DensePCP - Density-Driven Plausible Counterfactual and Path

Author1[1], Author2[2], and Author3[1]

Name and address of institution 1
Name and address of institution 2
email1, email2, email3

Abstract. Decisions obtained from classification models can significantly impact people's lives, and counterfactual explanations help to understand these decisions, indicating which attributes must be changed to reverse the class of interest. In this work, we proposed the DensePCP method, an extension of XMethod[1], which uses a genetic algorithm to adjust sparsity and similarity. Still, we added *density* as a metric to evaluate a generation of counterfactual candidates. Although XMethod guarantees validity, sparsity, similarity, diversity, and actionability, there is no guarantee of plausibility and viable paths for change. Thus, once the final counterfactuals are generated, the DensePCP finds the densest or most feasible path between the original instance and each generated counterfactual. As a result, we validate density as an indicator of plausibility through detailed analysis. Furthermore, we generate the shortest and densest path between the original instance and the counterfactual, indicating the necessary changes as sequential steps. The method implementation is available at xxxxxxx[1].

Keywords: Plausability · Contrafactual Explanations · Machine Learning Interpretability · Feasible Paths

1 Introduction

Machine learning (ML) models are increasingly used in decision-making processes in companies and large organizations. No matter how technical, the results of these classifications can profoundly impact individuals' lives [17]. For example, classifying whether a person is eligible for a bank loan can significantly affect their financial future and purchasing power. This scenario highlights the importance of understanding what led an algorithm to reach the final classification result and which attributes impacted its decision so individuals can understand their position, challenge it, or even act on that information to increase their chances of approval [17].

However, many ML models function as a "black box", where the internal decision-making mechanisms are hidden from users [17]. This obscurity can raise

[1] As the review process is double anonymous, we will add the method name and address after review if accepted.

questions about the decision's fairness, accuracy, and reliability [11, 17]. Therefore, counterfactual explanation methods have emerged in the literature. Counterfactuals are hypothetical scenarios showing which attributes would need to be changed for the classification result to differ given an input [5, 12]. Thus, a new world is created where, for example, an individual denied a loan would be approved if specific attributes were modified. A counterfactual would answer the question: "What would I need to change in my data to get the loan approved?"

Even though these methods have made significant advances in generating counterfactual explanations, aspects must be considered in their development. One of the main limitations is the need to guarantee plausibility in the generated changes concerning reality.

Plausibility is defined as the coherence that a generated counterfactual has with reality. This can be indicated by the deviation that the attribute values of a counterfactual have concerning the normal distribution of values of a set of actual instances. Using the same example of loan eligibility, suggesting that an individual increase their income to 1 billion reais in a database where salaries range from R$5,000.00 to R$100,000.00 can be considered an outlier. Such a change does not align with reality and can make a counterfactual infeasible. Similarly, it is not plausible to suggest that an individual from a lower social class with a substantially different economic reality from the majority in the database increases their salary from R$6,000.00 to R$80,000.00. Although this value is within the observed salary range in the database, it does not consider socioeconomic barriers, limited access to education, and professional opportunities that influence this individual's salary increase. Thus, a counterfactual that proposes such a salary adjustment for this person needs to pay attention to critical factors that shape their reality, making the suggestion incoherent with the lived context and, therefore, not plausible.

In this regard, this work proposes the DensePCP method, which adds the plausibility metric in generating counterfactual explanations. The main purpose is to understand how the relationship of the spatial distribution of attribute values in a database can be incorporated into the generation of counterfactuals to ensure the production of more coherent, informative, and applicable hypothetical scenarios. As a result, it was possible to analyze the density of counterfactuals initially generated by the XMethod method, verify density as an indicator of plausibility, add density to the model's optimization function, and develop the densest and shortest path between the original instance and the counterfactual.

This is a significant contribution in the context of counterfactual explanations since indicating the path taken from the original instance to the counterfactual can provide important insights to a user who wants to better understand the method's explanation.

This work is organized as follows: Section 2 provides the theoretical framework and related works that support this study. Section 3 presents the methodology used. Section 4 specifies the implementation of the DensePCP, Section 5 the experiments and results obtained, and Section 6 presents the conclusions and future work.

2 Theoretical Background

2.1 Counterfactual Explanations

Counterfactual explanations are a way to provide interpretability for various machine learning algorithms. Suppose a decision model that classifies job candidates for eligibility to participate in final interview rounds. If the model disqualifies a candidate x, it is necessary to explain to this candidate how the decision was made and the reason for the result [17]. This feedback is essential to: 1) increase understanding of how the decision-making process works and the reason for the obtained result; 2) allow the decision to be contested; 3) provide information on how to act to achieve a different result in future classifications;

However, opening the "black box" of the classification model to provide this information can mean exposing confidentiality about the method used and the criteria defined by the company and risking the security of other candidates' data. This high level of exposure often requires companies to explain the results obtained with the classification. Additionally, even if the decision-making logic is disclosed, some may only understand its highly technical language, making it difficult for a candidate to take appropriate action to change their current situation [18].

