ERASMUS UNIVERSITY ROTTERDAM
Erasmus School of Economics

Master Thesis Data Science & Marketing Analytics

IBACE: An Algorithm for Instance-Based Actionable Counterfactual Explanations

Name student: Rutger Thijmen Johannes Mostert
Student ID number: 637267

Supervisor: Dr. Eran Raviv
Second assessor: TBD

Date final version: 23-07-2024

Abstract

The increasing utilization of Machine Learning (ML) and Artificial Intelligence (AI) across various industries has led to the development of highly accurate predictive models that support decision-making processes. Despite their accuracy, many of these models function as black-boxes, providing little transparency into their decision-making processes. This lack of transparency can have significant negative consequences, driving the need for eXplainable Artificial Intelligence (XAI) methods. Among the various XAI approaches, Counterfactual Explanations (CEs) are particularly notable for their ability to suggest actionable changes to input variables to achieve a different prediction outcome. This research focuses on instance-based CEs, which are advantageous due to their coherence, plausibility, and model-agnostic nature.

However, existing instance-based approaches often lack actionability, a critical attribute for practical applications. To address this gap, this paper proposes the Instance-Based Actionable Counterfactual Explanations (IBACE) algorithm. IBACE aims to generate CEs that are valid, plausible, sparse, and actionable by leveraging the nearest unlike neighbors approach and integrating principles from multiple existing algorithms. The results demonstrate that IBACE achieves high coverage and computational efficiency, making it practical for real-world applications.

Despite its promising results, the study is limited by its reliance on a single dataset and ruleset, which restricts generalizability. Future research should focus on validating IBACE with diverse datasets and real-world applications and refining the interpretability of features for actionable CEs. This paper contributes to the field of XAI by demonstrating that actionable insights can be effectively integrated into instance-based counterfactual explanations, thereby enhancing the transparency of black-box models and providing actionable insights for data subjects.

Table of Contents

Table of Abbreviations

AI	Artificial intelligence
AUC	Area under the curve
CBCE	Case-based counterfactual explainer
CE	Counterfactual explanation
CRUDS	Counterfactual recourse using disentangled subspaces
FACE	Feasible and actionable counterfactual explanations
IBACE	Instance-based actionable counterfactual explanations
LIME	Local interpretable model-agnostic explanations
ML	Machine learning
NICE	Nearest instance counterfactual explanations
NNCE	Nearest-neighbor counterfactual explainer
NUN	Nearest unlike neighbor
ROC	Receiver operating characteristic curve
SHAP	Shapley additive explanations
XAI	Explainable artificial intelligence
XC	Explanation case

1. Introduction

Machine Learning (ML) and Artificial Intelligence (AI) are increasingly being utilized across various industries to build predictive models that support decision-making processes. These models have achieved high levels of accuracy in numerous applications (Angelov et al., 2021). However, many ML models function as black-boxes, providing predictions without explaining the underlying decision-making process in a way that is understandable to humans. This lack of transparency can have negative consequences (Rudin, 2019). Explanations of ML models are fundamental for both those implementing the models and the individuals affected by the decisions (Guidotti, 2022). As the adoption of ML models has surged due to their high accuracy, so has the need for model explanations. This issue has led to the emergence of the field of eXplainable Artificial Intelligence (XAI) (Angelov et al., 2021).

There are various approaches to explain predictive models. For example, SHapley Additive exPlanations (SHAP) (Lundberg & Lee, 2017) determine the contribution of each feature to a prediction using principles from game theory. Another approach is Local Interpretable Model-agnostic Explanations (LIME) (Ribeiro et al., 2016), which creates a surrogate model around the observation of interest, making it easier to interpret, such as a linear regression.

The method of interest in this paper is called "Counterfactual Explanations" (CEs). A CE is a local explanation, meaning that it explains a single data point of interest. It does so by generating a data point that is similar to the data point of interest, but with a different predicted outcome by the black-box model. In contrast to other explainability methods (such as SHAP and LIME), CEs can provide suggestions to achieve a desired outcome, by suggesting changes to the feature values (Verma et al., 2020). For example, assume an individual who applies for a loan at a bank. If the loan is denied, based on the prediction of a black-box model, CEs could potentially provide the individual with *actionable* feedback. The individual could make changes to their feature values based on the CE, helping them to effectively cross the decision boundary of the black-box model, i.e. receiving the loan. Hence, the question that can be answered by a CE is: "What changes to the input variables would have resulted in a different prediction by the black-box model?". These types of explanations come naturally to humans, as it helps establish a cause-effect relation (Byrne, 2019), where the cause is the feature values, and the effect is the predicted outcome, making them suitable in the realm of XAI. CEs are not limited to the field of finance; they are applicable across various domains that demand actionable insights. In marketing, for instance, CEs can clarify the reasons behind

customer churn and suggest measures for customer retention. They can also identify strategies that could transform a disengaging customer experience into an engaging one, among numerous other potential applications.

There are several strategies for retrieving CEs, categorized into five main approaches. The first is the brute force approach, which involves a grid search on feature values. Secondly, optimization strategies aim to minimize a loss function that accounts for various desirable properties of CEs, which will be detailed in this paper. These optimization strategies work on differentiable models, using the gradients, such as neural networks. Thirdly, heuristic search strategies are used to find good CEs. The remaining two strategies are subcategories of heuristic search. One involves approximating the black-box model with a decision tree and using its structure to find CEs. The other strategy, called instance-based CEs, leverages the most similar observations from the dataset to the observation of interest (Guidotti, 2022). Instance-based CE approaches have desirable properties, particularly when compared to other strategies, such as optimization strategies. They provide CEs based on actual instances in the dataset, ensuring coherence and plausibility. Additionally, instance-based approaches are model-agnostic, meaning they can be applied regardless of the underlying model architecture (Guidotti, 2022). Therefore, this paper will focus on instance-based CEs.

Despite their advantages, the existing instance-based approaches do not incorporate *actionability*. Given that CEs should ideally be *actionable*, this paper aims to develop an instance-based CE algorithm capable of providing *actionable* CEs. Therefore, the research question addressed in this paper is as follows:

How can actionable insights be added to case-based counterfactual explanations?

The research question is answered by presenting the Instance-Based Actionable Counterfactual Explanations (IBACE) algorithm. By leveraging a nearest unlike neighbors approach and integrating concepts from several existing algorithms, IBACE aims to generate CEs that are valid, plausible, sparse, and actionable. The results show that IBACE provides high coverage and computational efficiency, making it practical for real-world applications.

This paper is structured as follows. Chapter 2 presents a literature review, in which first the definition of CEs is provided. Additionally, desirable properties of CEs are discussed and different CE approaches are highlighted. Also, different methodologies to incorporate actionability are

discussed briefly. Chapter 3 introduces and discusses the data used in this study. Chapter 4 outlines the methodology, including the theoretical foundation for the algorithm, the proposed algorithm itself, and the method for testing and benchmarking the algorithm. In Chapter 5, the test results on the performance of IBACE are presented. Chapter 6 summarizes and discusses the results. Chapter 7 provides suggestions for improvements on IBACE, highlights limitations of this research as well as proposing avenues for future research. Chapter 8 concludes this paper.

2. Literature Review

2.1 An Introduction to Counterfactual Reasoning in Machine Learning

The concept of CEs in ML was formally introduced by Wachter, Mittelstadt, and Russell in 2017. In their paper titled "Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR", the use of CEs as a method to provide explanations for automated decisions is discussed. In contrast to existing methods that aim to reveal the internal logic of black-box algorithms, CEs focus on identifying which external factors need to change to achieve a desired outcome, rather than explaining the internal decision-making process. The authors proposed CEs with three main goals in mind. The first objective is for a data subject (i.e. an individual impacted by the decision of the black-box model) to understand why a particular decision is made. The second goal is to be able to contest the decision that is made, as CEs can shed light on the fairness of the decision. Thirdly, it enables the data subject to make adjustments, swaying the black-box prediction in its favor (Wachter et al., 2017). The concept of CEs in ML is also grounded in philosophical and psychological literature. For example, the philosopher Lewis (1973) published an article on the concept of counterfactuals, in which he outlines the core principles of counterfactuals. He formulates the question that is answered by a counterfactual as follows: "If it were the case that …, then it would be the case that …". In the context of ML, the central question that is answered by a CE is: "What changes to the input variables would have resulted in a different prediction by the black-box model?". Note that the answers to these questions do not constitute the entire and only truth. A change in the decision by the black-box algorithm could also possibly have occurred by making different changes to the input variables. The psychologist Byrne (2019) established that CEs can help make decisions made by black-box models intelligible to both model developers and data

subjects, as it can elicit causal reasoning in humans. Counterfactuals are also discussed by Kahneman and Miller (1986), who state that humans naturally construct counterfactual alternatives to reality. In their 1986 paper "Norm Theory: Comparing Reality to Its Alternatives", the authors give the example of an individual who is involved in an accident. This will lead to an analysis of the series of events that resulted in it, and this analysis will in turn involve creating counterfactual alternatives. For these reasons, the concept of CEs in ML has proven to be an interesting avenue for academic research. Most research in this area has focused on classification problems, as this problem is easier to conceptualize and implement compared to regression problems, due to the nature of counterfactual reasoning (Verma et al., 2020). Therefore, this paper will solely deal with a binary classification problem.

2.2 Desirable Counterfactual Explanation Properties

In recent years, many CE approaches for ML have been proposed. Both the review paper of Verma et al. (2020), and the review paper of Guidotti (2022) highlight desirable properties of CEs that are widely used and accepted within the CE literature.

The first, and arguably the most important property of a CE, is *validity*. A CE is valid if changes to the input features actually change the predicted class. If the CE is not valid, it does not contain the necessary feature value changes to change the predicted class.

The second property is *sparsity*. Sparsity refers to the number of features that are changed in the original observation to obtain a CE. Ideally, a CE should change a limited number of feature values in order to be the most effective. This idea stems from social sciences and psychological research, in which it is argued that it is easier for humans to understand shorter explanations, compared to longer explanations (e.g. Miller, 2019).

Thirdly, a CE should be in close *proximity* to the original observation. This means that, given a distance function, changes made to a particular feature value should be as small as possible. This property might consist of a trade-off between *sparsity*. Changing less feature values (i.e. generating a sparse solution) might require larger differences within the features that are changed.

The fourth property is *plausibility*. A CE is considered plausible if it is coherent with the reference population. This means that feature values of the CE should not be outliers compared to the rest of the data. If the CE adheres to this property, it ensures its trustworthiness by making the CE more

realistic. This property is also referred to as closeness to the data manifold. Again, if the CE lies far from the data manifold, it is hard to trust the CE.

