POLITECNICO DI TORINO

Master's Degree in DATA SCIENCE AND ENGINEERING

Master's Degree Thesis

Prescriptive Analytics: Review of Frameworks and Critical Evaluation of PrescrX

Supervisors

Prof. Daniele APILETTI

Dott. Gianmarco SABATINI

Candidate

Paolo FAVELLA

DECEMBER 2025

Summary

Prescriptive analytics represents the most advanced stage in the evolution of data analytics. While descriptive analytics aims to interpret past data and predictive analytics focuses on forecasting possible future outcomes, prescriptive analytics goes one step further by recommending specific actions to achieve desired objectives. The ability to not only predict but also prescribe makes it a powerful decision-support tool for both scientific research and industrial practice.

This thesis, developed during a professional internship at Aizoon, contributes to this emerging domain by providing both a critical review of existing frameworks and a detailed evaluation of a newly developed tool named **PrescrX**. PrescrX is a **proprietary and patented software** that proposes prescriptions through a local, model-agnostic approach inspired by eXplainable Artificial Intelligence (XAI). It operates independently of the underlying classifier, meaning that it can work with a wide range of models, from neural networks to decision trees. Given a trained classifier, a background dataset, and a target class, PrescrX generates minimal modifications to an input instance so that the modified point is classified into the desired category. In contrast to many existing optimizers that tend to maximize classification confidence regardless of plausibility, PrescrX is explicitly designed to prioritize minimal and interpretable changes that remain close to the original data distribution.

The work begins with a theoretical overview of the analytics spectrum, highlighting the progression from descriptive to predictive and finally to prescriptive methods. Particular emphasis is placed on the distinction between correlation and causality, a key requirement for actionable recommendations. This theoretical foundation is followed by a comprehensive literature review of methodological approaches to prescriptive analytics, including probabilistic models, machine learning integrations, mathematical programming, evolutionary computation, logic-based systems, and hybrid frameworks. The review highlights the absence of standardized evaluation criteria and points to interpretability, scalability, and comparability as open challenges.

The core of the thesis focuses on PrescrX. The system is introduced as a model-independent tool that constructs prescriptions by approximating the local

decision boundary of a classifier with a linear model and solving a quadratic programming problem. The algorithm is highly configurable, supporting constraints such as locked features, bounded values, discrete domains, and, uniquely, a cost-function constraint that restricts prescriptions within an admissible threshold. This functionality is particularly relevant in industrial contexts, where interventions are associated with financial costs or operational risks. The flexibility of PrescrX is demonstrated through two case studies: the first based on the MNIST dataset of handwritten digits, and the second on a real industrial process involving the cleaning of mechanical components.

The industrial case study represents the starting point of the experimental analysis. By applying PrescrX to operational parameters, it was possible to prescribe changes that transformed components classified as "dirty" into "clean" according to the predictive model. When compared to a traditional optimizer, however, some inconsistencies emerged. While the optimizer could generate points with high confidence in the target class, many of these points were not plausible or interpretable from an industrial perspective. PrescrX, in contrast, produced prescriptions that remained realistic and closer to the original operational ranges. These observations raised the question of whether the comparison between the two methods was fully equitable. To explore this issue in a more controlled environment, the MNIST dataset was introduced as a benchmark scenario, allowing the prescriptions to be validated not only through numerical metrics but also via human supervision, as the generated images could be visually inspected.

The experiments with MNIST confirmed the discrepancies observed in the industrial case. Traditional optimizers, which only use the classifier, often produced adversarial-like examples—images classified with high confidence but meaningless to a human observer. PrescrX, which leverages both the classifier and the background dataset, generated more natural and interpretable prescriptions. This highlighted a structural imbalance in the comparison: PrescrX exploits information from the background dataset, whereas the optimizer does not. To make the evaluation fairer, we proposed an improved version of the optimizer that integrates the background dataset into its process. This was achieved by introducing a **similarity factor** into the optimizer's objective function, penalizing solutions that deviated too far from the data manifold. This modification allowed the optimizer to produce prescriptions more comparable to those of PrescrX, enabling a more balanced and meaningful evaluation.

To assess the quality of prescriptions, the thesis also introduces a set of original evaluation metrics. These include the normalized distance from the starting point, the gain of neighbors, a feature preservation score, robustness to perturbations, and execution time. Together, these metrics allow prescriptions to be evaluated not only in terms of predictive accuracy but also in terms of plausibility, interpretability, and computational efficiency. Applied to both case studies, the metrics confirmed

that PrescrX tends to generate prescriptions that remain embedded in the data distribution, while standard optimizers frequently drift into low-density and implausible regions. The improved optimizer with similarity constraints reduced this gap, but PrescrX still demonstrated superior interpretability and reliability.

