COMILLAS
UNIVERSIDAD PONTIFICIA

ICAI

GRADO EN INGENIERÍA MATEMÁTICA E INTELIGENCIA ARTIFICIAL

TRABAJO FIN DE GRADO

INTERPRETING NEURAL NETWORKS - DEVELOPING A METHODOLOGY FOR BANKING CASE STUDIES USING EXPLAINABLE AI

Autor: Javier Prieto Domínguez

Co-Director: David Alfaya Sanchez

Co-Director: Jaime Pizarroso Gonzalo

Madrid, Junio

UNIVERSIDAD PONTIFICIA COMILLAS
Escuela Técnica Superior de Ingeniería (ICAI)
Grado en Ingeniería Matemática e Inteligencia Artificial

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título **Interpreting Neural Networks - Developing a Methodology for Banking Case Studies Using Explainable AI** en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2024/25 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: **Javier Prieto Domínguez** Fecha: 10/06/2025

Autorizada la entrega del proyecto

LOS DIRECTORES DEL PROYECTO

Fdo.: **David Alfaya Sánchez** Fecha: 10/06/2025
Fdo.: **Jaime Pizarroso Gonzalo** Fecha: 10/06/2025

AGRADECIMIENTOS

Abstract

Neural networks are increasingly employed in the banking sector for tasks ranging from credit scoring to fraud detection. Despite their powerful predictive capabilities, the opacity of neural network models poses significant challenges for their interpretability. This project aims to bridge the gap between complex neural network architectures and their practical interpretation by developing a comprehensive methodology utilizing state-of-the-art Explainable AI (XAI) techniques.

Contents

1 Introduction

Currently, the regulations many entities face on the use of AI are very restrictive. They must be able to "explain" the decisions made by their models to justify business decisions based on those AI systems [7]. Today, the vast majority of them use models such as decision trees, random forests, and logistic regressions due to the lack of interpretability of bigger and more powerful models [8, 9]. The aim of this project is to provide a solution to solve two main problems. First, neural networks and larger models are considered "black-box" models, so we cannot fully understand the decision-making processes behind specific outputs, and therefore result useless when facing regulations. Secondly, we wish to give *actionable* solutions for differentiable classification models, such as neural networks or logistic regressions, by identifying changes in input features that could alter the model output.

In the context of banking and loan applications, for example, if a loan application is denied, the solution should provide actionable insights, such as "reduce your debt by $10,000 to qualify" or "increase your salary by $5,000 to qualify." These are concrete changes in the individual's profile that can change the classification outcome. These *explanations* or actionable changes are known as *counterfactuals* [1, 2]. A counterfactual reveals what should have been different in an instance to observe a different outcome. These explanations are a clear and direct definition for local interpretability. It provides an insight into why a singular instance has been classified a certain way. For our specific case, our aim is not to solve or provide global interpretability, as the end user, or clients of a bank, do not need to know the reason a model behaves a certain way for the entire dataset.

Our objective is to devise a novel counterfactual technique based on an *optimal mathematical solution* to the problem, using optimization methods and the model's derivatives to find the minimal shift needed for a given instance to change its output. We propose a customizable solution in which the end user is able to weigh how easy or difficult is to change each of the original features.

1.1 Estructura del Trabajo

2 Related works

Research in counterfactual explanations often highlights a set of properties a good explanation should fulfill [2]. These are:

- **Validity:** The counterfactual truly changes the classification outcome.
- **Minimality:** The counterfactual changes only the smallest set (or magnitude) of features.

- **Similarity:** The counterfactual remains close to the original instance by some distance metric.
- **Plausibility:** The counterfactual resembles realistic feature values found in the reference dataset.
- **Actionability:** The counterfactual only changes features that can realistically be altered (e.g., debt reduction but not age).
- **Diversity:** A set of counterfactuals should offer varied options for achieving the desired outcome.

Other desirable properties of the explanation *itself* are efficiency, which speaks to the speed of the method of returning the explanations; stability, if two instances are similar, their counterfactual explanations should be similar as well; and fairness, explanation remains valid even if sensitive attributes (e.g., ethnicity) were altered [2].

