Privacy-Preserving Counterfactual Explanations for Federated AI

Sjoerd Berning[1][a], Vincent Dunning[1,2][b], Thijs Veugen[1,2][c] and Kevin Witlox[1][d]
[1]*TNO, The Netherlands*
[2]*University of Twente, Enschede, The Netherlands*
{*first name.last name*}*@tno.nl*

Keywords: XAI, Counterfactual Explanations, Privacy-Enhancing Technologies, Federated Learning

Abstract: As the usage of Artificial Intelligence (AI) for sensitive purposes increases, there is a growing need for privacy-aware explainable AI (XAI) tools. In this paper, we present a *privacy-preserving counterfactual explanation algorithm*. Our starting point is a decision-support model that is able to operate on vertically partitioned datasets, meaning that each party holds a different subset of datapoint attributes. The goal of a counterfactual algorithm is to find, given an *observation*, a datapoint from the (virtual) dataset that is closest to the observation but has a different label. Our algorithm fully preserves the privacy of the n datapoints belonging to the different parties by combining the strengths of *homomorphic encryption* and *secret sharing*. Through a number of experiments, we demonstrate the added value of combining multiple datasets in a realistic scenario and show that the privacy-preserving solution does not affect the accuracy. We fully implement our solution and demonstrate that it scales as to thousands of datapoints.

1 Introduction

More and more solutions are being developed to train machine learning models on datasets that are distributed amongst different parties. For example, Federated Learning (FL) can be used to build a system where various parties each train a model on their own datasets after which a central server aggregates these models (McMahan et al., 2017), (Yang et al., 2019). FL can be enhanced with other privacy-enhancing technologies to further protect the privacy of the participants (Bonawitz et al., 2017). Other options include techniques such as multi-party computation and homomorphic encryption (Yang et al., 2023). This is very beneficial in settings where multiple parties want to obtain a joint Machine Learning (ML) model even though their data is too sensitive to share, such as in healthcare or finance. Additionally, modern ML deployments typically benefit from and even require some form of explainability in applications such as in healthcare (AI Act, 2024), (Van Kolfschooten and Van Oirschot, 2024). We refer to this as eXplainable AI (XAI). For example, a patient who gets a prediction for his risk of developing a disease might want to know why the model thinks the patient is at risk. One promising technique for explaining precisely these questions is counterfactual explanations (Dandl et al., 2020). On a high level, a counterfactual explanation is another datapoint (a person) that is very close to the query subject (the patient) but has a lower risk prediction. The difference between the patient and the counterfactual could then pinpoint how the patient should change to reduce his risk as well.

By combining datasets from multiple sources, we get access to a more diverse set of attributes, which is expected to improve the accuracy and actionability of the explanation. However, these XAI techniques have to deal with the same privacy concerns as the data on which they operate, necessitating privacy-preserving explanation techniques. Previous work has focused on how the counterfactual explanations themselves might leak information about the people on which they are based (Goethals et al., 2023; Berning et al., 2024), but not how to deal with the privacy of the data that is needed to come up with the explanation in case the data are distributed among multiple parties.

In this paper, we extend the state of the art by generating the counterfactual explanations on datasets that are distributed over multiple parties. We focus on a scenario where the data are vertically partitioned, meaning that each party holds a different set of attributes about the same datapoints, patients in our

[a] https://orcid.org/0009-0009-1874-0713
[b] https://orcid.org/0009-0004-1148-3017
[c] https://orcid.org/0000-0002-9898-4698
[d] https://orcid.org/0009-0007-6890-6676

case. This is a natural scenario that occurs for example in healthcare, where a hospital might know the medical history of patients while a supermarket has information about the diet of (a subset of) patients and a gym might have data about their activity. Our motivation is that by combining these different datasets, a more relevant counterfactual can be presented, which we expect to lead to a better and more accurate treatment plan for the patient in practice.

1.1 Problem definition

We consider the problem of generating a counterfactual from vertically distributed data in a privacy-preserving way. Specifically, we take the algorithm given in (Berning et al., 2024) using the Multi-Objective Counterfactual (MOC) method by (Dandl et al., 2020) and adapt it into a privacy-preserving solution.

Let $\mathbf{D}$ denote a dataset with size $|\mathbf{D}| = n$. Let $\mathbf{x}$ be a vector $(x_0, \ldots, x_a)$ representing a datapoint, where x_0 is the identifier of the datapoint and $(x_1, \ldots, x_a)$ are the a attribute values. Let $A = \{1, \ldots, a\}$ be th e set of attribute indices. In general, we will use the notation $[a]$ to denote the set $\{1, \ldots, a\}$ for any integer $a \geq 1$. Let $f(\mathbf{x})$ denote the class label of $\mathbf{x}$.

The challenge is to find the best counterfactual $\mathbf{x} \in \mathbf{D}$ for observation $\mathbf{x}^*$ that minimizes the objective

$$\begin{aligned} o(\mathbf{x}) = \mathbf{x} \in \mathbf{D}\big(& o_P(\mathbf{x}, \mathbf{x}^*) \\ & + o_S(\mathbf{x}, \mathbf{x}^*, S) + o_{Fe}(\mathbf{x}, \mathbf{x}^*, S, F) \\ & + o_F(\mathbf{x}, Y) + o_K(\mathbf{x}, Y, k)\big), \end{aligned}$$

The overall objective function is an aggregation of the objectives Proximity (o_P), Sparsity (o_S), Feasibility (o_{Fe}), Fidelity (o_F) and k-purity (o_K) (Berning et al., 2024). The proximity objective measures general similarity of the attributes between a counterfactual and the observation. The sparsity objective penalizes data points that differ in many different attributes (taking into account a per-attribute threshold according to the sparsity model S). The feasibility objective penalizes data points that differ from the observation in attributes that cannot be changed (i.e. the age of a person) according to the feasibility model F. The fidelity objective penalizes data points that do not have the desired target label Y. Finally the k-purity objective captures how well a datapoint can be anonymized by counting whether the nearest neighbours of the datapoint have the desired target class Y, while k denotes the desired number of neighbours. We assume that the target class is chosen at time that the counterfactual search is performed and is the class that we want the observation to become, so $Y \neq f(\mathbf{x})$). All objectives are quantified between 0 and 1 except for fidelity which is either 0 or ∞. To aid notation later, we will denote the aggregation of all objective scores *except* o_K for a datapoint $\mathbf{x}$ as $o_{\neg K}$.

The data are vertically distributed among m databases, where party p_i, $i \in [m]$ has the attribute indices $A_i \subset A$. We assume each partial database contains the same ordered set of identifiers for the datapoints, but each contains distinct attributes for the same identifiers. There is one party that holds the class label $f(\mathbf{x})$ for each $\mathbf{x} \in \mathbf{D}$.

We do not consider the training of the model, e.g. classification by means of a XGBoostClassifier, which produces the class labels which are to be explained using counterfactuals. For a full privacy-preserving solution, the training of the model would also need to be performed in a privacy-preserving manner. Various methods, such as federating learning, are known to train a model on vertically distributed data in a privacy-preserving way.