In conclusion, this thesis contributes in three main ways. First, it provides a critical review of prescriptive analytics frameworks, clarifying current methods and highlighting open challenges. Second, it presents PrescrX, a proprietary software that bridges optimization and explainability through a model-agnostic, constraint-aware approach. Third, it introduces a set of novel evaluation metrics and applies them to rigorous experimental comparisons, including the proposal of an improved optimizer to ensure fairness. The results demonstrate that PrescrX generates plausible and interpretable prescriptions in both academic and industrial settings, offering a valuable alternative to traditional optimizers. Future work will focus on scaling the approach to larger datasets, integrating causal inference, and extending PrescrX toward dynamic, real-time decision-making environments.

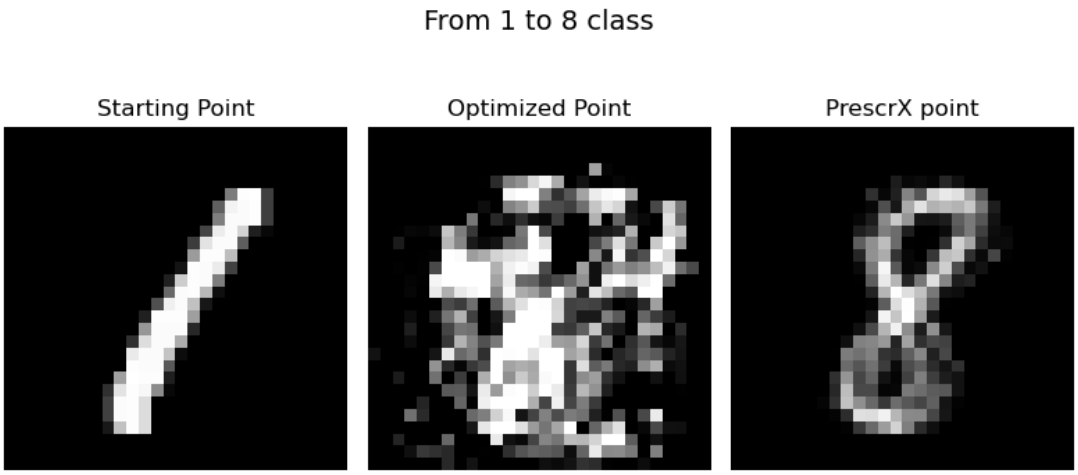

Figure 1: Comparison of starting point and generated prescriptions

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This project was developed during an internship at Aizoon, a technology consulting firm. The focus is on the PrescrX tool, designed to perform prescriptive analyses. The aim of the project is to contribute to the evolving field of prescriptive analytics, which continues to attract significant interest due to its powerful analytical capabilities. The work involves a thorough study of how PrescrX functions, including an analysis of its strengths and limitations. In addition, to address the lack of standardized evaluation criteria in the existing literature, a set of original metrics will be developed. These metrics are designed both to objectively assess the quality of the prescriptions generated by PrescrX and to enable a meaningful comparison with traditional optimizers. Proposed enhancements will also be introduced to improve the tool's versatility. Given that the academic literature in this domain is relatively limited—and primarily centered on the development of optimizers, which PrescrX is not—this study will also include a comparative analysis between PrescrX and a traditional optimizer. Before examining the tool in detail, the report will first provide an overview of prescriptive analytics: its definition, current applications, and the scope explored thus far.

1.1 What is prescriptive analytics

Prescriptive analytics represents a significant advancement in the field of data analysis and is often referred to as "the future of data analytics." Unlike other forms of analytics that primarily describe or predict outcomes, prescriptive analytics goes a step further by recommending optimal actions to achieve specific objectives. This characteristic makes it particularly valuable for data-driven decision-making.

Data analytics is typically categorized into three core types: descriptive analytics, which answers questions such as "What happened?" and "Why did it happen?"; predictive analytics, which addresses "What might happen next?" and "Why might

it happen?"; and prescriptive analytics, which focuses on "What should we do next?" and "Why should we do it?" [1]. Together, these approaches offer a comprehensive framework that enables organizations to analyze historical data, identify causality, forecast future outcomes, and take informed actions.

To fully grasp the concept of prescriptive analytics, it is essential first to understand descriptive and predictive analytics, as these provide the foundational insights upon which prescriptive analytics is built.

1.1.1 Definition of descriptive analytics

Descriptive analytics is primarily utilized in typical data analyst roles, and most analytics tools on the market do not extend beyond this stage. The main goal of descriptive analytics is to impose structure on raw data to facilitate the identification and visualization of relevant patterns [2]. This is achieved through data collection, categorization, and classification.

A wide body of literature is available online that supports this initial phase of the decision-making process, enabling users to derive meaningful insights from raw datasets. Descriptive analytics is the most prevalent and mature stage of data analysis, as the majority of current analytics tools fall within this category. The emphasis here is on processing and organizing data, as well as uncovering and presenting patterns in a clear and accessible way [2].

Techniques such as pattern recognition and clustering are commonly used at this stage to help users interpret and visualize the data more effectively. The descriptive analytics phase includes the creation of visualizations to summarize key data distributions, correlation analysis between features to detect linear dependencies, and an explained variance analysis to assess the structure and dimensionality of the dataset.