One way to ensure that, even within the example scenario, the candidate receives positive and understandable feedback without disclosing the decision criteria is through counterfactual explanations. They disclose why a decision was made by informing what changes in the input (attributes/characteristics of the instance/candidate) would lead to a different classification result [5]. They provide the minimum information necessary to alter a decision and do not require the person corresponding to the input instance to understand the internal logic of the model to use the information to their advantage [17].

Definition 1: *Given a classification model that results in $b(x) = y$ for an instance x, a counterfactual explanation is an instance x' where $b(x') = y'$, meaning the classification result is different for x' and x, with the difference between the attributes of x and x' being minimal.*

Thus, a counterfactual explanation can be described as follows: "You were disqualified from the final interview stage because you have 1 year of work experience. You would have been approved if you had 4 years of experience". For instance, a candidate aged 23, with 1 year of work experience, a skill level of 3 in the field, a current salary of 3000, and working 8 hours per day was disqualified. If the candidate had 4 years of work experience instead of 1, the result would have been approval, while all other attributes remained the same. Given this information, the candidate can understand the reason for their disqualification and which attributes led to this result instead of just receiving the message "you were disqualified" from the company. Additionally, the candidate can contest the company's decision with arguments based on the explanations, such as: "I understand that I would be qualified if I had 4 years of work experience, but the decision can be reviewed since this 1 year only corresponds to the period I have been officially employed. I have previous experience in internships that lasted

more than 3 full years, totaling 4 years of experience in the field". Finally, the knowledge of the explanation serves as a resource for the candidate to act on their reality to change the result in a future classification. Thus, counterfactual explanations act as a recommendation: *the candidate understands that they need to increase their experience to be qualified in the future.*

During the generation of counterfactual explanations, some crucial properties must be considered to verify if a counterfactual is truly useful. A result indicating that a candidate would be approved if they were 2 years younger while providing some understanding of the decision logic does not allow the individual to act on this information to change their reality, as it is impossible to decrease one's age. Therefore, there needs to be a more feasible counterfactual. Similarly, a counterfactual suggesting that a candidate would be approved if they had 5 more years of work experience, earned a salary of 5000, and worked 10 hours daily is also impractical. Many changes and the high-value difference make it difficult for the individual to act based on this information.

Thus, to produce good counterfactuals, models for generating counterfactual explanations must ensure validity, minimality/sparsity, similarity, plausibility, actionability, discriminative power, causality, and diversity [5]. This work will delve deeper into plausibility and the methods to ensure this property in generating counterfactuals.

2.2 Plausibility in the context of counterfactual explanations

Plausibility refers to how close to reality the attribute values of a counterfactual are [5]. For example, consider a decision model that classifies the health level of a home's garden. Suppose a garden has been classified as low health. A counterfactual could indicate that the health level would be high if the water sprinklers were activated every 5 minutes. However, if, in the original data, the activation period of the sprinklers varies from every 5 hours to every 12 hours, activating them every 5 minutes can be considered an outlier, distancing the change from feasible and practical reality. The drastic change between the indicated value and the actual attribute values can make the generated counterfactual implausible—if no one activates their sprinklers every 5 minutes, it is likely that there is a reason for this (e.g., financial constraints due to increased water consumption) that makes this action not applicable to reality.

According to [5], a counterfactual is realistic if it is similar to the known database and incorporates the correlational relationships between attributes. For [3], plausibility is defined as the likelihood of a counterfactual occurring in practice, an aspect that conflicts with the extent of modifications made to the model input. [19] state that the property reflects how achievable and believable a set of attributes is about population distribution. For [2], plausibility is similar to actionability and "reasonableness" [6]. According to the authors, counterfactuals must be legitimate and logical, with immutable variables (such as gender and race) not being modified nor carrying a bias such as "if you were a man, you would have been accepted for the loan".

Recent methods increasingly focus on ensuring plausibility during the generation of counterfactuals. Accurate data distribution is commonly used as a parameter to create feasible counterfactuals. The distribution or density of data in space represents how data points (instances) are distributed across a multidimensional space, reflecting the characteristics of the data.