1.2 Related work

The field of XAI is evolving and considering all kinds of privacy attacks. A good overview of various privacy attacks and defense mechanisms for explainable AI can be found in (Nguyen et al., 2025). There is a large line of work that aims at protecting the explanations themselves from leaking private information, typically using some form of differential privacy. Harder et al. explored differential privacy to explain classifications using locally linear maps (Harder et al., 2020). Mochaourab et al. consider counterfactual explanations for privacy-preserving support vector machines by means of differential privacy (Mochaourab et al., 2021). Vo et al. make their counterfactuals less privacy-sensitive by discretizing continuous attributes (Vo et al., 2023). Goethals et al. reduce the risks of a specific type of attack on counterfactual explanations, the linkage attack, by using k-anonymous counterfactual explanations (Goethals et al., 2023), a technique that relates to the notion of k-purity (Berning et al., 2024), that we also use in this work. Finally, Bozorgpanah and Torra use differential privacy to achieve a similar goal (Bozorgpanah and Torra, 2024).

Much less attention has been given to the challenge of actually generating (counterfactual) explanations in case the dataset is distributed. In the regime of explanations using feature importance, Jetchev and Vuille (Jetchev and Vuille, 2025) propose XorSHAP, a method to generate SHapley Additive exPlanations (SHAP) on both horizontally and/or vertically partitioned data using multi-party computation.

In the regime of generating explanations using Secure Multi-Party Computation (MPC), the only other

work on counterfactual explanation generation on distributed data is (Molhoek and Van Laanen, 2024). They propose a solution for two parties, each holding a part of the features. Their approach uses synthetic data to generate privacy-preserving explanations which are ranked using MPC to find the optimal explanation.

Our work is, as far as we know, the first paper that combines MPC with counterfactual explanations for arbitrary numbers of parties, to obtain an accurate and secure method for generating counterfactuals from vertically distributed data sources.

1.3 Our contribution

We extend the scientific state-of-the-art on privacy-preserving counterfactuals by developing cryptographic protocols to efficiently compute and minimize a number of objectives for counterfactual explanations over vertically distributed data in a secure way (see Section 3). In order to achieve this, we introduce novel strategies for computing most of the counterfactual objectives without interaction and a strategy for minimizing the interaction needed between the parties to compute the final k-purity objective. In Section 4 we explain the security model and the information we protect. We implement our solution and show its performance and accuracy (see Section 6). We finish with conclusions.

2 Preliminaries

In order to construct a privacy-preserving algorithm, we look at the set of cryptographic techniques known collectively as MPC (ICO, 2023). In this section, we will introduce the techniques and their notation as used in this paper.

2.1 Distributed Homomorphic Encryption

We adopt the additively homomorphic encryption scheme of Paillier (Paillier, 1999). This is a public-key encryption scheme, meaning anyone can use the public key to encrypt messages to obtain a ciphertext while only the party that knows the secret key can decrypt ciphertexts to retrieve the message. Paillier encryption works over inputs in some RSA-modulus n. The additively homomorphic property means that we can perform operations directly on two ciphertexts, each encrypting a value, such that the result is a ciphertext which is a valid encryption of the sum of the two values. Furthermore, we can combine a ciphertext with a scalar such that the result is a new ciphertext which encrypts the product of the known scalar with the secret value.

We will denote a Paillier encryption of $x \in \mathbb{Z}_n$ as $[\![x]\!]$. For ciphertexts $[\![x]\!], [\![y]\!]$ and scalar c, we can use the properties of the Paillier encryption scheme to compute $[\![x]\!] \cdot [\![y]\!] = [\![x+y]\!] \mod n^2$ and $[\![x]\!]^c = [\![x \cdot c]\!] \mod n^2$. When $\mathbf{x}$ is a vector $(\mathbf{x}_1, \ldots, \mathbf{x}_a)$, we denote $[\![\mathbf{x}]\!]$ to mean $([\![\mathbf{x}_1]\!], \ldots, [\![\mathbf{x}_a]\!])$.

To ensure the algorithm supports continuous data, we use fixed point encoding, where a rational number is encoded as an integer and a scaling factor, as in (Sangers et al., 2019). This also allows us to perform division by a public divisor. To divide an encrypted value $[\![x]\!]$ by a scalar c, we compute $[\![x]\!] \cdot \frac{1}{c}$. This is possible if $\frac{1}{c}$ is encoded as a fixed point. Note that multiplication of two fixed point numbers with the same precision results in an output with double the precision. As our algorithm requires a limited number of multiplications, we do not require rounding or truncation.

We use a *distributed* variant of the Paillier encryption scheme (Veugen et al., 2019), where the secret key is secret-shared amongst the participants, and therefore they can only decrypt if they work together. Furthermore, the system by Veugen et al. (Veugen et al., 2019) works by having the participants work together to generate the key pair at the beginning of the protocol instead of relying on a trusted dealer, which ensures that no single party ever knows enough information to decrypt ciphertexts on its own.

2.2 Secret Sharing

The other cryptographic building block we require is secret sharing. In particular, we will use Shamir's secret sharing scheme, which is a *linear* secret sharing scheme defined over some finite field $\mathbb{F}_q$ for prime q. A party who wishes to share a secret x can do so by splitting it up into multiple shares $x_1, x_2, ..., x_m$ and give those to parties $p_1, p_2, ..., p_m$ (keeping one share itself). We will denote the sharing as $\langle x \rangle$. The linearity of the scheme gives us similar properties as the encryption scheme, namely $\langle x \rangle + \langle y \rangle = \langle x+y \rangle \mod q$ and $\langle x \rangle \cdot c = \langle x \cdot c \rangle \mod q$ without any interaction. Furthermore, the scheme allows us to multiply ciphertexts, such that $\langle x \rangle * \langle y \rangle = \langle x \cdot y \rangle \mod q$, at the cost of more communication (rounds) among the parties. This will allow us to for example perform secure comparisons and secure sorting. Finally, the parties can reconstruct the secret by combining sufficient shares of the final result.

3 Secure Counterfactual Explanations

Algorithm 1: Protocol for performing the secure counterfactual explanation algorithm.