1.1.2 Definition of predictive analytics

Predictive analytics addresses the limitations of descriptive analytics in forecasting future outcomes. It leverages large volumes of historical data to uncover new insights through the use of machine learning, data mining, and statistical methods. The purpose of predictive analytics is to generate reliable forecasts that inform decision-making [3].

While descriptive tools focus on past events, predictive analytics introduces models that anticipate future developments. By applying methodologies from machine learning and statistics, it becomes possible to estimate probabilities of future events, identify recurring patterns, and uncover relationships between variables. This allows organizations to not only understand historical performance but also anticipate what is likely to happen next. Predictive analytics is widely

used in various domains such as marketing, financial services, healthcare, supply chain management, and capacity planning. In practice, this phase heavily relies on machine learning and deep learning techniques to train models capable of generating accurate forecasts and capturing complex, non-linear relationships within the data.

1.1.3 Definition of prescriptive analytics

Prescriptive analytics has already seen successful applications across various industrial and research contexts. It logically extends the progression established by the two preceding phases of Business Analytics: if the past has been analyzed (descriptive analytics) and future outcomes have been forecasted (predictive analytics), then it becomes possible to recommend—i.e., prescribe—the most effective actions to shape and adapt plans based on those forecasts.

Compared to the other stages of Business Analytics, prescriptive analytics empowers decision-makers not only to detect issues and opportunities across past, present, and future contexts, but also to directly suggest optimal courses of action aligned with specific objectives. It additionally allows for the evaluation of those decisions' potential outcomes. While optimization techniques have long been a core method for solving decision-making problems—leveraging mathematical tools—the integration of prediction with optimization introduces novel opportunities for decision support.

In real-world applications, prescriptive analytics often deals with considerable uncertainty. As a result, the effectiveness of optimization is strongly dependent on the accuracy of predictions and, in many cases, the capability to quantify uncertainty. To address this, optimization processes may incorporate statistical and simulation-based models to explicitly account for uncertainty in the domain.

To further illustrate and differentiate the three phases of Business Analytics, the following section will present a concrete use case that exemplifies their practical application.

Prescriptive analytics, the most recent among the three types of analytics, was introduced as a term by IBM in 2010 [4]. Although the body of literature dedicated to it is still limited compared to descriptive and predictive analytics—both of which have had more time to evolve and establish themselves—it is increasingly gaining traction across a wide range of research fields. Many approaches presented in the literature are either explicit implementations of prescriptive analytics or custom strategies tailored to solve particular problems.

1.2 Reichenbach's common cause principle

One of the reasons behind the scarcity of literature and the complexity of prescriptive analytics lies in the challenge of extracting the right information to support effective

decision-making. To properly address the questions "What should we do?" and "Why should we do it?", it is essential to clearly distinguish between correlation and causality.

In statistics, two variables may be correlated—meaning they exhibit a common pattern—but this does not imply a causal relationship between them. A formal articulation of this concept is provided by Reichenbach's Common Cause Principle [5], which, according to the philosophy of probability, states that if two variables are correlated, there exists a third variable (possibly latent or hidden) that causes both, unless there is a direct causal link between them.

A commonly cited example is the observed relationship between the number of ice creams sold and the number of shark attacks at a beach. These two events are positively correlated. If the goal is to reduce shark attacks, one might initially consider reducing ice cream sales. However, doing so would have no effect. The two events are correlated, but neither causes the other. Both are driven by a third factor: rising temperatures during the summer. Hypothetically, decreasing the temperature would lead to fewer shark attacks and fewer ice cream sales.

This example illustrates the importance of distinguishing correlation from causation when making decisions based on data. In the context of prescriptive analytics, having a strong predictive model is a necessary but not sufficient condition for effective prescription. If the model relies solely on proxies rather than true causal variables, it may still achieve high predictive accuracy but fail to identify actionable interventions. Therefore, without uncovering and modeling causal relationships, prescriptive analytics cannot reliably recommend actions, even when predictions are accurate.

1.3 Structure of the thesis

This introductory chapter has presented the context, objectives, and scope of the project carried out during an internship at Aizoon, with a particular emphasis on the PrescrX tool and its role within the field of prescriptive analytics. A conceptual framework distinguishing descriptive, predictive, and prescriptive analytics has been established, highlighting the progression from data interpretation to action-oriented recommendations. Furthermore, the importance of differentiating correlation from causality—an essential aspect of effective decision-making—has been introduced through Reichenbach's Common Cause Principle.

The remainder of the thesis is organized as follows:

- **Chapter 2** provides a literature review of prescriptive analytics, surveying the principal methodological families, including probabilistic models, machine learning approaches, mathematical programming, evolutionary computation, logic-based systems, and hybrid frameworks. The chapter also discusses key

challenges in the field, such as scalability, interpretability, and the lack of standardized evaluation criteria.