Data density can be calculated using Kernel Density Estimation (KDE) [13, 1]. This method creates a continuous function that smooths the data to represent the probability density across different points in the space. Counterfactuals generated in high-density regions are more plausible and realistic as they align with the actual data distribution, reflecting changes that could naturally occur in real life. Another method to incorporate data distribution into the generation of counterfactuals is using Variational Autoencoders (VAE) [15, 9, 4, 7]. The VAE learns the data representation in a latent space, enabling it to generate new data points consistent with the original data distribution or, given a new instance, measure whether it is within the calculated latent space to evaluate plausibility. Generative Adversarial Networks (GANs) [16] are commonly used to generate counterfactuals. This method involves two neural networks: a generator, which creates instances mimicking the data distribution, and a discriminator, which evaluates if an instance is real or fake. During training, the discriminator distinguishes between real and fake data, while the generator improves at producing realistic data that can deceive the discriminator. GANs are frequently mentioned in current literature for image classification [10, 8].

2.3 Related work

[9] propose the Example-Based Counterfactual Explainer (EBCF), which uses a causal proximity regularizer and optimizes a loss function based on a Structural Causal Model (SCM). It incorporates user feedback and VAE to generate plausible counterfactuals without a complete causal graph.

In [4], the authors propose a CSVAE method to generate counterfactuals based on labeled data. A genetic algorithm separates relevant latent attributes, and decision trees create simple rules. CRUDS requires labeled data and prior human knowledge of causal relationships.

[13] propose FACE, which ensures feasible counterfactuals by finding paths based on the shortest distance and data density. FACE constructs a graph with instances as nodes and uses Dijkstra's algorithm to find the shortest viable path, considering user-defined thresholds and densities.

[15] present REVISE+, which uses LSTM-VAE autoencoders to ensure counterfactuals are in high-density regions. It employs a Declare language template based on LTLf to capture sequential attribute changes, ensuring plausibility by incorporating patterns supported by the original dataset.

3 Methodology

As seen in the previous section, current literature attempts to develop different ways to ensure plausibility in generating counterfactual explanations.

XMethod[1] is a method to generate explanations that, although not guaranteeing plausibility, present significant advances to the current literature and ensure most of the other properties mentioned in Section 2. In the XMethod, a Genetic Algorithm (GA) generates a set of k counterfactuals and, in each generation, tries to approximate it to the desired class.

This method differs from others in several aspects. First, XMethod focuses on being a user-oriented approach, meaning that for each generated counterfactual, the differences between x and x' are extracted and transformed into an easy-to-read and understandable explanation for the user. Thus, the model's results are formatted as recommendations. Second, the algorithm has weight variables related to sparsity and similarity properties. Therefore, the user can adjust the minimality as desired, favoring counterfactuals with a smaller number of modified attributes (greater sparsity) or smaller distances (more remarkable similarity - smaller amplitude of change in each attribute's values). Third, XMethod has a static list that restricts the attributes used in generating counterfactuals. Making immutable attributes static in real life, such as age, gender, or some subjectively immutable characteristic in an individual's reality, guarantees the property of actionability. Finally, according to [14], using a genetic algorithm to generate counterfactuals is among the best techniques for dealing with optimization problems. Since ML involves various attributes - including continuous ones that can vary infinitely - the search space for an explanation becomes enormous, making choosing a suitable optimization method crucial.

The XMethod can generate counterfactual explanations with diversity, without prolixity, user-oriented, with highly efficient optimization, high confidence, ensuring actionability, and allowing parameterization according to the user's choice for a more accurate and, in a way, almost personalized explanation. Due to all these qualities, this method was chosen as the basis for advancing the plausibility study. Thus, this work proposes the DensePCP method, adding the characteristic of plausibility to the counterfactual explanations obtained. Furthermore, the DensePCP method advances by visualizing the path from the original instance to the counterfactual.

[13] present the FACE method to generate counterfactual explanations. According to the author, it is necessary to ensure not only the shortest possible distance between an instance and its counterfactual but also that this new instance is in a region of higher density and that the path between it and the original instance passes through regions of higher density (Figure 1). This way, we can ensure that the counterfactual is plausible and that an actual individual can follow a feasible (dense) path to achieve the desired class change.

The FACE model [13] represents data instances as nodes in a graph (DAG). Each node is connected by an edge, whose weight is calculated by k-NN, KDE, or ϵ-graph. In this work, we focus on using KDE as the only one among the three that uses density as a variable. In KDE (Kernel Density Estimator), a density kernel is calculated according to the actual data distribution in space. This kernel can be used to measure the density at a given point. In the graph, the weight of each edge is a combination of the density of the path between the

nodes (calculated at the midpoint between the two) and the Euclidean distance between the two instances. This calculation is replicated for every pair of cases in the dataset until the graph is entirely constructed.