1. **Initialisation:**
   (a) Together, the parties $p_1, p_2, ..., p_m$ pick the public parameters and generate the key material for the encryption- and secret sharing schemes.
   (b) Each party p_i sends the encoding schemes (explained in Section 3.1) for their attributes A_i to the query party p_1.
   (c) Party p_1 prepares the query $\mathbf{x}^*$ by preparing the encoded version of $\mathbf{x}^*$ and encrypting both the regular- and the encoded query.
   (d) Party p_1 sends to each party p_i the encrypted values and encoded versions for attributes A_i.
2. **Local Computations:**
   (a) Each party p_i computes the scores $[\![o^i_P(\mathbf{x},\mathbf{x}^*)]\!]$, $[\![o^i_S(\mathbf{x},\mathbf{x}^*)]\!]$, and $[\![o^i_{Fe}(\mathbf{x},\mathbf{x}^*)]\!]$ for each datapoint $\mathbf{x}$ and attributes A_i. The label party p_m additionally computes $[\![o_F(\mathbf{x},\mathbf{x}^*)]\!]$.
   (b) Each party p_i computes the sum of all objective scores to obtain the partial local objective scores $[\![o^i_{\neg K}(\mathbf{x},\mathbf{x}^*)]\!]$ for each $\mathbf{x}$ and sends this back to p_1.
   (c) Party p_1 sums for each datapoint $\mathbf{x}$ the $[\![o^i_{\neg K}(\mathbf{x},\mathbf{x}^*)]\!]$ received from each party p_i to obtain $[\![o_{\neg K}(\mathbf{x},\mathbf{x}^*)]\!]$.
3. **Minimization Search:**
   (a) The parties convert all $[\![o_{\neg K}(\mathbf{x},\mathbf{x}^*)]\!]$ to $\langle o_{\neg K}(\mathbf{x},\mathbf{x}^*)\rangle$ (explained in Section 3.6) s.t. each party p_i now holds the shares $\langle o_{\neg K}(\mathbf{x},\mathbf{x}^*)\rangle_i$.
   (b) Together, the parties use the sorting algorithm (explained in Section 3.6) to find the μ candidates with the lowest score $o_{\neg K}(\mathbf{x},\mathbf{x}^*)$.
4. **K-Purity:**
   (a) Together, the parties compute the k-purity score $\langle o_K(\mathbf{x},\mathbf{x}^*,k)\rangle$ for the μ candidates as in Section 3.7.
   (b) The parties compute $\langle o(\mathbf{x},\mathbf{x}^*)\rangle = \langle o_{\neg K}(\mathbf{x},\mathbf{x}^*)\rangle + \langle o_K(\mathbf{x},\mathbf{x}^*)\rangle$.
   (c) The parties perform another secure sort (explained in Section 3.6) to find the global best counterfactual $\langle \texttt{cf}\rangle$.
5. **Result:**
   (a) Finally, the parties send their shares of $\langle \texttt{cf}\rangle$ to the query party, who can reconstruct $\texttt{cf}$.

We will start this section with an intuitive explanation of the ideas underneath our secure counterfactual algorithm.

A naive approach for a secure counterfactual explanation algorithm would be to generate a virtual, complete database, e.g. by means of secret sharing, joining all vertically distributed databases together. This virtual database could then be used as input to a generic multi-party computation circuit implementing the algorithm.

However, we can leverage the fact that each party can use the properties of the additively homomorphic encryption scheme to locally compute most of the objectives on their own attributes to construct a more efficient solution. Concretely, we can adapt the proximity, sparsity, feasibility and fidelity objectives to be computed locally on encrypted values. The core idea of our solution is that all parties who own part of the attributes (*the data parties*) obtain the encrypted attributes $[\![\mathbf{x}^*_j]\!]$ of the observation attributes $\mathbf{x}^*_j$, $1 \leq j \leq a$ from some party which we refer to as the *query party*. Note that each data party p_i only obtains the encrypted attributes for their subset of attributes A_i. We also introduce the *label party*, one of the data parties, that knows the class for each of the datapoints. Furthermore, we assume that the target class Y can be publicly revealed. After the local computations, we can build a virtual, complete database containing only the local objective scores for all datapoints. This also saves on communication, as the parties now only have to share the sum of their locally computed objective scores instead of values for all the attributes in their datasets.

Now, the parties still have to compute the k-purity objective, which they cannot do on their own. Roughly speaking, they will have to compute, for every datapoint, its distance to all other datapoints and sort the list of distances. This is very expensive and therefore we do not want to do it for the entire dataset. Instead we introduce a parameter μ, which selects only μ *candidate counterfactual explanations* based on the local objectives. Only for these μ candidates, we compute the final k-purity objective after which we return the single best counterfactual from those μ candidates.

To achieve this, the parties convert the encryptions of the objective scores to secret-shares in order to perform a search for the μ candidates with lowest objective measure, compute the k-purity objective for those, add it to the other objective scores and return the overall best counterfactual.

Our protocol works for any number m of parties. W.l.o.g., we assume that p_1 acts as the query party, p_m as the label party and all m parties as data parties, where each party p_i has the subset A_i of attributes.

An overview of the full protocol is given in Algorithm 1. In the next sections, we describe the protocol in more detail. In section 3.1 we describe how attributes are encoded for efficient secure computation. In sections 3.2 through 3.5, we provide exact

definitions of the proximity (3.2), sparsity (3.3), feasibility (3.4) and fidelity (3.5) objectives and their secure implementations. In section 3.6 we describe the minimization search, and in section 3.7 we detail the calculation of the k-purity objective.

3.1 Encoding

A dataset can contain different types of attributes. We consider three attribute types: *Categorical*, *Discrete* and *Continuous*. In this section, we explain how the attribute types are represented and encoded for use in secure computation.

We capture the way that each attribute should be encoded in one *encoding scheme*, which is communicated to the query party. The query sent by the query party will consists of one vector where each of the attributes are encrypted directly ("the regular query"), and one vector where each of the attributes are encoded according to the encoding scheme ("the encoded query"). Which version of the query is used depends on the objective that is computed and will be clear from context.

Encoding continuous attributes A continuous attribute j has a real value in some range $[R'_j, R_j]$. W.l.o.g. we will assume that $R'_j = 0$ to ease notation. The distance between two continuous attribute values $\mathbf{x}_j$ and $\mathbf{x}^*_j$ is defined as

$$\delta(\mathbf{x}_j, \mathbf{x}^*_j)^{\text{con}} = \frac{|\mathbf{x}_j - \mathbf{x}^*_j|}{R_j}.$$

Later, in sections 3.3 and 3.4, we will need to calculate whether the distance exceeds a threshold S_j. To allow this computation to be performed locally, we introduce the following encoding.

The continuous attribute value $\mathbf{x}_j$ is encoded to a binary vector $(\mathbf{x}_{j,1}, \ldots, \mathbf{x}_{j,B})$ where $\mathbf{x}_{j,k} = 1$ if $\mathbf{x}_j$ is contained in bucket $k \in [B]$. The size of a bucket is determined by B and the range $[R'_j, R_j]$ of attribute j. W.l.o.g. we assume $R'_j = 0$ to simplify notation, meaning that a bucket has size R_j/B. We introduce the vector $\mathbf{b}$ to describe the boundaries of the buckets. The encoding function is described as follows:

$$\mathsf{Enc}(\mathbf{x}_j, R_j, B_j) = \left(\mathbf{x}_{j,1}, \ldots, \mathbf{x}_{j,B_j}\right),$$
$$\text{where } \mathbf{x}_{j,k} = \begin{cases} 1, & \text{if} \quad \mathbf{b}_{k-1} \le \mathbf{x}_j < \mathbf{b}_k, \\ 1, & \text{if} \quad \mathbf{x}_j = R_j, \\ 0, & \text{if} \quad \text{otherwise}, \end{cases}$$
$$\text{and } \mathbf{b}_k = \frac{k \cdot R_j}{B_j}.$$

The choice of B_j is a trade-off between overhead and accuracy. A larger B_j results in a higher granularity and thus more accuracy at the cost of additional computational and communication overhead.

