opaque classifier. Looveren and Klaise [12] adopted the use of class prototypes to generate sparse counterfactual explanations. The class prototypes are generated by an encoder. The use of class

prototypes sped up the search process for the counterfactuals. Fernández et al. [13] proposed the use of a partial fusion of tree predictors from a random forest into a single decision tree using a modification of the classification and regression tree algorithm. The proposed approach obtained a counterfactual set that guaranteed an optimal sparse counterfactual. Samoilescu et al. [14] proposed the use of a deep reinforcement learning approach that converts the optimization process into an end-to-end learnable process, allowing the generation of sparse counterfactual explanations in a single forward pass. The empirical results showed that the proposed approach can produce sparse, in-distribution counterfactual explanations across different datasets. All of the above approaches focused on generating sparse counterfactuals, and sparseness ensures that the affected individuals only have to focus on making a few changes to get the desired outcome.

It is not enough to generate counterfactuals that are only sparse; counterfactuals need to be plausible and actionable. A plausible counterfactual is realistic. In [15], a novel approach named PermuteAttack was proposed to generate counterfactual explanations for machine learning models used in credit scoring. This method leverages a gradient-free optimization technique inspired by genetic algorithms, ensuring that the perturbations applied to data remain realistic and valid. Downs et al. [16] proposed a Counterfactual Recourse Using Disentangled Subspaces (CRUDS) approach that generates algorithmic recourse to achieve more favorable outcomes. The CRUDS approach generated recourse that is more plausible, realistic, and actionable than those of baseline counterfactual explanations. The generated counterfactuals obeyed causal relations between features. Cito et al. [17] used Large Language Models to help generate plausible and actionable counterfactuals in software engineering, specifically for code reviews and code performance prediction, to help and inform developers about their code base. The results showed that the generated explanations are useful and software developers can gain insights about their code bases. Suffian et al. [18] generated counterfactual explanations using human feedback. The empirical results showed the user-centricity and human-friendliness of the generated counterfactuals. Albeit the study allows users to input data, the users define input ranges of the variables, which may introduce outliers, resulting in unrealistic counterfactuals. Na et al. [19] proposed a novel framework for generating practical and plausible counterfactuals by minimally changing the meaningful information of inputs in a lower dimensional space of a generative adversarial network (GAN). The empirical results on different domain tasks demonstrate the superiority and versatility of the proposed framework. The study is only applicable to tasks where classifiers are differentiable. Fernandez et al. [20] proposed a heuristic search approach to find plausible and actionable counterfactuals. The findings were that techniques that use feature weights/importances are not conducive to model explanations. They argue that it is not clear how feature weights influence decisions. Förster et al. [21] proposed a novel approach to generate coherent counterfactual explanations. A counterfactual is coherent if the counterfactual scenario is realistic and typical as well as suitable to explain the factual situation [21]. The results showed that the proposed approach produces explanations that are significant, realistic, and as well as suitable to explain the factual situation when compared to state-of-the-art counterfactual explanations.

One of the limitations of the counterfactuals is the lack of robustness over time, which is influenced by the changing nature of the data over time [2]. The change in data normally requires model retraining. Model retraining may result in the counterfactuals that were previously proposed for a recourse no longer being valid. However, Ferrario and Loi [22] tackled the challenge of instability of counterfactual explanations. The study proposed a method that is called counterfactual data augmentation. The proposed method improves the stability of counterfactual explanations over time, which could result in more real-world adoption of machine learning applications. The quality of the generated counterfactuals can also be assessed by domain experts. For example, Carlevaro et al. [23] proposed the use of Support Vector Data Description (SVDD) to generate counterfactuals. The results showed that the proposed approach is trustworthy and can be understood and tested by domain experts who do not possess prior knowledge of AI. It was also noticed in the literature that some of the studies ([24], [25]) did not consider any of the counterfactual properties that normally help in assessing the quality of the generated counterfactuals. The counterfactual properties are used to enhance the quality of the generated counterfactuals. Hence, it is important to generate counterfactuals that encapsulate more counterfactual properties. It is observed in the literature that some studies only focused on a few counterfactual properties, see Table 1.

III. METHODOLOGY

This section focuses on the methods that were used to generate counterfactuals. Before formulating the new optimization problem, definitions are provided. Suppose $\mathbf{x} \in \mathbb{R}^d$ is a feature vector, where d denotes the number of features. In the context of credit scoring, which is a binary classification, the black-box model g maps the feature vector $\mathbf{x}$ to a predicted class label $\hat{y} \in \{0, 1\}$, i.e.,

$$g : \mathbf{x} \rightarrow \hat{y}, \tag{1}$$

where 0 represents a non-default class, and 1 represents a default class. The probability of $\mathbf{x}$ being mapped to the non-default class and default class is $P(g(\mathbf{x}) = 0)$ and $P(g(\mathbf{x}) = 1)$, respectively. Next, a counterfactual explanation and a counterfactual explainer are defined.

- **Counterfactual explanation**: A counterfactual explanation for a feature vector $\mathbf{x}$, is a feature vector $\mathbf{c}$ such that the output $\hat{y}' = g(\mathbf{c})$ differs from that of $\hat{y} = g(\mathbf{x})$, and that the distance between $\mathbf{x}$ and $\mathbf{c}$ is minimal [1].

TABLE 1. Summary table of the selected papers from the literature. Abbreviation: (CE) Counterfactual Explanation.

Source	Year	Methods for generating CE	Advantage	Disadvantage	Properties
[10]	2018	ANOVA F-values, KNN	Generates more compact and intelligible explanations.	Interaction between features is not considered.	Validity, Sparsity
[11]	2018	Growing Spheres Generation	The framework is model-agnostic and data-agnostic.	Prone to using irrelevant areas of the input space for explanations.	Validity, Sparsity
[25]	2019	SHAP	Accounts for interactions between features and is model-agnostic.	Computational complexity for high dimensional data.	Validity
[12]	2019	Encoder or k-d decision trees	Fast and efficient for searching counterfactuals.	Prone to skewed or unbalanced data leading to biased explanations.	Validity, Sparsity
[8]	2019	Class-contrastive	Fairness & model debugging	Compromised data privacy	Actionability
[13]	2020	RF-OCSE (Random Forest Optimal Counterfactual Set Extractor)	Handles missing data.	Biased towards features with many categories.	Validity, Sparsity
[16]	2020	Counterfactual Recourse Using Disentangled Subspaces (CRUDS)	Computationally efficient.	Impractical for high dimensional data.	Validity, Actionability
[20]	2020	Evidenced-based Explainer (EBE)	Ability to deal with high-dimensional data.	Multiple explanations may overwhelm users.	Validity, Plausibility
[15]	2020	Genetic Algorithm	Model-agnostic & maintains data integrity	Computationally intensive & May not scale with large data	Actionability, Validity
[21]	2021	Gradient-Free Optimization	Non-parametric, no need for assumptions of the underlying distribution.	Computationally intensive and not capable of handling missing data.	Validity, Coherence
[14]	2021	Reinforcement Learning (RL)	Model-agnostic and fast generation of counterfactuals.	Computationally complex.	Validity, Sparsity
[17]	2022	Masked Language Modeling	Semantic understanding of code.	Resource intensive and slow during inference time.	Validity, Actionability, Plausibility
[23]	2022	Support Vector Data Description (SVDD)	Less sensitive to parameter tuning.	Sensitive to outliers and impractical for unbalanced data.	Validity
[18]	2022	Feedback Counterfactual Explanation (FCE)	Reduces computational cost.	Causation influenced by users (domain experts).	Validity, Plausibility, Actionability
[22]	2022	Counterfactual Data Augmentation (CDA)	Counterfactuals are not influenced by model retraining.	May struggle to capture the underlying distribution of data.	Validity, Robustness over time
[24]	2022	Genetic Algorithm	Non-parametric and adaptable to data distribution.	Memory intensive and computationally complex.	Validity
[19]	2023	Generative Adversarial Network (GAN)	High-quality data generation.	Resource intensive.	Actionability and Plausibility

- **Counterfactual explainer**: Is a function u that takes as input, the feature vector $\mathbf{x}$ of the dataset $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, the black-box model g, and it returns a set of counterfactuals $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_k\}$ [1].