The fifth property that a CE should ideally adhere to is *actionability*. In the existing literature, actionability refers to feature values that are either *mutable* or not. If a feature value is mutable, it is considered actionable. If the CE suggests a feature value change, even though the feature value is not mutable in practice, the CE is considered unactionable. For example, the country of origin of an individual is considered immutable, and therefore unactionable.

The sixth desirable CE property is *causality*. Causality refers to the fact that features in a dataset are rarely independent. Usually, changing one feature value also means that another feature value should be changed. For example, if a CE suggests that an individual should have one more year of working experience, the CE should also take into account the effect on the individual's age, which should also increase by a year. Note that when relationships between variables exist, they do not have to be causal. Therefore, the terminology can be confusing, hence in this paper a distinction is made between causal relationships and variable relationships in general.

Even though the last two properties are separated in the existing literature, in this paper, actionability will be referred to as the combination of mutability and causality, as a CE that does not adhere to both mutability and causality cannot be considered actionable in reality.

2.3 Desirable Counterfactual Explainer Properties

Next to desirable properties for a CE, the counterfactual explainer should ideally also adhere to multiple properties. First, the explainer should be *efficient*, in order to be useful in real life applications. Secondly, the explainer should be *stable*. This means that if there are similar observations of interest, for which a CE should be constructed, the CEs for both observations should also be similar. This property is also referred to as *robustness*. Thirdly, the explainer should ideally be *model-agnostic*. If the explainer is model-agnostic, it means that it will work with different types of ML models, regardless of their underlying architecture. If an explainer needs access to the internals of a ML model, it is considered to be a model-specific approach. This can for example be the case when the explainer needs a differentiable model (e.g. neural networks) when it is based on gradients. Lastly, the explainer should ideally be able to return *multiple* and *diverse* CEs. This could potentially help the data subject, as they might have a preference for different types of explanations.

2.4 Instance-Based Counterfactual Explanations Algorithms

There exist multiple instance-based CE algorithms, that each focus on different desirable CE properties. The first algorithm that will be discussed is the Nearest-Neighbor Counterfactual Explainer (NNCE) algorithm. This algorithm is based on nearest-neighbor classifiers, and works as follows. First, the observation of interest (i.e. the observation for which a CE should be constructed, or x) is determined. Then, all observations in the dataset that have the opposite predicted class are selected. The distances between the observation of interest and all other selected observations, from the dataset from which the original observation is taken, are calculated. Lastly, the closest observation to the original observation for which holds that the predicted class is not equal to the predicted class of the observation of interest is returned as a CE (or x'). This observation can also be referred to as the nearest unlike neighbor (NUN). This algorithm takes into account the validity and proximity properties. Wexler et al. (2020) present the What-If tool, which is a visualizer for CEs for small datasets. The counterfactual selection is performed using the NNCE algorithm and uses the L1 (also known as the Manhattan distance) or L2 (also known as the Euclidean distance) distance functions. The NNCE algorithm has several shortcomings. First, it can be computationally expensive to compute all distances between the observation of interest and all potential CEs. This can be mitigated by using only a subsample of the dataset, but this increases the probability of the returned CE to be less similar to the observation of interest (Guidotti, 2022). Additionally, the algorithm can only handle continuous features. However, this is a matter of the distance function used, and can be changed based on the needs of the dataset. The NNCE algorithm is the simplest algorithm in the instance-based CEs category, and other instance-based algorithms are based on this algorithm.

Another instance-based CE algorithm is proposed by Keane and Smyth (2020), and is called Case-Based Counterfactual Explainer (CBCE). The first thing to note about this proposed algorithm is that the authors are the only ones who put a hard cap on sparsity. In order for a CE to be good, the authors allow at most two feature value changes in the CE. They show that this hard cap limits the amount of observations for which there is a CE present in the dataset. Therefore, they propose the following algorithm. The first step is to find what the authors call “explanation cases” (XCs). These are pairs of a case (i.e. an observation) and its corresponding good counterfactual (i.e. an observation with at most 2 feature value changes and a different class prediction). Then, the XC whose query is most similar to the observation of interest is identified. Since the query of XC is most similar to the observation of interest, and it has a good CE, the intuition is that the good CE of

XC is a suitable basis for a CE for the observation of interest. The difference-features in XC are solely responsible for the class change and should play an important role in the construction of the CE. The second step is to copy the values of the observation of interest of the match-features in XC into the CE. Similarly, for each of the difference-features in XC, the values from the good counterfactual of XC are copied into the CE. This makes the resulting CE a combination of feature values from the observation of interest and the good counterfactual in XC. In this case, the CE differs from the observation of interest in the same way that the observations in XC differ. When the potential CE is constructed, it still needs to be checked for validity, by making a prediction on this observation with the black-box model. If the CE is valid, it is returned as a good CE. If not, the algorithm moves to the next nearest neighbor of the explanation cases, until a valid and good CE is found. Additionally, the authors argue that CEs need to be explicitly grounded in known cases (i.e. the training data) to ensure plausibility. This algorithm takes into consideration the CE properties of validity, sparsity, and plausibility. Proximity is taken into account by using the nearest neighbors for construction. The idea of moving to a next close neighbor in case the generated potential CE is not valid will be implemented in the proposed algorithm in this paper, and will be discussed in more detail in Chapter 4.

The Feasible and Actionable Counterfactual Explanations (FACE) algorithm is proposed by Poyiadzi et al. (2020), which emphasizes the creation of counterfactual examples that are both feasible and actionable. The algorithm identifies “feasible paths” by calculating shortest path distances using density-weighted metrics, ensuring that the counterfactuals are consistent with the input data distribution. The FACE algorithm constructs a graph over the data points using methods such as KDE, k-NN, or ϵ-graph. Users define the properties of the target counterfactual, including prediction confidence and density thresholds. The algorithm adapts the graph based on these constraints and employs Dijkstra’s algorithm to find the shortest path to target counterfactuals that meet the defined criteria. This approach allows for the generation of counterfactuals that are coherent with the data distribution and aligned with user-defined feasibility and actionability constraints. Actionability in this case refers to mutability, and it is possible to fix immutable features. Notably, in contrast to the other instance-based algorithms, which only work on tabular data, this approach is data-agnostic.

Lastly, the algorithm for Nearest Instance Counterfactual Explanations (NICE) is proposed by Brughmans et al. (2023). First, this algorithm can return the same CEs as NNCE, as it uses the NUN for the construction of CEs. However, this will be the worst case scenario for the NICE algorithm.

The authors also propose three different versions, or reward functions that can be optimized, in order to obtain better CEs. The authors include reward functions for sparsity, proximity, and plausibility. The underlying method used for optimizing each reward function is the same, and will work as follows. First, the NUN to the observation of interest is found in the dataset. Then, hybrid observations between *x* and the NUN are generated by iteratively permuting feature values one after the other. In the first iteration, only one feature value can be changed compared to *x*. On each of the generated hybrid instances, one of the reward functions is calculated. The hybrid instance that obtains the maximum reward is then used to again generate hybrid instances, but this time between the newly generated hybrid and the NUN. This process is repeated until a valid hybrid is found, and returned as a CE. When no valid hybrid is found, the NUN is returned, which is by definition valid, and corresponds to the solution provided by the NNCE algorithm. Because of the reward functions, this algorithm can optimize either for sparsity, proximity, or plausibility. The authors further demonstrate that there is a tradeoff between sparsity or proximity on the one hand, and plausibility on the other hand. The algorithm proposed in this paper will be based on the idea of permuting feature values as used by the NICE algorithm, hence this algorithm will be explained in more detail later.

2.5 Considering Feature Interactions

In the current literature, multiple approaches to deal with variable relationships are proposed, even though they are somewhat limited (Guidotti, 2022). Kanamori et al. (2021) construct, what they call, an interaction matrix. This matrix consists of estimated linear causal models, one for each causal relationship. Consider a causal relationship between feature *i* and *j*, where a change in *i* has a causal effect on *j*. If in the optimization technique used by the authors a change occurs to feature *i*, a change is made to feature *j* in accordance with the causal relationship estimated by the linear causal model.

Karimi et al. (2021b) use a similar approach, where the authors assume the existence of a known structural causal model, that captures all inter-variable causal dependencies. This model is then used in their optimization procedure to find CEs. Note that both these methods only work with variable relationships that are 1) truly causal and 2) occur in a single direction.

Lash et al. (2017b) take a slightly different approach to incorporating variable relationships. The authors divide all features used in the binary classification model into three categories;

unchangeable (U), directly changeable (D) and indirectly changeable (I). When optimizing the feature values, only the value for x_D can be determined and the values of x_I will depend on x_D and x_U . Therefore, the authors model the dependency of x_I on x_D and x_U as $x_I = H(x_D, x_U)$ where the mapping $H : \mathbb{R}^{|D|+|U|} \rightarrow \mathbb{R}^{|I|}$ is assumed to be differentiable. Even though this approach is slightly different from the previous two, it still captures causal relationships occurring in a single direction.

A completely different approach is proposed by Downs et al. (2020). Their algorithm Counterfactual Recourse Using Disentangled Subspaces (CRUDS) first generates a set of potential CEs, by using a Conditional Subspace Variational Autoencoder. Then, based on user given constraints and known variable relationships, the set is filtered based on these constraints. The example given by the authors for a known causal constraint is that more years of education necessitates an increase in age. The example given based on individual end-user preferences is that getting more education is impossible for one individual but not another. These examples are very similar to the variable relationships that will be discussed later in this paper. Additionally, the starting point of having a set of potential CEs corresponds with the proposed algorithm in this paper. Lastly, this approach is very flexible and straightforward to implement. For these reasons, this approach of handling variable relationships will be implemented in the proposed algorithm. Note that the CRUDS algorithm is not model-agnostic. Instead, it only works on differentiable models, such as neural networks, but not on ensemble models like random forests.

Immutable features are taken into consideration by many proposed algorithms, by fixing these feature values while generating CEs, regardless of the approach used (Guidotti, 2022). This method will also be used in the proposed algorithm in this paper.

3. Data

I opted for a widely used dataset within the subject of CEs, which is the South German Credit dataset (Verma et al., 2020). The dataset is retrieved from the “rchallange” package in R (Todeschini & Genuer, 2022), but comes originally from the UCI Machine Learning Repository (Dua & Graff, 2019). This dataset is a transformed version of the Statlog German Credit data, according to the suggestions from Grömping (2019). The dataset was originally donated by the German professor Hans Hofmann via the European Statlog project in 2019, but the data itself is from 1973 to 1975, Germany.