Counterfactual methods can be categorized by the strategy used to create the explanations. The main strategies include Optimization-based, Heuristic-based, Instance-based, and Decision-tree-based methods [2]. While optimization-based explainers usually have outstanding results in many of the individual properties highlighted above, they usually lack a good trade-off between all of them. Heuristic, instance and decision tree based methods are usually endogenous, meaning the counterfactual returned comes from the original dataset, which provides good results for plausibility and diversity, but not for similarity. On the other hand, optimization methods are in their majority exogenous, meaning they create a new sample based on the algorithm they follow, which usually performs good on minimality and similarity while possibly leaving other properties like plausibility and actionability unfulfilled. The current state-of-the-art does not yet include an explainer that fulfills all the desirable properties [2].

The task of finding an algorithm that fulfills all of the properties has been attempted in many papers. The most well-known technique in counterfactual space is WACH [1], one of the first papers to publish and propose these explanations. They propose a loss function, adopted by other explainers, consisting in a distance function and a cross-entropy loss, which they try to minimize with a common optimizer such as SGD [16] or Adam [17].

Other important algorithms in the literature include ACTREC (Actionable Recourse) [4], one of the first to rigorously address which features can be changed and which must remain fixed. ACTREC formulates the counterfactual generation as a discrete optimization problem using integer programming, ensuring feasibility, actionability, and minimal cost over a discretized space. It guarantees global optimality within its constraints but is limited to linear models and can become computationally intensive for high-dimensional problems.

In contrast, our method introduces a new optimization framework based on Lagrange multipliers and pseudo-Newton techniques, designed specifically for differentiable models such as neural networks and logistic regressions. By using NeuralSens [10] for efficient second-order derivative computation, our approach finds the minimal change required to change an instance classification output. Unlike ACTREC, which operates over a finite action grid, our method solves the continuous problem directly, offering high precision and smooth convergence in continuous feature spaces.

Another notable method is DiCE [19], which uses a differentiable loss function to generate a set of diverse counterfactuals. DiCE ensures validity and similarity while promoting diversity using Determinantal Point Processes (DPPs). While many methods focus on finding the closest counterfactual with high precision, DiCE offers users a broad range of actionable options.

Finally, SGNCE (Scaling Guarantees for Nearest Counterfactual Explanations) [20] guarantees the closest actionable counterfactual by formulating the search as a mixed-integer program (MIP), exploring the full solution space and supporting variables of mixed types. This approach provides formal guarantees on coverage, feasibility and similarity, making it robust but computationally demanding. Unlike SGNCE, our method achieves similar goals through a continuous, second-order optimization strategy that avoids combinatorial search. By doing so, we gain significant efficiency and maintain high accuracy, while still enabling tailored cost functions and domain-specific adaptations, an advantage in real-world sectors like finance.

Other common XAI methods like LIME [5] and SHAP [6] exist, but they provide explanations by assigning feature importance rather than producing clear, actionable changes in the input that could alter the classification. They both have adapted to the counterfactual world [18] but are mainly adapted to textual data and it has not been compared with many of the algorithms in the literature [2].

Our objective is to devise a novel counterfactual technique based on the *optimal mathematical solution* to the problem, using optimization methods and the model's derivatives to find the minimal shift needed for a given instance to change its output, fitting into the optimization-based category. We not only fulfill this similarity to the original instance by minimizing a distance function, validity and plausibility are guaranteed as well with the intrinsics of the method. Other features like minimality (of features changed), actionability, and diversity are achieved through a set of weights that determine the decisions and inner workings of the method in every step. This gives the ability, to either the entity using the algorithm or the client, to determine which features are actually changeable and how difficult it is to change it. In the field of diversity, while DiCE provides a discrete set of diverse counterfactuals, our method has the ability to provide a wide spectrum of explanations by changing the weights to adapt to different situations and the needs and interests of different

parties.

3 Methodology

The purpose of this project is to create a innovative XAI technique for differentiable models. We will initially restrict the analysis to differentiable models since the method requires the use of derivatives for optimization matters and for that, differentiability is needed. We will also assume that it is a binary classification model with a threshold This new technique, or method, falls into the Counterfactual Explanations family of XAI. As mentioned earlier in the motivation, a counterfactual reveals what should have been different in an instance to observe a diverse outcome.