To find a counterfactual $\mathbf{c}$, the following cost function h must be minimized:

$$h(\mathbf{x}, \mathbf{c}, g) = \gamma \times (1 - P(g(\mathbf{c}) = 0)) + \text{dist}(\mathbf{x}, \mathbf{c}) \quad (2)$$

subject to:

$$P(g(\mathbf{c}) = 0) \geq 0.5 \quad (3)$$

and

$$\frac{1}{1 + \exp^{-\zeta^i}} < \delta, \quad \forall i \in \{1, 2, \cdots, d\}, \quad (4)$$

and

$$c^i = \begin{cases} c^i, & \text{if } f^i \text{ is mutable} \\ f^i, & \text{otherwise,} \end{cases} \quad (5)$$

and

$$c^i \in [\min(\mathbf{f}^i), \max(\mathbf{f}^i)], \quad \forall i \in \{1, 2, \cdots, d\}, \quad (6)$$

where γ controls the contribution of the first term against the second term in Eq. (2), and $\gamma > 1$, f^i and c^i represent the i^{th} feature value for a given $\mathbf{x}$ and $\mathbf{c}$, respectively, $P(g(\mathbf{c}) = 0)$ denotes the probability of a counterfactual belonging to a non-default class, and $\mathbf{f}^i$ represents feature values of the i^{th} feature in dataset $\mathbf{X}$ (i.e., $\mathbf{f}^i$ is column i in $\mathbf{X}$). The choice

of distance is

$$\text{dist}(\mathbf{x}, \mathbf{c}) = \sum_{i=1}^{d^*} \frac{|c^i - f^i|}{MAD^i} + \sum_{i=d^*+1}^{d} \frac{|c^i - f^i|}{MD^i}, \tag{7}$$

where MAD^i and MD^i denote a mean absolute deviation and a mean deviation for the i^{th} feature, respectively, and d^* denotes the number of continuous features [6]. Please note that the first term of Eq. (7) is the calculation of distance for continuous features, and the second term is the distance calculation for categorical features. The mean absolute deviation for the i^{th} feature in $\mathbf{X}$ is

$$MAD^i = \frac{1}{n} \sum_{j=1}^{n} |f_j^i - \bar{\mathbf{f}}^i|, \tag{8}$$

where n denotes the number of records and $\bar{\mathbf{f}}^i$ denotes the mean or average of the i^{th} feature [6]. The mean deviation for the i^{th} feature is

$$MD^i = \frac{1}{m} \sum_{l=1}^{m} o_l |f_l^i - \tilde{\mathbf{f}}^i|, \tag{9}$$

where m is the number of categories, o_l is the frequency of each category and $\tilde{\mathbf{f}}^i$ is the median of the i^{th} feature [6].

The five counterfactual properties that are included in the optimization problem are validity, sparsity, similarity, actionability, and plausibility. Validity measures the ability of a counterfactual to change the decision of an outcome [1]. Sparsity measures the minimal changes in data features/variables that are required to effect the desired outcome [1]. Similarity measures how far a counterfactual is, from the original instance [1]. Actionability measures the ability of a counterfactual to change mutable features only [1]. Plausibility ensures that the counterfactual is realistic [1].

The validity property is captured in Eq. (3). The sparsity property is represented in Eq. (4), the similarity property is captured in the actual cost function in Eq. (2) by the distance calculation term, the actionability property is captured in Eq. (5), and the plausibility is represented in Eq. (6). The term ζ^i in Eq. (4) measures the absolute percentage change between the i^{th} feature value in the vector of interest $\mathbf{x}$ and the generated counterfactual $\mathbf{c}$ and is given as

$$\zeta^i = \frac{|c^i - f^i|}{|f^i| + \epsilon}, \quad \forall i \in \{1, 2, \cdots, d\}, \tag{10}$$

where ϵ is a small positive arbitrary number that guarantees a non-zero division. The absolute percentage change values ζ^i can be greater than 1. For ease of use, the logistic function (i.e., the left-hand side of Eq. (4)) is used to ensure the values are not greater than 1, where the range of the logistic function is [0.5, 1]. Since the absolute change values ζ^i are inputs to the logistic function and are non-negative, the lower value of the logistic function is 0.5.

To guarantee sparsity, a threshold value $0.5 < \delta < 1$ must be chosen when generating a counterfactual $\mathbf{c}$. Any feature values in $\mathbf{x}$ that result in the left-hand side of Eq. (4) to be greater than δ, those feature values will not be changed when generating the counterfactual $\mathbf{c}$.

In the sequel, details of how predictive features were selected, and optimization techniques that this study focuses on are provided.

A. FEATURE SELECTION

Feature selection, regarded as an important step in machine learning, is a process of selecting predictive features. The aim of feature selection is to improve model performance, reduce overfitting, and minimize training time. There are multiple methods for feature selection, broadly categorized into filter methods, wrapper methods, and embedded methods. A literature review by [5] explains in detail the methods of feature selection. In this study, an *Information Value* (IV) [26] (a filter method) was used to select predictive features. The IV uses weights of evidence (WOE) [26]. The WOE is a measure used primarily in credit scoring to quantify the predictive power of each feature concerning a binary target variable. Each feature is divided into various bins, and the weight of evidence is calculated for each bin. The WOE is denoted for bin b in feature f as $WOE_b^{(f)}$ which is calculated as follows:

$$WOE_b^{(f)} = \ln\left(\frac{N_b^{(f)}}{D_b^{(f)}}\right) \tag{11}$$

where $N_b^{(f)}$ is the frequency of non-defaults and $D_b^{(f)}$ is the frequency of defaults for bin b in feature f [4].

While WOE is traditionally used in credit scoring, it can also be applied in other domains where binary classification is required, as it helps measure each attribute's predictive power.

The IV is used to select predictive features and is calculated as follows for each feature f [4],

$$IV^{(f)} = \sum_{b=1}^{B^{(f)}} \left(N_b^{(f)} - D_b^{(f)}\right) WOE_b^{(f)} \tag{12}$$

where $B^{(f)}$ represents the number of bins in feature f. The following thresholds apply as a general rule of thumb when using IV for feature selection [26]:

T1) $IV^{(f)} < 0.02$: unpredictive;
T2) $0.02 \leq IV^{(f)} < 0.1$: weak predictor;
T3) $0.1 \leq IV^{(f)} < 0.3$: medium predictor;
T4) $0.3 \leq IV^{(f)} < 0.5$: strong predictor; and
T5) $IV^{(f)} \geq 0.5$: suspicious or too good to be true.

Both categorical and continuous variables were considered for WOE and IV analysis. For continuous features, binning was performed to create discrete intervals. WOE and IV were then calculated for these binned intervals. For categorical variables, the existing categories naturally formed the bins, and WOE and IV were computed directly on these categories.

B. GENETIC ALGORITHM (GA)

The genetic algorithm is an evolutionary search approach that is motivated by natural evolution [27]. The genetic algorithm

Algorithm 1 Genetic Algorithm (GA) [27]

Require: A population size, crossover rate, mutation rate, and termination criteria
Initialize a random population
Evaluate the fitness of each individual in the population
while The end condition is not met **do**
Select parent pairs based on fitness
Produce offspring through crossover of parents
Apply mutation to offspring with the given mutation rate
Evaluate the fitness of the offspring
Replace the old population with the new one
end while

return The best solution found in the population

process involves five phases i.e. the *initial population*, the *fitness function*, the *selection*, the *cross-over*, and the *mutation*. The genetic algorithm involves a selection of individuals (based on some defined fitness function) from an initial population for reproduction/mating (i.e., cross-over) purposes. The selected individuals produce diverse offspring (using mutation) that inherit the initial individual's characteristics. The diverse offspring gets included in the next generation of the population. The process repeats itself until it converges to a solution (a counterfactual explanation). In the context of credit scoring, an individual is a feature vector (or single record) that can potentially be a counterfactual that will be used to explain an original instance. The population is the set of possible counterfactuals. The fitness function is the neural network that is used to predict the output (i.e., default or non-default) of the original instance. The cross-over rate is the probability of two feature vectors exchanging values at a single point [27]. The mutation is responsible for creating diversity during the optimization to prevent early convergence.

C. PARTICLE SWARM OPTIMIZATION (PSO)

The PSO mimics the navigation of a flock of birds or a school of fish [28]. The original inventors of the PSO are Eberhart and Kennedy [29]. Since its inception in 1995, PSO has gained a huge amount of improvements over time [28].

The PSO is a stochastic search strategy and uses position vectors and velocities of the particles in a swarm [28]. The particles are assigned initial random position and velocity values. Each particle in the swarm has its own objective function, which is governed by the position of the particle and the velocity at which the particle moves [28]. In the context of credit scoring, a particle is a feature vector that serves as a potential candidate for being a counterfactual explanation. To find the optimal minimum for the objective function, the particle will have to consider its personal best-optimized value and the global best-optimized value in the entire swarm. The final global best value is found when all the particles have converged to the global best. The current position of particle j is represented by Z_j^t and the position of particle j in the next iteration/generation is

$$Z_j^{t+1} = Z_j^t + V_j^{t+1}, \tag{13}$$

where V_j^{t+1} represents the velocity of moving from position Z_j^t to position Z_j^{t+1} and t is the indexing/iteration value. The velocity at the next iteration is represented by the following formula

$$V_j^{t+1} = \omega V_j^t + a_1 r_1 (P_j^t - Z_j^t) + a_2 r_2 (G_j^t - Z_j^t). \tag{14}$$

The first part of Eq. (14) is the *inertia* where ω represents the weight parameter [28]. The second part of Eq. (14) is called *cognitive component* and P_j^t denotes the personal best-optimized value [28]. The third part of Eq. (14) represents the *social component* and G_j^t is the global best-optimized value [28]. The particle j is a feature vector and the aim is to find the feature values that result in the global best-optimized value (i.e., the minimal value of the objective function) in a search space. Eq. (14) has several parameters that may influence the performance of the PSO. The initial researchers who studied the impact of the weight parameter on the performance of the PSO were Shi and Eberhart [30]. The weight parameter ω balances exploration and exploitation [30]. The exploitation refers to the particles converging to a known solution in the search space [30]. The exploration refers to how well a particle explores other regions in a search space that may have possible optimized objective function values [30]. The value of ω should gradually decrease with time/iterations [28]. Shi and Eberhart [30] proposed that the weight parameter must change during the search from 0.9 to around 0.2.