- **Chapter 3** introduces the PrescrX tool in detail, explaining its underlying mechanisms and configurable parameters. Special attention is devoted to its local, model-agnostic approach inspired by LIME, the integration of constraints (locked features, bounded values, discrete domains), and the extension to cost function constraints. The chapter also presents two case studies—one based on the MNIST dataset and another on an industrial cleaning process—to illustrate the tool's versatility in both academic and real-world contexts.

- **Chapter 4** compares PrescrX with traditional optimization-based approaches. Through the MNIST and industrial case studies, it highlights the differences in objectives and outcomes between minimal-change prescriptions and maximization-based optimizers, with emphasis on interpretability, plausibility, and computational performance.

- **Chapter 5** proposes a set of original evaluation metrics designed to objectively assess prescriptive models. These metrics include normalized distance, gain of neighbors, feature preservation, and robustness. The chapter applies them to the experimental results obtained in earlier sections, enabling a structured and quantitative comparison between PrescrX and optimizer-based prescriptions.

Finally, Chapter 6 draws conclusions from the research, discussing its contributions, limitations, implications for scientific and industrial practice, and directions for future work.

Chapter 2

Literature review

As previously mentioned in Section 1.1, prescriptive methods are heavily influenced by and dependent on the descriptive and predictive models used in the earlier stages. In fact, many prescriptive models can often be seen as an extension of the predictive model, without a clear boundary between where one ends and the other begins. Furthermore, many models are not based on a single technique, but rather on a combination of methods and technologies, making strict categorization difficult. In the proposed literature review, a distinction among methods is made, although, as already noted, it is not particularly rigid.

2.1 Methods proposed in the literature

This section provides a review of the literature on models used to perform prescriptive analytics. While not all of these models were originally developed with a prescriptive purpose in mind, many have been effectively adapted for this domain. A notable example is the use of optimization techniques, which, despite their broader applications, currently represent the most widely adopted tools for generating actionable prescriptions. The reviewed literature includes contributions from a variety of application contexts and methodological approaches, although the number of studies is not uniformly distributed across them. Special attention is given to techniques that incorporate machine learning models and mathematical programming, as these have garnered substantial interest in recent research efforts focused on prescriptive analytics.

Probabilistic models

The main idea of a probabilistic model is to solve a problem using a statistical or distributional approach, where the analysis of variables goes beyond their actual values to also consider their uncertainty, variability, and distribution. As

a result, the output of probabilistic models consists of a set of possible events, each associated with a probability of occurrence, defined by the initial parameters and their uncertainties. The main limitation of these models is their scalability: the complexity of the analyzed problem increases in proportion to the number of variables considered. High-dimensional problems may not be fully captured or interpreted by probabilistic models, as the computational and analytical demands grow significantly with each additional variable[6].

Martinez, Cristaldi, and Grau [7] propose a comprehensive data analytics process aimed at maximizing penicillin productivity through a bioreactor. Prescriptive analytics represents the final phase of the analytical cycle. In the initial phase (descriptive analytics), the bioreactor parameters are modeled as probability distributions to capture uncertainty and correlations among them. Then, using a Bayesian framework, a distribution of potential penicillin yields is derived (predictive analytics). Finally, a run-to-run optimization is applied to determine the optimal parameter adjustments that enhance productivity while minimizing uncertainty.

In a different scenario, a probabilistic model is proposed to support Prescriptive Maintenance decisions under uncertainty[8]. In the aeronautical sector, a Monte Carlo simulation has been developed to estimate cost-benefit distributions. Subsequently, the expected cost variation ($E(\Delta C)$), the risk of incorrect decisions, the Expected Opportunity Loss (EOL), and the Expected Value of Information (EVI) have been computed in order to prescribe the optimal combination of cost, benefit, and risk.

Both proposed methods have the advantage of managing the uncertainties and correlations of the parameters in play, but these are solutions built ad hoc for the case studies, with limited scalability in the aeronautical case.

Machine learning

Machine learning (ML) is a branch of artificial intelligence that focuses on building systems capable of learning from data to make predictions or decisions without being explicitly programmed. Unlike probabilistic models, which rely on predefined statistical assumptions and focus on modeling uncertainty through probability distributions, ML models learn patterns directly from data, often capturing complex, nonlinear relationships. One major advantage of ML is its scalability: it can efficiently handle large, high-dimensional datasets where probabilistic models may struggle[9, 10]. Additionally, ML models often require less manual feature engineering and can adapt more flexibly to different types of data, making them more suitable for tasks like image recognition, natural language processing, and real-time decision-making. However, their black-box nature often hinders interpretability, making it difficult to fully answer the question, “Why should a specific action be

taken?" (as discussed in Section 1.1).

Despite this limitation, machine learning has been successfully applied in several prescriptive analytics contexts. For example, to support bulk ordering of critical spare parts in the shipping industry, a decision-making tool based on prescriptive analytics and machine learning was developed to improve the planning and execution of annual maintenance spare parts orders for a fleet of 75 ships [11]. Critical high-impact components were identified using clustering techniques, while the required quantities were predicted using a Random Forest model. Finally, all possible combinations proposed by the Random Forest were generated, and the minimum-cost solution was selected. The limited amount of data analyzed results in reduced scalability and versatility of the model, although the identification of high-impact orders has led to significant administrative benefits.