Following the definition of a counterfactual, and inspired by some of the main properties mentioned earlier, the innovative method aims to fulfill, or have, as many properties as possible. With the validity and similarity properties in mind, which refer to counterfactuals that succeed in changing the classification output with the minimum change needed, the mathematical optimization method of Lagrange multipliers comes up with a direct application to the problem. In this optimization method, it is possible to find local minimum (or maximum but it doesn't apply to the problem) of a function with certain equation constraints. In the case of the counterfactual, we want to minimize the distance between the original instance and the new instance, subject to the condition of changing its classification output. This would fulfill both of the properties mentioned above. The equation with the Lagrangian multiplier becomes:

$$\mathcal{L}(\mathbf{x}, \lambda) = \ C(\mathbf{x}, \mathbf{x}') + \lambda \Big(M(\mathbf{x}) - \theta \Big) \tag{3.1}$$

where

- x is the new instance (the counterfactual) with m features,
- x' is the original instance,
- $C(\cdot)$ is a cost function (distance) measuring how far x is from x',
- $M(\cdot)$ is the differentiable model,
- θ is the threshold for changing the classification, and
- λ is the Lagrange multiplier.

The base of the cost/distance function is the weighted $L2$ norm, which can be defined as follows:

$$C(\mathbf{x}, \mathbf{x}', \mathbf{w}) = \sum_{i=1}^{m} \mathbf{w}_i \cdot (\mathbf{x}_i - \mathbf{x}'_i)^2 \tag{3.2}$$

We will discuss added regularizers added to this cost function later, but the main idea is to have a distance function that can be weighted by the user, or the entity using the method, to determine how easy or difficult it is to change each feature. This is a key part of the method, as it allows for flexibility and customization in the counterfactual generation process.

3.1 Mathematical solution

The minimum value of $\mathcal{L}$ is found when the partial derivatives w.r.t. $\mathbf{x}$ and λ are equal to 0. This implies:

$$F(\mathbf{x}, \lambda) \;=\; \nabla\mathcal{L}(\mathbf{x}, \lambda) \;=\; \begin{cases} \nabla C(\mathbf{x}) \;-\; \lambda \cdot \big(\nabla M(\mathbf{x})\big) \;=\; 0, \\ -\,M(\mathbf{x}) \;+\; \varepsilon \;=\; 0. \end{cases} \tag{3.3}$$

To solve this problem we will use the *Newton-Raphson* method to solve this mathematical problem, ensuring we reach the optimal solution. Newton's methods are used to find the roots of a function, or the solution to the equation $f(x) = 0$. In optimization, with $f \in C^2$, we are trying to find the roots of f', or the solutions to $f'(x) = 0$, known as critical points. These points can be minima, maxima or saddle points, but in the case of this problem we will try to find the minima of the function f.

For this, Newton-Raphson's method of optimization uses the first and second term of the Taylor expansion of the function to find that point iteratively [11]. For 1 dimension, or 1 variable, the update rule is $x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$. As seen in (3.3), we have two variables, so the we have to use the generalization of the iterative scheme using the hessian as the second derivative. Thus, the new iterative scheme obtained becomes

$$\big(\mathbf{x}_{n+1},\, \lambda_{n+1}\big) \;=\; \big(\mathbf{x}_n,\, \lambda_n\big) \;-\; \mathbf{H}^{-1}\big(\mathcal{L}(\mathbf{x}_n, \lambda_n)\big) \;\cdot\; \nabla\mathcal{L}\big(\mathbf{x}_n, \lambda_n\big).$$

If we set $F(\mathbf{x}, \lambda) \;=\; \nabla\mathcal{L}(\mathbf{x}, \lambda)$, the equation simplifies to

$$\big(\mathbf{x}_{n+1},\, \lambda_{n+1}\big) \;=\; \big(\mathbf{x}_n,\, \lambda_n\big) \;-\; \mathbf{J}\big(F(\mathbf{x}_n, \lambda_n)\big) \;\cdot\; F\big(\mathbf{x}_n, \lambda_n\big),$$

where

$$\mathbf{J}\big(F(\mathbf{x}_n, \lambda_n)\big) \;=\; \begin{pmatrix} H\big(C(\mathbf{x}_n)\big) \;-\; \lambda_n \cdot H\big(M(\mathbf{x}_n)\big) & -\nabla M(\mathbf{x}_n) \\ -\nabla M(\mathbf{x}_n) & 0 \end{pmatrix}.$$

Above, $H(\cdot)$ denotes the Hessian (second derivative) with respect to $\mathbf{x}$, and $\nabla(\cdot)$ denotes the gradient (first derivative).