The impact of the parameters a_1 and a_2 (which denote the acceleration of the particles towards the personal best solution and the global best solution, respectively) is assessed by setting one parameter to 0 and setting the other parameter to 2. When $a_1 = 0$, there's no personal best in the search space, leading particles to converge to a minimum, maximizing exploitation but minimizing exploration. Conversely, for $a_2 = 0$, particles don't converge due to maximum exploration and minimal exploitation, as they rely solely on their personal best without sharing information. It is therefore key to balancing exploration and exploitation. In most cases, $a_1 = a_2 = 2$ to ensure that there is a balance between the exploration and the exploitation [28]. Carlisle and Dozier [31] conducted experiments and found that the optimal values for a_1 and a_2 are 2.8 and 1.3, respectively, and this was further confirmed by Schutte and Groenwold [32].

The parameters r_1 and r_2 are random numbers ranging between 0 and 1, and their purpose is to create randomness in the search space.

D. BAYESIAN OPTIMIZATION (BO)

Bayesian Optimization is another heuristic search approach. Bayesian optimization uses a notion of a surrogate function [34]. The purpose of the surrogate function is to

Algorithm 2 Particle Swarm Optimization (PSO) [33]

Require: The controlling parameters. The population of N particles.
  while The end condition is not satisfied **do**
    for Each particle **do**
      Calculate the objective function
      Update personal best (PBEST) if required
      Update global best (GBEST) if required
    end for
    Update the inertia weight
    for Each particle **do**
      Update the velocity (V)
      Update the position (Z)
    end for
  end while

  return GBEST

approximate the original cost function during optimization [34]. Figure 1 illustrates the phenomenon of Bayesian Optimization. The first frame (1) in Figure 1 shows the cost function $h(\mathbf{x}, \mathbf{c}, g)$ that needs to be optimized. The surrogate function is represented by the dashed line in frame (2) of Figure 1 and it is formed based on sampled data points. In each iteration, new counterfactual data points are sampled and the surrogate function is updated based on the new sampled counterfactual data points. When the iterations are exhausted, the process reaches a global minimum. The complexity (in terms of evaluation) of the cost function is taken away by the surrogate functions which are normally cheap (in terms of evaluation) [35]. In the following section, the Bayes Theorem is discussed, which serves as the basis for Bayesian Optimization.

1) BAYES THEOREM

Before the discussion of the Bayes decision rule, the following notation is introduced:

- $P(h(\mathbf{c})|\mathbf{c})$ denotes the *a posterior probability* [36] (i.e. the probability of the cost function given the counterfactual $\mathbf{c}$);
- $P(h(\mathbf{c}))$ denotes the *a priori probability* [36] (i.e. the probability distribution of the cost function);
- $p(\mathbf{c}|h(\mathbf{c}))$ represents the *conditional density* [36] (i.e. the distribution of the feature vector $\mathbf{c}$ given $h(\mathbf{c})$);
- $p(\mathbf{c})$ represents the *evidence* [36] (it can be viewed as the scaling factor that guarantees that the *posterior probabilities* sum to one).

The Bayes decision rule [36] is defined as follows:

$$P(h(\mathbf{c})|\mathbf{c}) = \frac{p(\mathbf{c}|h(\mathbf{x}))P(h(\mathbf{c}))}{p(\mathbf{c})}. \tag{15}$$

Note that $p(\mathbf{c}|g(\mathbf{c}))$ is a multivariate Gaussian distribution density function. The Gaussian distribution density function is given as:

$$p(\mathbf{c};\boldsymbol{\mu},\Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}}\exp\{-\frac{1}{2}(\mathbf{c}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\}$$

where the mean, $\boldsymbol{\mu}$, and covariance-matrix, Σ, are estimated by using Expectation Maximization (EM) [37].

2) SURROGATE AND ACQUISITION FUNCTIONS

The probability $P(h(\mathbf{c})|\mathbf{c})$ in Eq. (16) is known as the Gaussian Process which represents a surrogate model [35]. The acquisition function is applied to the surrogate function to find better-sampled data points. The commonly used acquisition function is the *Expected Improvement* (EI) [35]. The formula for EI is:

$$\mathbf{EI}_{h^*}(\mathbf{c}) := \int_{-\infty}^{\infty} \max(h^* - h(\mathbf{c}), 0)P(h(\mathbf{c})|\mathbf{c})\,dh(\mathbf{c}). \tag{16}$$

The h^* represents the best (i.e., lowest) cost function value observed so far. The $h(\mathbf{c})$ represents the cost function evaluated at counterfactual $\mathbf{c}$. The aim is to maximize the acquisition function [34] to find better data points (i.e., counterfactuals) when sampling, which will potentially result in a minimum for the cost function. Thereafter, the surrogate function is updated using the sampled data points. This is repeated until a global minimum is reached or the number of iterations is reached. The acquisition function is responsible for newly sampled data points and the notion of exploration and exploitation is applicable when looking for new data points [34]. The exploitation allows the acquisition function to sample where the surrogate function results in an optimized cost function. The exploration allows the acquisition function to sample in regions with lots of uncertainty in hopes that better data points will be found that will result in an optimized cost function. Thus, the acquisition function must balance the exploration and the exploitation when sampling for new data points.

Algorithm 3 shows the pseudo-code for the Bayesian Optimization. The algorithm requires several inputs and these include, the number of iterations, cost function, surrogate function, and acquisition function. The surrogate function M is an approximation of the cost function h. The acquisition function S, determines where to sample next during the search. The $\mathcal{H}$ is a set of pairs consisting of the counterfactual and the cost function, i.e., $(\mathbf{c}, h(\mathbf{c}))$. The set of pairs $\mathcal{H}$ is initialized to an empty set. The first line inside the iteration loop finds the input value $\mathbf{c}$ that minimizes the acquisition function S. This step determines where to sample the cost function next, based on the previous surrogate model M_{t-1}. The second line inside the iteration loop updates the set of pairs $\mathcal{H}$ by appending a new pair $(\mathbf{c}, h(\mathbf{c}))$. The last line in the iteration loop fits a new surrogate model M_t based on the updated set $\mathcal{H}$. The first line outside the iteration loop sorts the pairs from lowest in terms of the cost function $h(\mathbf{c})$ to highest. The last line returns a pair $(\mathbf{c}, h(\mathbf{c}))$ in $\mathcal{H}$ with the lowest cost function. Consequently, the $\mathbf{c}$ in the returned pair, is the counterfactual explanation.

E. STATISTICAL SIGNIFICANCE TESTING

Statistical significance testing of the counterfactual generating approaches is assessed. The aim is to see if there are performance differences in the way counterfactual explanations are generated. The counterfactual properties that were used for significance testing are validity, similarity, and sparsity. All statistical significance tests in this study were based on *Friedman test* [38] which is a non-parametric test. The Friedman test consists of four steps, i.e., a hypothesis, a test statistic, a rejection region, and a conclusion. Below are the steps involved in the Friedman test, Step-1)

1) State a null hypothesis H_0 and an alternative hypothesis H_1;
   - H_0: The counterfactual generating approaches do not differ in how they rank counterfactual performances for validity/sparsity/similarity.
   - H_1: At least one counterfactual generating approach differs in how it ranks counterfactual performances for validity/sparsity/similarity.
2) Calculate the test statistic [39];
   - Collect all the results for each counterfactual generating approach.
   - For each generated counterfactual i, rank values from 1 (denoting best performance) to o (denoting worst performance). The ranks are denoted as $r_i^j (1 \leq j \leq o)$ and i ranges between $(1 \leq i \leq q)$. Please note that in our study, the number of generated counterfactuals for Friedman test is $q = 30$, and a number of counterfactual generating approaches is $o = 4$.
   - For each counterfactual generating approach j, calculate the average obtained for all generated counterfactuals as $R_j = \frac{1}{q} \sum_i^q r_i^j$.
   - The F-score, which is the test statistic is

$$\text{F-score} = \frac{12}{oq(q+1)} \sum_{j=1}^{o} R_j^2 - 3o(q+1).$$

3) Define a rejection region;
   - Rejection region is defined by the level of significance $\alpha = 0.05$, using the upper critical values from the chi-square table.
   - The critical value is defined as a chi-squared value χ^2_{b-1}, where $b-1$ is the degrees of freedom.
4) Draw a conclusion;
   - The null hypothesis is rejected if F-score $> \chi^2_{b-1}$ or p-value < 0.05 and a conclusion is that there is no significant evidence that the counterfactual generating approaches rank counterfactual performances equally/perform similarly.

In essence, the Friedman test assesses whether there exist significant differences in how counterfactuals perform when generated by more than two approaches. To find exactly which counterfactual generating approaches differ in terms of counterfactual performances, a post-hoc test known as Nemenyi test [40] is used.

