BACHELOR'S DEGREE FINAL PROJECT

Explainable Machine Learning: Mathematical Optimization in Counterfactual Analysis

Presented by:
Agustín Martín Agüera

Supervised by:
Dr. Emilio Carrizosa Priego

FACULTY OF MATHEMATICS
Statistics and Operational Research Department
Sevilla, June 2022

Abstract

Counterfactual Analysis is a recent field of Explainable Machine Learning focused on the development of methods to provide human-understandable explanations for the results of predictive models. Further research and advancement in the area is deemed fundamental on account of the rapidly spreading usage of Machine Learning algorithms in human-impacting automated decision-making processes.

In this work we present an approach to Counterfactual Analysis through the formulation of optimization programs, posing single-objective and multi-objective problems that lead to counterfactual generation. We then particularize them for specific choices of classifier model and parameters, though further options are outlined. Computational experiments are conducted and reported for the multi-objective approach.

Resumen

El Análisis Contrafáctico es un área de creación reciente dentro del Aprendizaje Automático Explicable, que se centra en el desarrollo de técnicas que permitan obtener explicaciones aptas para interpretación humana para los resultados de modelos predictivos. Dada la rápida expansión de los algoritmos de Aprendizaje Automático en sistemas de decisión automática que tienen consecuencias directas sobre vidas humanas, la investigación en este campo se vuelve fundamental.

En este trabajo tratamos el Análisis Contrafáctico desde la formulación de programas de optimización matemática, proponiendo problemas de un solo objetivo y multiobjetivo. Posteriormente los particularizamos para elecciones específicas de modelo de clasificación y parámetros; aunque dejamos indicadas otras opciones. Por último, presentamos resultados numéricos de la aplicación a un caso práctico de problemas de la variedad multiobjetivo.

Contents

Abstract **3**

Resumen **5**

1 Introduction **9**
1.1 Machine Learning in automated decision-making 9
1.2 Counterfactual Analysis 10

2 Previous results **13**
2.1 Supervised Classification 13
2.2 Optimization problems 14
2.3 Multi-objective optimization 15
2.3.1 The Weighting Method 17

3 Counterfactual Optimization Problems **19**
3.1 A first approach 19
3.2 A multi-objective approach 20
3.3 On the flexibility of the problems 21

4 Classification models **25**
4.1 Variable encoding 25
4.2 Linear Classifiers 26
4.3 Logistic Regression 26
4.3.1 The binary case 27
4.3.2 Computation 27
4.3.3 Implementation 27
4.4 Support Vector Machines 28
4.4.1 Computation 29
4.4.2 Non-linear SVM and kernel methods 30
4.4.3 Implementation 31

5 Particularization of Counterfactual Problems **33**
5.1 The Simple Counterfactual Optimization Problem 35
5.2 The Multi-objective Counterfactual Optimization Problem 38

6 Numerical Illustration **41**
6.1 The German Credit Data dataset 41
6.2 Training the classifiers 42
6.3 Constraints and other settings 43
6.4 Results 44
6.5 Some comments on the results 49

References **51**

Chapter 1

Introduction

1.1 Machine Learning in automated decision-making

Machine Learning (ML) is the field that seeks to understand and build models with the ability to *learn*, that is, models that are *trained* on data to improve performance on some set of tasks [14]. As an extensively researched, computerized and effective tool, it is nowadays used in large-scale automation in many domains, which include but are not restricted to the objectives of Artificial Intelligence.

Its applications range from the extraordinary to the mundane, where ML algorithms are often used to support decisions that directly impact human lives. Such is the case of their usage in credit lending [17], medical treatment [6] or talent sourcing [16].

However, Machine Learning algorithms can be wrong. In spite of the cleanness of their theoretical formulation, training a ML model is a complex process which requires a deep dive in the data at hand, good unbiased error estimators, and many subtle adjustments of the parameters of the model. Even when correctly trained, sometimes models will learn biases that, while present in the data, are not acceptable as criteria for decision-making in certain domains, such as sex or race. When there are hard consequences on the line for humans, these biases need to be identified and accounted for.

More often than not this proves a challenging task due to the intricate complexity of some models, especially those trained over large volumes of data. This has led specialists in the field to coin the expression *black box model*, to refer to a predictive algorithm whose internal functioning is completely obscured. But even if no substantial bias is present, people deserve explanations about the machine-

informed decisions that affect their lives. In this sense, the EU recently extended its General Data Protection Regulation to include a form of this *right to explanation* [5].

The need for transparency in Machine Learning has led to the development of the field of *Explainable Machine Learning* (XML), referring both to (i) "inherently transparent" ML algorithms that produce results understandable to humans, and (ii) methods and techniques to generate post-hoc explanations for more opaque ML models. The main objectives of XML techniques in social applications include [18]:

- To shed light on the internal functioning of the algorithm, providing a way to detect and fix bugs and biases.
- To provide people with reasonable explanations about the decisions reached, weighting the role of each of their attributes.
- To challenge an decision felt unfair by the implied parts.
- To inform an interested party about the changes to be accomplished in order to alter an outcome.

Though the range of techniques included within XML is wide and varied, in this work we focus on a post-hoc explainability-enhancing technique named *Counterfactual Analysis*.

1.2 Counterfactual Analysis

The term *counterfactual* arises from the fields of philosophy, psychology and other social sciences, where it refers to a conditional statement for which the antecedent is false. It is an assessment of the supposed consequences of a situation or an action that never occurred, hence "counterfactual": "contrary to the facts". An example in this manner could be:

If there had been no World War II, then the UN would not exist today.

In those fields, counterfactuals are used as tools for the exploration of human reasoning and decision-making. Thus, the unreal situations need not actually be plausible at the time of the analysis, and can stay as removed from reality as one desires.

That is not the case in the setting of Machine Learning, where they were first proposed in 2017 by Wachter et al. [19]. As a method integrated in XML, counterfactual analysis tackles the question: "Given a trained classifier and an instance of data, how would its predicted class change if the data was slightly

different?". Or, more often and conversely: "How different must the datapoint be (and in what manner) for the predicted class to be different?"

Thus, the counterfactuals we are interested in are not removed from reality, but the opposite: they must represent realistic, attainable states, similar enough to the original datapoint to be worth considering, but assigned a different label in classification. A counterfactual statement of this type would be akin to:

If the applicant had a higher disposable income, they would be granted the credit.

An instance of counterfactual analysis for a particular datapoint offers new datapoints in the vicinity of the original whose predicted class is different from that of the original. Such points are termed *counterfactual explanations* or *counterfactual instances* of the original. They should present a set of desirable properties, which include [18]:

(1) **Validity**. We say that a counterfactual instance is *valid* if its predicted class is different to that of the original instance. If the number of possible classes is greater than two, we will often designate a *desired class*. Validity is fundamental to the definition of counterfactual instance.

(2) **Proximity to the original instance**. A counterfactual instance must stay close to the original instance in order to be meaningful to the corresponding user.

(3) **Feasibility**. We term a counterfactual instance *feasible* if it represents a realistic combination of features. Surely this is essential for the analysis to hold any meaning at all. Feasibility is enforced by banning all non-realistic combinations and by attending to the *causal relationships* between features. It can be further enhanced by accounting for *data manifold closeness*.

(4) **Actionability**. A feature of the data is actionable if it represents an attribute that can reasonably be changed. Thus, it must be mutable and under control of the person that it applies to. An example would be "daily protein consumption" as opposed to "ethnicity". A counterfactual instance is likewise termed *actionable* if it only differs from the original instance in actionable features. Actionability is necessary for counterfactual instances to be useful to users.

(5) **Sparsity**. A counterfactual instance is termed *sparse* if it only differs from the original in a few features. It has been found [13] that people understand shorter explanations more easily, and they provide a clearer guideline towards label change. The prioritization of features may also offer clearer insight about the classifier's functioning and biases.

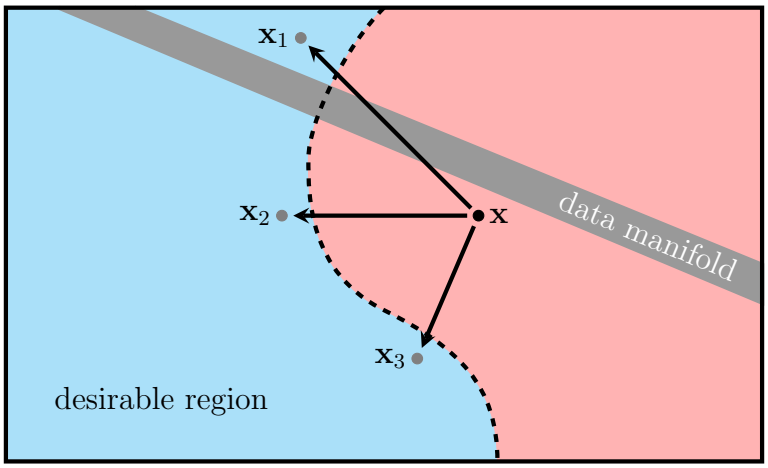

Figure 1.1: Counterfactual explanations for a given instance.

We present a simplified visualization of Counterfactual Analysis in figure 1.1, where three counterfactual explanations are given for instance $\mathbf{x}$. All three of them are valid, though $\mathbf{x}_3$ is the closest for the euclidean distance, $\mathbf{x}_2$ is the sparsest and $\mathbf{x}_1$ is closest to the data manifold, and thus probably the most realistic. It is obvious that criteria must be developed to weight the desired properties in order to choose an optimal counterfactual instance.

In this work we intend to describe general methods to generate counterfactual instances that reasonably comply with all required properties, through the formulation of appropriate mathematical optimization problems.