Encoding discrete attributes Discrete attributes have integer values in the range $[R'_j, R_j]$. W.l.o.g. we assume values in $[R_j]$ to simplify notation. The distance between two discrete attribute values $\mathbf{x}_j$ and $\mathbf{x}^*_j$ is defined as

$$\delta(\mathbf{x}_j, \mathbf{x}^*_j)^{\text{dis}} = \frac{|\mathbf{x}_j - \mathbf{x}^*_j|}{R_j - 1}.$$

The encoding of a discrete attribute value $\mathbf{x}_j$ is defined as

$$\mathsf{Enc}(\mathbf{x}_j, R_j) = \left(\mathbf{x}_{j,1}, \ldots, \mathbf{x}_{j,R_j}\right),$$
$$\text{where } \mathbf{x}_{j,k} = \begin{cases} 1, & \text{if} \quad \mathbf{x}_j = k, \\ 0, & \text{if} \quad \text{otherwise}. \end{cases}$$

Encoding categorical attributes Categorical attributes have the distance between two categorical attribute values $\mathbf{x}_j$ and $\mathbf{x}^*_j$ defined as

$$\delta_h(x, y) = \begin{cases} 0, & \text{if} \quad x = y, \\ 1, & \text{if} \quad x \neq y, \end{cases}$$
$$\delta(\mathbf{x}_j, \mathbf{x}^*_j)^{\text{cat}} = \delta_h(\mathbf{x}_j, \mathbf{x}^*_j).$$

Categorical attributes are encoded in the same fashion as discrete attributes, assuming the attribute categories are numbered as $1, \ldots, R_j$.

3.2 Proximity

The proximity objective measures the general closeness of the attributes of datapoint $\mathbf{x}$ to the attributes of the observation $\mathbf{x}^*$, where a lower score means a higher proximity. The proximity objective is defined as the inverse of Gower's distance[1]

$$o_P(\mathbf{x}, \mathbf{x}^*) = \frac{1}{|A|} \sum_{j \in A} \delta(\mathbf{x}_j, \mathbf{x}^*_j), \text{ where}$$
$$\delta(\mathbf{x}_j, \mathbf{x}^*_j) = \begin{cases} \delta(\mathbf{x}_j, \mathbf{x}^*_j)^{\text{con}} & \text{if} \quad \mathbf{x}_j \text{ is } \textit{continuous}, \\ \delta(\mathbf{x}_j, \mathbf{x}^*_j)^{\text{dis}} & \text{if} \quad \mathbf{x}_j \text{ is } \textit{discrete}, \\ \delta(\mathbf{x}_j, \mathbf{x}^*_j)^{\text{cat}} & \text{if} \quad \mathbf{x}_j \text{ is } \textit{categorical}. \end{cases}$$

A naive conversion of this objective to a secure computation would require computing the absolute value of an encrypted number (in the case of continuous and discrete attributes), or a secure equality

[1] Gower's distance is a similarity measure where 1 is maximally similar and 0 is maximally dissimilar. We require the opposite.

(in the case of categorical attributes), both of which require additional protocols with communication assuming partial homomorphic encryption.

Instead, we choose to slightly adjust the definition of the distance for continuous attributes and discrete attributes to compute the squared distance instead of the absolute distance, as

$$\delta(\mathbf{x}_j, \mathbf{x}_j^*)^{\text{con}} = \frac{(\mathbf{x}_j - \mathbf{x}_j^*)^2}{R_j^2},$$

$$\delta(\mathbf{x}_j, \mathbf{x}_j^*)^{\text{dis}} = \frac{(\mathbf{x}_j - \mathbf{x}_j^*)^2}{(R_j - 1)^2}.$$

Notice that we can write $(\mathbf{x}_j - \mathbf{x}_j^*)^2 = \mathbf{x}_j^2 - 2\mathbf{x}_j \cdot \mathbf{x}_j^* + (\mathbf{x}_j^*)^2$, letting a party p_i holding $\mathbf{x}_j$ and $[\![\mathbf{x}_j^*]\!]$ locally compute

$$[\![\delta(\mathbf{x}_j, \mathbf{x}_j^*)^{\text{con}}]\!] = \left[\!\!\left[\frac{\mathbf{x}_j^2}{R_j^2}\right]\!\!\right] \cdot [\![\mathbf{x}_j^*]\!]^{\frac{-2\mathbf{x}_j}{R_j^2}} \cdot \left[\!\!\left[\frac{(\mathbf{x}_j^*)^2}{R_j^2}\right]\!\!\right]$$

and similarly for $[\![\delta(\mathbf{x}_j, \mathbf{x}_j^*)^{\text{dis}}]\!]$.

The encryption of $[\![(\mathbf{x}_j^*)^2/R_j^2]\!]$ cannot be computed from $[\![\mathbf{x}_j^*]\!]$ non-interactively (under additively homomorphic encryption), and is required to ensure the result of the objective is constrained to the interval $[0,1]$. Instead, the *query party* (knowing $\mathbf{x}^*$) precomputes this value and sends it in encrypted form to the other parties.

For categorical attributes we use our encoding from Section 3.1 to compute the distance in the encrypted domain as

$$[\![\delta(\mathbf{x}_j, \mathbf{x}_j^*)^{\text{cat}}]\!] = \sum_{k \in [R_j]} [\![\mathbf{x}_{j,k}^*]\!]^{\mathbf{x}_{j,k}}.$$

Using the rewritten calculations, each party p_i can locally compute for their attributes A_i

$$[\![o_P^i(\mathbf{x}, \mathbf{x}^*)]\!] = \prod_{j \in A_i} [\![\delta(\mathbf{x}_j, \mathbf{x}_j^*)]\!]^{\frac{1}{|A|}}.$$

3.3 Sparsity

The sparsity objective measures the number of attributes in which the datapoint $\mathbf{x}$ and the observation $\mathbf{x}^*$ differ, where attribute j is said to differ if the distance $\delta(\mathbf{x}_j, \mathbf{x}_j^*)$ crosses a threshold S_j. The objective is defined as

$$o_S(\mathbf{x}, \mathbf{x}^*, S) = \frac{1}{|A|} \sum_{j \in A} \delta_s(\mathbf{x}_j, \mathbf{x}_j^*, S_j),$$

i.e. the normalized sum over the distance δ_s between $\mathbf{x}_j$ and $\mathbf{x}_j^*$ for each attribute j where distance δ_s is

$$\delta_s(\mathbf{x}_j, \mathbf{x}_j^*, S_j) = \begin{cases} 0, & \text{if} \quad \delta(\mathbf{x}_j, \mathbf{x}_j^*) \leq S_j, \\ 1, & \text{if} \quad \delta(\mathbf{x}_j, \mathbf{x}_j^*) > S_j, \end{cases}$$

and S is a vector of thresholds for the different attributes. For a categorical attribute j we say that $S_j = 0$.

Translating the sparsity objective into a secure computation, each party p_i locally computes

$$[\![o_S^i(\mathbf{x}, \mathbf{x}^*, S)]\!] = \prod_{j \in A_i} \delta_s(\mathbf{x}_j, [\![\mathbf{x}_j^*]\!], S_j)^{\frac{1}{|A|}}.$$

The distance δ_s can be computed by a secure comparison. However, in order to avoid communication, we opt to use the encoding introduced in Section 3.1 as shown in the next paragraphs.

Continuous Recall that the attribute value $\mathbf{x}_j$ can be encoded as an indicator vector $(\mathbf{x}_{j,1}, \ldots, \mathbf{x}_{j,B_j})$ and that $\mathbf{b}$ is the vector of the boundaries of the buckets. Using this encoded representation, the party holding $\mathbf{x}_j$ can locally compute δ_s as

$$[\![\delta_s(\mathbf{x}_j, \mathbf{x}_j^*, S_j)]\!] = \prod_{k \in K} [\![\mathbf{x}_{j,k}^*]\!]^{\frac{1}{|A|}},$$

$$\text{where } K = \left\{ k \in [B_j] \;\middle|\; \begin{array}{l} \mathbf{b}_{k-1} - \mathbf{x}_j > S_j \\ \vee\, \mathbf{x}_j - \mathbf{b}_k > S_j \end{array} \right\}.$$

We sum over the subset K of entries in the indicator vector $\mathbf{x}_{j,1}^*, \ldots, \mathbf{x}_{j,B_j}^*$ whose elements correspond to a bucket that differs with $\mathbf{x}_j$ by at least S_j.