Algorithm 3 Bayesian Optimization (BO) [35]

Require: Number of iterations T, cost function h, surrogate function M and acquisition function S
$\mathcal{H} = \emptyset$
for Each iteration t **do**
  $\mathbf{c} \leftarrow \text{argmin } S(\mathbf{c}, M_{t-1})$
  $\mathcal{H} \leftarrow \mathcal{H} \overset{\mathbf{c}}{\cup} (\mathbf{c}, h(\mathbf{c}))$
  Fit new model to M_t
end for
Sort $\mathcal{H}$ using $h(\mathbf{c})$ in ascending order

return First pair of $\mathcal{H}$

IV. EXPERIMENTS

A. BLACK-BOX MODEL

The black-box model that was used is a deep neural network consisting of three hidden layers. The first hidden layer had 20 neurons, the second hidden layer had 10 neurons and the third hidden layer had 6 neurons. The input layer and the output layer had 12 and 2 neurons, respectively. The hidden layers' activation functions were *Rectified Linear Unit* (ReLU). The ReLU returns a maximum between 0, and the sum of the multiplication of the inputs, weights, and biases. The choice of the optimizer was Adam (Adaptive Moment Estimation). An optimizer is a method of adjusting the weights and biases of a neural network to ensure the loss function is minimized. Since the classification task is binary (either default or non-default), the choice of the last hidden layer activation function was a sigmoid function. Since a sigmoid function maps any input to values between 0 and 1, a decision boundary was created by choosing a threshold of 0.5, to classify input data with prediction outputs above 0.5 as defaults and below 0.5 as non-defaults. The batch size of the neural network was set to 50 and the number of epochs was 1000. Please see Table 2 for parameter details of deep neural network. To optimize the performance of the artificial neural network (ANN), we conducted a grid search to identify the best set of hyperparameters in Table 2. Grid search is normally performed using K-fold cross validation on the training set. For each combination of hyperparameters, the ANN was trained using K-fold cross validation. The combination of the parameters that resulted in the best accuracy on the validation set was considered best hyperparameters.

B. DATA

The dataset that was used in this study is the German credit dataset [41]. The credit dataset is publicly available on *Kaggle* website. The dataset has 21 features, where 7 of the features are continuous and the other 14 are categorical. The total number of records on the German dataset is 1000.

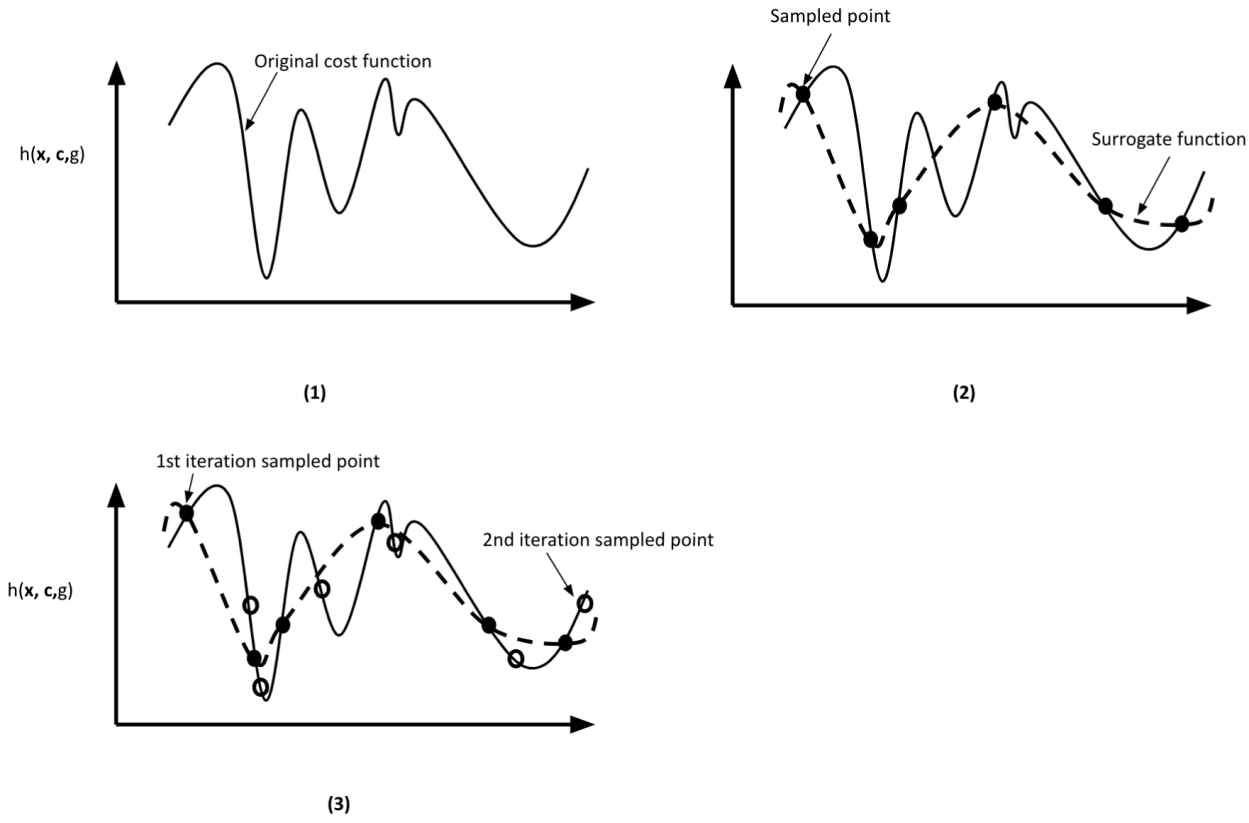

FIGURE 1. Bayesian optimization process.

TABLE 2. Artificial neural network parameters and grid search values.

Model (Artificial Neural Network)	Parameters	Grid Search Values
No. of Hidden Layers	3	{2, 3, 4}
Input Layer (Neurons)	12	-
First Hidden Layer (Neurons)	20	{10, 20, 30}
Second Hidden Layer (Neurons)	10	{10, 20, 30}
Third Hidden Layer (Neurons)	6	{5, 6, 7}
Output Layer (Neurons)	2	-
Activation Functions	Rectified Linear Unit (ReLu)	{ReLU, Sigmoid, Tanh}
Optimizer	Adam	{Adam, SGD}
Batch Size	50	{32, 50, 64}
Epochs	1000	{500, 1000, 1500}
Last Hidden Layer Activation Function	Sigmoid	-

The German credit dataset has features such as `status of existing checking account`, `duration in month`, `credit history`, `purpose`, `Age (years)`, `Sex & Marital Status` and `Length of current employment` to mention the least. Features such as `Age (years)` and `Sex & Marital Status` are treated as immutable. All other features are mutable. A feature selection was performed to select predictive features (features that have information values between 0.1 (inclusive) and 0.5 (exclusive)), and this resulted in 12 features. The target variable is binary (i.e., default/non-default). Please see Table 3 and Table 4 for feature descriptions and feature min-max values, respectively.

C. OPTIMIZATION ALGORITHMS' PARAMETERS

The genetic algorithm had a population size of 100, the mutation probability was set to 0.01, the crossover probability was 0.2 and the number of iterations was 500. The choice of the mutation rate of 0.01, was based on the experiments done by Grefenstette [42]. Further, Grefenstette [42] posited that mutation rates that exceed 0.05 are not good for the optimal performance of Genetic Algorithms. Hassanat et al. [43] noted that the use of 0.9 (usually used as a predefined parameter setting to encourage broader exploration search) for cross-over rate does not always perform best. Hence, the choice for cross-over rate was 0.2 to reduce too much exploration.

TABLE 3. Features descriptions of the german credit data [41].

Feature Name	Description
Age (years)	Age in years (e.g., 25, 30, 40)
Length of current employment	Employed since (e.g., < 1 year, >= 7 years)
Job	Job type (coded as integers, e.g., 0, 1, 2, 3)
Housing	Type of housing (e.g., own, rent, free)
Value Savings/Stocks	Savings account status (e.g., little, moderate, quite rich, rich)
Most valuable available asset	Checking account status (e.g., little, moderate, rich)
Credit amount	Amount of credit in DM (e.g., 1500, 2000)
Duration of Credit	Duration for the credit in months (e.g., 12, 24, 36)
Purpose	Purpose of the credit (e.g., radio/TV, education, furniture/equipment)
Risk	Good or bad risk (e.g., good, bad)
Sex & Marital Status	Personal status and sex (e.g., male single, female divorced/separated)
Debtors/Guarantors	Whether there are debtors or guarantors (e.g., none, guarantor)
Residence Duration	Duration of residence at the current location (e.g., 2, 3, 4 years)
Type of apartment	Type of property (e.g., real estate, life insurance)
Installment rate	Installment rate in percentage of disposable income (e.g., 2, 3, 4)
Other installment plans	Existence of other installment plans (e.g., bank, stores)
Concurrent credits	Number of credits at this bank (e.g., 1, 2, 3)
Job type	Type of job (e.g., skilled, unskilled resident)
Payment status of previous credit	Credit history (e.g., paid on time, delayed), coded as 1, 2
Telephone	Whether the individual has a telephone registered (e.g., yes, no)
Foreign worker	Whether the individual is a foreign worker (e.g., yes, no)

TABLE 4. Selected features with their min-max values.