The data is about loan applicants from a large regional bank in southern Germany. Each of the applicants in the dataset has a corresponding credit risk ("good" or "bad", also referred to as good and bad credits), along with other features describing the applicant. Therefore, a binary classification model could be used for predicting whether or not an individual has good or bad credit risk. Based on this prediction, the bank could decide whether or not to issue the loan to the individual.

The dataset is a stratified sample of 1000 credits, where the bad credits (i.e. applicants with a bad credit risk) are oversampled heavily, with 700 good credits and 300 bad credits. In reality, only about 5% of the observations consist of bad credits. A good credit risk means that the customer perfectly complied with the conditions in the contract, while customers with bad credit risk did not (Grömping, 2019). The credit risk is considered the dependent variable in this paper. Next to the dependent variable, there are 20 independent features included in the dataset. Three of these features are quantitative and 17 are categorical. The variable descriptions are given in Table 1 (UCI Machine Learning Repository).

Table 1
Variable descriptions

Variable name	Variable description
Credit Risk (Dependent variable)	Has the credit contract been complied with (good) or not (bad)?
Duration (quantitative)	Credit duration in months
Amount (quantitative)	Credit amount in DM
Age (quantitative)	Age in years
Status (categorical)	Status of the debtor's checking account with the bank
Credit History (categorical)	History of compliance with previous or concurrent credit contracts
Purpose (categorical)	Purpose for which the credit is needed
Savings (categorical)	Debtor's savings

Employment Duration (categorical/ordinal)	Duration of debtor's employment with current employer
Installment Rate (categorical/ordinal)	Credit installments as a percentage of debtor's disposable income
Personal Status Sex (categorical)	Combined information on sex and marital status
Other Debtors (categorical)	Is there another debtor or a guarantor for the credit?
Present Residence (categorical/ordinal)	Length of time (in years) the debtor lives in the present residence
Property (categorical)	The debtor's most valuable property
Other Installment Plans (categorical)	Installment plans from providers other than the credit-giving bank
Housing (categorical)	Type of housing the debtor lives in
Number of Credits (categorical/ordinal)	Number of credits including the current one the debtor has (or had) at this bank
Job (categorical/ordinal)	Quality of debtor's job
People Liable (categorical)	Number of persons who financially depend on the debtor (i.e. are entitled to maintenance)
Telephone (categorical)	Is there a telephone landline registered on the debtor's name?
Foreign Worker (categorical)	Is the debtor a foreign worker?

Table 2 illustrates the summary statistics for the quantitative features. Specifically, the credit duration ranges from a minimum of 4 months to a maximum of 72 months, with a mean duration of 20.9 months. The credit amount spans from a minimum of 250 DM to a maximum of 18,424 DM, with an average amount of 3,271 DM. The age of credit applicants varies from a minimum of 19 years to a maximum of 75 years, with a mean age of 35.5 years. All three variables exhibit right skewness.

Table 2

Summary statistics of quantitative features

Variable name	Min.	1st Quantile	Median	Mean	3rd Quantile	Max.	St. dev
Duration	4	12	18	20.9	24	72	12.1
Amount	250	1366	2320	3271	3972	18424	2822.8
Age	19	27	33	35.5	42	75	11.4

In Table 3, the corresponding categories for each categorical feature, along with summary statistics, are presented. The data exhibits a notable imbalance, with the *credit risk* variable showing 700 good credits and 300 bad credits, despite this variable being heavily oversampled, as previously noted. Most other features also display a lack of balance among their categories. Analyzing the proportion of good and bad credits within each variable's categories shows a relationship between *credit history* and *credit risk*. For instance, 62.5% of observations with the *credit history* category "delay in paying off in the past" are classified as bad credits. Similarly, 57.1% of observations with the credit history category "critical account/other credits elsewhere" are bad credits, though these categories are relatively infrequent.

Table 3

Summary statistics of categorical features

Variable name	Categories	Count	Good	Bad
Credit Risk	good	700	100	0
	bad	300	0	100
Status	no checking account	274	50.7	49.3
	... < 0 DM	269	61.0	39.0
	0 <= ... < 200 DM	63	77.8	22.2
	... >= 200 DM / salary for at least 1 year	394	88.3	11.7
Credit History	delay in paying off in the past	40	37.5	62.5
	critical account/other credits elsewhere	49	42.9	57.1
	no credits taken/all credits paid back duly	530	68.1	31.9
	existing credits paid back duly till now	88	69.7	31.8
	all credits at this bank paid back duly	293	82.9	17.1
Purpose	others	234	62.0	38.0
	car (new)	103	83.5	16.5
	car (used)	181	68.0	32.0
	furniture/equipment	280	77.9	22.1
	radio/television	12	66.7	33.3
	domestic appliances	22	63.6	36.4

	repairs	50	56.0	44.0
	vacation	9	88.9	11.1
	retraining	97	64.9	35.1
	business	12	58.3	41.7
Savings	unknown/no savings account	603	64.0	36.0
	... < 100 DM	103	67.0	33.0
	100 <= ... < 500 DM	63	82.5	17.5
	500 <= ... < 1000 DM	48	87.5	12.5
	... >= 1000 DM	183	82.5	17.5
Employment Duration	unemployed	62	62.9	37.1
	< 1 yr	172	59.3	40.7
	1 <= ... < 4 yrs	339	69.3	30.7
	4 <= ... < 7 yrs	174	77.6	22.4
	>= 7 yrs	253	74.7	25.3
Installment Rate	>= 35	136	75.0	25.0
	25 <= ... < 35	231	73.2	26.8
	20 <= ... < 25	157	71.3	28.7
	< 20	476	66.6	33.4
Personal Status Sex	male : divorced/separated	50	60.0	40.0
	female : non-single or male : single	310	64.8	35.2
	male : married/widowed	548	73.4	26.6
	female : single	92	72.8	27.2
Other Debtors	none	907	70.0	30.0
	co-applicant	41	56.1	43.9
	guarantor	52	80.8	19.2
Present Residence	< 1 yr	130	72.3	27.7
	1 <= ... < 4 yrs	308	68.5	31.5
	4 <= ... < 7 yrs	149	71.1	28.9
	>= 7 yrs	413	70.0	30.0
Property	unknown / no property	282	78.7	21.3
	car or other	232	69.4	30.6
	building soc. savings agr./life insurance	332	69.3	30.7
	real estate	154	56.5	43.5
Other Installment Plans	bank	139	59.0	41.0
	stores	47	59.6	40.4
	none	814	72.5	27.5
Housing	for free	179	60.9	39.1
	rent	714	73.9	26.1
	own	107	58.9	41.1
Number of Credits	1	633	68.4	31.6
	2-3	333	72.4	27.6

	4-5	28	78.6	21.4
	>= 6	6	66.7	33.3
Job	unemployed/unskilled - non-resident	22	68.2	31.8
	unskilled - resident	200	72.0	28.0
	skilled employee/official	630	70.5	29.5
	manager/self-empl./highly qualified employee	148	65.5	34.5
People Liable	3 or more	155	70.3	29.7
	0 to 2	845	69.9	30.1
Telephone	no	596	68.6	31.4
	yes (under customer name)	404	72.0	28.0
Foreign Worker	yes	37	89.2	10.8
	no	963	69.3	30.7

Note. The columns "good" and "bad" describe the percentage of observations with a good and bad credit risk for each observed category.

The relevance of this dataset to the present research is twofold. Firstly, it comprises real data with an intuitive narrative, where the predicted outcome has significant implications for individuals, thereby making the concept of actionability both applicable and comprehensible. Secondly, the dataset features variables that 1) may be immutable for an individual and 2) exhibit interrelationships. For instance, an individual's *Personal Status and Sex* can be considered immutable, as changing between these categories is not easily actionable. One notable relationship in the data exists between *employment duration* and *age*; if a CE suggests increasing the *employment duration* (e.g., from 1 <= ... < 4 years to 4 <= ... < 7 years), it must also consider the corresponding impact on *age*. Another significant relationship is between *present residence* and *age*; thus, if a CE recommends an increase in the *present residence* (e.g., from 1 <= ... < 4 years to 4 <= ... < 7 years), it should also account for the effect on *age*.

4. Methodology

This section outlines the methodology applied in this paper. In the first subsection, the development of the algorithm is explained. This includes an overview of the Nearest Instance Counterfactual Explanations (NICE) algorithm by Brughmans, Leyman, and Martens (2023), which is discussed in detail. This will lay the theoretical foundation for the proposed algorithm, as it will build on the NICE algorithm. This is followed by the limitations that NICE faces in handling certain

scenarios, which the proposed algorithm aims to address and overcome. Then, the proposed algorithm is introduced and laid out in detail, taking into account the approaches from other algorithms as highlighted in Chapter 3. The pseudocode of the proposed algorithm is included in this section. In the next subsection, the binary classification model that is used for the analysis is briefly discussed. The last subsection describes the method of implementing, testing and benchmarking the algorithm.

4.1 Algorithm Development

4.1.1 Nearest Instance Counterfactual Explanations

As mentioned earlier, Nearest Instance Counterfactual Explanations is an algorithm proposed by Brughmans et al. (2023), and makes use of real instances in a dataset to construct the CEs. More specifically, it creates hybrid instances between the observation of interest (x_0) and the NUN (x_n), for which holds: $\hat{y}_0 \neq \hat{y}_n$ and $y_n = \hat{y}_n$. This means that the predicted class between the instance of interest and the nearest neighbor are not the same, making it the nearest *unlike* neighbor. Additionally, the algorithm only considers the nearest unlike neighbor that is classified correctly. Once the NUN is selected, hybrid instances are created by replacing one non-overlapping feature (between x_0 and x_n), in x_0 with the corresponding value from x_n. This is done for every non-overlapping feature. When there are 3 non-overlapping features, 3 hybrid instances are created. For these three hybrid instances, one of their reward functions is calculated. The authors proposed reward functions based on sparsity, proximity, and plausibility. Sparsity relates to the number of features that are changed between the observation of interest and the CE. This is the same as the L0 distance. Proximity refers to the distance between the observation of interest and the CE (L1 and/or L2 distance). Plausibility relates to the closeness of the CE to the data manifold. The hybrid instance that achieves the highest reward is stored as the best hybrid instance, and is tested for validity. If the hybrid instance is valid, it is returned as the CE (x_c), and the algorithm terminates. If the hybrid instance is not valid, the stored best hybrid is used again for replacing non-overlapping features in the same way as described above. Again, the reward function is calculated, and the best hybrid is updated. This process repeats until a valid CE is found and returned. A simplified example of this procedure is shown in Table 4, based on the described dataset. The non-overlapping features are highlighted in **bold**. Also, the best hybrid instance per iteration is highlighted by denoting the reward in **bold**. In this example, x_2 of the second iteration is returned as the CE, as it has the desired predicted class (as well as the highest reward). By definition, the model's last possible hybrid is x_n,

which is also a valid CE. x_n can also be returned as a CE immediately, without taking a reward function into account. The algorithm works on tabular data and can handle any classification model, and requires access to the classification model, for which only the inputs and outputs of the model are needed, making the algorithm model-agnostic.