With the graph created, the user must determine the minimum local density of the counterfactual (*density_threshold*) and the number of paths they wish to obtain (*how_many_paths*). From a given original instance and desired classification, instances in the dataset with the target classification and minimum local density are candidates for counterfactuals. According to the user's choice in *how_many_paths*, that number of counterfactuals is selected, and the shortest path (Dijkstra) is calculated between the original instance and the counterfactuals through the created graph (Figure 1). According to the authors, the fact that the generated path is also the densest means that actual instances obtain those combinations of attribute values. Therefore, the changes an individual x would need to implement to become x' would be more feasible, ensuring plausibility, actionability, and causality.

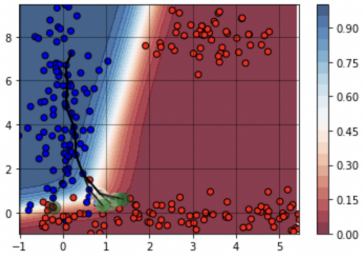


Fig. 1. The result was obtained with the original FACE model. The graph shows the shortest possible path (black lines) between the original instance (point in the upper left corner, blue region) and the generated counterfactuals (points in green in the lower left corner, red region). Although counterfactuals in the red region of the upper right corner are closer, [13] argue that they are not in a dense location. Therefore, counterfactuals in the lower left corner are more plausible, achieved by a denser and more feasible path.

As a methodology for this project, the application of these FACE characteristics to the original XMethod algorithm was verified. At each generation of counterfactuals, it will be checked whether plausibility was truly ensured. The results of each approach will be qualitatively compared to evaluate the existence of the other necessary properties of a good counterfactual (mentioned in Section 2), whether the XMethod model maintained the properties already indicated in its study, and finally, whether there was an evolution in developing plausible, actionable, realistic, and minimally changing explanations.

It is worth noting that counterfactual generation by FACE is endogenous (selected from the dataset itself) as opposed to XMethod, which is exogenous (synthetically generated using GA). This characteristic makes it plausible by the natural reality of the data. Therefore, this study focuses on verifying whether the application of density - calculated by KDE - as a quality parameter in creating counterfactuals will help ensure plausibility in the XMethod method, and

wheather calculating the densest path between the original instance and generated counterfactuals produces plausible recommendations.

The project was developed locally using Visual Studio Code (VS Code) as the IDE, with Python 3.9 as the primary programming language. Essential libraries and tools used include pandas (2.0.0) for data manipulation, scipy.spatial.distance for distance calculations and scikit-learn (0.0) with GridSearchCV for hyperparameter tuning, KernelDensity for density estimation, and RandomForestClassifier for classification. Additionally, custom functions from a module created by the FACE authors were used for Dijkstra's algorithm and graph construction.

4 DensePCP Method

The DensePCP method uses a genetic algorithm to generate counterfactuals in the same way that the XMethod does. First, an initial synthetic population is generated, and new generations are developed from it, always aiming to improve the "quality" of individuals. This quality translates directly through the parameters of similarity and sparsity. Iteratively, new generations of instances are generated from a previous generation, and all individuals are evaluated based on these parameters to determine their continuation to the next generation, analogous to a selection of the best individuals. Only individuals with the best metrics are retained for the next generation. This process is repeated iteratively, promoting the evolution of the population in each cycle.

The DensePCP method contributes by introducing density as another metric for evaluating the quality of the population's individuals. In each generation, the counterfactuals are analyzed and evaluated for their similarity, distance, and density x_dens in space. The three values are normalized and used in the genetic evaluation (whose objective function is described in Equation 1).

$$C(x) = \arg\min_{c} \left(\lambda_1 \cdot \mathrm{x\_dist}(c, x) + \lambda_2 \cdot \mathrm{x\_nchg}(c, x) + \mathrm{y\_dist}(p(c), y') + \lambda_3 \cdot \mathrm{x\_dens}(c)\right) \quad (1)$$

In this equation, we have: 1) x is the original instance, 2) c is its candidate counterfactual, 3) $p(c)$ is the classification model's response for c, 4) y is the desired class, 5) $x\_dist(c, x)$ is the distance between the instance and the counterfactual candidate, 6) $x\_nchg(c, x)$ is the number of attributes modified in x to generate c, 7) $y\_dist(p(c), y')$ is the Euclidean distance from c to the desired class y, and 8) x_dens is the density of the counterfactual candidate. Thus, x_dist and x_nchg keep the counterfactual candidate c close to the original instance, y_dist aims to direct c towards the desired class, and x_dens tries to keep c in a denser region.