Feature Name	Minimum	Maximum
Age (years)	19	75
Concurrent credits	1	3
Credit Amount	250	18424
Duration of Credit	4	72
Foreign Worker	1	2
Length of current employment	1	5
Most valuable available asset	1	4
Payment status of previous credit	0	4
Purpose	0	10
Sex & Marital Status	1	4
Type of apartment	1	3
Value Savings/Stock	1	5

TABLE 5. Parameters for different optimization techniques. Legend: EI (Expected Improvement), GP (Gaussian Process), NFE (Number of Function Evaluations).

Technique	Parameters
Genetic Algorithm	Population Size: 100 Mutation Probability: 0.01 Crossover Probability: 0.2 Iterations: 500 NFE: 50 000
Particle Swarm Optimization	Population Size: 40 Weight Parameter: 0.9 to 0.2 Acceleration: 2 Iterations: 50 NFE: 2 000
Bayesian Optimization	Iterations: 100 Acquisition Function: EI Surrogate Function: GP NFE: 100

The particle swarm optimization had a population size (i.e., the number of particles) of 40, the weight parameter was decreasing from 0.9 to 0.2, the acceleration parameters were set to 2 and the number of iterations was 50. Shi and Eberhart [30] suggested that the weight parameter should change during the search from 0.9 to around 0.2. The acceleration parameters were set to 2, to ensure that there is a balance between exploration and exploitation [28] during the counterfactual explanation search.

The number of iterations for the Bayesian Optimization was set to 100. Please refer to Table 5 for parameter details of the optimization algorithms.

Each of the optimization problems must take into account sparsity. The level of sparsity is controlled by the threshold value which ranges between $0.5 < \delta < 1$. The threshold value was set to 0.9. If δ is set too low, the counterfactual will be very similar to the original instance, since the majority of the feature values will not be changed during the search of counterfactuals. Thus, high values of δ, allow the original instance feature values to be changed during the search of counterfactuals. This is illustrated in Figure 2. The impact of the value of δ is illustrated in Figure 2, when δ is set low, even minor absolute percentage changes in features will cause them not to be changed. The balanced threshold is at 0.8. However, a balanced threshold of 0.8 might not always guarantee a desired prediction (i.e., non-default) for the counterfactual. A higher delta optimally balances sparsity with the likelihood of flipping the counterfactual's outcome to the desired prediction.

D. NUMBER OF FUNCTION EVALUATIONS

Another important metric is the number of function evaluations(NFE) [44] which evaluates the computational complexity of an optimization problem. The NFE is calculated as follows:

$$\text{NFE} = \text{Number of Iterations} \times \text{Population Size}.$$

The higher the NFE the more computationally intensive the algorithm. However, this is also driven by the complexity of the objective function. For example, if each evaluation of the objective function is expensive, even if the NFE is relatively

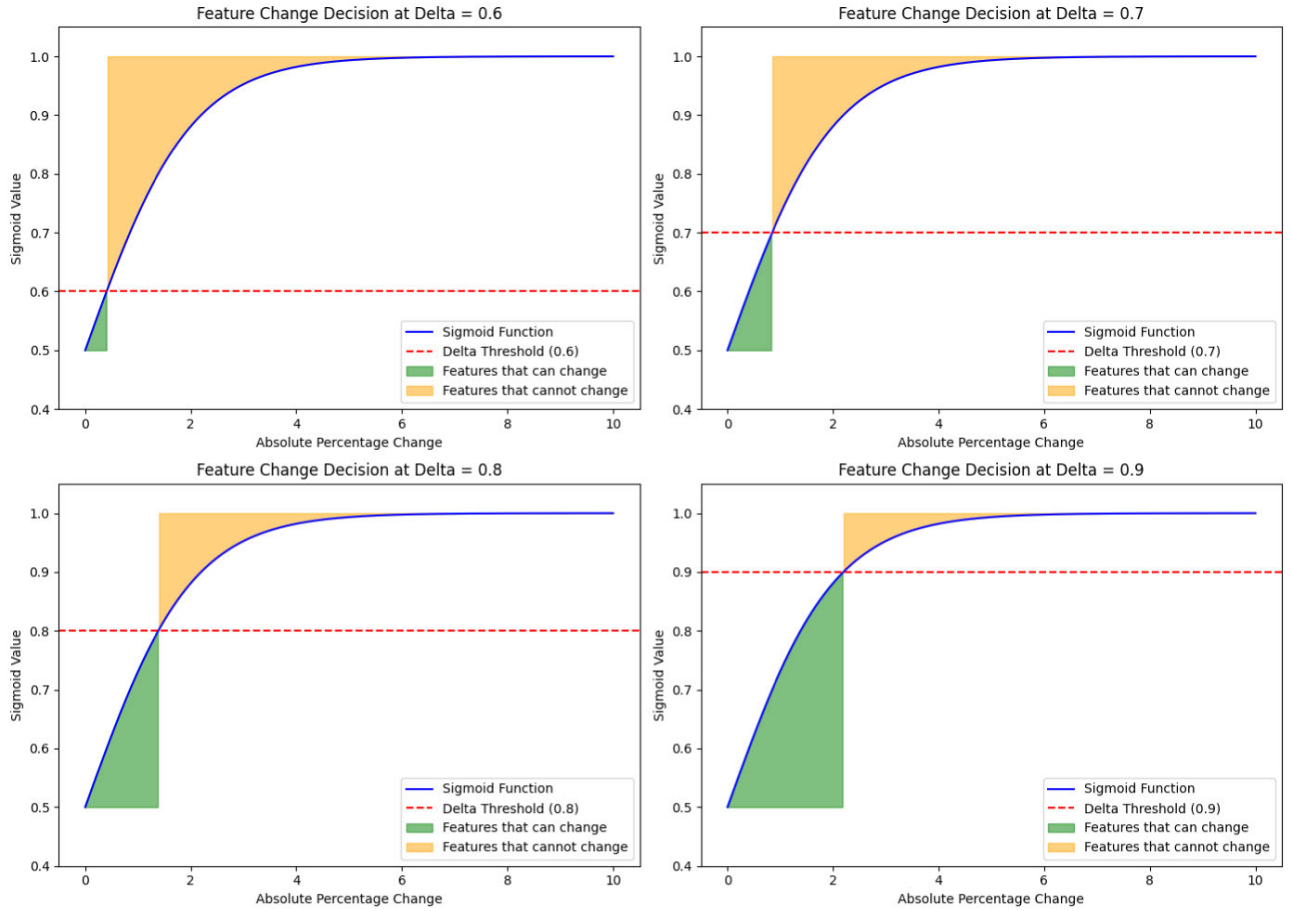

FIGURE 2. Impact of delta (δ) threshold on feature changes. The x-axis represents the absolute percentage change, and the y-axis represents the sigmoid value. The green region illustrates where the sigmoid values fall below the set threshold (δ), indicating acceptable feature changes. The yellow region illustrates where the sigmoid values are above the threshold, indicating no feature changes. As (δ) increases from 0.5 to 1, the size of the green region increases, illustrating lower sparsity. The red dotted line represents the threshold value that controls the sparsity level. These regions are for illustration purposes only and do not represent specific features.

low, it can result in high computational demand for compute resources [44].

E. SELECTION OF COUNTERFACTUALS

It is important to explain how the final counterfactuals were obtained. Since the optimization methods used in this study generate different results for each optimization run due to randomness. Given the extended processing durations for generating counterfactuals, each optimization algorithm was executed 10 times. Consequently, 10 potential counterfactual candidates were produced. The final counterfactual was chosen among the 10 candidates using Eq. (7) which calculates the L1-norm distance between the original instance and the counterfactuals. A counterfactual that resulted in a small distance was chosen as the final counterfactual.

V. RESULTS

This section begins with an examination of the performance of the black-box model. Subsequently, we delve into counterfactual explanations of the black-box model, generated using a range of optimization algorithms. It is important to clarify that using numeric values for categorical features is not appropriate for interpretability. While numeric values were utilized to generate counterfactuals, for the purpose of explanation, we will present the actual categorical values. This ensures that the counterfactuals remain explainable and meaningful. For instance, instead of saying ''Payment status must decrease by 4'' we will use the corresponding categorical descriptions to provide clear and understandable explanations. This approach aligns with the aim of making the counterfactuals comprehensible. To clarify, the numerical codes assigned to each category are used strictly as identifiers for ease of generating counterfactual explanations.

Following this, we conducted a performance assessment of three selected optimization algorithms using explainer properties. Thereafter, a qualitative comparison of the counterfactual generating approaches was conducted. In this context, a counterfactual generating approach that exhibits a greater number of counterfactual properties is considered superior. Additionally, we have undertaken a quantitative evaluation of various counterfactual generating methodologies. Notably, in this study, Particle Swarm

Optimization (PSO) was the chosen optimization algorithm, owing to its efficiency in solving optimization problems.

A. MODEL PERFORMANCE

The Receiver Operating Characteristic Area Under the Curve (ROC AUC) metric quantifies the overall ability of a model to distinguish between the default (i.e., positive) and non-default (i.e., negative) classes, irrespective of the threshold [45]. An AUC of 1 indicates perfect classification, whereas an AUC of 0.5 implies that the model performs no better than random guessing. The ROC AUC is given as:

$$\text{ROC AUC} = \frac{1}{2}\left(1 + \frac{TP}{TP+FN} - \frac{FP}{FP+TN}\right)$$

where:

- TP (True Positives) is the number of correctly predicted positive instances;
- FN (False Negatives) is the number of positive instances incorrectly predicted as negative;
- FP (False Positives) is the number of negative instances incorrectly predicted as positive;
- TN (True Negatives) is the number of correctly predicted negative instances.