Table 4

Simplified example of the NICE algorithm

Instance	***Age***	***Employment Duration***	***Savings***	***Other Debtors***	**Predicted Class**	**Reward Function**
x_0	35	<1 year	100	none	Bad	NA
x_n	**36**	<1 year	**120**	**guarantor**	Good	NA
			ITERATION 1			
x_1	**36**	<1 year	100	none	Bad	0.10
x_2	35	<1 year	**120**	none	**Bad**	**0.16**
x_3	35	<1 year	100	**guarantor**	Bad	0.13
			ITERATION 2			
x_1	**36**	<1 year	**120**	none	Bad	0.14
x_2	35	<1 year	**120**	**guarantor**	**Good**	**0.20**

Note. Before the first iteration, the observation of interest and the NUN are selected. In the first iteration, all possible hybrids between x_0 and x_n are generated by only permuting a single feature value. In this example, there are three possible hybrids (x_1, x_2, x_3). The reward function of choice is than calculated and the class predictions are made. In this example, all hybrids are invalid (i.e. the predicted class is "bad"). Therefore, a second iteration is initialized, using the hybrid instance that achieved the highest reward (x_2). All possible hybrids between x_2 and x_n are generated (x_1, x_2). Again, the reward function is calculated and the class prediction is made. In this case, the hybrid x_2 is predicted as "good", and is therefore returned as the CE.

Now, consider another, but similar, simplified example. Imagine that an individual is applying for a loan at a bank, but does not receive it, due to the prediction of the black-box model used by the bank. The individual wants to know why it does not receive the loan, but more importantly, what they can do about it (i.e. what features can be changed to obtain a change in the predicted class). The bank can use the NICE algorithm to find a CE for this case (x_0). Assume the individual is 35 years old, has less than a year of work experience at the current employer, and has $100 of savings. The nearest unlike neighbor in the dataset (x_n) is also 35 years old, has less than a year of work

experience at the current employer, but has $120 of savings. Note that the only feature that differs is the savings amount. In this case, x_n has the opposite predicted class of x_0 by definition, and is therefore returned as a CE (x_c) by the NICE algorithm. This means that; if the individual's savings would have been $120 (instead of $100), they would have received the loan. In this case, the individual could work towards increasing their savings by $20.

4.1.2 Identifying Limitations of NICE

As discussed in the previous section, it is clear that the NICE algorithm only considers the NUN. Remember the last scenario presented in the previous section. Again, assume the same individual (x_0), but now the individual of the nearest unlike neighbor (x_n) is 34 years old, instead of 35 years old. A problem with the NICE algorithm arises when only hybrid instances between x_0 and x_n with age of 34 change the predicted class. If this is the case, the provided CEs are not actionable, as an individual cannot reduce their age. A different problem can arise when the hybrid instances do not adhere to relationships between variables. For example, if the CE suggests to increase age by 1 year (e.g. from 35 to 36 years), and at the same time suggests to increase work experience by 2 years, it creates an impossible scenario. Again, this means that the CEs are not actionable. The proposed algorithm aims to address and overcome these limitations.

4.1.3 Instance-Based Actionable Counterfactual Explanations

The goal of the proposed algorithm is to generate Instance-Based Actionable Counterfactual Explanations (IBACE). Henceforth, it will be referred to as the IBACE algorithm. CEs should be actionable for individuals that are adversely impacted by a prediction outcome of a black-box model (e.g. the individual that does not receive a loan from the bank). The limitations outlined in the previous section are addressed by making the following changes to the NICE algorithm:

1. Impose a ruleset that x_c should adhere to in relation to x_0, making the CEs actionable (e.g. x_c *age* >= x_0 *age*). These rules also capture the interdependencies between variables. This change is based on the CRUDS algorithm, as discussed in Chapter 2.
2. Do not restrict the search space to only the nearest unlike neighbor. As outlined in the previous section, if no valid actionable hybrids can be constructed, the provided CEs will not be useful. If there are no actionable hybrids possible between x_0 and x_n that adhere to the ruleset, the algorithm should move to the second NUN and try again. This procedure can be followed until actionable CEs are found. This change is based on the CBCE algorithm, as discussed in Chapter 2.

3. Do not use the NICE reward functions, as these are not able to capture variable actionability and relationships effectively. Consider the following simple example: x_0 *age* > x_n *age*, and changing the age feature value results in the highest reward. This results in all potential CEs generated by NICE being non-actionable, as x_c *age* will always be smaller than x_0 *age*. Instead, IBACE generates all possible hybrids between x_0 and x_n, but restricts the number of features that at most are changed in x_0. Not restricting the number of features changed results quickly in an unmanageably large number of potential CEs for larger datasets with many features, which will be 2^n - 1 potential CEs, where n is the number of differing features between x_0 and x_n. On the other hand, restricting the number of features changed to k features will result in a number of potential CEs of the following sum of the binomial coefficient: $\sum_{i=1}^{k}\binom{n}{i}$, which in this case is $\sum_{i=1}^{k}(\frac{n!}{i!(n-i)!})$. Note that when $n = k$, this function reduces to 2^n - 1. Not only does this improve computation time for datasets with many features (compared to a 2^n - 1 solution), it is also beneficial for the sparsity of the CE, of which the importance is previously highlighted. Especially for datasets with many numeric features, the chance of having a large n is relatively big even if the instances are relatively close, compared to datasets with mostly categorical features. Additionally, if there are immutable features, n equals the number of differing features minus the number of immutable features. This further reduces the number of hybrids and increases computational efficiency.

The imposed static ruleset based on the dataset used for testing is the following:

1. x_c *personal status sex* = x_0 *personal status sex*
2. x_c *foreign worker* = x_0 *foreign worker*
3. x_c *purpose* = x_0 *purpose*
4. x_c *property* = x_0 *property*
5. x_c *credit history* = x_0 *credit history*
6. x_c *people liable* = x_0 *people liable*
7. x_c *housing* = x_0 *housing*
8. x_c *job* >= x_0 *job*
9. x_c *savings* >= x_0 *savings*
10. x_c *employment duration* >= x_0 *employment duration*
11. x_c *present residence* >= x_0 *present residence*
12. x_c *age* >= x_0 *age*

Rules 1-7 ensure that immutable features are not changed from x_0 to x_c. Rule 8 ensures that an individual does not have to reduce the quality of their job. Rule 9 ensures that an individual does not have to reduce their savings. Rules 10-12 ensure that years cannot be reversed (i.e. no time traveling to the past).

The rules that describe all variable relationships in the dataset are presented in Table 5 and Table 6, where the rules for all possible scenarios in a hybrid instance for both *present residence* (Table 5) and *employment duration* (Table 6) are presented. Two details are key to understanding these rules. Firstly, as *present residence* and *employment duration* are categorical variables in the dataset, x_0 *present residence* and x_0 *employment duration* are assumed to be the average of the category (e.g., if x_0 *present residence* is *1 <= … < 4 yrs*, x_0 *present residence* is assumed to be 2.5 years). Secondly, the rules assume that the individual wants to take action as soon as possible. For example, if the individual is unemployed, and a potential CE also suggests that the individual can stay unemployed, *age* could theoretically increase all the way to the highest *age* in the dataset. However, this is assumed to be undesirable. Therefore, the rule in this scenario is x_c *age* = x_0 *age*.

The last rule regards the number of features that should at most be changed in the CE (k). In their review paper, Verma et al. (2020) conclude that there is no consensus on a hard cap on the number of features that are changed in the current literature, but mention that shorter explanations are more comprehensible to humans. This means that only a small number of features should be changed in the CEs in order to keep them actionable. Keane and Smyth (2020), however, cap the number of changed features to two in their CBCE algorithm. On the other hand, restricting k too much can result in long computation times, as it becomes harder to find CEs. Additionally, this might also reduce the coverage. To balance this trade-off, a sparse solution in this paper is considered to have at most three features changed in the CE (k = 3). Note that the ideal value of k depends on the dataset, computational power, and the individuals' needs and capabilities, and can be experimented with. Keep in mind that the larger the dataset is, the more impact increasing k has on computation times.

Importantly, the provided CEs should adhere to all applicable rules. Note that these rules are highly customizable, and can be changed depending on the dataset, computational power, and the individuals' needs and capabilities. The fact that customized rules can be incorporated into the algorithm is one of its main strengths. This means that x_c can be made actionable, by taking into

account immutable variables, as well as relationships between variables, and by returning sparse solutions.

Table 5

Rules for all scenarios of present residence

Scenario	Rule
x_c *present residence = < 1 yr* **&** x_0 *present residence = < 1 yr*	x_c *age =* x_0 *age*
x_c *present residence =* 1 <= ... < 4 yrs **&** x_0 *present residence = < 1 yr*	x_c *age >=* x_0 *age* **&** x_c *age <=* x_0 *age + 3*
x_c *present residence =* 4 <= ... < 7 yrs **&** x_0 *present residence = < 1 yr*	x_c *age >=* x_0 *age + 4* **&** x_c *age <=* x_0 *age + 6*
x_c *present residence = >=* 7 yrs **&** x_0 *present residence = < 1 yr*	x_c *age >=* x_0 *age + 7*
x_c *present residence =* 1 <= ... < 4 yrs **&** x_0 *present residence =* 1 <= ... < 4 yrs	x_c *age >=* x_0 *age* **&** x_c *age <=* x_0 *age + 1*
x_c *present residence =* 4 <= ... < 7 yrs **&** x_0 *present residence=* 1 <= ... < 4 yrs	x_c *age >=* x_0 *age + 2* **&** x_c *age <=* x_0 *age + 4*
x_c *present residence = >=* 7 yrs **&** x_0 *present residence =* 1 <= ... < 4 yrs	x_c *age >=* x_0 *age + 5*
x_c *present residence =* 4 <= ... < 7 yrs **&** x_0 *present residence =* 4 <= ... < 7 yrs	x_c *age >=* x_0 *age* **&** x_c *age <=* x_0 *age + 1*
x_c *present residence= >=* 7 yrs **&** x_0 *present residence =* 4 <= ... < 7 yrs	x_c *age >=* x_0 *age + 2*
x_c *present residence = >=* 7 yrs **&** x_0 *present residence = >=* 7 yrs	x_c *age >=* x_0 *age*

Note. The scenarios show what the *present residence* categories are for x_c and x_0. For each scenario, a corresponding rule for the effect on *age* is shown. A provided CE by IBACE should adhere to these rules.