Just as the *similarity* (distance) and *sparsity* (number of changes) factors have user-adjustable weights in the original XMethod model (λ_1 and λ_2, respectively), in the DensePCP method, the density can also be adjusted by λ_3.

In addition to introducing density as a quality metric for counterfactual candidates, DensePCP implements a search for the shortest and densest path between counterfactuals and the original instance. After the genetic algorithm successfully generates the final counterfactual instances, a graph is created similarly

to FACE, where nodes are data instances (including the new counterfactual instances), and the weights $P(no_1, no_2)$ of the edges between each pair of instances no_1 and no_2, calculated from Equation 2:

$$P(no_1, no_2) = \text{weight\_function}(\exp(\text{dens\_midpoint}(no_1,\, no_2))) \cdot \text{dist}(no_1,\, no_2) \quad (2)$$

The *weight_function* was initialized as in Equation 3, following the FACE code standard when using the KDE method.

$$\text{weight\_function(x)} = -\log(x) \quad (3)$$

dens_midpoint is the density of the point located precisely halfway along the distance between instances no_1 and no_2, and *dist* is the Euclidean distance between them. Thus, the edge weight between the two instance nodes is weighted by the path density (*dens_midpoint*) and the distance (*dist*) between the instances.

Once the graph is created, Dijkstra's algorithm finds the shortest possible path between the original instance x and counterfactual c. In this case, the shortest path will be the path with the smallest sum of edge weights and, therefore, the highest trade-off between distance and path density.

Thus, DensePCP contributes to improving the user experience when using the model by translating the paths found between each instance x and c into recommendations. The results of these recommendations can be analyzed in the next section of this document.

5 Experiments and Results

To validate the proposed method's performance, experiments were conducted with the real-world dataset *German Credit*[2], which XMethod also used in its paper. This dataset includes 1000 loan applicants. Each candidate is described by a set of 20 features in addition to the class. Each candidate is assigned a credit risk rating of "good" (0) or "bad" (1). In the dataset, 700 instances belong to the class of good credit candidates and 300 to the bad candidates.

Five counterfactuals (Table 2) were generated for a random instance (Table 1) in the original XMethod model without applying density as a parameter yet.

The density measure of each counterfactual was obtained using KDE. Counterfactual 2 achieved the highest density, while counterfactual 4 had the lowest density. By analyzing the attributes modified for counterfactual 2, it is noted that:

- Changing *account_check_status* (current account status) from 4 (no current account) to 2 (has a positive and low balance) is plausible. Creating a bank account and maintaining a low balance is feasible—significant changes such as having an account with an extremely high balance would not be viable.

[2] Available at https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

Table 1. Original Instance - Class 0

Attribute	Value	Attribute	Value	Attribute	Value
account_check_status	4	property	3	present_res_since	4
duration_in_month	6	age	52	other_debtors	1
credit_history	4	other_installment_plans	1	foreign_worker	1
purpose	0	housing	2	telephone	1
credit_amount	362	credits_this_2	2	people_under_maintenance	1
savings	2	job	2	personal_status_sex	2
present_emp_since	3	installment_as_income_perc	4		

Table 2. XMethod Original Results

		Values		
	Attribute	Before	After	Density
Counterfactual 1	account_check_status	4	2	8.37589398e-08
	duration_in_month	6	34	
Counterfactual 2	account_check_status	4	2	**0.00446028**
	credit_history	4	1	
Counterfactual 3	duration_in_month	6	21	1.43585866e-09
	credit_amount	362	12024	
	other_installment_plans	1	2	
	telephone	1	0	
Counterfactual 4	duration_in_month	6	21	1.81490991e-12
	credit_history	4	0	
	credit_amount	362	11414	
Counterfactual 5	account_check_status	4	2	2.05037314e-10
	credit_amount	362	12024	
	other_installment_plans	1	2	
	telephone	1	0	

- Changing *credit_ history* (credit history) from 4 (critical account/credits existing in other banks) to 1 (all credits in this bank were duly paid) is also plausible for an individual who has improved their financial condition and is consistent with the change in *account_ check_ status*. Paying off debts without significantly increasing income is feasible, normalizing their bank situation.

On the other hand, the attributes modified for counterfactual 4 shows that:

- Changing *duration_ in_ month* (duration in months) from 6 to 21 could be plausible depending on the renegotiation of payment terms, but it is not realistic for banks to accept such a high renegotiation.
- Changing *credit_ history* (credit history) from 4 (critical account/credits existing in other banks) to 0 (no credit taken) might be plausible, depending on financial improvement.
- Changing *credit_ amount* (credit amount) from 362 to 11414 is not plausible. It is unrealistic for the credit request to increase so much without justification.