The present work is organized as follows. In chapter 2 we present some results and notation to be used through the document. Then in chapter 3 we formulate two different approaches to the counterfactual optimization problem, through single-objective and multi-objective optimization respectively. In chapter 4 we briefly describe and discuss three classifier models, for which we particularize the previously posed problems in chapter 5. Finally, in chapter 6 we give practical examples by applying the particularized problems to real data.

Chapter 2

Previous results

In this chapter we present some previous results and notation to be used throughout this work.

2.1 Supervised Classification

A fundamental problem in Machine Learning [9] is that of *Supervised Classification*, the task of defining procedures for classifying objects in a finite set Ω into a set $\mathcal{C}$ of *classes* or *labels*.

Objects in Ω are usually described by a set of *variables* $\{X_j\}, 1 \leq j \leq M$, each of them taking real, integer or categorical values. We note the vector $\mathbf{x} = (x_1, \ldots, x_M)$ the *predictor vector* of each element of Ω.

If we note X the space of predictor vectors, then the task at hand is to arrive at a *classification rule*, a function $g : X \longrightarrow \mathcal{C}$ assigning a class or label $g(\mathbf{x}) \in \mathcal{C}$ to each vector $\mathbf{x} \in X$.

Rule g is sought to accurately label a set of given instances whose predictor vectors and class membership are known, the *training sample* $I \subset \Omega$. However, g has the broader objective of correctly classifying any further instance of Ω.

From here on, we will relax the notation by referring to the classes as natural numbers, this is, $\mathcal{C} = \{1, ..., K\}$ in multi-class classification; and $\mathcal{C} = \{-1, 1\}$ in binary classification. We will also leave Ω behind to work directly with the space X.

Models and training

A supervised classification model defines a rule $g_{\mathbf{w}}$ dependent on a vector of parameters, termed *weights* $\mathbf{w}$. It also defines a loss function $\mathcal{L} : \mathcal{C} \times \mathcal{C} \longrightarrow \mathbb{R}$ mea-

suring the accuracy of the classification that $g_{\mathbf{w}}$ induces on the training sample, to which a regularization term –measuring complexity of the rule– is frequently added. An optimization problem is then posed to find the optimal parameter vector $\hat{\mathbf{w}}$, which minimizes the cost $\mathcal{L}$. The classifier is then said to be trained, and rule $g_{\hat{\mathbf{w}}}$ will be useful in classifying further elements of X, provided that the training sample was representative enough.

As a foundational problem of the field, today we find a wide and rich range of approaches to the supervised classification problem [3]. Nonetheless, all of them require the usage of techniques from mathematical optimization to arrive at the optimal rule $g_{\hat{\mathbf{w}}}$ within the specific method.

Scoring functions

A common approach to devising a classification rule is the usage of *scoring functions*: a function $f_k : X \longrightarrow \mathbb{R}$ is built for each class $k \in \{1, ..., K\}$ as a measure of the likelihood of point $\mathbf{x}$ belonging in class k. The classification rule is then to assign the class corresponding to the highest score:

$$g : \mathbf{x} \mapsto g(\mathbf{x}) = \arg \max_{1 \leq k \leq K} f_k(\mathbf{x}) \tag{2.1}$$

If ties were to be found, a class would be chosen randomly among those at which the maximum is attained.

The binary case is susceptible to further simplification: the prediction is encoded in the *sign* of $f_1(\mathbf{x}) - f_{-1}(\mathbf{x})$. Thus, we may define $f : \mathbf{x} \longrightarrow f(\mathbf{x}) = f_1(\mathbf{x}) - f_{-1}(\mathbf{x})$ and state the classification rule as:

$$g : \mathbf{x} \mapsto g(\mathbf{x}) = sign(f(\mathbf{x})) \tag{2.2}$$

These methods are of interest to us because of how the scores offer a measure of the certainty of classification, rather than just a label. These will be the focal point of our multi-objective optimization problems.

2.2 Optimization problems

Mathematical Optimization problems –usually called *programs*– are frequently classified in terms of their *objective function* and their *constraints*. This classification allows for the formulation of general results and algorithms, so that they may be applied to the resolution of problems that are similar enough to each other.

In this work we aim not only to formulate optimization programs, but to obtain problems that are (i) solvable by existing and off-the-shelf tools and (ii) faithful to the situations that may arise in practical applications. With this in sight, we will be formulating Mixed-Integer Convex Quadratic Programs (MICQP); since they are general enough for our purposes while sufficiently documented in the literature.

Convex optimization problems are of the form (2.3), $f_0, f_1, \dots f_j$ being convex functions. Convexity is enough to assure that any local optimum is a global optimum [1], and this guarantees convergence and optimality of the local-search algorithms employed.

$$\begin{array}{ll} \text{minimize} & f_0(\mathbf{x}) \\ \text{subject to} & f_j(\mathbf{x}) \leq 0 \quad j = 1, ..., p \end{array} \tag{2.3}$$

Convex Linearly-Constrained Quadratic Programs also follow form (2.3), while requiring f_0 to be a convex quadratic function; and f_i to be affine functions. A more transparent formulation would be (2.4).

$$\begin{array}{ll} \text{minimize} & (1/2)\mathbf{x}^T P\mathbf{x} + \mathbf{q}^T\mathbf{x} + r \\ \text{subject to} & G\mathbf{x} \leq \mathbf{h} \\ & A\mathbf{x} = \mathbf{b} \end{array} \tag{2.4}$$

where P is a symmetric and positive semi-definite matrix. As a subset of convex problems, convex quadratic programs share the property of local optima being global optima.

Mixed-integer programs are those in which some decision variables are only allowed to take integer values. This condition requires the usage of completely different algorithms than the continuous case. Nevertheless the inclusion of integer-valued variables in our description proves fundamental, as it is the natural way of dealing with categorical and ordinal attributes.

2.3 Multi-objective optimization

As will be shown throughout this document, it is in our interest to consider Multi-objective Optimization problems and some methods for their reduction to scalar optimization problems.

We will consider a multi-objective optimization problem of the form

$$\begin{array}{ll} \text{minimize} & \{f_1(\mathbf{x}), f_2(\mathbf{x}), ...f_K(\mathbf{x})\} \\ \text{subject to} & \mathbf{x} \in S \end{array} \tag{2.5}$$

in which we have $K \geq 2$ *objective functions* to minimize over a *feasible set* S. For this problem to be of interest, the objective functions must be conflicting in some way, which does not allow for the simultaneous minimization of them all.

We usually denote the image of the feasible region, the *outcome set*, by $Z = \mathbf{f}(S)$, and its elements by $\mathbf{z} = (z_1, z_2, ..z_K)^T$. These are the *outcome vectors*.

We say a objective vector $\mathbf{z}^*$ *dominates* another objective vector $\mathbf{z}$ if $z_i^* \leq z_i$ $\forall i = 1...K$ and there is at least one j such that $z_j^* < z_j$. This is to say, $\mathbf{z}^*$ offers a better value for objective function f_j, without prejudice to any other objective function.

An ideal outcome would be feasible ($\mathbf{z}^* \in Z$) while dominating all other feasible solutions. Such vector is, more often than not, non-existent. This is the case in figure 2.1: $\mathbf{z}^*$ dominates all feasible outcomes, but is not feasible itself.

Nonetheless, the notion of dominance gives rise the *Pareto optimal set* (POS). An objective vector $\mathbf{z}$ is said to be Pareto optimal if it feasible and it is not dominated by any other feasible objective vector. In figure 2.1, the Pareto optimal set is represented in red, while vector $\hat{\mathbf{z}}$ is Pareto optimal.

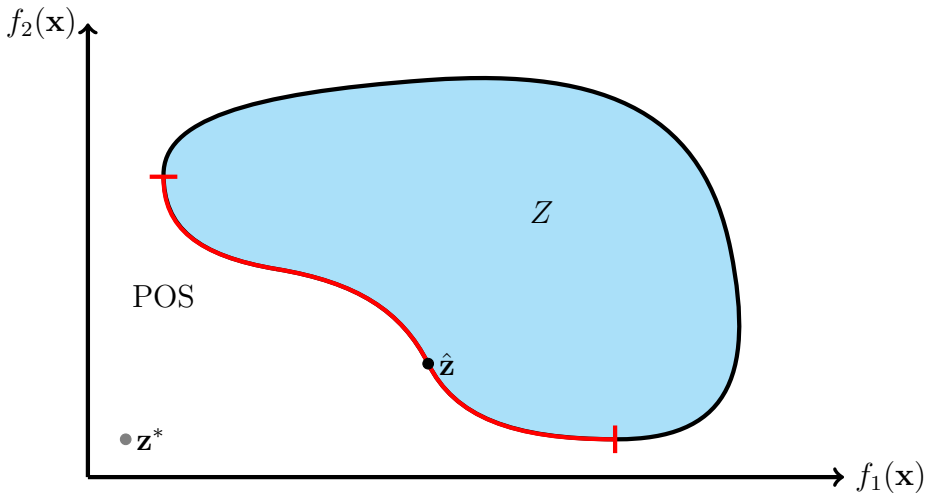

Figure 2.1: Representation of an outcome set.

Even when no ideal all-minimizing objective vector is available, it is sensible to search for a solution to the multiobjective problem within the POS, since all other

feasible outcomes are dominated, and so they would give rise to less-than-optimal solutions.

2.3.1 The Weighting Method

A wide range of strategies used to obtain satisfactory solutions to problem (2.5) involve the generation of (part of) the POS to choose a point from. These are named *a posteriori methods* [12], since the choice of optimal solution is made after the computations.