Table 6

Rules for all scenarios of employment duration

Scenario	Rule
x_c *employment duration = unemployed* **&** x_0 *employment duration = unemployed*	x_c *age =* x_0 *age*
x_c *employment duration = < 1 yr* **&** x_0 *employment duration = unemployed*	x_c *age =* x_0 *age*
x_c *employment duration =* 1 <= ... < 4 yrs **&** x_0 *employment duration = unemployed*	x_c *age >=* x_0 *age + 1* **&** x_c *age <* x_0 *age + 4*
x_c *employment duration =* 4 <= ... < 7 yrs **&** x_0 *employment duration = unemployed*	x_c *age >=* x_0 *age + 4* **&** x_c *age <=* x_0 *age + 6*
x_c *employment duration =* >= 7 yrs **&** x_0 *employment duration = unemployed*	x_c *age >=* x_0 *age + 7*
x_c *employment duration = < 1 yr* **&** x_0 *employment duration = < 1 yr*	x_c *age =* x_0 *age*
x_c *employment duration =* 1 <= ... < 4 yrs **&** x_0 *employment duration = < 1 yr*	x_c *age >=* x_0 *age* **&** x_c *age <=* x_0 *age + 3*
x_c *employment duration =* 4 <= ... < 7 yrs **&** x_0 *employment duration = < 1 yr*	x_c *age >=* x_0 *age + 4* **&** x_c *age <=* x_0 *age + 6*
x_c *employment duration =* >= 7 yrs **&** x_0 *employment duration = < 1 yr*	x_c *age >=* x_0 *age + 7*
x_c *employment duration =* 1 <= ... < 4 yrs **&** x_0 *employment duration =* 1 <= ... < 4 yrs	x_c *age >=* x_0 *age* **&** x_c *age <=* x_0 *age + 1*
x_c *employment duration =* 4 <= ... < 7 yrs **&** x_0 *employment duration =* 1 <= ... < 4 yrs	x_c *age >=* x_0 *age + 2* **&** x_c *age <=* x_0 *age + 4*
x_c *employment duration =* >= 7 yrs **&** x_0 *employment duration =* 1 <= ... < 4 yrs	x_c *age >=* x_0 *age + 5*
x_c *employment duration =* 4 <= ... < 7 yrs **&** x_0 *employment duration =* 4 <= ... < 7 yrs	x_c *age >=* x_0 *age* **&** x_c *age <=* x_0 *age + 1*
x_c *employment duration =* >= 7 yrs **&** x_0 *employment duration =* 4 <= ... < 7 yrs	x_c *age >=* x_0 *age + 2*
x_c *employment duration =* >= 7 yrs **&** x_0 *employment duration =* >= 7 yrs	x_c *age >=* x_0 *age*

Note. The scenarios show what the *employment duration* categories are for x_c and x_0. For each scenario, a corresponding rule for the effect on *age* is shown. A provided CE by IBACE should adhere to these rules.

Based on these changes, IBACE will work in the following way. The pseudocode of the algorithm is first presented in Table 7, and is followed by a detailed explanation.

Table 7
Pseudocode for IBACE

IBACE (x_0, *dataframe*, *ruleset*, *black-box model*)

Line	Pseudocode
1:	**Input**: x_0: Instance for which to find counterfactual; *dataframe*: All observations including the predicted class; *ruleset*: All rules that counterfactual instance $x_{c,i}$ should adhere to; *black-box model*: Model used to make predictions on the dependent variable
2:	**Output**: X_c: Dataframe containing n counterfactual instances for x_0 ($X_c = \{x_{c,i} \mid i=0,1,\ldots,n\}$)
3:	**Step 1: Calculate distance between x_0 and all other observations**
4:	**for** each observation j **in** *dataframe*:
5:	$dataframe[distance]_j \leftarrow \text{distance}(x_0, x_j)$ --- *Note*: Gower's distance is used
6:	*unlike neighbors* ← **filter**($dataframe$, $class(x_j) \neq class(x_0)$)
7:	**order** *unlike neighbors* by *distance* --- *Note*: *Unlike neighbors* is ordered in ascending order
8:	**Step 2: Initialization**
9:	$i \leftarrow 1$ --- *Note*: i keeps track of the nearest unlike neighbor that is being used to construct hybrid instances
10:	$k \leftarrow 3$ --- *Note*: k determines the maximum number of feature value changes and can be set to any value between 1 and n
11:	$X_c \leftarrow [\,]$ --- *Note*: Empty dataframe that will be filled with counterfactual instances for x_0
12:	**Step 3: Find nearest unlike neighbor**
13:	$x_n \leftarrow unlike\ neighbors_i$
14:	**Step 4: Create hybrid instances**

15: *hybrids* ← all possible hybrids between x_0 and x_n with 1 to k feature value changes

--- *Note*: Method as described in section 4.1.3

16: **Step 5: Check if hybrid instances adhere to ruleset and if hybrid instances are valid**

17: **for** each hybrid instance h **in** *hybrids*:

18: **if** $hybrids_h$ adheres to *ruleset*:

19: $prediction$ ← **predict** $hybrids_h$

--- *Note*: Use the black-box model

20: **if** $class(prediction) \neq class(x_0)$:

21: **append** $hybrids_h$ to X_c

22: **Step 6: Check if X_c contains counterfactual instances**

23: **if** nrow(X_c) = 0:

24: **if** i < nrow(*unlike neighbors*):

25: $i \leftarrow i + 1$

26: **repeat** from Step 3

--- *Note*: With the next closest unlike neighbor

27: **else**:

28: **return** X_c

--- *Note*: In this case no counterfactual instances are found

29: **else**:

30: **return** X_c

Note. The pseudocode demonstrates the sequence of steps taken by the IBACE algorithm to find CEs. A dataframe (X_c) is initialized and filled with CEs that are both valid and actionable, which is returned in the final step of the algorithm.

Firstly, the algorithm requires access to multiple inputs. The dataset with the features that the black-box model is trained on is used to construct the CEs, and should therefore be accessible. Additionally, the black-box model itself needs to be accessible. As potential CEs are generated, their validity needs to be checked, which is done by passing the generated potential CE to the black-box model and predicting the class of the dependent feature. However, IBACE does not need access to the internal structure of the black-box model, making IBACE model-agnostic. Also, a ruleset (such as

the ruleset described above) should be provided, if it is needed to ensure that the CEs will be actionable. Lastly, an instance of interest should be determined (x_0), for which one or more CEs should be found. These inputs are then used to construct the CEs.

The first step in the algorithm is to calculate the distance between x_0 and all other observations in the dataset. In contrast to the paper of Brughmans et al. (2023), in which the Heterogeneous Euclidean Overlap Method is used as the distance metric, in this paper Gower's distance is used. The main reason for this is that the implementation of NICE in the R package "counterfactuals" uses this distance metric. As will be described in section 4.3.1, this package is used for benchmarking purposes. In order to make a fair comparison between the two algorithms, Gower's distance is implemented in this paper. Additionally, Gower's distance is also used in different CE algorithms that can deal with mixed data types, as shown in Verma et al. (2020), making it a valid metric. It should be noted that any metric that can handle mixed variable types can be implemented in IBACE.

The next step is to select all unlike neighbors, meaning all the observations in the dataset for which the predicted class is the desired class, and therefore the opposite class of x_0. Additionally, only correctly classified observations are used, similar to NICE. It should be noted that it is computationally more efficient to swap step one and two, but this is the order implemented in the counterfactuals package. As the Gower's distance also performs scaling, the order of steps impacts the calculated distances. Again, for a fair comparison, the order of steps is kept similar to the approach in the counterfactuals package.

Next, the NUN (x_n) to x_0 is selected using Gower's distance, which will be the first instance that is used for generating the hybrid instances between x_0 and x_n.

Then, the hybrids are generated by generating all possible feature value combinations between x_0 and x_n, while keeping overlapping and immutable features fixed. The maximum total allowed number of features changed in the hybrids compared to x_0 is equal to k (k=3).

These hybrids are then filtered based on the rest of the ruleset (i.e. everything else than the immutable features).

The hybrids that remain are then tested for validity, by making predictions with the black-box model on each of them. If there are valid hybrids, these are returned as CEs, and are by design actionable. The algorithm is terminated. Note that IBACE is capable of returning multiple CEs, as all

possible hybrids are computed at the same time. However, IBACE does not optimize for multiple or diverse CEs.

As soon as there are no hybrid instances left as a result of the ruleset, or when there are no valid hybrids, the algorithm continues the search with the next NUN. This process can repeat as many times as necessary and as there are unlike neighbors. In the case that no valid CE can be constructed with any of the unlike neighbors, the algorithm terminates without returning any CEs. Note that this iterative process can also accommodate the need for a specified number of CEs. However, this potentially comes at the cost of increased computation times if there are not enough hybrids constructed with the unlike neighbor that returns at least one good CE.

Consider the simplified example in Table 8. In the first iteration, the NUN to x_0 is selected. The changes with respect to x_0 are highlighted in **bold**. Then, all possible hybrids are created, while fixing the values of the immutable feature *purpose* and the overlapping feature *employment duration*, and making no more than three changes from x_0. Then, the resulting hybrids are filtered according to the ruleset. As can be seen in Table 8, only x_2 is actionable, as only for this hybrid the relationship between *age* and *employment duration* is adhered to. The validity of this hybrid is checked, and turns out to be invalid (i.e. the predicted class does not change). Therefore, the algorithm does a second iteration. The second closest unlike neighbor is selected and the process repeats. In this iteration, the actionable hybrid is also valid and is returned as a CE.