Machine learning techniques are often integrated with other methods, particularly with mathematical programming and optimization algorithms. Pessach Dana, Singer Gonen, and their colleagues [12] propose an analytics framework that applies machine learning models to support HR recruitment decisions and improve hiring outcomes. Using a Variable-Order Bayesian Network (VOBN) model, which remains interpretable for recruiters, they predict which candidates are most likely to succeed. Subsequently, the selected candidates are assigned to job positions through a Mixed Integer Linear Programming (MILP) model, aiming to maximize hiring success. Unlike the previous case, the availability of a large volume of data and the interpretability of the models facilitate broader adoption and practical use.

To address inefficiencies in outpatient appointment scheduling systems, exacerbated by patient no-shows, Sharan Srinivas and A. Ravi Ravindran propose a stacking approach that combines several machine learning models [13]. The idea is to predict the no-show risk for each patient and schedule appointments accordingly, following prescriptions specifically designed for the problem and validated through prior simulations. The proposed solution proves effective due to the generalization capabilities of the stacking technique; however, model interpretability is limited, and local adaptations are required for each clinic.

Mathematical programming

Mathematical programming, also known as mathematical optimization, is a discipline that formulates decision-making problems as mathematical models. These models consist of an objective function to be optimized (either maximized or minimized), decision variables, and a set of constraints that represent the problem's limitations or requirements [14]. In predictive contexts, mathematical programming can be used to model and forecast outcomes based on certain inputs. In prescriptive contexts, it goes a step further by not only predicting outcomes but also recommending optimal decisions to achieve desired objectives, considering the

constraints and possible scenarios.

Compared to ML, which excels at identifying patterns and making predictions from large datasets, mathematical programming provides a structured approach to decision-making by explicitly modeling the relationships and constraints within a problem. While ML can handle complex, high-dimensional data and adapt to new patterns, it often lacks the explicit constraint-handling capabilities of mathematical programming. Therefore, mathematical programming is particularly advantageous in scenarios where decisions must adhere to strict rules or limitations, such as resource allocation, scheduling, and logistics. Conversely, ML is more suitable for problems where learning from data to make predictions is paramount, such as image recognition or natural language processing[15].

The popularity of these methods stems from their interpretability, the extensive literature supporting them, and their broad applicability. Common forms include linear and nonlinear programming, as well as their subtypes: linear integer programming, mixed-integer programming, and binary linear programming.

Advanced strategies such as stochastic, adaptive, and robust optimization have also been explored. Recent developments include Bayesian Optimization and Constrained Bayesian Optimization, though these are more complex to implement in real-world scenarios.

Examples of applications include:

- In the context of solution selling, where assigning the right working team to a customer account is crucial for maximizing revenues, a recent study proposes an information system that integrates predictive analytics through the use of a Multilayer Perceptron Neural Network (MLP) to estimate the probability of a team's success with a specific client. Subsequently, an Integer Linear Programming (ILP) model is used to optimally compose the team to meet customer needs [16]. Although a Neural Network model is proposed, most of the work focuses on the optimization phase, aiming to offer alternative scenarios beyond the optimal solution;

- The simplicity of formulating problems solved through linear programming has enabled the development of one of the first commercial products for prescriptive analytics. Using LogiQL—a unified, declarative language that integrates queries, rules, triggers, statistical models, and optimization models—it is possible to formulate problems that are automatically solved via Linear Programming or Mixed Integer Programming. Although using this commercial platform still requires domain expertise, it guarantees a high degree of flexibility [17];

- Hosting events like the FIFA World Cup requires careful lodging planning due to the influx of international visitors. The specific case analyzed concerns

Qatar 2022, where the pre-existing hotel infrastructure was insufficient to meet the expected demand. To optimize the overall accommodation plan, Integer Programming models were developed to assign groups to stadiums while maximizing balanced occupancy of the facilities [18]. These models not only predict the number of expected spectators but also prescribe: how to form groups, how to assign them to stadiums, and how to scale lodging capacities under uncertain scenarios;

- Electric utilities face growing challenges in managing aging, geographically dispersed assets while balancing reliability, regulations, and budgets. To address this, advanced analytics solutions have been developed to assess asset health and network reliability by optimizing maintenance planning through stochastic programming models and sequential optimization based on minimizing the expected total cost. This approach supports decision-making regarding when and how to perform maintenance or replace assets, prioritizing interventions based on geographic location and criticality [19]. Although the proposed solution successfully handles a high level of problem complexity, maintaining it requires constant, difficult-to-manage, and non-automatable updates.

Methods based on Machine Learning and Mathematical Programming have been increasingly developed in recent years, particularly for decision-making processes. As shown in Figure2.1, these two approaches are experiencing a growing number of publications, as tracked by Wissuchek Christopher and Zschech Patrick, who categorized the publications accordingly [20].