A similar method of optimizing a loss function has been done in the past (BFGS [12]), but due to the computational complexity of calculating second derivatives, most of the work has been with its approximations. In this case, we will employ a recently created system, called *NeuralSens* [10], to calculate these second derivatives efficiently, eliminating the need

for approximations.

All of this will be implemented programmatically in `PyTorch`, using neural networks (or logistic regressions if there are no hidden layers), which, using activations like *sigmoid* or *tanh*, are at least twice-differentiable.

Notwithstanding, this mathematical approach has some challenges. First, in some iterations of some datapoints, the hessian can become singular or ill-conditioned, meaning that the inverse does not exist or it is very big and the update step explode. This causes problems to the overall convergence of the method for that certain point. Second, many real-life datasets include variables like integer and categorical columns. The method works very well with continuous variables but it does not have a direct and easy solution for dealing with these column types.

3.2 Singular Matrices

As discussed, one of the main problems when taking this optimization-based generation is the possibility of singular or ill-conditioned Hessians. This can most tipically happen because of two main things. If we have two variables that have a high correlation, they most likely will have a very small eigenvalue, which means that the matrix could become ill-conditioned or even singular. This is very easily solved by performing a bit of exploratory data analysis and removing highly correlated variables. This is already a common practice in the data science world, so it is assumed that it should not produce any problems.

The second problem is harder to solve, and it is the one we will be discussing in this section. This problem is when the gradient of the model is very close to 0, or the model has a flat curvature around a particular input. This leads again to a small eigenvalue, and the matrix becomes ill-conditioned. We explored some techniques like using the pseudo-inverse, damping techniques or switching to first-order updates for the affected iterations, but the experiments were not successful.To mitigate this issue, we implement an alternative update strategy when this type of ill-conditioning in the Hessian is detected.

The ulterior objective behind this method is to find a valid counterfactual, which ideally is a point close to the original with a different classification output. If the

To solve this problem, the alternative update step devised consists of using the normalized gradient of the model's output with respect to the input features to update.

$$\begin{cases} \mathbf{x}_{n+1} = \mathbf{x}_n - \frac{\nabla M(\mathbf{x_n})}{|\nabla M(\mathbf{x_n})|} \\ \lambda_{n+1} = \lambda_n \end{cases} \tag{3.4}$$

3.3 How Weights Shape the Distance

In our counterfactual explanation framework, the distance function (3.2) quantifies how "far" a candidate counterfactual $\mathbf{x}'$ lies from the original instance $\mathbf{x}$, where $\mathbf{w}_i \geq 0$ is a nonnegative *weight* that encodes the relative "cost" or difficulty of changing feature i.

- **High weights** ($w_i \gg 1$) make changes in feature i "expensive", discouraging the optimization from selecting that feature unless it is indispensable for flipping the model output.
- **Low weights** ($w_i \approx 0$) make changes in feature i "cheap", biasing counterfactuals toward modifying x_i first.

By tuning the vector $\mathbf{w}$, practitioners can encode domain knowledge about which features are easy or hard to change in the real world, and users can tune their explanations by encoding their desirability or actionability to alter some features over others.
Because each term in (3.2) is multiplied by w_i, a unit change for a certain feature i where $\left(x_i - x_i'\right)^2 = 1$ *costs* exactly $2w_i$.

$$\frac{\partial C}{\partial x_i'} = \frac{\partial}{\partial x_i'}\left(w_i \cdot \left(x_i - x_i'\right)^2\right) = -2 \cdot w_i \cdot (x_i - x_i')$$

This means that the higher w_i, the larger the gradient (in magnitude), penalising deviations in x_i' more aggressively.

In practice, one might begin with all $w_i = 1$ (equal cost), inspect the resulting explanation, and then increase w_i for features the end user finds unrealistic or non-actionable, thereby refining the counterfactuals to be truly *actionable*.