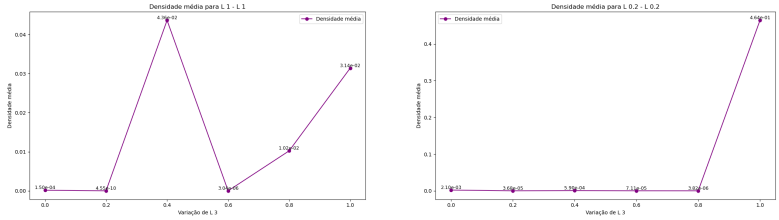
Thus, the higher density for counterfactual 2 and the lower density for counterfactual 4 are consistent with the recommendations' plausibility level. Therefore, its use as an indicator of plausibility is validated.

As another experiment, tests were conducted on DensePCP - with density applied - to validate its contributions. In other words, a third metric representing density was also added besides the adjustments that consider the minimum number of changes and minimum distance. With density as a parameter on DensePCP, tests were performed with a randomly selected instance from the *German Credit* dataset for a closer analysis of the model's behavior. The tests were divided into two cases:

1. Fixing λ_1 and λ_2 at value '1' and varying λ_3 from 0.2 to 1, with a gradual increase of 0.2.
2. Fixing λ_1 and λ_2 at value '0.2' and varying λ_3 from 0.2 to 1, with a gradual increase of 0.2.

With these experiments, it was expected that, in both cases, the final density value of the counterfactuals would gradually increase as λ_3 increased and that, in the second case, the total final density value of the explanations would be higher than in the first case. This expectation is because, in the first case, the fixed value of λ_1 and λ_2 gives more weight to sparsity and similarity, while in the second case, this weight is lower.

Figure 2 shows the variation in the average density of the counterfactuals generated by DensePCP, according to the variations of λ_3 (see Equation 1).



(a) Average density of counterfactuals. Variation of $\lambda 3$ for λ_1=1 and λ_2=1

(b) Average density of counterfactuals. Variation of $\lambda 3$ for λ_1=0.2 and λ_2=0.2

Fig. 2. Evaluation of density values as a function of λ_1 and λ_2

Figure 2(a) shows that, with the increase of λ_3, the density of the generated counterfactuals became higher but did not follow the expected gradual behavior. Instead, a peak is observed in Figure 2(a) when $\lambda_3 = 0.4$, and an almost constant line in Figure 2(b), followed by a peak when $\lambda_3 = 1$. Additionally, the average density values between both cases did not show considerable variation, which was different from what was expected.

As a potential explanation for this behavior, we analyze that the XMethod was designed to emphasize diversity in counterfactual generation, and DensePCP may have inherited this characteristic. Consequently, generating five counterfactuals from a single original instance in each test case likely resulted in a diverse set of counterfactuals with varying density values, reducing the average density

per test case. Furthermore, we try to balance many constraints (similarity, sparsity, diversity, and now density), even conditioning on a specific metric (density); perhaps the genetic algorithm is prioritizing the other metrics.

Finally, as a last experiment, DensePCP generated the densest and shortest path in the form of recommendations for two other random instances from the *German Credit* dataset. For each instance, only one counterfactual and path were generated.

The first original instance x_1 and its attributes can be analyzed in Table 3.

Table 3. Instance 1 (original) - 743

Attribute	Value	Attribute	Value	Attribute	Value
account_check_status	1	property	1	present_res_since	4
duration_in_month	24	age	22	other_debtors	1
credit_history	1	other_installment_plans	3	foreign_worker	1
purpose	3	housing	2	telephone	2
credit_amount	2483	credits_this_2	1	people_under_maintenance	1
savings	3	job	3	personal_status_sex	3
present_emp_since	3	installment_as_income_perc	4		

Figure 4 shows the path generated by DensePCP as a recommendation to reach the counterfactual c_1. Each instance in this table represents a data point an individual should pass through to achieve the desired class change. The numbers written in front of the instances correspond to their position in the utilized dataset.

It is noted in Table 4 that the generated path goes directly from x_1 to c_1. This behavior is understood by the counterfactual's small final distance value (0.5491169358628116 - normalized value)^2 and high density (0.32538687914437875). These values indicate that the path density and the distance trade-off were very favorable - marked by the graph edge - and did not require passing through other instances along the way. In Figure 4, the path between x_1 and c_1 is shown but cannot be visualized for the small distance value, which made the instances overlap.