Table 8
Simplified example of the IBACE algorithm

Instance	***Age***	***Purpose***	***Employment Duration***	***Savings***	**Predicted Class**
x_0	28	Car (new)	<1 year	1000	Bad
			ITERATION 1		
x_n	**29**	**Furniture**	<1 year	**1500**	Good
			Generate hybrids		
x_1	**29**	Car (new)	<1 year	1000	*NA*
x_2	28	Car (new)	<1 year	**1500**	*NA*
x_3	**29**	Car (new)	<1 year	**1500**	*NA*

Filter for actionability					
x_2	28	Car (new)	<1 year	**1500**	Bad
ITERATION 2					
x_n	**30**	**Retraining**	<1 year	**2000**	Good
Generate hybrids					
x_1	**30**	Car (new)	<1 year	1000	*NA*
x_2	28	Car (new)	<1 year	**2000**	*NA*
x_3	**30**	Car (new)	<1 year	**2000**	*NA*
Filter for actionability					
x_2	28	Car (new)	<1 year	**2000**	**Good**

Note. Before the first iteration, the observation of interest is selected. In the first iteration, the NUN is selected and all possible hybrids between x_0 and x_n are generated. In this example, there are three possible hybrids (x_1, x_2, x_3). The generated hybrids are then filtered for actionability, based on the ruleset. The resulting hybrids (in this case only x_2) are tested for validity. In this example, x_2 is invalid (i.e. the predicted class is "bad"). Therefore, a second iteration is initialized, using the second NUN. Again, all possible hybrids between x_0 and the new x_n are generated (x_1, x_2, x_3). These hybrids are then also filtered for actionability. In this example, x_2 is the only actionable hybrid. This hybrid is tested for validity, and has the predicted class "good". Therefore, this hybrid is returned as a CE.

Note that when the dataset has a large amount of observations, the algorithm becomes most computationally intensive in two stages. The first stage is where all distances are calculated to obtain the nearest unlike neighbor, which is the same for the NICE algorithm. The second stage is where the algorithm iterates over multiple unlike neighbors. It could be that the algorithm has to try many unlike neighbors before finding a solution, but this does not have to be the case. However, as suggested by Guidotti (2022) for similar approaches, a sample of the entire dataset can be used to generate CEs. This reduces computational complexity, but on the other hand increases the probability of the CE being more different from x_0 and coverage is possibly reduced. This trade-off should be considered by the user.

4.2 The Binary Classification Model

The IBACE algorithm assumes the existence of a binary classification black-box model that is used for the prediction of a binary dependent variable. Since the IBACE algorithm needs the predictions to be available, a black-box model is trained. Note that for the objectives of this paper, the specific binary classification model does not need to be a black-box model, but could also be a glass-box model, as its sole purpose is to provide the proposed algorithm with predictions of the dependent variable. Therefore, a logistic regression is trained. The dataset is split into a training set (80%) and a testing set (20%). The area under the receiver operating characteristic curve (AUC-ROC), which is calculated based on the results of the testing data, is 0.739. This is above the threshold of 0.6, which is used by Brughmans et al. (2023) as the requirement for the binary classification model. Also, the AUC of 0.739 is similar to the AUC obtained by the models used in their paper on the same dataset (0.718 and 0.758, for their neural network and random forest, respectively). No feature selection is performed, and in order to account for the class imbalance, the classification threshold is moved up from 0.5 to 0.671. This optimal value is found using the AUC-ROC. This means that the probability of a "good" credit score prediction should be at least 67.1% for an observation to be classified as such. As a result, sensitivity is brought down, while specificity is brought up.

4.3 Implementing, Testing and Benchmarking of IBACE

4.3.1 Implementation of IBACE

The implementation of the IBACE algorithm, as well as the training of the black-box model, is performed in the R programming language (version 4.3.1). The used hardware is a laptop with the Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 2.00 GHz, and 8GB of RAM, and runs Windows 11. For the implementation of the IBACE algorithm in R, the following packages are used:

- tidyverse - for data manipulation
- data.table - for faster computation
- cluster - for the Gower's distance metric

For the training of the logistic regression model, the following package is used:

- caret - for training a logistic regression model
- pROC - for calculating the AUC-ROC and finding the optimal classification threshold

For the benchmark algorithm (NICE), the following packages are used:

- counterfactuals - for the implementation of the NICE algorithm

- iml - for storing the data and the trained logistic regression model in an iml object, as needed for the counterfactuals package

4.3.2 Testing IBACE

The test statistic of interest for the IBACE algorithm is coverage (c), which is specified as the percentage of observations with a bad credit risk for which at least one actionable and valid CE is found. The confidence interval for the coverage is obtained using a bootstrap procedure. This involves creating 1,000 bootstrapped datasets by resampling with replacement from the original dataset 1,000 times. For each bootstrapped dataset, the coverage is calculated. The distribution of these 1,000 bootstrapped statistics is then used to construct a confidence interval by taking the 2.5th and 97.5th percentiles for a 95% confidence interval. The following hypotheses for this specific dataset are tested:

- H_0: The true coverage is equal to the observed coverage ($c_T = c_O$)
- H_A: The true coverage is not equal to the observed coverage ($c_T \neq c_O$)

If the observed coverage falls within the 95% confidence interval, the null-hypothesis (H_0) is not rejected. This means that there is no significant evidence to suggest that the true coverage is different from the observed coverage. If the observed coverage falls outside the 95% confidence interval, the null-hypothesis (H_0) is rejected, and the alternative hypothesis (H_A) is accepted, indicating that there is significant evidence to suggest that the true coverage is different from the observed coverage.

Additionally, the distribution of the number of unlike neighbors used to construct the CEs is examined. This is only done for the cases in which the nearest unlike neighbor did not suffice, so where at least the second nearest neighbor is used. This provides an insight into how close the solutions lie to the observation of interest and how useful and efficient it is to use more observations than only the nearest unlike neighbor to find CEs.

4.3.3 Benchmarking IBACE

The IBACE algorithm is benchmarked against the NICE algorithm. First, an example of CEs provided by the NICE algorithm that do not adhere to the ruleset is highlighted and compared to the resulting CE of the IBACE algorithm for the same observation of interest. The counterfactuals package in R is able to provide multiple CEs if they are available. All returned CEs are taken into consideration and are filtered according to the ruleset. Second, the test statistic of interest for the comparison

between IBACE and NICE is the difference in their coverage ($\Delta c = c_{IBACE} - c_{NICE}$), where the coverage of NICE is calculated based on the same bootstrapped datasets as described in the previous section. Again, the distribution of these 1,000 bootstrapped statistics is then used to construct a confidence interval by taking the 2.5th and 97.5th percentiles for a 95% confidence interval. The following hypotheses for this specific dataset are tested:

- H_0: There is no difference in coverage between IBACE and NICE ($\Delta c = 0$)
- H_A: There is a difference in coverage between IBACE and NICE ($\Delta c \neq 0$)

If the 95% confidence interval covers 0, the null-hypothesis (H_0) is not rejected. This means that there is no significant evidence to suggest that there is a difference in coverage between IBACE and NICE. If the 95% confidence interval does not cover 0, the null-hypothesis (H_0) is rejected, and the alternative hypothesis (H_A) is accepted, indicating that there is significant evidence to suggest that the coverage of IBACE is different from the coverage of NICE. If the values within the 95% confidence interval are positive, IBACE has a larger coverage, and if the values within the 95% confidence interval are negative, IBACE has a smaller coverage.

The same procedure is followed for the difference in runtime of both algorithms. The runtime is calculated as the total runtime for each entire bootstrapped dataset in minutes. The test statistic in this case is Δt ($\Delta t = t_{IBACE} - t_{NICE}$), and the following hypotheses for this specific dataset are tested:

- H_0: There is no difference in runtime between IBACE and NICE ($\Delta t = 0$)
- H_A: The difference in coverage between IBACE and NICE is larger than 0 ($\Delta t \neq 0$)

If the 95% confidence interval covers 0, the null-hypothesis (H_0) is not rejected. This means that there is no significant evidence to suggest that there is a difference in runtime between IBACE and NICE. If the 95% confidence interval does not cover 0, the null-hypothesis (H_0) is rejected, and the alternative hypothesis (H_A) is accepted, indicating that there is significant evidence to suggest that the runtime of IBACE is different from the runtime of NICE. If the values within the 95% confidence interval are positive, IBACE has a longer runtime, and if the values within the 95% confidence interval are negative, IBACE has a shorter runtime.

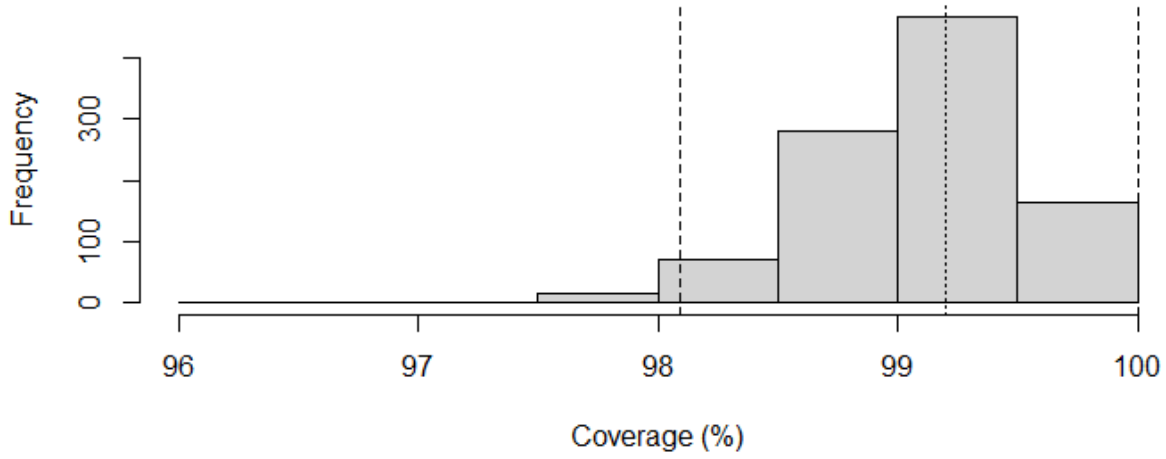
5. Results

5.1 Test Results of IBACE

First, the coverage of IBACE is tested, for which the bootstrap results are presented in Figure 1. The dashed lines show the 95% confidence interval. The lower bound is at 98.1% and the upper bound is at 100%. This means that with a 95% certainty, the coverage of IBACE lies between these values for this particular dataset with this particular ruleset. The observed coverage in the dataset with the particular ruleset is 99.2%, and is presented in Figure 1 with the dotted line. This means that it lies within the confidence interval. Therefore, the null-hypothesis (H_0) is not rejected. This means that there is no significant evidence to suggest that the true coverage is different from the observed coverage.