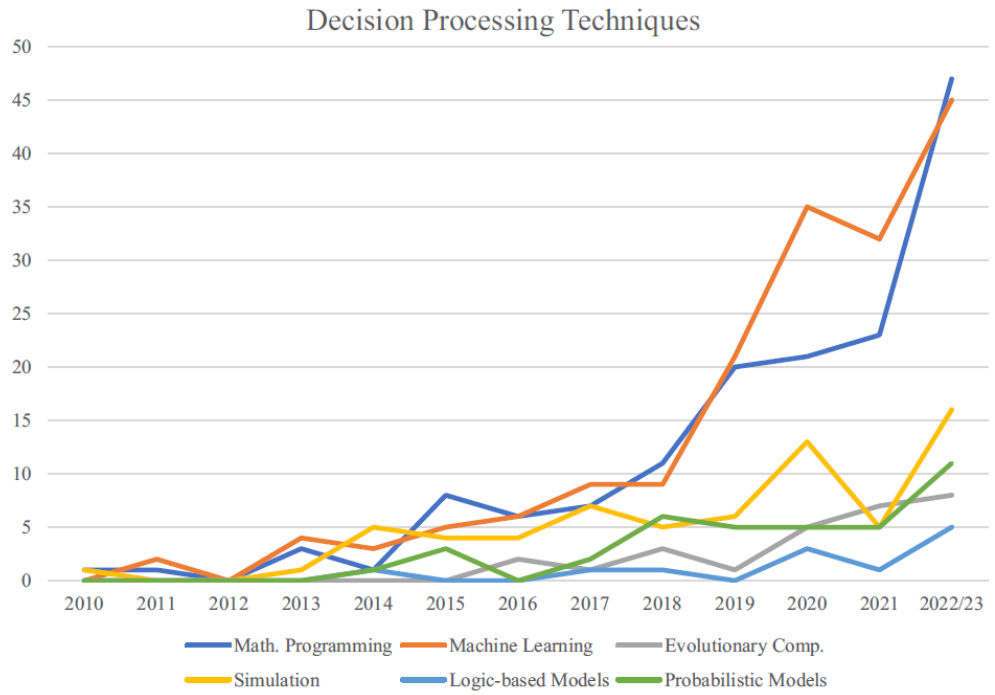

Figure 2.1: Number of publications divided by category and year.

Statistical analysis and evolutionary computation

When data becomes too large or complex for traditional mathematical programming, hybrid models combining statistical analysis and evolutionary computation are often employed [21]. These methods are especially effective when fast, approximate solutions are required.

Statistical models, such as regression and Bayesian inference, provide a structured framework for analyzing data variability and uncertainty. They enable hypothesis testing and causal inference, especially when data is noisy or incomplete. However, their effectiveness can be limited with large-scale or unstructured data and when underlying assumptions are not met.

Evolutionary computation, inspired by natural selection, includes algorithms like genetic algorithms and particle swarm optimization. These are well-suited for optimizing complex, non-linear, or non-differentiable functions. They can explore vast solution spaces and adapt over iterations [22, 23], though they may require many evaluations and do not guarantee global optima.

In contrast, machine learning excels at processing large datasets and identifying complex patterns, often achieving high predictive performance. Yet, ML models can lack interpretability and typically demand large amounts of training data.

Furthermore, in many scenarios testing decisons can be expensive, especially

when the complexity of the case is elevated (marketing multicanale, medical treatments). With the purpose to solve this dynamic has been proposed ESP method (Evolutionary Surrogate-Assisted Prescription) where a Surrogate model (es: random forest o neural network) imita il comportamento reale a partire da dati storici and a Prescriptor (una rete neurale) evoluto tramite algoritmi genetici massimizza gli outcome previsti dal surrogate. ESP non si limita a predire gli esiti delle azioni, ma prescrive direttamente quali azioni compiere ottimizzando gli obiettivi desiderati[24]. Questo approccio garantisce un'alta efficienza e versatilità, ma comunque nell'elaborato si sottolinea quanto sia importante la bontà del surrogate per ottenere una prescrizione di qualità; inoltre è necessario definire dei confini precisi per lo spazio di ricerca delle policy, altrimenti l'Evoluzione genetica potrebbe risultare troppo lenta e inefficace.

In a different case study, evolutionary algorithms have been applied to optimize nonpharmaceutical intervention strategies (NPI) for controlling COVID-19. It's reproposed the ESP technique with a surrogate model LSTM (Long Short-Term Memory) predict l'andamento dei casi COVID-19 a partire da dati storici su contagi e NPI and prescriptor (rete neurale evoluta tramite algoritmi genetici tipo NSGA-II) genera strategie ottimali di NPIs in ordeto to maximize the afficacy of these actions minimizing costs [25].

In yet another distinct case, advanced analytics were applied to optimize university admissions within Thailand's multi-round admission system. Predictive models—such as generalized linear models, deep learning, and gradient boosted trees—were used to forecast student performance. These predictions informed a prescriptive model using evolutionary optimization to identify optimal admission criteria and student allocation across engineering majors and admission rounds [26].