3.4 Discrete features

To address discrete variables, we found that imposing a regularizer to the distance function worked the best for the task. The regularizer is designed to take value 0 at the integer points. We explored different trigonometric funcions, such as sines and cosines with periods matching the integers, but some features were getting stuck on the maxima of that function, as it has derivative 0 as well and Newton's method is designed to find critical points in general. The function we finally proposed for solving the discrete challenge faced was

$$tan(\pi * x) \tag{3.5}$$

that looks something like

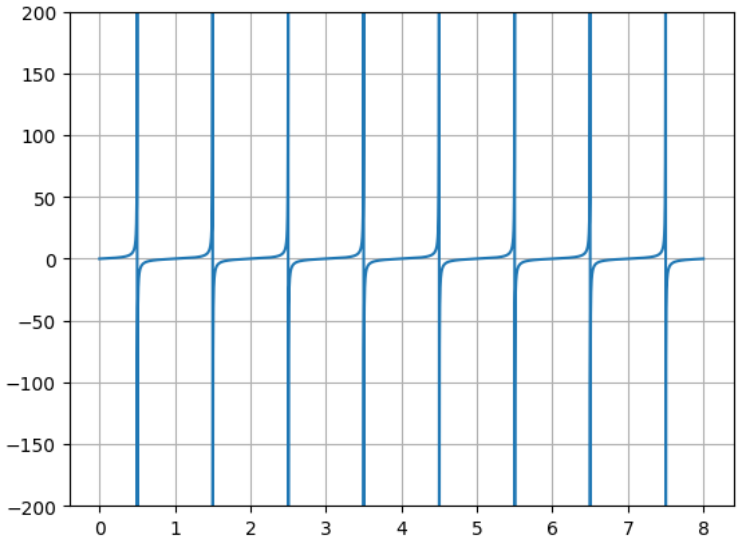

Figure 1: Tangent regularizer function

This function takes the value 0 in the integer points, but more importantly, they are saddle points, which are critical points in the newton's method. This means that iteratively, the features where the regularization is enforced will end up in one of those points, rather than in the possible optimum that could be a non-integer value. This regularizer is added to the cost function, so the new cost function becomes

$$C(\mathbf{x}, \mathbf{x}', \mathbf{w}) = \sum_{i=1}^{m} \mathbf{w}_i \cdot (\mathbf{x}_i - \mathbf{x}'_i)^2 + \sum_{j=1}^{n} tan(\pi \cdot x'_j) \tag{3.6}$$

where n is the number of discrete features.

A common practice in data-science is to standardize the data, meaning that the features are transformed to have mean 0 and standard deviation 1. This helps the model to converge faster and better. In the case of (3.6), it only works if the features are not standardized, as the periodic function is designed to have these critical points in the integers. If the features are standardized, the regularizer will not work as expected and the counterfactuals will not be valid. In this case, we can use a scaled version of the regularizer, which is designed to work with standardized features. The scaled version of the regularizer is defined as

$$C(\mathbf{x}, \mathbf{x}', \mathbf{w}) = \sum_{i=1}^{m} \mathbf{w}_i \cdot (\mathbf{x}_i - \mathbf{x}'_i)^2 + \sum_{j=1}^{n} tan(\pi \cdot (x'_j \cdot std_j + mean_j)) \tag{3.7}$$

where std_j and $mean_j$ are the standard deviation and mean of the feature j respectively, previously calculated with the whole dataset and before training the model. This way, the regularizer will work as expected and the counterfactuals will be valid.

Now we can use the regularizer to penalize the distance function for discrete features, ensuring that the counterfactuals generated will have integer values for those features. This is a key part of the method, as it allows for flexibility and customization in the counterfactual generation process, while still ensuring that the explanations are valid and actionable.
Nevertheless, we will not use this regularizer in every iteration. We want the algorithm to come close to the optimal solution without taking into account the integer features, so that the solution with integers comes close to the optimal solution. Then, we incude the regularizer in the cost function, and the algorithm will converge to the closest integer solution PROBAR SI FUNCIONA DESDE EL PRINCIPIO. When it is reaching the optimal solution with the regularizer, we *turn off* the regularizer, round the features to the closest integer and freeze them by setting the weights to ∞ so that it is virtually impossible to change that feature.