Figure 1

Bootstrap results for the coverage of IBACE



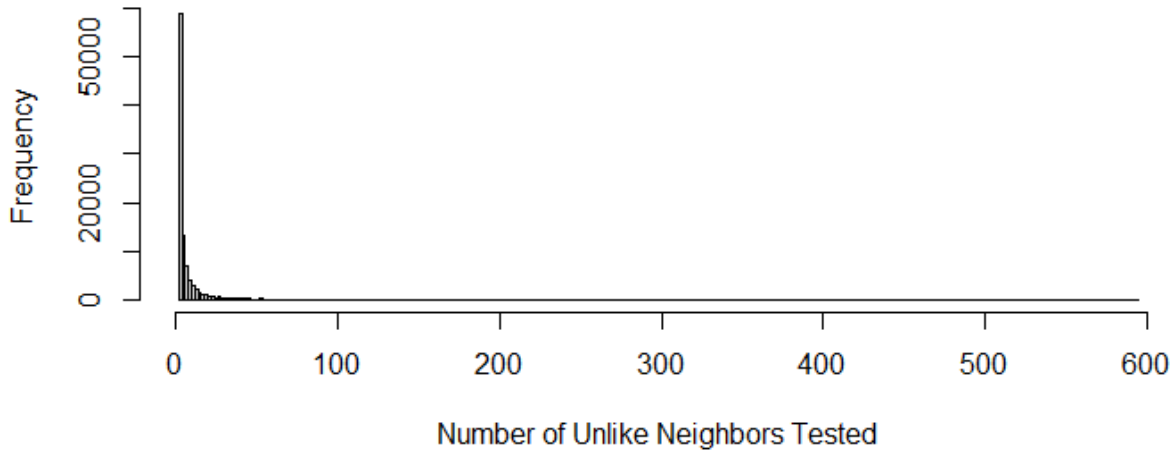
Note. This figure displays the distribution of the bootstrap results for the coverage of IBACE. The coverage of IBACE lies between 98.1% and 100% for 95% of the bootstrapped datasets (dashed lines). The observed coverage in the original dataset is 99.2%, and is presented by the dotted line. The coverage is the percentage of observations in the dataset for which at least a single valid and actionable CE is found.

Additionally, the distribution of the number of unlike neighbors tested to construct the CEs is examined. This is only done for the cases in which the nearest unlike neighbor did not suffice, so where at least the second nearest neighbor is used. The results are presented in Figure 2. This figure clearly illustrates that, in the cases the NUNs do not suffice, the solutions mostly still lie

relatively close to the observations of interest. In 76.7% of these cases, a CE can be constructed with one of the 10 closest unlike neighbors. Including the cases where the NUN suffices, a CE can be constructed for 93.2% of the cases using one of the 10 closest unlike neighbors. Again, these results are specifically for the used dataset with the used ruleset.

Figure 2

The Number of Unlike Neighbors Tested



Note. This figure displays the distribution number of unlike neighbors that had to be tested for the construction of the CEs for cases where the NUN did not suffice. The y-axis shows how many times a particular number of unlike neighbors was used in the construction of the CEs. In most cases, at least one CE could be constructed by using one of the 10 closest unlike neighbors. These results are based on all bootstrapped datasets.

5.2 Benchmark Results of IBACE

An example case for which the NICE algorithm could not find a solution that adheres to all rules is presented in Table 9. The second column represents the observation of interest, and all immutable features are left out of the table. The following columns include a selection of the solutions provided by the IBACE algorithm. From the results it is clear that in this instance, an increase in employment duration would result in a change in the predicted class. For x_1, only employment duration is increased, without increasing the age. However, remember that less than a year's increase would already result in the increase in employment duration. Therefore, this CE is still actionable. In CE x_2, age is also increased, with 3 years. This amount lies within the boundaries of the increase in employment duration, and therefore still adheres to the ruleset. CE x_3 also adheres to the ruleset and shows, in this case, that the bank values the customer having a checking account (albeit with a

negative balance) over having no checking account at all. In total, IBACE returned 16 CEs for the observation of interest, while the NICE algorithm did not find solutions that adhere to the ruleset.

Table 9
Example of IBACE solutions

Feature	x_0	x_1	x_2	x_3
Status	no checking account	-	-	… < 0 DM
Duration	6	-	-	-
Amount	662	-	-	-
Savings	unknown / no savings account	-	-	-
Employment Duration	< 1 yr	1 <= … < 4 yrs	1 <= … < 4 yrs	-
Installment Rate	20 <= … < 25	-	-	-
Other Debtors	none	-	-	-
Present Residence	>= 7 yrs	-	-	-
Age	41	-	44	-
Other Installment Plans	none	-	-	-
Number of Credits	1	-	-	-
Job	unskilled - resident	-	-	-
Telephone	yes	-	-	-

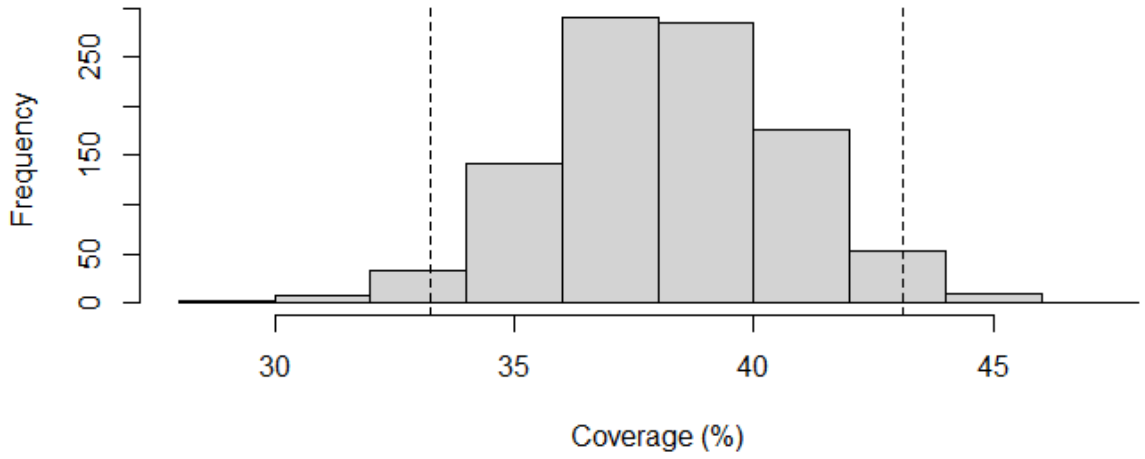
Note. The column of x_0 is the observation of interest with its corresponding feature values. x_1, x_2, x_3 are some of the provided CEs by IBACE for this observation. A "-" indicates that no changes are made.

The bootstrap results for the difference in coverage between IBACE and NICE are presented in Figure 3. Again, the dashed lines show the 95% confidence interval. The lower bound is at 33.2% and the upper bound is at 43.1%. This means that with a 95% certainty, the difference in coverage between IBACE and NICE lies between these values for this particular dataset with this particular ruleset. The 95% confidence interval does not include 0, hence the null-hypothesis (H_0) is rejected, and the alternative hypothesis (H_A) is accepted, indicating that there is significant evidence to suggest that the coverage of IBACE is different from the coverage of NICE. Additionally, the values

within the 95% confidence interval are positive, meaning that IBACE has a larger coverage, compared to NICE.

Figure 3

Bootstrap results for the difference in coverage between IBACE and NICE

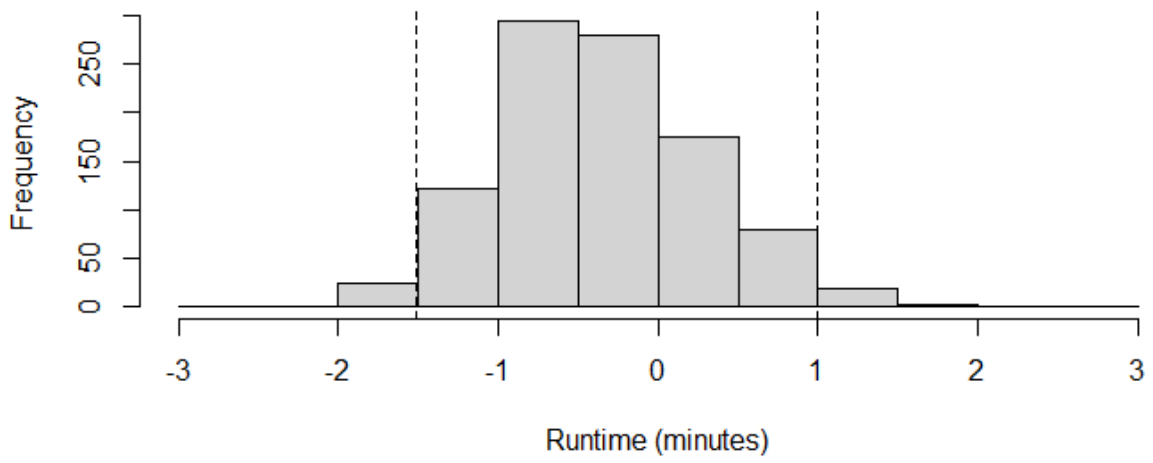


Note. This figure displays the distribution of the bootstrap results for the difference in coverage between IBACE and NICE. For 95% of the bootstrapped datasets, the difference in coverage lies between 33.2% and 43.1% (dashed lines), meaning that in 95% of all bootstrapped datasets, the coverage of IBACE is between 33.2% and 43.1% higher compared to the coverage of NICE. The difference in coverage is the percentage of observations in the dataset for which at least a single valid and actionable CE is found by IBACE minus the percentage of observations in the dataset for which at least a single valid and actionable CE is found by NICE.

The difference in runtime between the two algorithms is tested in the same way. The bootstrap results are presented in Figure 4. The dashed lines show the 95% confidence interval. The lower bound is at -1.5 minutes and the upper bound is at 1 minute. This means that with a 95% certainty, the difference in runtime between IBACE and NICE lies between these values for this particular dataset with this particular ruleset, using the specific hardware and software. The 95% confidence interval does include 0, hence the null-hypothesis (H_0) is accepted. This means that there is no significant evidence to suggest that there is a difference in runtime between IBACE and NICE. The average runtime of IBACE for finding CEs for a single observation of interest is found to be approximately 0.47 seconds.

Figure 4

Bootstrap results for the difference in runtime between IBACE and NICE



Note. This figure displays the distribution of the bootstrap results for the difference in runtime between IBACE and NICE. For 95% of the bootstrapped datasets, the difference in runtime lies between -1.5 minutes and 1 minute (dashed lines) for an entire bootstrapped dataset. The difference in runtime is the total runtime for each entire bootstrapped dataset in minutes of IBACE minus the total runtime of NICE.