We will be applying one of these: the *Weighting Method*, where each objective function is assigned a weight coefficient; and the weighted sum of the objectives is minimized. In this way the problem is reduced to a scalar one, to be solved through usual means. Problem (2.5) is thus modified into the following *weighting problem*, where coefficients ω_i are real, non-negative numbers that add up to one.

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{K} w_i f_i(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in S \end{array} \tag{2.6}$$

There are results assuring Pareto-optimality of the solution of problem (2.6) if all weights are positive, or if the solution is unique. Furthermore, if the multi-objective optimization problem is convex, then every Pareto optimal point is solution to a weighting problem with an appropriate weighting vector [12]. Then, under convexity assumptions, one can generate any PO vector by solving the related weighting problem.

In the bi-objective case there is only one degree of freedom in the values of the weights. This allows for an easy numeric exploration and representation of the POS, in such a way that a decision maker could easily contrast PO solutions to their preference.

Chapter 3

Counterfactual Optimization Problems

In this chapter we develop an approach to Counterfactual Analysis through mathematical optimization, proposing two alternative problems to pose and solve in order to get counterfactual explanations [2].

3.1 A first approach

A first idea is to find, for a misclassified instance, the closest (for some notion of closeness) feasible point in the desired region. Membership of such counterfactual to the desired class may be somewhat weak, as it is very near or even sitting on the border between classification regions. Thus, it might not be something to recommend to a user searching to change their classification, but it would offer insight on the classifier's internal working.

Definition 3.1.1 *Let X be a space of predictor vectors, and $\mathcal{C} = \{1, \ldots, K\}$ a set of K labels. Let $g : X \longrightarrow \mathcal{C}$ be a classification rule for the corresponding supervised classification problem. Furthermore, let us have:*

(1) A set of inequality constraints $\{h_j(\mathbf{x}) \leq 0\}_{1 \leq j \leq p}$ to be imposed on the counterfactual.

(2) A distance $D : X \times X \longrightarrow \mathbb{R}$

*Finally, let us have a point $\mathbf{x_0} \in X$ and a desired label $k \in \mathcal{C}$. Then the **Simple Counterfactual Optimization Problem** (SCOP) for $\mathbf{x_0}$ into class k is*

$$\begin{array}{ll} \arg\min_{\mathbf{x} \in X} & D(\mathbf{x}, \mathbf{x_0}) \\ \text{subject to} & g(\mathbf{x}) = k \\ & h_j(\mathbf{x}) \leq 0, \qquad 1 \leq j \leq p \end{array} \tag{3.1}$$

Observation 3.1.1 *When the classifier in question makes use of scoring functions, the class condition in problem (3.1) may be written in terms of those scoring functions:*

$$g(\mathbf{x}) = k \iff k = \arg\max_{1\leq k\leq K} f_k(\mathbf{x}) \iff f_k(\mathbf{x}) \geq f_i(\mathbf{x}), \quad 1 \leq i \leq K, \quad i \neq k \tag{3.2}$$

Furthermore, if it is a binary classification problem, then the class condition is written as a single inequality:

$$g(\mathbf{x}) = 1 \iff f(\mathbf{x}) \geq 0 \tag{3.3}$$

3.2 A multi-objective approach

We will propose another way of handling the validity condition; since as already stated, solutions to problem (3.1) are not likely to be robust. In the score-based case, one or many of the $K-1$ inequalities are likely to end up saturated while solving the optimization problem. Counterfactual $\mathbf{x}$ is then on the border of the desired region, and any slight change in the classifier's internal settings may be capable of letting $\mathbf{x}$ back into misclassification.

This is a concern, for instance, if the classifier in question is trained by a Online Machine Learning algorithm. Online Learning methods –as opposed to the usual Batch Learning methods– involve the sequential training of the model as new data becomes available. It is a common technique in situations where the classifier has a need to dynamically adapt to changes in the patterns of data [7].

Fortunately, we can find a solution for score-based classifiers, where scoring functions offer a measure of the *certainty* of classification in each class. Thus, we propose other formulation for the counterfactual optimization problem in these situations: to (i) minimize the distance to the original point *while* (ii) maximizing the difference between scores of the desired class and the rest.

Definition 3.2.1 *Let X be a space of predictor vectors, and $\mathcal{C} = \{1, \ldots, K\}$ a set of K labels. Let $\{f_k : X \longrightarrow \mathbb{R}\}_{1\leq k\leq K}$ be the respective scoring functions for a supervised classification problem. Furthermore, let us have:*

(1) A set of inequality constraints $\{h_j(\mathbf{x}) \leq 0\}_{1\leq j\leq p}$ to be imposed on the counterfactual.

(2) A distance $D : X \times X \longrightarrow \mathbb{R}$

. Finally, let us have a point $\mathbf{x_0} \in X$; and for the sake of simplicity we will take K as our desired label, without loss of generality. Then the ***Multi-objective***

Counterfactual Optimization Problem *(MCOP) for* $\mathbf{x_0}$ *into class* K *is*

$$\begin{array}{ll} \arg\min_{\mathbf{x}\in X} & \{D(\mathbf{x},\mathbf{x_0}), f_1(\mathbf{x}) - f_K(\mathbf{x}), \ldots, f_{K-1}(\mathbf{x}) - f_K(\mathbf{x})\} \\ \text{subject to} & h_j(\mathbf{x}) \leq 0, \quad 1 \leq j \leq p \end{array} \tag{3.4}$$

This is a multi-objective optimization problem with K *objectives.*

Observation 3.2.1 *The binary case is written rather simply in terms of the only scoring function as a two-objective optimization problem:*

$$\begin{array}{ll} \arg\min_{\mathbf{x}\in X} & \{D(\mathbf{x},\mathbf{x_0}), -f(\mathbf{x})\} \\ \text{subject to} & h_j(\mathbf{x}) \leq 0, \quad 1 \leq j \leq p \end{array} \tag{3.5}$$

We propose the use of the Weighting Method as a tool to scalarize and solve to MCOP. Iterative solving through a sweep in the space of possible weights gives a variety of solutions from which to choose, attending to the trade-off between distance and scores. Figure 3.1 shows a simplified representation of this method for instance $\mathbf{x}$.

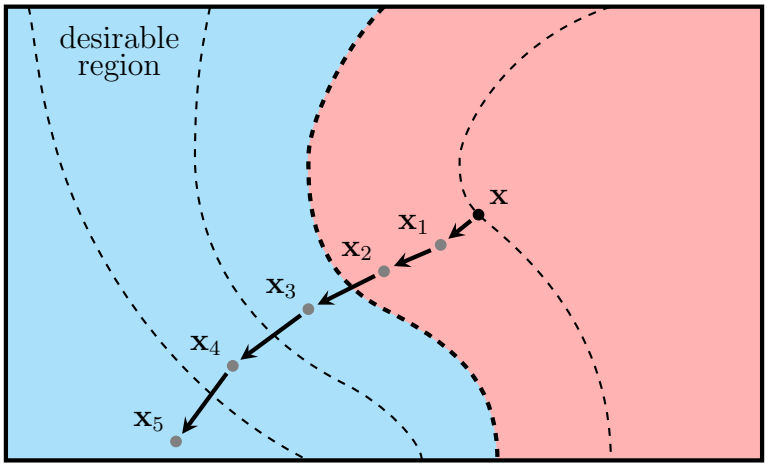

Figure 3.1: Different solutions to the MCOP for a given instance.

3.3 On the flexibility of the problems

The definitions given above are general enough to accommodate for a variety of situations and approaches to the problem, as the choices of distance and constraints for a given problem are not unique; and different methods may report results of varying interest. Moreover, they will likely change the character of problems (3.1), (3.4) and (3.5) as optimization programs, and thus the algorithms that may be used to solve them. We first need to define space X, distance D, and constraints h_j.

The predictor space X. In order to better handle and solve the classification problem, categorical and ordinal variables are usually encoded as binary and integer variables, respectively. Space X is thus a product of the form $\mathbb{R}^{m_1} \times \mathbb{Z}^{m_2} \times \{0,1\}^{m_3}$. This is, the (almost certain) presence of categorical or non-continuous variables always implies a mixed-integer character of the proposed problems.

Distance D. The choice of distance between points is not free of consequence. If we focus on Minkowski metrics, we find an interesting array of features:

- The $L2$ distance seems a natural choice of distance based on the geometric intuition of proximity. If we use $(L2)^2$ then the distance term is quadratic in the predictor variables, which is a desirable feature for the optimization problem.

$$L2(\mathbf{x}, \mathbf{x}_0) = \sum_i (x_i - x_{0i})^2 \tag{3.6}$$

- The $L1$ distance is similar to $L2$, but it assigns a greater distance to situations where the difference between points is spread between variables. This makes it an asset for promoting sparsity of the counterfactual.

$$L1(\mathbf{x}, \mathbf{x}_0) = \sum_i |x_i - x_{0i}| \tag{3.7}$$

  The $L1$ distance can be linearized by means of a norm cone, in order to keep the problem linear and convex. However, this introduces further real variables t_i and new constraints.

$$\begin{array}{ll} \text{minimize} & L1(\mathbf{x}, \mathbf{x_0}) \\ \text{subject to} & \mathbf{x} \in X \end{array} \quad \equiv \quad \begin{array}{lll} \text{minimize} & \sum_{i=1}^{M} t_i & \\ \text{subject to} & \mathbf{x} \in X & \\ & t_i \in \mathbb{R} & 1 \leq i \leq M \\ & x_i - x_{i0} \leq t_i & 1 \leq i \leq M \\ & -(x_i - x_{i0}) \leq t_i & 1 \leq i \leq M \end{array} \tag{3.8}$$