Discrete and Categorical The calculation for both discrete and categorical attributes is the same. Using the encoding of $\mathbf{x}_j^*$ we sum over the entries $\mathbf{x}_{j,k}^*$ where the difference between datapoint $\mathbf{x}_j$ and k is greater than the threshold S_j. For categorical attributes, S_j is 0.

$$[\![\delta_s(\mathbf{x}_j, \mathbf{x}_j^*, S_j)]\!] = \prod_{k \in K} [\![\mathbf{x}_{j,k}^*]\!]^{\frac{1}{|A|}},$$

$$\text{where } K = \left\{ k \in [R_j] \;\middle|\; |\mathbf{x}_j - k| > S_j \right\}.$$

3.4 Feasibility

The feasibility objective measures how feasible it would be to cross the difference from observation $\mathbf{x}^*$ to the data point $\mathbf{x}$. The feasibility measure is given by

$$o_{Fe}(\mathbf{x}, \mathbf{x}^*, S, F) = \frac{1}{|A|} \sum_{j \in A} \delta_s(\mathbf{x}_j, \mathbf{x}_j^*, S_j) \cdot F_j,$$

where the feasibility vector F determines whether it is feasible for the observation to change into the counterfactual: $F_j = 0$, if attribute j could be changed (no penalty), and $F_j = 1$ otherwise.

The feasibility measure is very similar to the sparsity measure, with the same local computations. Given the encoded representation, party p_i can compute their local feasibility score as

$$[\![o^i_{Fe}(\mathbf{x},\mathbf{x}^*,S,F)]\!] = \prod_{j\in A_i} \delta_s(\mathbf{x}_j, [\![\mathbf{x}^*_j]\!], S_j)^{F_j \cdot \frac{1}{|A|}}.$$

3.5 Fidelity

The fidelity objective simply penalizes datapoints whose label is not equal to the target label, i.e. $f(\mathbf{x}) \neq Y$, as these would not result in actual counterfactuals.

We assume the class of all datapoints is locally known (as an additional categorical attribute) to one of the parties, as well as the target class Y. The *query party* can thus locally compute the objective as

$$o^i_F(\mathbf{x},Y) = \delta_h(f(\mathbf{x}),Y)\cdot v.$$

We add a penalty v equal to infinite if any datapoint $\mathbf{x}$ with $f(\mathbf{x}) \neq Y$ is not a counterfactual.

3.6 Selecting candidates

Each party p_i can now compute a local objective score for each datapoint $\mathbf{x}$ as

$$\begin{aligned}[\![o^i_{\neg K}(\mathbf{x},\mathbf{x}^*)]\!] = &[\![o^i_P(\mathbf{x},\mathbf{x}^*)]\!] \cdot [\![o^i_S(\mathbf{x},\mathbf{x}^*,S)]\!] \\ &\cdot [\![o^i_{Fe}(\mathbf{x},\mathbf{x}^*,S,F)]\!] \cdot [\![o^i_F(\mathbf{x},\mathbf{x}^*)]\!].\end{aligned}$$

The parties then exchange their encrypted local partial scores $[\![o^i_{\neg K}(\mathbf{x},\mathbf{x}^*)]\!]$ and accumulate them to obtain $[\![o_{\neg K}(\mathbf{x},\mathbf{x}^*)]\!]$.

The partial score $o_{\neg K}(\mathbf{x},\mathbf{x}^*)$ excludes the k-purity objective o_K. As explained in the overview of section 3, we introduce a parameter μ before continuing the calculation of the k-purity objective. To prevent us from having to calculate o_K for each datapoint, we first find the top-μ candidates based on the partial objective score.

Computing a minimum is preferably computed by means of secret-sharing instead of partial homomorphic encryption, for efficiency reasons. Therefore, we first need to convert these values to secret sharings.

To transform a locally computed encrypted value $[\![y]\!]$ into a jointly held secret-shared value $\langle y\rangle$ with modulus q, where $q > y$, we execute the following protocol.

1. Each party $p_i \in [m]$ generates a random value R_i, such that $R_i \gg y$ (at least κ bits more than y, where κ is the statistical security parameter).
2. Each party $p_i \in [m]$ computes $r_i = R_i \bmod q$, and generates a secret sharing $\langle r_i\rangle$.
3. All parties $p_i \in [m]$ exchange a share of $\langle r_i\rangle$ with each other, and the encryption $[\![R_i]\!]$.
4. The parties compute $[\![R]\!] = [\![\sum_i R_i]\!] = \prod_i [\![R_i]\!]$ and $\langle r\rangle = \sum_i \langle r_i\rangle$.
5. The parties compute $[\![z]\!] = [\![y+R]\!] = [\![y]\!]\cdot[\![R]\!]$ and jointly decrypt it.
6. The parties compute $\langle y\rangle = z \bmod q - \langle r\rangle$

The plain text size of homomorphic encryption is usually large enough to guarantee the lack of carry-overs in the computation of z in step 5. In fact, it allows us to combine multiple conversions into one protocol (Veugen et al., 2025, Section 4).

With this protocol, the parties jointly convert the encrypted scores $[\![o_{\neg K}(\mathbf{x},\mathbf{x}^*)]\!]$ to secret sharings $\langle o_{\neg K}(\mathbf{x},\mathbf{x}^*)\rangle$ for each datapoint $\mathbf{x} \in \mathbf{D}$.

We then sort the list of secret-shared scores using Batcher's odd-even merge sort (Batcher, 1968). When sorting the list of secret-shared scores, we can use the fact that we only need to find the μ candidates to speed up the sorting by ignoring the parts of the list that are guaranteed to be outside of the top μ candidates. In literature, this is typically referred to as *truncated sorting*. Our solution is essentially a secure implementation of the truncated sorting algorithm defined in (Sismanis et al., 2012). In general, we refer to this variant returning only the top-t elements as $\mathsf{min_t}$.

3.7 Purity

The k-purity objective o_K measures the ratio of datapoints in the neighbourhood of datapoint $\mathbf{x}$ that do not have the target label Y.

$$o_K(\mathbf{x},Y,k) = \frac{1}{k}\sum_{n=1}^{k} \delta_h(f(K_n(\mathbf{x})),Y),$$

where k is the number of neighbours and K_n denotes the n^{th} closest neighbour (defined by Gower's distance). This penalizes outliers, datapoints whose neighborhood label does not agree with the class label.

We will compute this objective securely for the list of μ candidates. For each candidate, we need to find the k closest neighbours based on Gower's distance and then count the number of closest neighbours with a different class. Afterwards, the k-purity score for each candidate needs to be added to the partial objective score $o_{\neg K}$. With this final objective score, we perform a last round of truncated sorting to find the overall best counterfactual.