6. Discussion

The results of the IBACE algorithm presented in Chapter 5 seem to be rather promising, and provide a relatively large improvement on actionability compared to the NICE algorithm, without a compromise on computation time. The summarized results are as follows:

- The coverage of IBACE lies between 98.1% and 100% at a 95% confidence interval.
- For 76.7% of the cases where no CE can be constructed using the NUN, a CE can be constructed using one of the 10 closest unlike neighbors. Including the cases where the NUN suffices, this value jumps up to 93.2%.
- The difference in coverage between IBACE and NICE lies between 33.2% and 43.1% at a 95% confidence interval, in favor of IBACE.
- There is no statistically significant difference in runtime between IBACE and NICE.

The high coverage of IBACE is a direct result of the construction of the algorithm. Therefore, a high coverage is to be expected. IBACE will continue its search until at least one good CE is found. Though, it should be noted that a sufficiently large and diverse dataset is needed to achieve a high coverage.

The low number of unlike neighbors that need to be tested to construct a CE is also expected. The unlike neighbors are sorted, so that the closest unlike neighbors are tested first. Since these observations are relatively close to the observation of interest, but have the desired predicted class, it is likely that with a minimum amount of feature value changes a CE can be found. This reduces the chance of making unactionable changes, too. Additionally, the iterative process of moving to the next unlike neighbor means that the feature value changes become slightly larger for each tested unlike neighbor, again increasing the chance of finding a CE.

The reason for the increased coverage of actionable CEs is also the design of the algorithm. Actionability can be defined by the end-user by means of the ruleset that can be imposed on the CEs. In the worst case scenario, the coverage of IBACE will be equal to the coverage of NICE, meaning that no valid hybrid can be constructed with any of the unlike neighbors in the dataset.

The fact that there seems to be no difference in computation time between the two algorithms is most surprising, but can have multiple underlying reasons. At a first glance, it seems that the IBACE algorithm should take longer than the NICE algorithm, as it 1) can generate more hybrid instances for each unlike neighbor, and 2) can consider more unlike neighbors than only the NUN. The first reason that no significant difference in computation time is observed could be due to the specific implementation of the NICE algorithm in the counterfactuals package in R. It is possible that the underlying code is not as optimized as the custom code written for this paper. A second reason could be that computing the reward functions for each generated hybrid is relatively computationally expensive. However, it should be noted that the results are dependent on the specific dataset and ruleset used. It is possible that for larger datasets, with more features and more observations, a significant difference in computation time can be observed, in favor of the NICE algorithm.

Additionally, the results depend on the parameter k, which specifies the number of features that are allowed to be permuted at most. On the one hand, decreasing the number k will decrease the number of hybrids created per unlike neighbor used, in turn decreasing the computation time for each iteration. On the other hand, decreasing k also reduces the probability of finding CEs, prompting the algorithm to try more unlike neighbors, which in turn increases computation time. Therefore, the optimal value of k does not only depend on the actionability criteria as mentioned before, but also on the dataset. This parameter should therefore be experimented with in real-life applications in order to make the algorithm as useful as possible.

There also exists a trade-off between the complexity of the ruleset and the coverage. The more complex and restrictive the ruleset, the harder it becomes to find CEs. Again, the exact impact depends on the dataset that IBACE is used on and the ruleset that is imposed.

7. IBACE Improvements, Limitations and Future Research

The IBACE algorithm as proposed in this paper might be improved upon in terms of computation time. For example, instead of generating all possible hybrids at once, it could be changed by first generating all hybrids with only one permuted feature value. These can then be checked for actionability and validity. If CEs are found, these can be returned and the algorithm can be terminated, eliminating the need for the generation of more hybrids. If no CEs are found, the algorithm can continue by permuting an extra feature value, and repeat the process. Additionally, there could be a cap placed on the number of unlike neighbors that should be tested. As seen from the results, a good CE is mostly found within the first 10 NUNs. For larger datasets, it might be worth incorporating this cap, at the cost of slightly reducing the coverage. This will result in outliers not taking up too much time in real-life situations. The exact number of NUNs that should be tested is highly dependent on the end-user's preference, the computational power that is accessible, and the dataset. Lastly, the order of computing the distances and selecting the unlike neighbors should ideally be swapped.

The proposed algorithm might also be improved in terms of proximity. Even though the provided CEs are relatively close by design, it could still be the case that a closer solution is possible. For example, consider the constructed CE to suggest a decrease of the credit amount of 1,000 DM, as this is the feature value of the NUN, and suppose the CE is both actionable and valid. It could still be the case that decreasing the credit amount by only 500 DM would also result in an actionable and valid CE. A possible solution could be to include a grid search for continuous features, in order to nudge the CE even closer to the decision boundary, reducing the distance between the observation of interest and the CE. The exact implementation might be an interesting avenue for future research.

As can be noted, the biggest limitation of this research is that only a single dataset with a single ruleset is tested. Even though the IBACE algorithm shows its potential, no generalizable statements

can be made. In order to establish the usefulness of this algorithm in real-life applications, it should be tested on more datasets, both on freely available datasets and in real-life applications.

Another consideration for the real-life application of the IBACE algorithm is that for large datasets, it might be time consuming to construct a complete ruleset. Even though a custom ruleset provides a lot of flexibility, it is hard to automate this with, for example, an R package.

A caveat with all CEs is that in order for them to be actionable, the features themselves need to be interpretable. For IBACE it is needed for constructing a ruleset, but it is also needed to know what actions should be taken. For example, if the model makes use of principal components, an increase or decrease in the feature values is not informative enough to prompt specific actions. Therefore, if one wants to utilize CEs in the context of actionability, the features of the underlying black-box model need to be taken into consideration.

8. Conclusions

The use of predictive ML and AI models in decision-making processes is increasingly prevalent across various industries. However, many of these models function as so-called black-boxes, lacking transparency in their prediction mechanisms. In numerous applications, understanding the inner workings of these black-box models is not just useful but often essential. This need has driven significant interest in the field of XAI. Among the various proposed approaches to elucidate black-box models, such as SHAP and LIME, CEs stand out for their ability to specify changes to input variables that would lead to different model decisions. This is particularly applicable to binary classification models.

This research identifies instance-based CEs as possessing unique advantages over other CE methods, such as optimization-based approaches. Instance-based methods rely on actual instances from the dataset, ensuring coherence and plausibility, and they are model-agnostic, making them applicable regardless of the underlying model architecture. However, current instance-based approaches fall short in ensuring that the CEs they provide are actionable, a critical property for practical applicability.

To address this gap, this paper proposes the Instance-Based Actionable Counterfactual Explanations algorithm (IBACE). The primary objective of IBACE is to generate CEs that are not only *valid*, *plausible*, and *sparse* but also *actionable*. By leveraging the nearest unlike neighbors

approach, IBACE ensures that CEs are relatively close to the observation of interest, even though it does not explicitly optimize for *proximity*. IBACE integrates principles from multiple existing algorithms, including NICE, CBCE, and CRUDS, culminating in a model-agnostic solution that meets many of the desired properties of CEs.

The results of this study are promising, demonstrating that IBACE achieves high coverage, effectively providing good CEs for a wide range of observations. Furthermore, the algorithm exhibits computational efficiency, enhancing its practicality for real-world applications.

However, it should be noted that the primary limitation of this study is its reliance on a single dataset and ruleset, restricting generalizability. To validate IBACE 's practical utility, it must be tested on diverse datasets and in real-world applications. Additionally, constructing a comprehensive ruleset for large datasets can be time-consuming, and the interpretability of features remains crucial for actionable CEs. Future research should focus on these aspects to further refine and evaluate the IBACE algorithm.

In conclusion, this research contributes to the field of XAI by addressing the research question: "*How can actionable insights be added to case-based counterfactual explanations?*". Through the development and evaluation of IBACE, this paper has shown that it is indeed possible to create CEs that are actionable while maintaining validity, plausibility, and sparsity. This is achieved by imposing a ruleset that the CEs should adhere to, and by utilizing additional unlike neighbors when the NUN does not suffice. This advancement not only enhances the transparency of black-box models, but also provides data subjects with actionable insights that can be readily applied in real-life applications.

References

Angelov, P. P., Soares, E. A., Jiang, R., Arnold, N. I., & Atkinson, P. M. (2021). Explainable artificial intelligence: an analytical review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *11*(5), e1424.

Byrne, R. M. (2019). Counterfactuals in explainable artificial intelligence (XAI): Evidence from human reasoning. In *IJCAI* (pp. 6276-6282).

Downs, M., Chu, J. L., Yacoby, Y., Doshi-Velez, F., & Pan, W. (2020). Cruds: Counterfactual recourse using disentangled subspaces. *ICML WHI*, *2020*, 1-23.

Guidotti, R. (2022). Counterfactual explanations and how to find them: literature review and benchmarking. Data Mining and Knowledge Discovery, 1-55.

Kahneman, D., & Miller, D. T. (1986). Norm theory: Comparing reality to its alternatives. *Psychological review*, *93*(2), 136.

Kanamori, K., Takagi, T., Kobayashi, K., Ike, Y., Uemura, K., & Arimura, H. (2021). Ordered counterfactual explanation by mixed-integer linear optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 35, No. 13, pp. 11564-11574).

Karimi, A. H., Schölkopf, B., & Valera, I. (2021). Algorithmic recourse: from counterfactual explanations to interventions. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency* (pp. 353-362).

Keane, M. T., & Smyth, B. (2020). Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable AI (XAI). In *Case-Based Reasoning Research and Development: 28th International Conference, ICCBR 2020, Salamanca, Spain, June 8–12, 2020, Proceedings 28* (pp. 163-178). Springer International Publishing.

Lewis, D. (1973). Counterfactuals, Blackwells.

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, *30*.

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

Poyiadzi, R., Sokol, K., Santos-Rodriguez, R., De Bie, T., & Flach, P. (2020). FACE: feasible and actionable counterfactual explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society* (pp. 344-350).

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). " 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).

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature machine intelligence, 1(5), 206-215.

Verma, S., Boonsanong, V., Hoang, M., Hines, K. E., Dickerson, J. P., & Shah, C. (2020). Counterfactual explanations and algorithmic recourses for machine learning: A review. *arXiv preprint arXiv:2010.10596*.

Wachter, S., Mittelstadt, B., & Russell, C. (2017). Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.*, *31*, 841.