- The $L0$ distance, as defined by expression (3.9), only reflects whether variables have the same value or differ. This makes it our best tool to promote sparsity.

$$L0(\mathbf{x}, \mathbf{x}_0) = \sum_i I(x_i, x_{0i}), \quad \text{where} \quad I(x, x') = \begin{cases} 0 & \text{if} \quad x = x' \\ 1 & \text{if} \quad x \neq x' \end{cases} \tag{3.9}$$

To linearize the L0 distance, we may use the big-M method. This way, binary decision variables s_i and new constraints are introduced. The idea is that for every variable i, $s_i = 1$ if $x_i \neq x'_i$, $s_i = 0$ otherwise.

$$\begin{array}{ll} \text{minimize} & L0(\mathbf{x}, \mathbf{x_0}) \\ \text{subject to} & \mathbf{x} \in X \end{array} \quad \equiv \quad \begin{array}{lll} \text{minimize} & \sum_{i=1}^{M} s_i & \\ \text{subject to} & \mathbf{x} \in X & \\ & s_i \in \{0,1\} & 1 \leq i \leq M \\ & x_i - x_{i0} \leq M' s_i & 1 \leq i \leq M \\ & -(x_i - x_{i0}) \leq M' s_i & 1 \leq i \leq M \end{array} \tag{3.10}$$

where M' needs to be large enough to account for every possible variation in a variable. A possible choice for it would be:

$$M' = 2 \max |(x_i)_j| \tag{3.11}$$

A convex combination of the three of them will be used in the formulation of the concrete problems that we will propose, so as to benefit from the features of all three.

Furthermore, there is the question of standardization. Usually variables come in very different units and numeric ranges, which are not indicative of the relative difficulty of changing the value of the variable. If we think of the credit-lending example, the difficulty in changing one's disposable income by 10 (euros) is not equivalent to that of changing one's age by 10 (years). In the computation of distances, variables must be weighted to account for the units scales that they are expressed in –standardization– and their "resistance to movement". Thus, we will consider the weighted versions of $L2$ and $L1$ distances. But far from being an obstacle, this opens the door to new ways to further tailor the problems.

One could, for example, ask users about their perceived relative difficulty in changing each variable; and use the input as a guide when computing weights for the variables. This human-supervised method would account for users' preferences, which is a desirable feature for counterfactual generation.

On the other hand, there is an argument for replacing the weighted $L2$ distance with the *Mahalanobis distance*, as the latter rewards closeness to the data manifold; and thus is likely to present counterfactuals that are more realistic than the former.

Constraints h_j reflect conditions that must hold true for a counterfactual to be feasible. For example, it would be nonsensical to suggest to an user that they decrease their age; or get a higher-income job while maintaining the duration of their current post.

Constraints can also be used to impose limits on the change of variable values, to prevent an excess of variation in any given one. They may also help enforce sparsity, by setting a hard maximum on the number of features that can change. For instance, a recent paper [11] sets the limit at 2 variables for a "good" counterfactual. Finally, some variables may not be considered actionable, and so their values must stay fixed in the counterfactual generation.

Chapter 4

Classification models

In this chapter we give a brief presentation of the supervised classification models on which we will apply counterfactual analysis. We begin by assessing the case of logistic regression, as one of the simplest and most easily interpretable models. We then keep on with the Support Vector Machine model (SVM), a powerful and flexible method stemming from statistical learning. In SVMs, kernels methods are used to increase the model's flexibility, altering the shape of the boundary that separates the two classes, and thus modifying the form of the validity condition in our counterfactual optimization problems.

4.1 Variable encoding

Most classification models are designed to only accept numerical values as inputs, the implication being that categorical and ordinal variables must be encoded prior to their processing. Ordinal values are usually subject to *integer encoding*, which maps the ordered categories to integers so as to preserve the order relations. In contrast, categorical variables pass through *one hot encoding*.

One-hot encoding maps a categorical variable c with categories $1 \dots n$ to a vector $(c_1, \dots, c_n) \in \{0,1\}^n$, in such a way that $c_j = 1$ if $c = j$, $c_j = 0$ otherwise. This ensures a numerical codification in which no order relation is unwillingly introduced. It is also a common approach to get rid of one of the new variables, and let the null vector $(0, \dots, 0)$ represent the corresponding category.

Subsequently, the predictor space X is mapped to a subset of $\mathbb{R}^{M'}$. We will take this to be the case from now on.

4.2 Linear Classifiers

Definition 4.2.1 *We define a linear (score-based) classifier as one whose scoring functions are affine transformations of the predictor vector. This is, the classification rule is of the form:*

$$\exists\{\beta_k\}_{1\leq k\leq K} \in \mathbb{R}^M, \quad \{\beta_{k0}\}_{1\leq k\leq K} \in \mathbb{R} \qquad f_k(x) := \beta_k^T \mathbf{x} + \beta_{k0} \quad \forall \mathbf{x} \in X \tag{4.1}$$

$$g : \mathbf{x} \in X \mapsto f(\mathbf{x}) = \arg\max_{1\leq k\leq K} f_k(\mathbf{x}) \tag{4.2}$$

Observation 4.2.1 *In the binary case only one affine function is needed, in accordance with 2.2.*

$$\exists\beta \in \mathbb{R}^M, \quad \beta_0 \in \mathbb{R} \qquad g(\mathbf{x}) = sign(\beta^T \mathbf{x} + \beta_0) \quad \forall \mathbf{x} \in X \tag{4.3}$$

When using a linear score-based classifier, the locus of points classified in any class k is a polyhedron. Thus, it is convex and its boundary is piecewise linear. If the bounds and further constraints to be added to the SCOP keep this convexity and linearity, the resulting problem will be, at most, a Mixed-Integer Convex Quadratic Program (MICQP). Thus, there is a variety of optimization techniques that can be used to solve the problem which would not be applicable otherwise.

Similarly, affine scoring functions imply linear classification objectives for the MCOP, so that any single-objective problem resulting from the weighting method will also be, at most, a MICQP.

Thus, linearity plays an important enough role in our counterfactual problems that it is desirable to classify methods by this criterion. Examples of linear classifiers are the Logistic Regression classifier and the Support Vector Machine with a linear kernel. The other (non-linear) classifiers present boundaries of diverse shapes, which require either particular attention to each, or the usage of more general (and less accurate) solving algorithms.

4.3 Logistic Regression

Logistic Regresion models are linear score-based classifiers based on the modelling of posterior probabilities of the classes ($\mathrm{P}[k|X = \mathbf{x}]$) by means of the *logit transformation* [9].

In multi-class classification problems, a class is taken as reference (here we will take class K), in such a way that the logarithm of the odds of any class to the reference class is modelled as an affine function of $\mathbf{x}$.

$$\log\left(\frac{\mathrm{P}[k|X=\mathbf{x}]}{\mathrm{P}[K|X=\mathbf{x}]}\right) = f_k(\mathbf{x}) = \beta_k^T\mathbf{x} + \beta_{k0}, \qquad 1 \leq k \leq K-1 \tag{4.4}$$

This gives us $K-1$ functions, though to comply with our original definition of a score-based classifier, we could consider a K-th, which would be the constant null function. It is easy to check that these functions preserve the order relations showcased by the posterior probabilities that they correspond to:

$$\mathrm{P}[k|X=\mathbf{x}] \geq \mathrm{P}[k'|X=\mathbf{x}] \iff f_k(\mathbf{x}) \geq f_{k'}(\mathbf{x}), \qquad 1 \leq k,\, k' \leq K-1 \tag{4.5}$$

$$\mathrm{P}[k|X=\mathbf{x}] \geq \mathrm{P}[K|X=\mathbf{x}] \iff f_k(\mathbf{x}) \geq 0, \qquad 1 \leq k \leq K-1 \tag{4.6}$$

Thus, we use these $K-1$ affine functions for the first $K-1$ scores, and the constant null function for the last one. Note that this result is similar to the simplification we found for score-function methods in the binary case, where one of the classes was used as reference, leaving one less function (and its parameters) to be obtained through optimization.

4.3.1 The binary case

The binary classification case is completely analogous to the multi-class one, though now we may keep just one scoring function:

$$f(\mathbf{x}) = \beta^T\mathbf{x} + \beta_0 = \log\left(\frac{\mathrm{P}[1|X=\mathbf{x}]}{\mathrm{P}[-1|X=\mathbf{x}]}\right) = \log\left(\frac{\mathrm{P}[1|X=\mathbf{x}]}{1-\mathrm{P}[1|X=\mathbf{x}]}\right) \tag{4.7}$$

4.3.2 Computation

As we are able to express the posterior probabilities of the classes in terms of the parameters $\{\beta_k, \beta_{k0}\}_k$, it is appropriate to estimate these parameters by maximum likelihood. This is the most common approach when training logistic classifiers, giving rise to a non-linear optimization problem that is solved iteratively. A regularization term is frequently added to prevent weights from getting too large [9].

4.3.3 Implementation

For numerical illustration purposes, we will be working with Logistic Regression classifiers in Python as implemented in the Scikit-Learn library [15], in particular in module `sklearn.linear_model` as function `LogisticRegression()`.

4.4 Support Vector Machines

Support Vector Machines (SVM) are statistical learning models first formulated by Boser, Guyon and Vapnik in 1992. The have been successfully applied to a variety of classification problems, such as handwritten digit recognition, text categorization, cancer classification and protein secondary-structure prediction [10].

They are score-based models, available in both linear and non-linear versions, by means of *kernel methods*. Here we will first describe the original (linear) formulation, which was built for binary classification, and then expand on the usage of kernels.