Let index α (and β) denote (the attribute vector of) the α^{th} (and β^{th}) candidate $\mathbf{x}^\alpha$ (and $\mathbf{x}^\beta$) where $1 \leq \alpha,\beta \leq n$. First, the parties construct a $n \times n$ matrix of Gower's distances G where $G_{\alpha,\beta} = o_P(\mathbf{x}^\alpha,\mathbf{x}^\beta)$,

containing the Gower's distance between each pair of datapoints. Using this matrix, the parties calculate the k-neighbourhood for each candidate. The steps are as follows.

1. The parties secret-share and aggregate $\langle G_{\alpha,\beta} \rangle = \sum_{i \in [m]} \langle o_P^i(\mathbf{x}^\alpha, \mathbf{x}^\beta) \rangle$.
2. Using the sorting result of Section 3.6, the matrix is reduced to a $\mu \times n$ matrix where row $1 \leq \alpha \leq \mu$ contains the distances for the α^{th} candidate.
3. The parties compute the top-k smallest distances for all μ candidates with outcome a secret binary matrix $\langle \delta^K_{\alpha,\beta} \rangle$, such that $\sum_\beta \delta^K_{\alpha,\beta} = k$ for all candidates α.
4. The class holder generates the secret-shared column $\langle \delta^H_\beta \rangle$, such that $\delta^H_\beta = \delta_h(f(\mathbf{x}^\beta), Y)$, and distributes the shares.
5. They compute $\langle o_K(\mathbf{x}^\alpha, Y, k) \rangle = \sum_\beta \langle \delta^K_{\alpha,\beta} \rangle \cdot \langle \delta^H_\beta \rangle$ through one inner product for each candidate α.
6. They add $\langle o_K(\mathbf{x}^\alpha, Y, k) \rangle$ to $\langle o_{\neg K}(\mathbf{x}^\alpha, \mathbf{x}^{*\alpha}) \rangle$.

This computation can be entirely done with secret-sharing, which saves conversions.

Finally, the parties perform another truncated sorting of the μ candidate scores to find the overall best counterfactual.

4 Security model and implementation

Our cryptographic protocols are secure in the semi-honest model (Goldreich et al., 1987). This means we assume that the parties are interested to try and learn as much information as possible from the messages they see, but follow the protocol description honestly. Furthermore, only the query party gets to see the attribute values of the observation, while no party sees anything besides the attribute values of the datapoints that they own themselves. Additionally, nobody learns anything about the result of the sorting nor who the μ candidates are for which they are computing the k-purity objective. Also the class labels of the datapoints are only visible to the label party. Only the attributes of the final, best counterfactual explanation are returned to the query party.

5 Dataset and Model

The experiment makes use of the Whitehall II Dataset of roughly 10.000 British civil servants. Out of this dataset, 43 attributes were chosen. These attributes were divided into three main categories for the experiment: hospital attributes containing medical information, supermarket attributes containing dietary information and gym attributes, containing activity-related information. The dataset was accessed through the DPUK platform (Bauermeister et al., 2020). For the prediction task, we used the same AI model as in (Berning et al., 2024). It is a XGBoost model with the goal of predicting whether a person (the query) is at risk of developing diabetes type 2.

For the experiments, we sampled the counterfactuals from a dataset consisting of approximately $n = 128$ datapoints.

6 Experiments

This section dives into the performance of the proposed secure solution. The performance of the solution was tested on its run time and the quality of the counterfactual explanations that it generates.

First, we perform a series of experiments to determine the effect of adding cryptographic tools on the accuracy (quality) of the explanations. Here, we use the aggregated objective scores as a proxy for the quality of the explanation (the *explanation score*). The minimization function searches for the minimum objective score. Therefore, a lower score is associated with a higher quality of explanation. Concretely, we perform two experiments to assess the explanation quality of our solution:

- **Explanation quality by combining multiple datasets:** The key benefit of our solution is that it enables counterfactuals based on multiple datasets, which leads to additional attributes in the explanation and therefore richer, more specific explanations.
- **Effect μ-parameter on the explanation quality:** An important parameter of the secure solution is the μ-value. This cut-off parameter is expected to influence the quality of the explanation. Intuitively, there is a risk of not finding the optimal solution if only a small selection of candidates is made before purity is taken into account.

After this, we evaluate the performance of our solution by developing a proof-of-concept implementation and measuring its runtime over various configurations of the parameters of our solution.

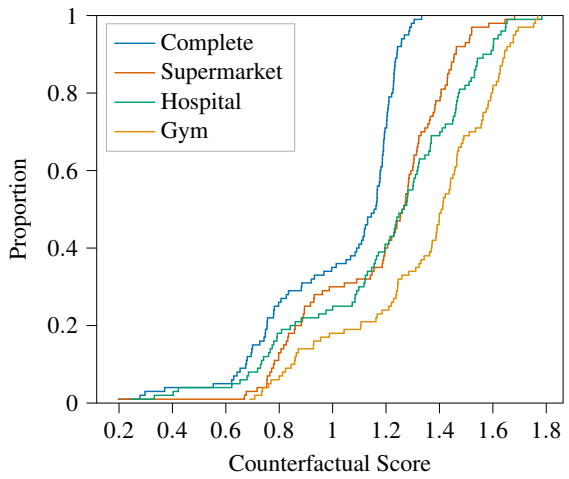

Figure 1: Quality of counterfactual explanations based on different datasets.

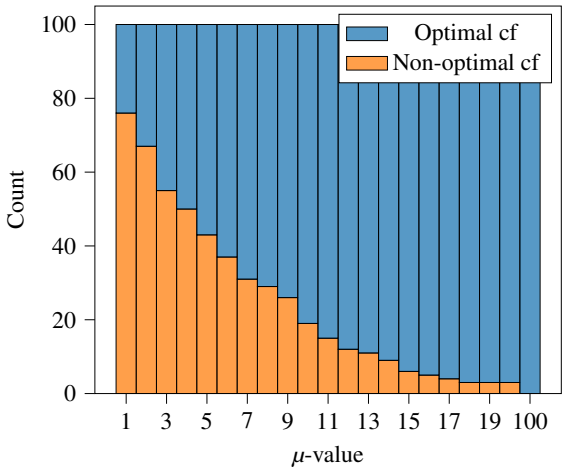

Figure 2: The proportion of optimal counterfactuals per μ-value.

6.1 Benefit of Combining Multiple Databases

In the first experiment, we evaluate the result of combining different datasets. As discussed in Section 5, we have made a selection of three datasets: medical attributes ('hospital', 12 features), dietary attributes ('supermarket', 27 features) and activity-related attributes ('gym', 4 features). We furthermore compare to a 'complete' dataset that combines the other three (and should therefore only exist in encrypted form to preserve the privacy of the parties). As previously mentioned, our assumption is that a larger and more diverse set of features will lead to more useful counterfactual explanations in practice. Therefore, the counterfactual explanations found in the 'complete' dataset are considered as the ground truth. We sample 100 random datapoints that will be used as the observations. For all 100 observations, we perform the counterfactual search in each of the partial datasets separately and evaluate the quality of the found counterfactuals. Figure 1 shows the explanation quality for different datasets. The x-axis shows the score of the counterfactuals while the y-axis depicts the proportion of counterfactuals that have an equal score or lower. By presenting the (accumulative) proportion of counterfactuals that have an equal score or lower, we can easily observe how qualitative the counterfactuals for each dataset are in general. For example, we can observe that 50% of the explanations found in the complete dataset have a quality score of 1.2 or lower, while this is 1.4 when searching for the explanations in only the 'gym' dataset. In general, the figure shows that the hospital and supermarket datasets are creating counterfactuals with a higher quality score compared to the gym dataset, but the optimal counterfactuals are typically only found when combining the three datasets. This is a straightforward result of our assumption that the complete dataset with all features is the ground truth. Therefore, this experiment should be considered as a theoretical motivation for the benefits of combining multiple datasets (using MPC). Our assumption that more features indeed leads to more practical explanations should be validated in a practical experiment.