3.5 Entrenamiento y Validación

4 Experimentos

4.1 Objetivos

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Equation 5.4

4.2 Conjunto de Datos

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. (5.4)

4.3 Configuración

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. (5.2) (5.3)

4.4 Análisis de Rendimiento

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

```
import torch
import torch.nn as nn
import torch.optim as optim

model = MyNetwork()
optimizer = optim.Adam(model.parameters(), lr=1e-3)
criterion = nn.MSELoss()

for epoch in range(100):
```

```
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, targets)
        loss.backward()
        optimizer.step()
```

Algorithm 1: Entrenamiento de la red con Adam

4.5 Visualización

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

4.5.1 Graficas

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

5 Resultados

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

$$u_{tt} - c^2\,\Delta u = 0 \tag{5.1}$$

$$u_t = k\,u_{xx} + f(x,t), \tag{5.2}$$

$$u(0,x) = u_0(x),$$

$$u(t,0) = 0, \quad u(t,L) = 0. \tag{5.3}$$

$$\rho\big(u_t + u\cdot\nabla u\big) + \nabla p = \mu\nabla^2 u, \tag{5.4a}$$

$$\nabla\cdot u = 0. \tag{5.4b}$$

6 Conclusiones y Trabajos Futuros

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

References

[1] S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. *SSRN Electronic Journal*, 2017.

[2] R. Guidotti. Counterfactual Explanations and How to Find Them: Literature Review and Benchmarking. *Data Mining and Knowledge Discovery*, 38:2770–2824, 2024.

[3] F. Bodria, F. Giannotti, R. Guidotti, F. Naretto, D. Pedreschi, and S. Rinzivillo. Benchmarking and survey of explanation methods for black box models. *Data Mining and Knowledge Discovery*, 37(5):1719–1778, 2023.

[4] B. Ustun, A. Spangher, and Y. Liu. Actionable Recourse in Linear Classification. *Proceedings of KDD '19*, 2019.

[5] M. T. Ribeiro, S. Singh, and C. Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *arXiv preprint arXiv:1602.04938*, 2016.

[6] S. Lundberg and S.-I. Lee. A Unified Approach to Interpreting Model Predictions. *arXiv preprint arXiv:1705.07874*, 2017.

[7] S. N. Cohen, D. Snow, and L. Szpruch. Black-Box Model Risk in Finance. *SSRN Electronic Journal*, 2021.

[8] N. Ghatasheh. Business analytics using random forest trees for credit risk prediction: a comparison study. *International Journal of Advanced Science and Technology*, 72:19–30, 2014.

[9] Anonymous. Point of View: Using Random Forest for credit risk models (Machine learning and Credit Risk: a suitable marriage?), 2019.

[10] J. Pizarroso, J. Portela, and A. Muñoz. NeuralSens: Sensitivity Analysis of Neural Networks. *Journal of Statistical Software*, 102(7):1–36, 2022.

[11] J. Fliege, L. M. G. Drummond, and B. F. Svaiter. Newton's Method for Multiobjective Optimization. *SIAM Journal on Optimization*, 20(2):602–626, 2009.

[12] J. M. Papakonstantinou. Historical Development of the BFGS Secant Method and its Characterization Properties. 2009.

[13] H, M. Y. (2022). Loan default dataset. Kaggle. `https://www.kaggle.com/datasets/yasserh/loan-default-dataset`.

[14] Dutta, G. (2020). Loan defaulter. Kaggle. `https://www.kaggle.com/datasets/gauravduttakiit/loan-defaulter/data`.

[15] Coursera. Data Science Coding Challenge: Loan Default Prediction. `https://www.coursera.org/projects/data-science-coding-challenge-loan-default-prediction`.

[16] H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[17] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[18] Y. Ramon, D. Martens, F. Provost, and T. Evgeniou. A comparison of instance-level counterfactual explanation algorithms for behavioral and textual data: SEDC, LIME-C and SHAP-C. *Advances in Data Analysis and Classification*, 14(4):801–819, 2020.

[19] R. K. Mothilal, A. Sharma, and C. Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT* '20)*, pages 607–617, 2020.

[20] K. Mohammadi, A.-H. Karimi, G. Barthe, and I. Valera. Scaling Guarantees for Nearest Counterfactual Explanations. *arXiv preprint arXiv:2010.04965*, 2021.