The idea behind linear SVMs is to find an optimal separating hyperplane in the predictor space. This hyperplane would not only completely separate instances according to their class, but also maximize the margins between the plane and the instances. If the datasets are not linearly separable, then intermission of points into the margins or the opposite region is allowed but minimized; an approach that has been dubbed *soft-margin*, as opposed to the original *hard-margin* strategy.

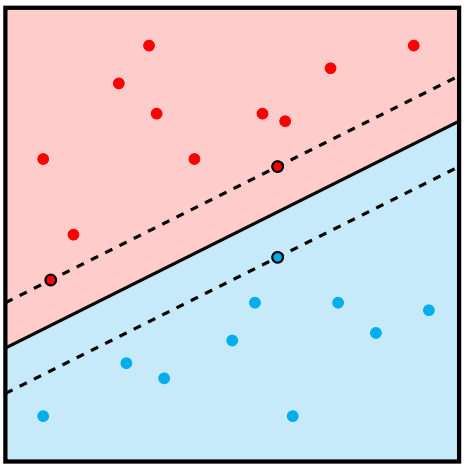

Figure 4.1: Hard-margin linear SVM

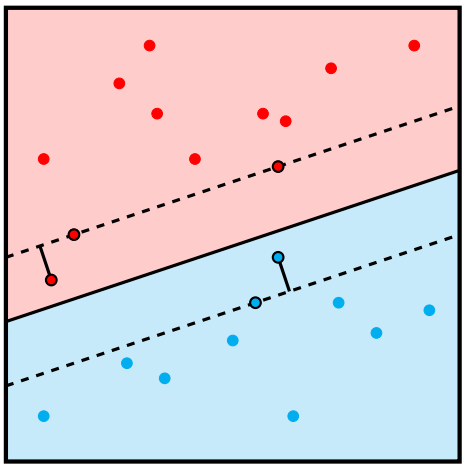

Figure 4.2: Soft-margin linear SVM

As a binary linear score-based classification method, its classification rule is of the form given by equation 2.2, this is:

$$g(\mathbf{x}) = sign(f(\mathbf{x})) = sign(\beta^T\mathbf{x} + \beta_0), \qquad \beta \in \mathbb{R}^M, \quad \beta_0 \in \mathbb{R} \tag{4.8}$$

4.4.1 Computation

As stated, parameters β and β_0 in 4.8 are supposed to maximize the margin between datapoints and the separating hyperplane. The optimization problem to be solved is the following:

$$\begin{array}{lll} \arg\min_{\beta,\beta_0} & \frac{1}{2}\|\beta\|^2 + C\sum_{j=1}^{n}\xi_j & \\ \text{subject to} & y_j(\beta^T\mathbf{x}_j + \beta_0) + \xi_j \geq 1 & 1 \leq j \leq N \\ & \xi_j \geq 0 & 1 \leq j \leq N \end{array} \tag{4.9}$$

where the slack variables ξ_i measure the instances' violation of the margins, and are non-negative. C is a positive tuning constant which balances the width of the margin and the number and depth of violations, affecting the "softness" of the margins.

The solving of problem (4.9) is of special interest, as it gives insight into the idea of kernel methods. It is a quadratic linearly-constrained program, and so we can describe a solution by using Lagrange multipliers. [9]

By means of the Lagrange primal and dual functions and the Karush-Kuhn-Tucker conditions, the solution of problem (4.9) is characterized by expressions (4.10 - 4.16).

$$\beta = \sum_{j=1}^{N} \alpha_j y_j \mathbf{x}_j \tag{4.10}$$

$$0 = \sum_{j=1}^{N} \alpha_j y_j \tag{4.11}$$

$$\alpha_j = C - \mu_j, \qquad 1 \leq j \leq N \tag{4.12}$$

$$\alpha_j \left[y_j(\beta^T\mathbf{x}_j + \beta_0) - (1 - \xi_j) \right] = 0, \qquad 1 \leq j \leq N \tag{4.13}$$

$$\mu_j \xi_j = 0, \qquad 1 \leq j \leq N \tag{4.14}$$

$$y_j(\beta^T\mathbf{x}_j + \beta_0) - (1 - \xi_j) \geq 0, \qquad 1 \leq j \leq N \tag{4.15}$$

$$\alpha_j, \mu_j, \xi_j \geq 0, \qquad 1 \leq j \leq N \tag{4.16}$$

We see from equation (4.10) that β is written as a linear combination of the observed instances $\mathbf{x}_j$, with nonzero coefficients α_j only for those observations that exactly meet constraint (4.15). These observations are the ones touching or invading the margins, and are labeled *support vectors*. Those on the edge of the margin verify $\xi_j = 0$, and so any of them can also be used to solve for β_0 in the equalities derived from (4.15):

$$\beta_0 = y_j - \beta^T \mathbf{x}_j \qquad \text{for any } \mathbf{x}_j \text{ lying on the edge of the margin} \tag{4.17}$$

An interesting remark to be made is about the absence of non-support observations in the expressions for β and β_0. As it turns out, training instances far enough from points from the other class have no influence whatsoever on the separating hyperplane. This feature characterizes SVMs, and is what gives support vectors their name [10].

4.4.2 Non-linear SVM and kernel methods

The idea behind nonlinear SVM is to find an optimal separating hyperplane in a high-dimensional feature space, where more flexible classifiers can be trained. Computations would get much more expensive if it were not for the use of *kernel methods*, which allow for the computation of the non-linear scoring function using the original variable space [9].

Firstly, let us derive an expression for the value of the scoring function f at any point $\mathbf{x}$ in the linear case. Using (4.10), we reach

$$f(\mathbf{x}) = \beta^T \mathbf{x} + \beta_0 = \sum_{j=1}^{N} \alpha_j y_j \mathbf{x}_j^T \mathbf{x} + \beta_0 \tag{4.18}$$

The fundamental realization is about the way that the training points $\mathbf{x}_j$ appear in equations (4.11) - (4.18): only through inner products between pairs.

This way, there is an alternative to mapping the training sample into some high-dimensional space and then working with its inner product: we only need to know how to compute the higher-dimensional product in terms of the original vectors.

Such function is termed a *kernel function*, and should be symmetric and positive semi-definite. This way calculations are reduced, even so that infinite-dimensional spaces are available to improve the flexibility of the classifier.

Indeed, when using a kernel $K(\cdot, \cdot) : X \times X \longrightarrow \mathbb{R}$, the scoring function is written as

$$f(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}) + \beta_0 \tag{4.19}$$

where β_0 is determined, analogously to (4.17), by

$$\beta_0 = y_j - \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) \tag{4.20}$$

for any $\mathbf{x}_j$ lying on the edge of the margin.

Finally, some of the most common choices for kernel functions are

$$\text{linear: } K(x, x') = x^T x' \tag{4.21}$$

$$\text{dth-degree polynomial: } K(x, x') = (1 + x^T x')^d \tag{4.22}$$

$$\text{radial basis: } K(x, x') = \exp\left(-\gamma \|x - x'\|_2^2\right) \tag{4.23}$$

These ideas are naturally extended to the multi-class classification problem through the One-versus-one or the One-versus-rest approaches [10].

4.4.3 Implementation

For numerical illustration purposes, we will be working with Support Vector Machines in Python as implemented in the Scikit-Learn library [15], in particular in module `sklearn.svm` as function `SVC()`.

Chapter 5

Particularization of Counterfactual Problems

In this chapter we specify the forms of Simple Counterfactual Optimization Problem (3.1) and the Multi-objective Counterfactual Optimization Problem (3.4) for the aforementioned classification methods. Furthermore, we aim to write formulations that explicitly showcase every aspect of the problems, so as to ease numerical implementation.

The feasible set Let us firstly consider the predictor variables. As stated in section 4.1, features are codified so that only numerical variables remain. The methods used to achieve this impose inherent constraints to the values that the newly created variables can take. This is, just by considering the result of encoding, we have for our problems:

- R continuous decision variables $\{r_1, \ldots, r_R\}$.
- Z integer decision variables $\{z_1, \ldots, z_Z\}$.
- O ordinal-turned-integer decision variables $\{o_1, \ldots, o_O\}$.
- For each original ordinal feature, an lower and a upper bound to the values that the corresponding integer variable may take, typically 0 and *number of levels - 1*. This makes $2O$ bounds.
- For each original categorical feature c_j with N_j classes, N_j binary decision variables $c_{j,1}, \ldots, c_{j,N_j}$; making a total of $B \coloneqq \sum_j N_j$ binary variables.
- For each original categorical feature c_j, a linear equality restriction stating

that exactly one of the categories must be applicable in each case:

$$\sum_{l=1}^{N_j} c_{j,l} = 1 \qquad 1 \leq j \leq C \tag{5.1}$$

This gives us a feasible set that is a subset of $\mathbb{R}^R \times \mathbb{Z}^{Z+O} \times \{0,1\}^B$, with $2O$ bounds[1] and C linear equality restrictions, that may be transformed into $2C$ linear inequality restrictions if so desired. With these, we ensure that solutions to the counterfactual problem are interpretable in terms of the original features.

Actionability and feasibility constraints These enforce the viability of the counterfactual instances. We may face three different kinds in our formulation: bounds, inequalities and *immutability conditions*. The former two can be directly introduced in our model, while the last are a consequence of some features not being actionable, and their presence makes it worth reconsidering the need for some decision variables.

For instance, an immutable categorical feature c_j needs not give rise to any binary decision variable or extra constraint. It can just be used to calculate the values of the scoring functions, and so the problems lose decision variables and constraints, and become simpler.

We will choose to only formulate *linear inequalities* as part of the constraints, in order to keep the boundaries piece-wise linear.