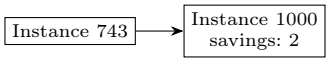


Fig. 3. Path of Changes from original instance to counterfactual, generated by DensePCP

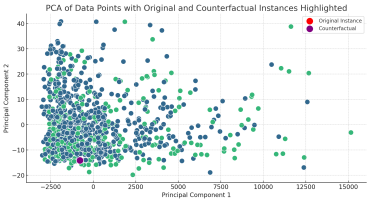


Fig. 4. The most dense and shortest path from the original instance (743) to the counterfactual (1000), generated by DensePCP.

The second original instance, x_2, and its attributes can be analyzed in Table 4. Figure 6 shows the path generated as a recommendation to reach the

counterfactual c_2. It is noted that the route passes through several instances before reaching c_2. This behavior is due to the higher distance value between x_2 and c_2 (1.6004559363871782 - normalized value) and the low density of x_2 (6.196900036803995e-08). These values indicate that the trade-off between density and distance was unfavorable, and the individual needs to undergo other changes before reaching the final class change. In Figure 6, the path between x_2 and c_2 is shown.

Table 4. Instance 2 (original) - 676

Attribute	Value	Attribute	Value	Attribute	Value
account_check_status	4	property	3	present_res_since	4
duration_in_month	24	age	33	other_debtors	1
credit_history	4	other_installment_plans	1	foreign_worker	1
purpose	3	housing	2	telephone	2
credit_amount	5150	credits_this_2	1	people_under_maintenance	1
savings	1	job	3	personal_status_sex	3
present_emp_since	5	installment_as_income_perc	4		

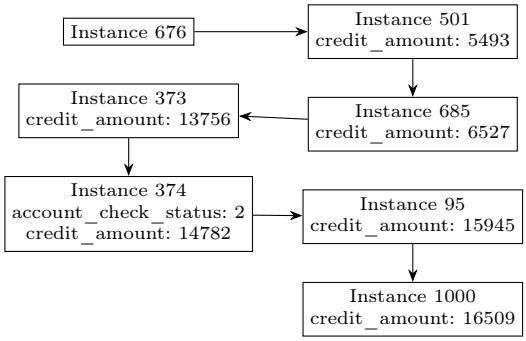


Fig. 5. Path of Changes from original instance to counterfactual, generated by DensePCP

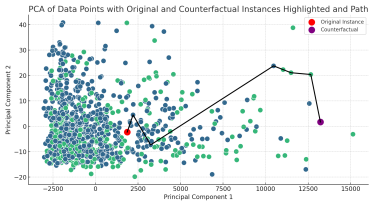


Fig. 6. Path from original instance (676) to counterfactual (1000) generated by DensePCP. Using PCA to visualize multidimensional data can lead to loss of information during dimensionality reduction. Thus, the density of the path cannot be analyzed entirely on the image

6 Conclusion and Future Work

In this study, we propose DensePCP, which uses density to generate plausible counterfactuals and paths between counterfactual and original instances. We first validate density as an indicator of plausibility in an experiment conducted with the XMethod method. It was observed that counterfactuals generated in denser regions are more plausible. Thus, density was confirmed as an indicator of plausibility. Then, we evaluate the proposed DensePCP method and the impact of

[2] The presented distance values are the normalized values already calculated by XMethod during the genetic algorithm to evaluate the counterfactuals in each generation.

density as a metric on counterfactual generation. During this evaluation, counterfactuals were generated with a variation of values for weights λ_1, λ_2, and λ_3 to test the sensitivity of these weights and how they could be better adjusted for increasing density importance. Finally, the DensePCP method was used to generate the densest path between two different counterfactuals and their original instances and present the path as a "step-by-step" recommendation of changes to get to the desired classification result.

In the first experiment, density was validated as an indicator of plausibility on XMethod. It was observed that counterfactuals generated in denser regions are more plausible, as shown in the results presented in Table 2. The second experiment's results highlight the need for more tests with diverse datasets to understand the density behavior in the DensePCP XMethod. While high values for density (λ_3) and low values for similarity (λ_1) and sparsity (λ_2) were expected to produce counterfactuals in denser regions, the introduction of density as a parameter led to random behavior. This randomness persisted regardless of the weight variations for λ_1, λ_2, and λ_3, necessitating further analysis to comprehend its impact on DensePCP's plausibility. Additionally, experiments generating one counterfactual per instance in a set of ten instances and choosing the densest one to analyze in each lambda variation could help diminish similarity, sparsity, and diversity factors, enabling a more precise analysis of the density effect. In the third experiment, DensePCP generated the shortest and densest path, demonstrating how an individual can act on sequential changes to reach the expected decision/classification, highlighting a feasible path. The generated path analysis for two instances showed that the trade-off between density and distance plays a crucial role in defining the sequence of recommended changes.