Logic-based models

Logic-based models use formal logic to represent knowledge and perform reasoning through structures like first-order logic, rule-based systems, and ontologies [27]. They are especially valuable in domains requiring transparency and rigorous inference, such as legal reasoning, semantic web, and safety-critical systems. Implemented through rule-based frameworks, these models integrate expert knowledge into prescriptive systems. Their deterministic nature and high interpretability contrast with the "black box" nature of many machine learning models. While logic models excel in explainability and logical rigor, they face challenges with scalability and handling ambiguous or noisy data. In such contexts, statistical or evolutionary models may offer better adaptability[28].

In the work by Cindy G. De Jesus and Mark Kristian C. Ledda [29], a support system for interventions targeting students at risk of dropping out (SARDO) in public schools is proposed, using a prescriptive analytics model based on fuzzy

logic. The system predicts dropout risk based on familial, individual, school, and community factors, and prescribes personalized interventions based on fuzzy rules that combine applicability and historical effectiveness. Results show that this approach improves the timeliness and accuracy of corrective actions.

To enhance decision support in agriculture in France, several methods have been proposed to address the same problem from different perspectives, all starting from a minimalist agricultural taxonomy (seven main categories) [30]. Different machine learning models are combined through a Boltzmann Machine to predict various agricultural scenarios. The best decision (maximizing productivity, minimizing risk) is determined using Particle Swarm Optimization (PSO), which simulates a "swarm" of possible solutions seeking the optimal one. However, since the objective functions are difficult to formalize in these contexts, fuzzy rules derived from expert knowledge are used. Two main approaches are proposed: Fuzzy Linear Programming (FLP), which performs linear optimization under data uncertainty; and Fuzzy Rule-Based Systems (FRBS), which manage approximate logic through expert systems.

Other methods and approaches proposed do not follow a rigorous mathematical formulation; instead, most of the attention is devoted to developing user-friendly and interactive tools, so they can be used even by non-domain experts, although this compromises the quality of the models. For example, one application involves a mapping method that extracts domain ontology information from CVs (skills, education, experience) [31]. The analyzed problem is addressed through a distributed architecture for job-to-curriculum matching using semantic technologies. Compared to other studies (e.g., those using MILP or genetic algorithms), here the prescription is built more on an advanced semantic architecture and intelligent matching rather than relying on a heavy mathematical solver. It is a lighter and more scalable approach, though less rigorous from the perspective of "formal optimization".

Another example is the EventAction system, which presents and explains time-based recommendations. The paper addresses the problem of recommending optimal actions based on temporal event sequences, helping users define action plans that increase the probability of achieving a desired outcome. It represents an interactive form of prescriptive analytics: the system does not automatically decide but supports users in making better decisions through data, visualizations, and suggestions [32].

From this analysis, it becomes clear that machine learning and deep learning models are widely used as background analyses of the problem under examination and for the evaluation of plausible scenarios. Furthermore, optimization concepts (such as Linear Programming and related techniques) are commonly applied to identify the best decision to be made, aiming to minimize or maximize the objective function.

2.2 Challenges

According to the literature, prescriptive analytics is trending toward models that can operate in real-time, streaming environments, especially in the context of the Internet of Things (IoT) [1]. The goal is to develop systems and tools that minimize the need for human intervention and dynamically update in response to new data streams.

Key challenges and future directions include:

- Developing real-time, sensor-driven information systems and recursive algorithms for large-scale applications;
- Leveraging distributed computing to handle high data volumes and enable rapid decision-making in IoT environments;
- Managing uncertainty stemming from both predictive models and data quality;
- Establishing generic metrics for evaluating individual prescriptions or overall prescriptive model performance—currently, these are mostly assessed through human judgment;
- Transitioning from expert-based models to AI-based systems capable of higher abstraction and autonomy.

There is also ongoing interest in integrating prescriptive analytics into broader data management and analytics frameworks [4]. This includes developing standardized workflows and software libraries at both managerial and technical levels, as well as embedding prescriptive methods into software engineering processes.

In this work, we propose a prescriptive model inspired by the principles of eXplainable Artificial Intelligence (XAI). The objective is to create interpretable models based on machine learning or deep learning, and to compare these with optimization-based models using specific metrics for evaluating individual prescriptions.

Chapter 3

What is PrescrX and How it works

PrescrX is a tool that enables prescriptive analysis even in multi-class classification problems. Given a specific point, a trained classifier, a background dataset, and a target class, the algorithm returns the minimal changes needed to the point's features so that the model classifies it into the desired category.

PrescrX is based on ideas from eXplainable Artificial Intelligence (XAI), particularly inspired by the LIME (Local Interpretable Model-agnostic Explanations) technique [**lime**]. LIME explains predictions by creating samples around the input point and learning a local decision boundary that reflects how the model behaves nearby. Usually, the classifier relies on a complex and unknown function $f(x)$ to separate the classes. Because of this complexity, it is not possible to reason analytically or geometrically about how to change the point to alter its classification.