6.2 Influence of μ on Counterfactual Quality

For this experiment, we again sample 100 random observations from the entire dataset and search for their corresponding counterfactuals for different values of μ. Figure 2 shows how often the secure solution with a particular μ-value was able to find the global best counterfactual explanation. The global best counterfactual was defined as the counterfactual that was found by the algorithm if the μ-value was set to n. The blue bars represent the number of times the optimal counterfactual was found, while the orange bars represent how often a non-optimal counterfactual was retrieved. The figure shows that especially for lower μ-values, a non-optimal counterfactual was often found. For μ-values of 18 and higher (only 14% of the total dataset), the algorithm almost always chose the optimal counterfactual. Therefore, we conclude that the μ-parameter introduced in our solution does not (significantly) harm the quality of the counterfactuals returned by the algorithm if (experimentally) picked to be high enough.

Finally, Figure 3 shows a more detailed compari-

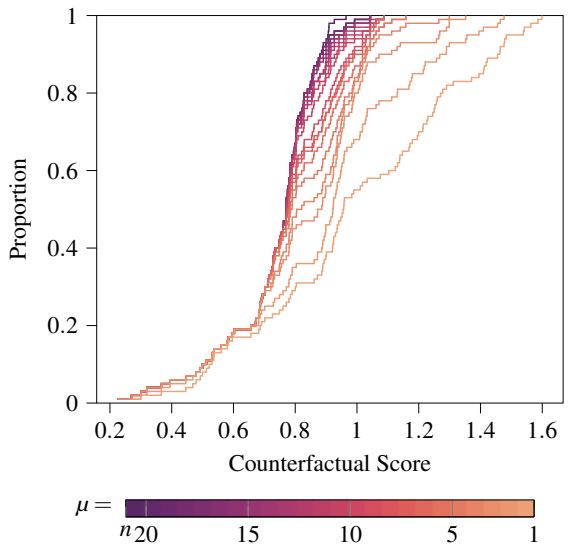

Figure 3: The cumulative proportion of counterfactuals that have at least the counterfactual score stated on the x-axis for different μ-values.

son of how different μ-settings influence the explanation scores of the returned counterfactuals by presenting the actual scores of the non-optimal counterfactuals that are found. For this experiment, explanations were again generated for 100 randomly selected observations per μ-value. Also, a μ-value of n was again included to know what the counterfactual without the μ-parameter is (i.e. the global best counterfactual explanation, in line with the 'complete' dataset in experiment 1). The figure shows on the x-axis the counterfactual score and on the y-axis the proportion of counterfactuals that correspond with at least the score on the x-axis. What can be observed is that for lower μ-values, the curves move to the right, indicating that a larger portion of the generated counterfactuals has a higher explanation score. Especially for lower μ-values, this has a significant impact on the quality of the explanations, while it converges towards almost optimal counterfactuals for higher values of μ.

6.3 Performance

To understand the performance characteristics of the algorithm, we implemented the solution and benchmarked its performance. The solution was implemented in Python using both the Privacy Enhancing Technologies (PET) Lab framework (TNO PET Lab, 2025) and the MPyC framework (Schoenmakers, 2018). The implementation has not been optimized for minimal runtime, but the scaling behaviour should follow those of an optimised version of our algorithm.

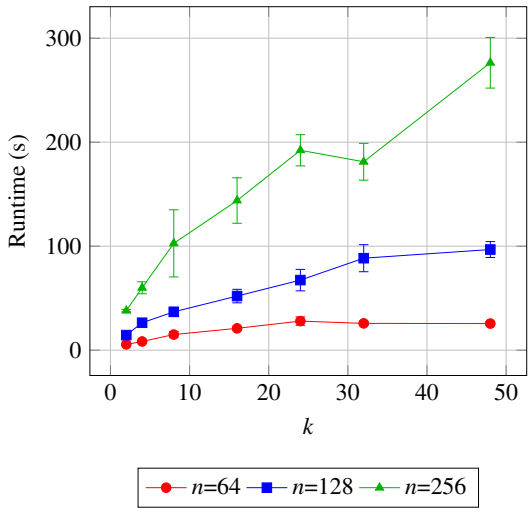

Figure 4: Runtime (s) vs k (#neighbors) with #attributes=3, $\mu = 16$.

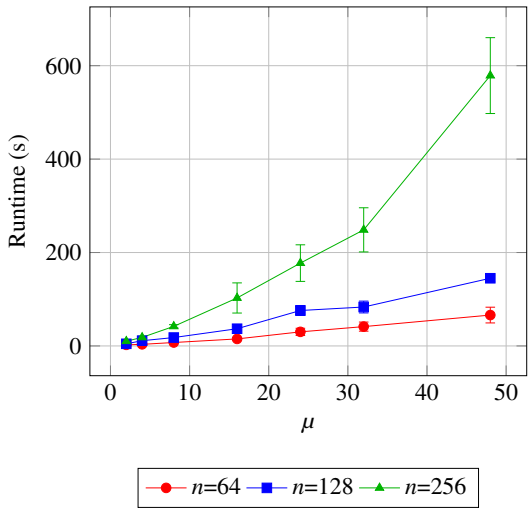

Figure 5: Runtime (s) vs μ (#candidates) with #attributes=3, $k = 8$.

The steps in the algorithm that take up most of the runtime are those involving sorting. To understand the performance of the algorithm, we must consider the scaling of these steps. Batcher's odd-even merge sort has a round complexity of $O(\log^2 n)$ and uses $O(n \log^2 n)$ comparisons (Jönsson et al., 2011). The $\min_t$ variant that returns t items, has a round complexity of only $(\log t)\log n$ using $n \log^2 t$ comparisons.

The algorithm uses sorting in three stages. First, the n objective scores are sorted using the $\min_\mu$ to find the μ candidates. Next, for each of the μ candidates, a $\min_k$ (k-neighborhood) search is performed

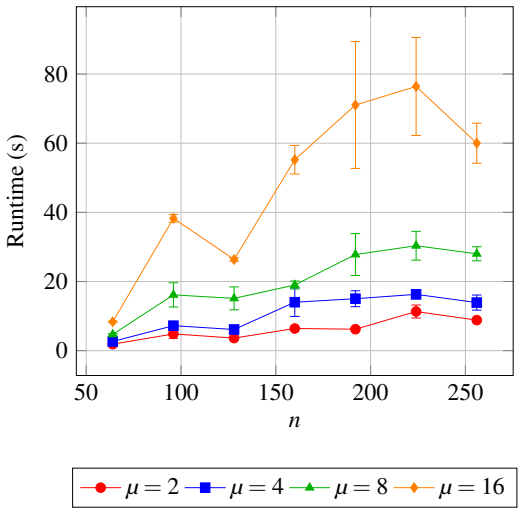

Figure 6: Runtime (s) vs n (#rows) with #attributes=3, $k = 4$.

on n distances to find the closest neighbours. Preliminary testing showed that this phase dominates runtime. Finally, to find the best counterfactual, a min_1 sort is performed over the final scores.