For future work, we suggest conducting more extensive tests with variations of datasets in the DensePCP method. In this study, we used only the German Credit dataset, and some specific characteristics of this dataset may be causing the observed random behavior. We recommend using datasets with different numbers of attributes, various quantities of instances, and distinct spatial distributions. For example, datasets with a wide range of categorical and numerical variables, other sizes, and population characteristics can provide more comprehensive conclusions about the model's performance. Additionally, it would be interesting to alternate the choice of the original instance within the same dataset, as its position, distance relative to other instances, and local density can interfere with the calculations. Including sensitivity and robustness analyses of the instances through variation of weights λ_1, λ_2, and λ_3 can help identify the factors that most influence the generation of counterfactuals. These approaches can provide a more robust validation and in-depth understanding of the DensePCP model, contributing to generating more consistent and plausible counterfactuals.

References

1. Chen, Y.C.: A tutorial on kernel density estimation and recent advances (2017)

2. Chou, Y.L., Moreira, C., Bruza, P., Ouyang, C., Jorge, J.: Counterfactuals and causability in explainable artificial intelligence: Theory, algorithms, and applications (2021)
3. Del Ser, J., Barredo-Arrieta, A., Díaz-Rodríguez, N., Herrera, F., Saranti, A., Holzinger, A.: On generating trustworthy counterfactual explanations. Inf. Sci. **655**(C) (feb 2024)
4. Downs, M., Chu, J., Yacoby, Y., Doshi-Velez, F., WeiWei, P.: Cruds: Counterfactual recourse using disentangled subspaces. ICML Workshop on Human Interpretability in Machine Learning pp. 1–23 (2020)
5. Guidotti, R.: Counterfactual explanations and how to find them: literature review and benchmarking. Data Min Knowl Disc (2022)
6. Keane, M.T., Smyth, B.: Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable ai (xai). In: Watson, I., Weber, R. (eds.) Case-Based Reasoning Research and Development. pp. 163–178. Springer International Publishing, Cham (2020)
7. Klys, J., Snell, J., Zemel, R.: Learning latent subspaces in variational autoencoders (2018)
8. Lang, O., Gandelsman, Y., Yarom, M., Wald, Y., Elidan, G., Hassidim, A., Freeman, W.T., Isola, P., Globerson, A., Irani, M., Mosseri, I.: Explaining in style: Training a gan to explain a classifier in stylespace. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 693–702 (October 2021)
9. Mahajan, D., Tan, C., Sharma, A.: Preserving causal constraints in counterfactual explanations for machine learning classifiers (2020)
10. Mindlin, D., Schilling, M., Cimiano, P.: Abc-gan: Spatially constrained counterfactual generation for image classification explanations. In: Longo, L. (ed.) Explainable Artificial Intelligence. pp. 260–282. Springer Nature Switzerland, Cham (2023)
11. Pearl, J., Glymour, M., Jewell, N.: Causal Inference in Statistics: A Primer. Wiley (2016)
12. Pearl, J.: Structural counterfactuals: A brief introduction. Cognitive Science **37**(6), 977–985 (2013)
13. Poyiadzi, R., Sokol, K., Santos-Rodriguez, R., Bie, T.D., Flach, P.: FACE - Feasible and Actionable Counterfactual Explanations. In: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society. ACM (feb 2020)
14. Shahab, M., Azizi, F., Sanjoyo, B., Irawan, M., Hidayat, N., Rukmi, A.: A genetic algorithm for solving large scale global optimization problems. Journal of Physics: Conference Series **1821**, 012055 (03 2021)
15. Stevens, A., Ouyang, C., Smedt, J.D., Moreira, C.: Generating feasible and plausible counterfactual explanations for outcome prediction of business processes (2024)
16. Verma, S., Boonsanong, V., Hoang, M., Hines, K.E., Dickerson, J.P., Shah, C.: Counterfactual explanations and algorithmic recourses for machine learning: A review (2022)
17. Wachter, S., Mittelstadt, B., Russell, C.: Counterfactual explanations without opening the black box: Automated decisions and the gdpr. vol. 31, pp. 842–887. Harvard Journal of Law & Technology (2018)
18. Wachter, S., Mittelstadt, B., Russell, C.: Counterfactual explanations without opening the black box: Automated decisions and the gdpr (2018)
19. Yetukuri, J., Hardy, I., Vorobeychik, Y., Ustun, B., Liu, Y.: Providing fair recourse over plausible groups. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 21753–21760 (2024)