This formula provides a simplified method for calculating the ROC AUC in binary classification scenarios. It balances the true positive rate (sensitivity) and false positive rate (1 - specificity) to give an overall performance measure of the classifier.

The dataset was split into training and test sets using a 70/30 split. The training set was used to train the neural network, and the test set was used to assess the neural network's performance. The metric of choice to assess the neural network's performance was ROC AUC, and the ROC AUC on the test set was 0.71. This indicates that the neural network performs better than random guessing.

B. GENETIC ALGORITHM

The counterfactual explanation suggests that for the feature vector of interest $\mathbf{x}$ to qualify for a loan application, the `Purpose` must change from buying a new car to buying a used car, the `Payment Status of Previous Credit` must change from critical account to existing account paid back fully till now and the `Length of current employment` must change from being unemployed to employed between 1 and 3 years.

C. PARTICLE SWARM OPTIMIZATION

The counterfactual explanation suggests that for the feature vector of interest $\mathbf{x}$ to qualify for a loan application, the `Purpose` must change from wanting to buy a new car to buying furniture/equipment and `Concurrent Credits` must decrease by 2, the `Type of apartment` must change from owning to staying for free and the `Duration of Credit (month)` must increase by 38.

D. BAYESIAN OPTIMIZATION

The counterfactual explanation suggests that for the feature vector of interest $\mathbf{x}$ to qualify for a loan application, the `Purpose` must change from wanting to buy a new car to buying a radio/television, the `Payment Status of Previous Credit` must change from critical account to no credit taken/all credits paid back duly, the `Most valuable available asset` must change from life insurance to car or other and the `Concurrent Credits` must decrease by 2.

E. OPTIMIZATION ALGORITHMS IN EXPLAINER PROPERTIES

Three distinct optimization algorithms were employed to solve optimization problems. This section compares three optimization algorithms by using explainer properties. According to [1], explainers have properties such as *efficiency*, *fairness*, and *stability*. The efficiency property is defined as the time it takes to generate a counterfactual. The fairness property is defined in Guidotti [1] as the ability of the explainer to give the same decision based on the same feature changes in $\mathbf{x}$ irrespective of the demographic group. Suppose a feature vector $\mathbf{x}$ is rejected for a loan application (i.e., $g(\mathbf{x}) = 1$), and that $\mathbf{c}_s$ and $\mathbf{c}_t$ are counterfactuals for feature vector $\mathbf{x}$ where the *gender* feature value for $\mathbf{c}_s$ is male and for $\mathbf{c}_t$ is female. The explainer h is fair if and only if the features that are changed in $\mathbf{x}$ are the same for $\mathbf{c}_s$ and $\mathbf{c}_t$ and that

$$g(\mathbf{c}_s) = g(\mathbf{c}_t) = 0. \tag{17}$$

The stability (also known as *robustness*) property is defined in [1] as the ability of the explainer to generate similar counterfactuals (i.e. $\mathbf{c}_1$ and $\mathbf{c}_2$) when the feature vectors (i.e. $\mathbf{x}_1$ and $\mathbf{x}_2$) of interest are similar and their predictions are the same (i.e. $g(\mathbf{x}_1) = g(\mathbf{x}_2)$). The similarity will be measured by the cosine similarity given as

$$cosine(\mathbf{c}_1, \mathbf{c}_2) = \frac{\mathbf{c}_1 \cdot \mathbf{c}_2}{\| \mathbf{c}_1 \| \| \mathbf{c}_2 \|}. \tag{18}$$

The cosine similarity has values between 0 and 1 (inclusive). A cosine similarity value closer to 1 indicates greater similarity between feature vectors, while a value approaching 0 suggests increased dissimilarity between them. The results of fairness and robustness are shown in Table 6. The results show that all three optimization algorithms perform similarly for explainer fairness and robustness. However, for efficiency, there are differences. The optimization algorithm that results in making the explainer more efficient in this study is the particle swarm optimization. Please note that there was only one experiment run, hence, the results are not statistically significant. For clarity, values in Table 6 for robustness are rounded to two decimal places, though actual values may differ in precision.

TABLE 6. Comparison of all three optimization techniques using explainer properties. The legend: GA (Genetic Algorithm), PSO (Particle Swarm Optimization), BO (Bayesian Optimization).

	Optimization Algorithms		
Properties	GA	PSO	BO
fairness	yes	yes	yes
robustness	0.99	0.99	0.99
efficiency	30 minutes	5 minutes	50 minutes

TABLE 7. Comparing state-of-the-art counterfactual methods with our approach.

Source	Year	Validity	Similarity	Sparsity	Actionability	Plausibility
[46]	2019	✓	✓		✓	
[47]	2019	✓	✓	✓		
[48]	2020	✓	✓			✓
[49]	2021	✓	✓			✓
[6]	2022	✓	✓	✓		✓
Our Approach	2023	✓	✓	✓	✓	✓

F. COMPUTATIONAL COMPLEXITY

Since particle swarm optimization resulted in low computational time, it is worth looking at its complexity. We specifically looked at time complexity using the Big O notation $O(n)$. The time complexity depends on the number of particles in a swarm, the dimension d of the feature vector $\mathbf{x}$, and the number of iterations the algorithm runs. Overall the complexity of the proposed approach is:

$$\text{complexity} = O(d * n_i * n_p)$$

where n_i and n_p denote the number of iterations and number of particles, respectively. The Big O notation $O(d * n_i * n_p)$ of our proposed approach essentially means that the running time of the particle swarm optimization algorithm increases linearly with the number of dimensions, the number of iterations, and the number of particles. If you increase any of these variables, the running time will increase proportionally.

G. QUALITATIVE COMPARISON OF COUNTERFACTUALS

Using counterfactual properties for qualitative comparison is a straightforward method to evaluate counterfactual performances. A counterfactual is deemed superior if it has multiple counterfactual properties. The counterfactual properties such as validity, similarity, sparsity, actionability, and plausibility were considered. Table 7 shows that most methods focus on validity and similarity. Our approach looks at all the suggested counterfactual properties. There is only one method [46] that focuses on the actionability of the generated counterfactuals. Please note that the list of sources in Table 7 is not exhaustive.

H. QUANTITATIVE COMPARISON OF COUNTERFACTUAL GENERATING APPROACHES

Statistical significance tests of the counterfactual generating approaches in terms of counterfactual performances using validity, similarity, and sparsity were performed. The validity property is assessed using the probability of counterfactual $\mathbf{c}$ belonging to the non-default class, i.e., $P(g(\mathbf{c}) = 0)$. The higher the probability, the more valid the counterfactual is. The similarity property is assessed using the distance between the record of interest $\mathbf{x}$ and the generated counterfactual $\mathbf{c}$, i.e., dist($\mathbf{x}$, $\mathbf{c}$). The lower the distance, the more similar $\mathbf{x}$ and $\mathbf{c}$ are. The sparsity property is measured by using the following formula

$$\text{sparsity} = \frac{\sum_{(c^i = f^i)} 1}{d}, \quad \forall i = \{1, 2, \cdot, d\}. \tag{19}$$

Eq. (19) counts the number of features that have the same values between $\mathbf{x}$ and $\mathbf{c}$. The higher the sparsity value, the more sparse the generated counterfactual is. This study is compared quantitatively to Dastile et al. [6], Sharma et al. [46], and Wachter et al. [50] using counterfactual properties (i.e. similarity, validity, and sparsity). There were 30 defaulted records that were randomly selected in the dataset, and counterfactuals were generated for the randomly selected records using approaches from Dastile et al. [6], Sharma et al. [46], and Wachter et al. [50]. Figure 3 shows the results for each of the counterfactual properties.

By looking at Figure 3, it is not clear how to tell which method performs better when it comes to counterfactual properties. Hence, a statistical significance test is required to assess if the methods perform differently from each other. The statistical significance test is performed using Friedman test. For Friedman test, there are three hypotheses that correspond to each of the counterfactual properties: Hypothesis-1)

1) For validity;
   - H_0: The counterfactual generating approaches do not differ in how they rank counterfactual performances for validity.
   - H_1: At least one counterfactual generating approach differs in how it ranks counterfactual performances for validity.
2) For similarity;
   - H_0: The counterfactual generating approaches do not differ in how they rank counterfactual performances for similarity.
   - H_1: At least one counterfactual generating approach differs in how it ranks counterfactual performances for similarity.
3) For sparsity;
   - H_0: The counterfactual generating approaches do not differ in how they rank counterfactual performances for sparsity.
   - H_1: At least one counterfactual generating approach differs in how it ranks counterfactual performances for sparsity.