Note that for now our feasible space is a polyhedron intersected with a grid; as all constraints imposed are affine.

The distance function As stated in section 3.3, we will propose a convex combination of weighted distances $L0$, $L1$ and $L2$, appropriately converted to provide a quadratic expression. The distance between an instance x_0 and the to-be-calculated counterfactual constitutes an objective function to be minimized.

A slight tweak is needed for the application of the L0 distance on the categorical variables. In our current formulation, a change in value of a categorical feature is represented as a change in two binary variables, practically giving double weight to categorical features in the L0 distance. Thus, we will have to compensate for this effect by introducing a 0.5 factor in the corresponding terms.

These comments are valid for all problems that we intent to formulate; but when tackling validity of the counterfactual, we must specify to keep on developing.

[1] We distinguish bounds from linear inequalities because of the difference that most solver softwares make between them.

5.1 The Simple Counterfactual Optimization Problem

The validity restriction In the Simple Counterfactual Optimization Problem, classification in the desired class is ensured by a hard restriction, which in the score-based case took the form of $K-1$ inequality constraints, as expressed in (3.2). We will take the K-th class as the desired one in the multi-class case, for the sake of simplicity of notation.

The Simple Counterfactual Optimization Problem is then stated as:

$$
\begin{aligned}
&\text{minimize} \quad a_0 \textstyle\sum_{i=1}^{M} c_i s_i + a_1 \sum_{i=1}^{M} b_i t_i + a_2 \sum_{i=1}^{M} b_i^2 (x_i - x_{0,i})^2 \\
&\text{subject to} \quad
\begin{cases}
r_j \in \mathbb{R} & 1 \le j \le R \\
z_j \in \mathbb{Z} & 1 \le j \le Z \\
o_j \in \mathbb{Z} & 1 \le j \le O \\
c_{j,l} \in \{0,1\} & 1 \le j \le C, \quad 1 \le l \le N_j
\end{cases} \\
&\qquad\qquad
\begin{cases}
0 \le o_j \le u_j & 1 \le j \le O \\
\sum_{l=1}^{N_j} c_{j,l} = 1 & 1 \le j \le C \\
A\mathbf{x} \le \mathbf{b}
\end{cases} \\
&\qquad\qquad
\begin{cases}
f_k(\mathbf{x}) \le f_K(\mathbf{x}) & 1 \le k \le K-1
\end{cases} \\
&\qquad\qquad
\begin{cases}
s_i \in \{0,1\} & 1 \le i \le M \\
x_i - x_{i0} \le M' s_i & 1 \le i \le M \\
x_i - x_{i0} \ge -M' s_i & 1 \le i \le M
\end{cases} \\
&\qquad\qquad
\begin{cases}
t_i \in \mathbb{R} & 1 \le i \le M \\
x_i - x_{i0} \le t_i & 1 \le i \le M \\
x_i - x_{i0} \ge -t_i & 1 \le i \le M
\end{cases}
\end{aligned}
\tag{5.2}
$$

where we have noted $\mathbf{x} = (r_1, \ldots, r_R, z_1, \ldots, z_Z, o_1, \ldots, o_O, c_{1,1}, \ldots, c_{C,N_C})^T$ to simplify the expressions. M is the dimension of $\mathbf{x}$, and M' is the big-M constant that we proposed in (3.11). Bounds u_j reflect the number of categories of each ordinal feature, as already stated. Weights b_i are given and reflect the ranges and resistance to change of the variables, as stated in section 3.3, while c_i take values on account of the interaction between the L0 distance and one-hot encoding.

$$c_i = \begin{cases} 0.5 & \text{if } x_i \text{ is a binary variable arising from a categorical feature} \\ 1 & \text{if it is not} \end{cases} \tag{5.3}$$

Coefficients a_0, a_1, a_2 for the distances are also presumed to be known.

For clarity, we have broken up the constraints into brackets, in such a way that:

- The first one states the nature of the decision variables.
- The second states the affine inequalities and bounds required for the well-definition of the ordinal and categorical variables, as well as for feasibility and actionability.
- The third presents the validity constraints, ensuring the desired classification.
- The fourth and fifth are required for the linearization of norms $L0$ and $L1$, respectively.

It is at the validity constraints that the problems for different classifiers behave separately.

In the case of linear classifiers such constraints are affine, and so the resulting feasible space keeps its polyhedral character. In a multi-class Logistic Regression classifier setting, the validity condition takes the shape

$$f_k(\mathbf{x}) = \beta_k^T \mathbf{x} + \beta_{k0} \leq 0, \qquad 1 \leq k \leq K-1 \tag{5.4}$$

In the binary Logistic Regression and the binary linear SVM settings, taking $+1$ as the desirable class, we get

$$f(\mathbf{x}) = \beta^T \mathbf{x} + \beta_0 \geq 0 \tag{5.5}$$

In both cases, problem (5.2) is a Mixed-Integer Convex Quadratic Program.

In non-linear classifiers however these constraints may not even be convex, resulting in the loss of both the linear and convex characters of the set. In the binary non-linear SVM setting, the form of the validity constraint is

$$f(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}) + \beta_0 \geq 0 \tag{5.6}$$

and it is the kernel $K(\cdot, \cdot)$ that determines the shape of the feasible region. In general, problem (5.2) will not exhibit convexity or any other characteristic that facilitates its solving.

Resolution may be accomplished by heuristic or local search methods, but these do not ensure optimality of the reached counterfactual.

5.2 The Multi-objective Counterfactual Optimization Problem

The validity objective In the MCOP, classification in the desired class is not enforced but encouraged through maximization of the difference between scoring functions, which appear as separate objectives. The Multi-objective Counterfactual Optimization Problem is then stated as:

$$
\begin{aligned}
\text{minimize} \quad & \Big\{ a_0 \textstyle\sum_{i=1}^{M} c_i s_i + a_1 \sum_{i=1}^{M} b_i t_i + a_2 \sum_{i=1}^{M} b_i^2 (x_i - x_{0,i})^2, \\
& \qquad f_1(\mathbf{x}) - f_K(\mathbf{x}), \ldots, f_{K-1}(\mathbf{x}) - f_K(\mathbf{x}) \Big\}
\end{aligned}
$$

$$
\text{subject to} \quad \begin{cases}
r_j \in \mathbb{R} & 1 \leq j \leq R \\
z_j \in \mathbb{Z} & 1 \leq j \leq Z \\
o_j \in \mathbb{Z} & 1 \leq j \leq O \\
c_{j,l} \in \{0,1\} & 1 \leq j \leq C, \quad 1 \leq l \leq N_j
\end{cases}
$$

$$
\begin{cases}
0 \leq o_j \leq u_j & 1 \leq j \leq O \\
\sum_{l=1}^{N_j} c_{j,l} = 1 & 1 \leq j \leq C \\
A\mathbf{x} \leq \mathbf{b}
\end{cases} \tag{5.7}
$$

$$
\begin{cases}
s_i \in \{0,1\} & 1 \leq i \leq M \\
x_i - x_{i0} \leq M' s_i & 1 \leq i \leq M \\
x_i - x_{i0} \geq -M' s_i & 1 \leq i \leq M
\end{cases}
$$

$$
\begin{cases}
t_i \in \mathbb{R} & 1 \leq i \leq M \\
x_i - x_{i0} \leq t_i & 1 \leq i \leq M \\
x_i - x_{i0} \geq -t_i & 1 \leq i \leq M
\end{cases}
$$

The comments we made about the brackets in problem (5.2) are also in order here, with the exception of those about validity constraints.

When reducing problem (5.7) to a single-objective one through the weighting method, the objective function takes the form

$$\begin{aligned} C(\mathbf{x}) \coloneqq \; & w_0 \left[a_0 \sum_{i=1}^{M} c_i s_i + a_1 \sum_{i=1}^{M} b_i t_i + a_2 \sum_{i=1}^{M} b_i^2 (x_i - x_{0,i})^2 \right] + \\ & + w_1 \Big[f_1(\mathbf{x}) - f_K(\mathbf{x}) \Big] + \ldots + w_{K-1} \Big[f_{K-1}(\mathbf{x}) - f_K(\mathbf{x}) \Big] \end{aligned} \tag{5.8}$$

with weights $w_0, \ldots, w_{K-1}$.

In the case of linear classifiers all scoring functions are affine, and so the objective is a quadratic function of $\mathbf{x}$. The resulting problem is then a Mixed-Integer Convex Quadratic Program. Furthermore, if the classification problem is binary, expression (5.8) is simplified to

$$C(\mathbf{x}) \coloneqq w_0 \left[a_0 \sum_{i=1}^{M} c_i s_i + a_1 \sum_{i=1}^{M} b_i t_i + a_2 \sum_{i=1}^{M} b_i^2 (x_i - x_{0,i})^2 \right] - w_1 \Big[\beta^T \mathbf{x} + \beta_0 \Big] \tag{5.9}$$

The optimal values for w_0 and w_1 can be determined *a posteriori*, as a user decides how to balance both objectives. In this sense, it is useful to account for the interpretation of the scoring function in each setting:

- The scoring function of a Logistic Regression classifier is a monotonous transformation of the probability of belonging to the desired class.

  $$\mathrm{P}[1|X = \mathbf{x}] = \frac{\exp(\beta^T \mathbf{x} + \beta_0)}{1 + \exp(\beta^T \mathbf{x} + \beta_0)} \tag{5.10}$$

  Expressing the counterfactual instances in terms of "changes to be made" and "probability of belonging to the desired class" may then be the most intuitive way to approach the choice.

- The scoring function of a SVM takes value 0 at the border between classes, and ± 1 at the edge of the margins; so a score in (0,1) is classified in the desirable class, but remains somewhat atypical; while a score greater than 1 rests safely inside the desirable region.