Figure 4 shows that the algorithm scales favorably with the neighborhood k, which can be attributed to the fact that the min_k operation in the calculation of o_K scales with $\log k$. The parameter μ also influences the runtime of the o_K calculation, figure 5 shows this scaling which initially appears linear. In our implementation, the rounds of the min_t sorting step are not executed concurrently for the μ candidates, meaning that the min_t algorithm is performed sequentially for μ candidates, which explains this behaviour.

The algorithm can also be seen to scale well initially over the number of datapoints n (figure 6). This agrees with round complexity of o_K (dominating phase) which is $O(\mu \cdot (\log k \log n))$ (finding the closest k neighbours for each of the μ candidates). However, for larger values of n the limiting factor becomes the memory usage of our implementation, which is $O(n^2)$ due to the matrix of pairwise Gower distances.

Though a plot is not included, the runtime sees very little variance depending on the number of attributes present in the datasets of the parties. This aligns with our expectations, as most computations that involve the attributes are performed locally using additively homomorphic encryption, requiring no communication between the parties.

7 Conclusion

In this paper, we present a new approach for generating counterfactual explanations for federated AI models in a privacy-preserving way. Our approach combines different PETs in order to use the strengths of each particular technology and introduces algorithm-level optimizations to further enhance the performance. We show theoretically that combining multiple data sources with different attributes can lead to better explanations and therefore motivate the use of PETs for XAI. Furthermore, we show that our solution scales well for large datasets while not hurting the quality of the explanations (significantly). With a μ-value of 18 (out of $n = 128$), we are able to find the best counterfactual explanation in 97% of the cases while saving approximately 80% of the runtime compared to a naive version of the algorithm without μ.

While the current experiments show that a μ-value of 18 is sufficient to not degrade the quality of explanations for most observations, this is likely heavily dependent on the dataset on which it is used and the observation for which an explanation needs to be found. Namely, for datasets of different nature for particular observations, the k-purity objective might have a larger influence on the counterfactual scores. Nevertheless, our experiments provide a good starting point for using our solution in various contexts.

REFERENCES

AI Act (2024). Regulation (eu) 2024/1689 of the european parliament and of the council of 13 june 2024 laying down harmonised rules on artificial intelligence and amending regulations (ec) no 300/2008, (eu) no 167/2013, (eu) no 168/2013, (eu) 2018/858, (eu) 2018/1139 and (eu) 2019/2144 and directives 2014/90/eu, (eu) 2016/797 and (eu) 2020/1828 (artificial intelligence act).

Batcher, K. E. (1968). Sorting networks and their applications. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference (AFIPS '68 (Spring))*, pages 307–314. ACM.

Bauermeister, S., Orton, C., Thompson, S., Barker, R. A., Bauermeister, J. R., Ben-Shlomo, Y., Brayne, C., Burn, D., Campbell, A., Calvin, C., et al. (2020). The dementias platform UK (DPUK) data portal. *European journal of epidemiology*, 35:601–611.

Berning, S., Dunning, V., Spagnuelo, D., Veugen, T., and Van Der Waa, J. (2024). The trade-off between privacy & quality for counterfactual explanations. In *Proceedings of the 19th International Conference on Availability, Reliability and Security*, pages 1–9.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. (2017). Practical secure aggregation for

privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191.

Bozorgpanah, A. and Torra, V. (2024). Explainable machine learning models with privacy. *Progress in Artificial Intelligence*, 13(1):31–50.

Dandl, S., Molnar, C., Binder, M., and Bischl, B. (2020). Multi-objective counterfactual explanations. In *International conference on parallel problem solving from nature*, pages 448–469. Springer.

Goethals, S., Sörensen, K., and Martens, D. (2023). The privacy issue of counterfactual explanations: explanation linkage attacks. *ACM Transactions on Intelligent Systems and Technology*, 14(5):1–24.

Goldreich, O., Micali, S., and Wigderson, A. (1987). How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, page 218–229, New York, NY, USA. Association for Computing Machinery.

Harder, F., Bauer, M., and Park, M. (2020). Interpretable and differentially private predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4083–4090.

ICO (2023). Privacy-enhancing technologies (pets). https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-sharing/privacy-enhancing-technologies/.

Jetchev, D. and Vuille, M. (2025). Xorshap: Privacy-preserving explainable ai for decision tree models. *IACR Communications in Cryptology*, 1(4).

Jönsson, K. V., Kreitz, G., and Uddin, M. (2011). Secure multi-party sorting and applications. *Cryptology ePrint Archive*.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. Pmlr.

Mochaourab, R., Sinha, S., Greenstein, S., and Papapetrou, P. (2021). Robust counterfactual explanations for privacy-preserving svm. In *International Conference on Machine Learning (ICML 2021), Workshop on Socially Responsible Machine Learning*.

Molhoek, M. and Van Laanen, J. (2024). Secure counterfactual explanations in a two-party setting. In *2024 27th International Conference on Information Fusion (FUSION)*, pages 1–10. IEEE.

Nguyen, T. T., Huynh, T. T., Ren, Z., Nguyen, T. T., Nguyen, P. L., Yin, H., and Nguyen, Q. V. H. (2025). Privacy-preserving explainable ai: a survey. *Science China Information Sciences*, 68(1):111101.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer.

Sangers, A., van Heesch, M., Attema, T., Veugen, T., Wiggerman, M., Veldsink, J., Bloemen, O., and Worm, D. (2019). Secure multiparty pagerank algorithm for collaborative fraud detection. In *International Conference on Financial Cryptography and Data Security*, pages 605–623. Springer.

Schoenmakers, B. (2018). Secure multiparty computation in python.

Sismanis, N., Pitsianis, N., and Sun, X. (2012). Parallel search of k-nearest neighbors with synchronous operations. In *2012 IEEE Conference on High Performance Extreme Computing*, pages 1–6. IEEE.

TNO PET Lab (2025). Paillier encryption scheme implementation. https://github.com/TNO-MPC/encryption_schemes.paillier.

Van Kolfschooten, H. and Van Oirschot, J. (2024). The eu artificial intelligence act (2024): implications for healthcare. *Health Policy*, 149:105152.

Veugen, T., Attema, T., and Spini, G. (2019). An implementation of the paillier crypto system with threshold decryption without a trusted dealer. *Cryptology ePrint Archive*.

Veugen, T., Dunning, V., Marcus, M., and Kamphorst, B. (2025). Secure latent dirichlet allocation. *Frontiers in Digital Health*, 7:1610228.

Vo, V., Le, T., Nguyen, V., Zhao, H., Bonilla, E. V., Haffari, G., and Phung, D. (2023). Feature-based learning for diverse and privacy-preserving counterfactual explanations. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2211–2222.

Yang, L., Chai, D., Zhang, J., Jin, Y., Wang, L., Liu, H., Tian, H., Xu, Q., and Chen, K. (2023). A survey on vertical federated learning: From a layered perspective. *arXiv preprint arXiv:2304.01829*.

Yang, Q., Liu, Y., Chen, T., and Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19.