The obtained p-values for validity, similarity, and sparsity are 5.09e-11, 3.19e-12, and 1.91e-08, respectively. Given that all these p-values fall below the 0.05 significance threshold, we reject the null hypotheses. This suggests that at least one counterfactual generating method ranks performances differently for each counterfactual property. To pinpoint which methods show significant differences, a post-hoc Nemenyi

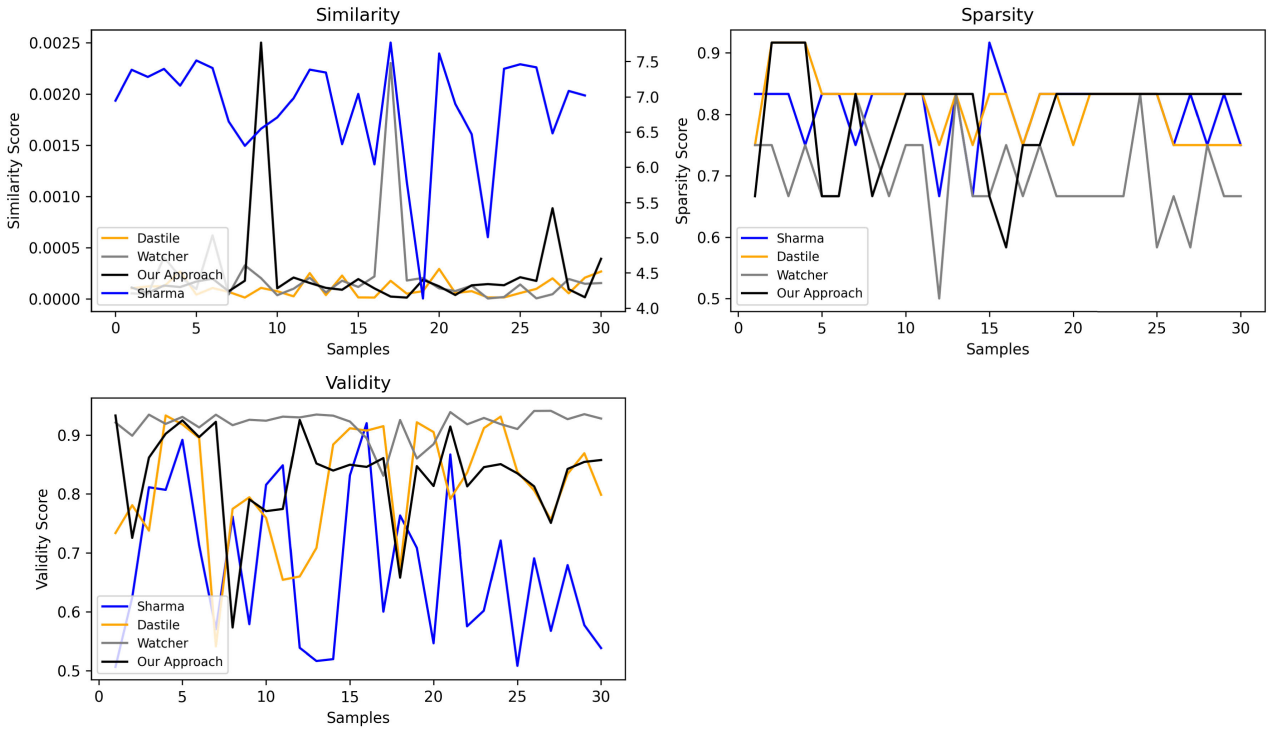

FIGURE 3. Performance analysis of our approach and that of Wachter et al. [50], Sharma et al. [46], and Dastile et al. [6]. Three counterfactual properties (i.e. sparsity, validity, and similarity) were compared using 30 samples that were selected randomly in the dataset.

test is conducted. This test produces a table of p-values. When comparing any two methods, if the intersecting p-value is below the significance level, it indicates a significant difference between their counterfactual performances.

From Table 8, we observe that at intersections involving Wachter et al. [50] with all other methods, p-values remain below 0.01. Given our 0.05 significance level, we deduce that Wachter et al. [50] has counterfactual validity performances that significantly deviate from the rest. This is corroborated by the validity graph in Figure 3, where Wachter et al. [50] registers high validity scores.

For the similarity property, Table 9 indicates that Sharma et al. [46] diverges notably from other methods, as evidenced by their intersection p-values being 0.00. This finding is reflected in the similarity graph within Figure 3, where Sharma et al. [46] shows high similarity scores. It's essential to note that a higher similarity score signifies greater dissimilarity from the original data point. Consequently, in terms of the similarity property, Sharma et al. [46] performs worse than other methods.

When analyzing the sparsity property in Table 10, Sharma et al. [46], Dastile et al. [6], and our proposed method show significant variances from Wachter et al. [50] when tested at the 0.05 significance level.

In summary, the results show that Wachter et al. [50] is more valid but less sparse as evidenced in Figure 3. Conversely, Dastile et al. [6] is more sparse but less valid as depicted in Figure 3. This suggests that there is a trade-off between validity and sparsity. From Figure 3, it's evident that while our proposed method sacrifices validity to some degree, it strikes a good balance between sparsity and validity.

VI. ADVANTAGES AND LIMITATIONS

The proposed approach has several advantages, and these include, model agnostic, transparency of decisions, and applicability to industry. The approach is model-agnostic, meaning it can be applied across different machine learning models without needing customization. A significant advantage is the approach's ability to provide transparency to individuals impacted by machine learning decisions, specifically in the context of loan applications. It not only explains why an application was rejected but also advises on possible recourse actions. This aspect is crucial for ethical AI practices, as it promotes fairness and accountability. Classification tasks involving tabular data are common in many industries, this makes our proposed approach highly relevant and effective for a significant portion of machine learning applications.

While our approach has positive aspects, it also faces certain limitations. Specifically, our method is tailored to tabular data and cannot be easily extended to other data forms, which restricts its applicability in scenarios requiring diverse data types. Furthermore, the assumption that training data remains static over time introduces challenges in maintaining the accuracy of the explanations, as it does not account for the dynamic nature of real-world data which requires model retraining. This makes our approach both data and

TABLE 8. The p-values from the Nemenyi test pertain to the validity property. Intersection values below the significance level of 0.05 indicate a significant performance difference between counterfactual generating methods.

Methods	Sharma et al. [46]	Dastile et al. [6]	Wachter et al. [50]	Our Approach
Sharma et al. [46]	1.00	0.01	0.00	0.00
Dastile et al. [6]	0.01	1.00	0.00	0.90
Wachter et al. [50]	0.00	0.00	1.00	0.00
Our Approach	0.04	0.90	0.00	1.00

TABLE 9. The p-values from the Nemenyi test pertain to the similarity property. Intersection values below the significance level of 0.05 indicate a significant performance difference between counterfactual generating methods.

Methods	Sharma et al. [46]	Dastile et al. [6]	Wachter et al. [50]	Our Approach
Sharma et al. [46]	1.00	0.00	0.00	0.00
Dastile et al. [6]	0.00	1.00	0.84	0.38
Wachter et al. [50]	0.00	0.84	1.00	0.84
Our Approach	0.00	0.38	0.84	1.00

TABLE 10. The p-values from the Nemenyi test pertain to the sparsity property. Intersection values below the significance level of 0.05 indicate a significant performance difference between counterfactual generating methods.

Methods	Sharma et al. [46]	Dastile et al. [6]	Wachter et al. [50]	Our Approach
Sharma et al. [46]	1.00	0.90	0.00	0.90
Dastile et al. [6]	0.90	1.00	0.00	0.90
Wachter et al. [50]	0.00	0.00	1.00	0.00
Our Approach	0.00	0.90	0.00	1.00

model dependent, necessitating careful consideration of its applicability based on the characteristics of the data and the classification models.

VII. CONCLUSION

In this article, we tackled the challenge of making machine learning models more understandable by using counterfactual explanations, which are "what-if" scenarios. We tested three different techniques; genetic algorithm, particle swarm optimization, and Bayesian optimization, to find the best way to create counterfactual explanations. Our tests on the German credit dataset, a common choice for studying credit scoring, showed that our method can generate explanations that meet several important counterfactual properties.

We compared our approach with other methods based on how valid, similar, and sparse the explanations were. Our findings suggest that there is a delicate balance in getting a counterfactual that is valid and sparse. Although our method does prioritize sparsity over absolute validity, it strikes a good balance between the two. The results indicated that the existing state-of-the-art explanation methods struggle to find a good balance between sparsity and validity. By generating counterfactuals that are both valid and sparse, this study provides explanations that are easier for end-users (loan applicants) and stakeholders (lenders) to understand and trust. Simpler explanations that effectively change the outcome can help bridge the gap between AI systems and human decision-makers. With sparser explanations, users are more likely to engage with the AI system and take actionable steps. As a result, users will find the recommendations more manageable and straightforward to implement. As regulations around AI transparency increase (like the EU's GDPR), the ability to provide clear, concise, and effective explanations will be crucial. This study's approach can help organizations comply with such regulations by offering explanations that fulfill legal requirements for explainability. Looking ahead, we are interested in exploring how different features in the data might cause certain outcomes in the explanations we generate.

ACKNOWLEDGMENT

Xolani Dastile would like to thank Prof. Turgay Celik, for making funds available for the Ph.D. study. The authors would like to thank the anonymous reviewers for providing valuable feedback on the initial versions of this article.

REFERENCES

[1] R. Guidotti, "Counterfactual explanations and how to find them: Literature review and benchmarking," *Data Mining Knowl. Discovery*, pp. 1–55, Apr. 2022, doi: 10.1007/s10618-022-00831-6.