For non-linear classifiers the scoring objectives may not be convex, which again prevents an easy approach to the problem. In the binary SVM case, we have

$$\begin{aligned} C(\mathbf{x}) \coloneqq \; & w_0 \left[a_0 \sum_{i=1}^{M} c_i s_i + a_1 \sum_{i=1}^{M} b_i t_i + a_2 \sum_{i=1}^{M} b_i^2 (x_i - x_{0,i})^2 \right] + \\ & - w_1 \Big[\sum_{j=1}^{N} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}) + \beta_0 \Big] \end{aligned} \tag{5.11}$$

Optimal weights w_0 and w_1 can be determined through the same strategy as the linear case; as the interpretation of the SVM scoring function does not vary when non-linear kernels are introduced.

Chapter 6

Numerical Illustration

In this chapter we apply the aforeposed problems to the generation of counterfactuals for a concrete dataset and classifiers. In order to do this, we previously encode categorical features and train a linear classifier of each type considered: Logistic Regression and linear SVM. Multi-objective Counterfactual Problems are then formulated and solved for various training instances and different values of the weights, offering a range of possible counterfactuals with associated measures of the certainty of classification.

Our purpose here is to showcase a direct implementation of these problems and the obstacles and peculiarities that may arise as a result.

6.1 The German Credit Data dataset

We have worked with the Statlog (German Credit Data) Data Set, as found at the Machine Learning Repository of the University of California, Irvine [4]. This dataset was collected by professor Dr. Hans Hofmann at Universität Hamburg in 2000. It classifies people described by a set of attributes as good or bad credit takers, intending to help banks and other credit-lending institutions predict whether an application for a credit is worth the risk.

The dataset contains 1000 instances of data, 700 of them labeled as *good* credit takers with the remaining 300 being *bad*. Variables included in it are categorical, ordinal, integer or continuous in nature; as described in table (6.1).

Categorical features labeled with (*) lack instances in one of the categories. This absence of data may cause some coefficients to be undetermined during training, and so we have removed the empty categories for the sake of simplicity.

	Variable	Type	Levels
1	Status of existing checking account	ordinal	4
2	Duration in months	integer	-
3	Credit history	ordinal	4
4	Purpose of the credit	categorical	11*
5	Credit amount in DM	continuous	-
6	Savings account	ordinal	5
7	Duration of present employment	ordinal	5
8	Installment rate in percentage of disposable income	continuous	-
9	Personal status and sex	categorical	5*
10	Other debtors / guarantors	categorical	3
11	Duration of present residence	integer	-
12	Property	ordinal	4
13	Age in years	integer	-
14	Other installment plans	categorical	3
15	Housing	categorical	3
16	Number of existing credits at this bank	integer	-
17	Job level	ordinal	4
18	Number of people to provide maintenance for	integer	-
19	Telephone number?	categorical	2
20	Foreign worker?	categorical	2
21	*Goodness* of the credit application	target	2

Table 6.1: German Credit Data variables

The dataset also suggests the use of specific costs for misclassifications during training: a *bad* client classified as *good* is considered five times worse than a *good* client classified as *bad*. The direct consequence is an imbalance in the resulting classifier's accuracy, as will be shown shortly; but we have decided to abide by the suggestion. We find it to be a more realistic problem, since it is sensible that a real credit-lending institution would want to penalize mistakes in such a way.

6.2 Training the classifiers

We train a Logistic Regression classifier and a linear SVM. Variables are previously standardized to ease convergence of the training processes, and the suggested misclassification errors are applied. The parameters and accuracy of training –tested optimistically on the training set– can be read in table (6.2).

Classifier	Settings	(-) class accuracy	(+) class accuracy	Total accuracy
Logistic Regression	$L2$ regulariz. C=1	0.89	0.56	0.66
Linear SVM	C=0.1	0.91	0.57	0.67

Table 6.2: Accuracy of different classifiers

6.3 Constraints and other settings

We give a brief retell of the actionability and feasibility constraints, weights and other parameters that we have considered for the Counterfactual Problems.

Constraints. Firstly, we have considered variables *purpose of the credit* (4), *personal status and sex* (9), *number of people to provide maintenance for* (18) and *foreign worker* (20) to be immutable; as they represent things that an applicant cannot change or at least is not likely to change in order to get a credit. The only exception being the change between *Purpose: new car* and *Purpose: old car* as we believe it to be a sensible suggestion.

We have imposed logical bounds: *age in years* (13) may only increase its value, while *credit amount* (4) may not step below 90% its original value, as it is reasonable that applicants may not be interested in a credit under such conditions.

After some preliminary results, we found some quirks of the classifiers that moved us to introduce new constraints. In the linear case, there is a tendency to suggest a change *for the worse* in variables *credit history* (4) and *job level* (17). This is likely due to correlation between variables causing indeterminate coefficients in the training problems, and could be solved by enforcing sparsity on the classifier. However, our main concern here is not the training of supervised classifiers, but the generation of realistic counterfactual instances. For this reason we will keep using the classifiers are they are, and impose bounds on these variables: *credit history* and *job level* cannot be worsened.

There is also a clear mathematical dependence between *credit amount* (5), *duration in months* (2), *installment rate as percentage of disposable income* and the *disposable income*, which is not itself a variable. If we want these to stay coherent, a quadratic bound should be introduced in the form

$$CreditAmount - C \times InstallmentRate \times Duration = 0 \tag{6.1}$$

where C represents the disposable income as calculated from the original instance. The introduction of quadratic equality bounds is not prohibited in the problems'

original formulation in chapter 3, but it is not desirable in the concrete versions we have developed since. Luckily, the preliminary results show that there is little to no tendency to alter the *duration in months* variable in the counterfactual instances; and so we propose a simpler (and linear) constraint: proportionality between *credit amount* and *installment rate*. This enforces relation (6.1) as long as there is no change in the duration of the credit, which is the norm in the generated counterfactual instances.

Other settings. We have used the inverse of standard deviations as weights b_i for the variables within weighted distances, in the non-binary variables. As stated in section 3.3, these could be tweaked to account for resistance to movement and/or personal preference of the applicant; but we have kept them as they are for the sake of simplicity. Finally, coefficients $(a_0, a_1, a_2) = (1, 1, 2)$ are used, as they have been found to give reasonable results.

6.4 Results

MCOP problems have been posed for a variety of instances with different values for the pair of weights (w_0, w_1), setting $w_0 = 1 - w_1$, so as to explore the trade-off between the proximity and validity objectives. The validity weight w_1 has taken values in $[0.8, 0.99)$, as every lower value has been found not to encourage validity enough, and thus gave the original point as counterfactual.

Optimal solutions have been attained by means of the Gurobi Optimizer solver [8], under a named-user student academic license. Library `gurobipy` in Python 3.9 has been used as interface for the formulation.

Typical sets of counterfactual instances are presented below in tables (6.3) - (6.10). These have been calculated on training instances that were classified as *bad credit applications*, with the intent to show what changes need to be made in order to increase probability of classification as *good credit applications*. The first counterfactual to be valid within each set is marked in green.

Interestingly, all sets have been found to be stable: as the validity weight increases its value, counterfactual explanations get further from the original instance in a monotonic way. All changes proposed by a counterfactual are also proposed – and expanded upon– by those with higher validity weights. This is an extremely desirable characteristic for the results.

Instance A Married or divorced woman, 22 years old, asking for a 5951 DM credit in order to buy a radio or television. Installment rate is 2% of disposable income. The actual label is *bad application*.

w_1	Changes to be made	Probability
0.00	None	0.13
0.87	Level increase in *checking account status*	0.22
0.90	Get a guarantor	0.40
0.92	Level increase in *checking account status*	0.55
0.93	Reduction in *credit amount*: 5355.90 DM Reduction in *installment rate*: 1.80%	0.59
0.95	Supply a telephone number	0.67
0.96	Level increase in *savings account status*	0.71
0.97	Level increase in *current employment time*	0.74
0.98	Resolve 1 of 1 existing credits Level increase in *savings account status* Level increase in *current employment time*	0.84

Table 6.3: Counterfactuals for instance A, Logisitic Regression classifier.

w_1	Changes to be made	Score f
0.000	None	-1.38
0.895	Level increase in *checking account status*	-0.85
0.910	Get a guarantor	-0.14
0.935	Level increase in *checking account status*	0.37
0.945	Reduction in *credit amount*: 5355.90 DM Reduction in *installment rate*: 1.80%	0.48
0.960	Level increase in *savings account status*	0.64
0.975	Level increase in *current employment time* Supply a telephone number	0.95
0.980	Level increase in *savings account status*	1.12
0.985	Level increase in *current employment time*	1.39
0.990	Level increase in *savings account status* 1-year increase in *current residence time*	1.76

Table 6.4: Counterfactuals for instance A, Lineal SVM classifier.