[2] S. Verma, J. P. Dickerson, and K. Hines, "Counterfactual explanations for machine learning: A review," 2010, *arXiv:2010.10596*.

[3] A. E. Khandani, A. J. Kim, and A. W. Lo, "Consumer credit-risk models via machine-learning algorithms," *J. Banking Finance*, vol. 34, no. 11, pp. 2767–2787, Nov. 2010.

[4] X. Dastile and T. Celik, "Making deep learning-based predictions for credit scoring explainable," *IEEE Access*, vol. 9, pp. 50426–50440, 2021.

[5] X. Dastile, T. Celik, and M. Potsane, "Statistical and machine learning models in credit scoring: A systematic literature survey," *Appl. Soft Comput.*, vol. 91, Jun. 2020, Art. no. 106263.

[6] X. Dastile, T. Celik, and H. Vandierendonck, "Model-agnostic counterfactual explanations in credit scoring," *IEEE Access*, vol. 10, pp. 69543–69554, 2022.

[7] C. Molnar, *Interpretable Machine Learning*, 2nd ed., 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book

[8] K. Sokol and P. A. Flach, "Counterfactual explanations of machine learning predictions: Opportunities and challenges for AI safety," in *Proc. SafeAI@AAAI*, 2019, pp. 1–4.

[9] S. Goethals, K. S rensen, and D. Martens, "The privacy issue of counterfactual explanations: Explanation linkage attacks," *ACM Trans. Intell. Syst. Technol.*, vol. 14, no. 5, pp. 1–24, 2022.
[10] R. M. Grath, L. Costabello, C. L. Van, P. Sweeney, F. Kamiab, Z. Shen, and F. Lecue, "Interpretable credit application predictions with counterfactual explanations," 2018, *arXiv:1811.05245*.
[11] T. Laugel, M.-J. Lesot, C. Marsala, X. Renard, and M. Detyniecki, "Comparison-based inverse classification for interpretability in machine learning," in *Proc. Int. Conf. Inf. Process. Manag. Uncertainty Knowl.-Based Syst.*, 2018, pp. 100–111.
[12] A. V. Looveren and J. Klaise, "Interpretable counterfactual explanations guided by prototypes," 2019, *arXiv:1907.02584*.
[13] R. R. Fernández, I. M. de Diego, V. Aceña, A. Fernández-Isabel, and J. M. Moguerza, "Random forest explainability using counterfactual sets," *Inf. Fusion*, vol. 63, pp. 196–207, Nov. 2020.
[14] R. Samoilescu, A. V. Looveren, and J. Klaise, "Model-agnostic and scalable counterfactual explanations via reinforcement learning," 2021, *arXiv:2106.02597*.
[15] M. Hashemi and A. Fathi, "Permuteattack: Counterfactual explanation of machine learning credit scorecards," 2020, *arXiv:2008.10138*.
[16] M. Downs, J. Chu, Y. Yacoby, F. Doshi-Velez, and P. WeiWei, "CRUDS: Counterfactual recourse using disentangled subspaces," in *Proc. ICML Workshop Hum. Interpretability Mach. Learn.*, 2020, pp. 1–23.
[17] J. Cito, I. Dillig, V. Murali, and S. Chandra, "Counterfactual explanations for models of code," in *Proc. IEEE/ACM 44th Int. Conf. Softw. Eng. Softw. Eng. Pract. (ICSE-SEIP)*, May 2022, pp. 125–134.
[18] M. Suffian, P. Graziani, J. M. Alonso, and A. Bogliolo, "FCE: Feedback based counterfactual explanations for explainable AI," *IEEE Access*, vol. 10, pp. 72363–72372, 2022.
[19] S.-H. Na, W.-J. Nam, and S.-W. Lee, "Toward practical and plausible counterfactual explanation through latent adjustment in disentangled space," *Expert Syst. Appl.*, vol. 233, Dec. 2023, Art. no. 120982.
[20] C. Fernández-Loría, F. Provost, and X. Han, "Explaining data-driven decisions made by AI systems: The counterfactual approach," 2020, *arXiv:2001.07417*.
[21] M. Förster, P. Hühn, M. Klier, and K. Kluge, "Capturing users' reality: A novel approach to generate coherent counterfactual explanations," in *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, 2021.
[22] A. Ferrario and M. Loi, "The robustness of counterfactual explanations over time," *IEEE Access*, vol. 10, pp. 82736–82750, 2022.
[23] A. Carlevaro, M. Lenatti, A. Paglialonga, and M. Mongelli, "Counterfactual building and evaluation via eXplainable support vector data description," *IEEE Access*, vol. 10, pp. 60849–60861, 2022.
[24] A. C. Bueff, M. Cytryński, R. Calabrese, M. Jones, J. Roberts, J. Moore, and I. Brown, "Machine learning interpretability for a stress scenario generation in credit scoring based on counterfactuals," *Expert Syst. Appl.*, vol. 202, Sep. 2022, Art. no. 117271.
[25] S. Rathi, "Generating counterfactual and contrastive explanations using SHAP," 2019, *arXiv:1906.09293*.
[26] N. Siddiqi, *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring*. Cary, NC, USA: SAS Publishing, 2005.
[27] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1996.
[28] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: An overview," *Soft Comput.*, vol. 22, no. 2, pp. 387–408, Jan. 2018.
[29] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. MHS 6th Int. Symp. Micro Mach. Human Sci.*, Jun. 1995, pp. 39–43.
[30] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Int. Conf. Evol. Comput. World Congr. Comput. Intell.*, May 1998, pp. 69–73.
[31] A. Carlisle and G. Dozier, "An off-the-shelf PSO," in *Proc. Workshop Part. Swarm Optim.*, 2001, pp. 1–6.
[32] J. F. Schutte and A. A. Groenwold, "A study of global optimization using particle swarms," *J. Global Optim.*, vol. 31, no. 1, pp. 93–108, Jan. 2005.
[33] F. vandenBergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.
[34] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, Jan. 2016.
[35] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 24, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2011, pp. 1–7.
[36] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed., Hoboken, NJ, USA: Wiley, 2001.
[37] D. Reynolds, *Gaussian Mixture Models*. Boston, MA, USA: Springer, 2015, pp. 827–832.
[38] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *J. Amer. Stat. Assoc.*, vol. 32, no. 200, p. 675, Dec. 1937.
[39] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.
[40] P. Nemenyi, "Distribution-free multiple comparisons," Ph.D. dissertation, Dept. Math., Princeton Univ., Princeton, NJ, USA, 1963.
[41] D. Dua and C. Graff. (2019). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml
[42] J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Syst. Man, Cybern.*, vol. SMC-16, no. 1, pp. 122–128, Jan. 1986.
[43] A. Hassanat, K. Almohammadi, E. Alkafaween, E. Abunawas, A. Hammouri, and V. B. S. Prasath, "Choosing mutation and crossover ratios for genetic algorithms—A review with a new dynamic approach," *Information*, vol. 10, no. 12, p. 390, Dec. 2019.
[44] A. Kazikova, M. Pluhacek, and R. Senkerik, "How does the number of objective function evaluations impact our understanding of metaheuristics behavior?" *IEEE Access*, vol. 9, pp. 44032–44048, 2021.
[45] J. M. Tomczak and M. Zieba, "Classification restricted Boltzmann machine for comprehensible credit scoring model," *Expert Syst. Appl.*, vol. 42, no. 4, pp. 1789–1796, Mar. 2015.
[46] S. Sharma, J. Henderson, and J. Ghosh, "CERTIFAI: Counterfactual explanations for robustness, transparency, interpretability, and fairness of artificial intelligence models," 2019, *arXiv:1905.07857*.
[47] A. Lucic, H. Oosterhuis, H. Haned, and M. de Rijke, "Actionable interpretability through optimizable counterfactual explanations for tree ensembles," 2019, *arXiv:1911.12199*.
[48] R. Poyiadzi, K. Sokol, R. Santos-Rodriguez, T. De Bie, and P. Flach, "FACE: Feasible and actionable counterfactual explanations," in *Proc. AAAI/ACM Conf. AI, Ethics, Soc.*, Feb. 2020, pp. 344–350.
[49] F. Yang, S. S. Alva, J. Chen, and X. Hu, "Model-based counterfactual synthesizer for interpretation," 2021, *arXiv:2106.08971*.
[50] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR," *Harvard J. Law Technol.*, vol. 31, no. 2, pp. 1–47, 2017.

XOLANI DASTILE received the Ph.D. degree from the University of the Witwatersrand, Johannesburg, South Africa, in 2023. He is a Senior Machine Learning Specialist for a streaming company in South Africa. Previously, he was with major banks in South Africa under credit risk departments as a Quantitative Analyst and a Data Scientist. His research interests include eXplainable Artificial Intelligence (XAI) and causal inference.

TURGAY CELIK received the second Ph.D. degree from the University of Warwick, Coventry, U.K., in 2011. His research interests include computer vision, (explainable) artificial intelligence, (health) data science, data-driven optimal control, and remote sensing. He is an Associate Editor of *BMC Medical Informatics and Decision Making*, IET ELL, IEEE Access, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, and SIVP (Springer).