Instance B Married or widowed man, 28 years old, asking for 5234 DM in order to buy a new car. Installment rate is 4% of disposable income. The actual label is *bad application*.

w_1	Changes to be made	Probability
0.00	None	0.06
0.80	Change *purpose* to *used car*	0.29
0.87	Level increase in *checking account status*	0.44
0.90	Reduction in *credit amount*: 4710.6 DM Reduction in *installment rate*: 3.6% Get a guarantor	0.69
0.92	Level increase in *checking account status*	0.80
0.95	Supply a telephone number	0.86
0.96	Level increase in *savings account status*	0.88
0.97	Level increase in *current employment time*	0.89
0.98	Resolve 1 of 2 existing credits Level increase in *savings account status* Level increase in *current employment time* Level increase in *property*	0.84

Table 6.5: Counterfactuals for instance B, Logisitic Regression classifier.

w_1	Changes to be made	Score f
0.000	None	-2.04
0.870	Change *purpose* to *used car*	-0.97
0.885	Level increase in *checking account status*	-0.45
0.910	Get a guarantor	0.26
0.925	Reduction in *credit amount*: 4710.6 DM Reduction in *installment rate*: 3.6%	0.39
0.935	Level increase in *checking account status*	0.91
0.960	Level increase in *savings account status*	1.07
0.975	Supply a telephone number Level increase in *current employment time*	1.38
0.980	Level increase in *savings account status*	1.54
0.985	Resolve 1 of 2 existing credits Level increase in *current employment time*	1.82
0.99	Supply a telephone number Level increase in *current employment time* Level increase in *savings account status* 1-year increase in *current residence time*	2.30

Table 6.6: Counterfactuals for instance B, Lineal SVM classifier.

Instance C Married or divorced woman, 44 years old, asking for a 12579 DM credit in order to buy a used car. Installment rate is 4% of disposable income. Actual label is *bad application*.

w_1	Changes to be made	Probability
0.00	None	0.15
0.86	Reduction in *credit amount*: 11591,57 DM Reduction in *installment rate*: 3,69%	0.18
0.88	Reduction in *credit amount*: 11321,1 DM Reduction in *installment rate*: 3,6% Level increase in *checking account status*	0.31
0.90	Get a guarantor	0.51
0.92	Level increase in *checking account status*	0.66
0.96	Level increase in *savings account status* Change *housing* from *renting* to *owning*	0,77
0.98	Resolve 1 of 1 existing credits Level increase in *savings account status* Level increase in *property*	0.85

Table 6.7: Counterfactuals for instance C, Logistic Regression classifier

w_1	Changes to be made	Score f
0.000	None	-1.27
0.885	Level increase in *checking account status*	-0.75
0.900	Reduction in *credit amount*: 11516,12 DM Reduction in *installment rate*: 3,66%	-0.57
0.905	Reduction in *credit amount*: 11408,73 DM Reduction in *installment rate*: 3,63%	-0.55
0.910	Reduction in *credit amount*: 11321,1 DM Reduction in *installment rate*: 3,60% Get a guarantor	0.18
0.935	Level increase in *checking account status*	0.70
0.960	Level increase in *savings account status*	0,86
0.975	Change *housing* from *renting* to *owning*	1.04
0.980	Level increase in *savings account status*	1.21
0.985	Resolve 1 of 1 existing credits	1.37
0.990	Level increase in *savings account status* 1-year increase in *current residence time*	1.74

Table 6.8: Counterfactuals for instance C, Linear SVM Classifier

Instance D Single man, 36 years old, asking for a 1374 DM credit in order to buy furniture or equipment. Installment rate is 1% of disposable income. Actual label is *good application*.

w_1	Changes to be made	Probability
0.00	None	0.38
0.87	Level increase in *checking account status*	0.53
0.90	Get a guarantor Resolve other installment plans (banks)	0.86
0.92	Level increase in *checking account status*	0.92
0.95	Level increase in *savings account status*	0.96
0.96	Reduction in *credit amount*: 1236.6 DM Reduction in *installment rate*: 0.9% Level increase in *savings account status*	0.96
0.97	Level increase in *current employment duration*	0.97
0.98	Resolve 1 of 1 existing credits Level increase in *savings account status* Level increase in *current employment duration*	0.98

Table 6.9: Counterfactuals for instance D, Logistic Regression classifier

w_1	Changes to be made	Score f
0.00	None	-1.35
0.87	Resolve other installment plans (banks)	-0.30
0.885	Level increase in *checking account status*	0.22
0.910	Get a guarantor	0.93
0.935	Level increase in *checking account status*	1.45
0.955	Level increase in *checking account status*	1.97
0.960	Level increase in *savings account status*	2.14
0.975	Reduction in *credit amount*: 1236.6 DM Reduction in *installment rate*: 0.90% Level increase in *current employment duration*	2.29
0.980	Level increase in *savings account status*	2.45
0.985	Resolve 1 of 1 existing credits Level increase in *current employment duration*	2.73
0.990	Level increase in *savings account status*	3.01

Table 6.10: Counterfactuals for instance D, Lineal SVM classifier

6.5 Some comments on the results

The Logistic and the linear SVM classifiers are sufficiently alike to offer very similar counterfactual instances. The most popular recommendations for the first steps seem to be:

- Increasing the level of one's checking account.
- Getting someone to act as a guarantor for the credit.
- Reducing the credit's amount.

Not that there are not minor-scale trends: all applicants stating *new car* as the purpose of the credit are recommended to change it to *used car* as their first suggestion, as is the case of instance B. Similarly, applicants with other installment plans are generally told soon to resolve them, as we see in instance D.

Note that while these may seem obvious suggestions, they happen to achieve the greatest improvement in the score function with the smallest change in input.

For instance, increasing the level of one's savings account or resolving other existing credits are also reasonable courses of action, but appear consistently later as suggestions since they do not imply a sufficiently large increment in the score.

An important aspect of the generation of these counterfactuals is the *step size* in the w_1 sweep. It would be ideal if each new counterfactual introduced changes in just one variable, so that we could isolate and weight the effects of each further change. Nevertheless this is often not the case. As an example we have table (6.5), wherein the same counterfactual is obtained for values $w_1 = 0.87, 0.88, 0.89$ but there is a sudden change in three variables for $w_1 = 0.90$. A smaller step size may have been able to generate an intermediate counterfactual, but would also have increased significantly the number of computations.

We have chosen different values for our classifiers –0.1 for the Logistic, 0.05 for the SVM– to further showcase this effect, which is visibly more prevalent in the Logistic tables than in the SVM ones. Moreover, higher values of w_1 seem to need a smaller step size.

The numerical results here given are supposed to serve as an illustration of both the capabilities of the method and the process of implementation. There is surely room for improvement in a finer tuning of the distance weights $a0, a1, a2$, a greater insight in the causal relations between variables, or in a deeper assessment of the resistance to movement of variables.

We aim to show that, in general, the direct application of Counterfactual Problems to concrete classifiers is not uniquely defined, as there is room for subjectivity. Familiarization, direct handling of the database and a complete assessment of the situation at hand have proved to be essential to arrive at this formulation, able to give rise to sensible results.

References

[1] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN: 9780521833783.

[2] Emilio Carrizosa, Jasone Ramírez-Ayerbe, and Dolores Romero Morales. "Generating Collective Counterfactual Explanations in Score-Based Classification via Mathematical Optimization". In: (July 2021). DOI: 10.13140/RG.2.2.22996.12168/1.

[3] Emilio Carrizosa and Dolores Romero Morales. "Supervised Classification and Mathematical Optimization". In: *Computers & Operations Research* 40.1 (2013). DOI: 10.1016/j.cor.2012.05.015. URL: https://www.sciencedirect.com/science/article/pii/S0305054812001190.

[4] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[5] Official Journal of the European Union. *Recital 71 EU GDPR*. Last accessed 10 June 2022. URL: https://www.privacy-regulation.eu/en/r71.html.

[6] Daniel Faggella. "Machine Learning for Medical Diagnostics - 4 Current Applications". In: (2020). URL: https://emerj.com/ai-sector-overviews/machine-learning-medical-diagnostics-4-current-applications/.

[7] Oscar Fontenla-Romero et al. "Online Machine Learning". In: *Efficiency and Scalability Methods for Computational Intellect*. Ed. by Boris Igelnik and Jacek M. Zurada. IGI Global, 2013.

[8] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2022. URL: https://www.gurobi.com.

[9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001. ISBN: 0387952845.

[10] Alan Julian Izenman. *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. Springer texts in statistics. Dordrecht: Springer, 2006. DOI: 10.1007/978-0-387-78189-1. URL: https://cds.cern.ch/record/1338317.

[11] Mark T. Keane and Barry Smyth. "Good Counterfactuals and Where to Find Them: A Case-Based Technique for Generating Counterfactuals for Explainable AI (XAI)". In: *CoRR* abs/2005.13997 (2020). DOI: 10.48550/ARXIV.2005.13997. URL: https://arxiv.org/abs/2005.13997.

[12] Kaisa Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999. ISBN: 0792382781.

[13] Tim Miller. "Explanation in artificial intelligence: Insights from the social sciences". In: *Artificial Intelligence* 267 (2019). DOI: 10.1016/j.artint.2018.07.007. URL: https://www.sciencedirect.com/science/article/pii/S0004370218305988.

[14] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997. ISBN: 0-07-042807-7.

[15] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[16] Kumba Sennaar. "Machine Learning for Recruiting and Hiring - 6 Current Applications". In: (2019). URL: https://www.aitimejournal.com/@saurav.singla/machine-learning-to-predict-credit-risk-in-lending-industry.

[17] Saurav Singla. "Machine Learning to Predict Credit Risk in Lending Industry". In: (2020). URL: https://www.aitimejournal.com/@saurav.singla/machine-learning-to-predict-credit-risk-in-lending-industry.

[18] Sahil Verma, John Dickerson, and Keegan Hines. "Counterfactual Explanation for Machine Learning: A Review". In: *CoRR* abs/2010.10596 (Oct. 2020). URL: https://arxiv.org/abs/2010.10596.

[19] Sandra Wachter, Brent Mittelstadt, and Chris Russel. "Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the CDPR". In: *Harvard Journal of Law & Technology* 31.2 (Nov. 2018). DOI: 10.2139/ssrn.3063289. URL: https://ssrn.com/abstract=3063289.