LIME deals with this by approximating the local decision boundary with a simpler linear function $g(x)$. It perturbs the input point to generate a neighborhood of synthetic data, uses the model to predict their classes, and then fits a linear classifier to separate them. This results in a local, linear, and interpretable approximation of $f(x)$. PrescrX applies this same approach: it approximates $f(x)$ locally with a linear classifier $g(x)$ to find the region where points are classified as belonging to the target class.

An important characteristic of PrescrX is that it is model-independent. It does not rely on the internal structure of the predictive model but instead operates on its outputs, meaning that it can be applied regardless of the classification algorithm used, whether neural networks, decision trees, support vector machines, or other classifiers.

Furthermore, PrescrX is a patented proprietary software, developed and legally protected to safeguard its originality and industrial applicability. Its deployment in

industrial contexts therefore requires explicit licensing, ensuring that the methodology is both rigorously defined and commercially regulated.

3.1 How PrescrX works

PrescrX is designed to generate prescriptive recommendations by identifying the minimal changes needed to reclassify a given data point into a desired target class. It achieves this by constructing a local and interpretable approximation of the classifier's decision boundary, which enables actionable suggestions even when the underlying model is complex or opaque.

A comprehensive understanding of how PrescrX operates requires the introduction of key parameters that govern its behavior. This section provides a general overview of these parameters and outlines the algorithm's core logic, while more detailed explanations will follow in subsequent sections.

- $X_{\textbf{background}}$: The set of background points used for analysis.
- **Classifier model**: The classification model responsible for assigning labels to $X_{\text{background}}$.
- **Starting point** (sp): The initial instance for which a prescription is to be generated.
- **Target class** (tc): The desired class to which the starting point should be reassigned.
- **Locked indices**: Feature dimensions of sp that remain unchanged throughout the process.
- **Bounds**: The upper and lower limits imposed on each feature of sp.
- **Discrete indices**: Feature dimensions of sp that must take integer values.
- **Trust threshold** ($trust_t$): When the classifier outputs a probability distribution, a point is considered trustworthy if its assigned probability exceeds this threshold.
- **Regularization parameter** (α): The l_1-norm regularization parameter for the linear approximation step.

In general terms, PrescrX analyzes the space defined by $X_{\text{background}}$ in the vicinity of a given starting point sp, with the goal of moving the point closer to a region populated by points that belong to the target class tc. These target points are selected from the dataset when the probability of belonging to the target class

exceeds the threshold $trust_t$. To ensure a reliable fit of the linear model, a dense sampling is performed between the initial point *sp* and the selected points. Once the linear model is established, it becomes possible to define the minimal changes required for the point to reach *tc*.

There are scenarios in which specific feature values are subject to constraints. For instance, features may need to stay within certain upper and lower bounds, take only discrete (integer) values, or remain fixed. These conditions can be handled within PrescrX by specifying *Locked Indices*, *Bounds*, and *Discrete Indices*. For instance, in the medical field, drug dosages must remain below a certain limit to avoid harmful effects. Similarly, in industrial settings, operational parameters such as machine temperature may need to be maintained within predefined thresholds to ensure safety and functionality.

Now that a general overview of the functioning has been provided, we can examine each step in detail to understand the rationale behind their implementation and their specific roles.

3.1.1 Selection of target points

From $X_{\text{background}}$, a subset of *candidate points* is selected—these are the points assigned to the target class *tc*. At this stage, the threshold $trust_t$ is not considered: even if the classifier outputs a probability distribution, the class with the highest predicted value is used.

Next, the Euclidean distance is computed between the starting point *sp* and each *candidate point*. An iterative process then begins, governed by two additional PrescrX parameters: n_{closest} and m, where n_{closest} defines the number of *target points* to be selected, and m indicates the number of neighbors considered for each candidate.

In the first iteration, the closest n_{closest} candidate points are selected. A candidate point becomes a valid *target point* if at least $\frac{m}{2}$ of its neighbors also belong to *tc*. In subsequent iterations, $\frac{n_{\text{closest}}}{2}$ additional candidate points are considered at a time. The process stops once n_{closest} valid target points have been selected.

The choice of n_{closest} and m significantly influences the outcome of PrescrX. The number of target points determines the size of the explored region in the space: a larger value increases the coverage, but it may also slow down execution and, in cases where $X_{\text{background}}$ is not large enough, may result in an insufficient number of valid points. The parameter m controls the consistency of the selected target points. Higher values ensure more reliable target points, as they are surrounded by a greater number of neighbors from the same class *tc*. However, setting m too high may make it difficult to find valid target points, again depending on the density and size of $X_{\text{background}}$.

In cases involving large datasets, high dimensionality, or both, execution time

and computational cost can become significant. The time complexity of computing Euclidean distances is $\mathcal{O}(n \cdot d)$, where n is the number of candidate points (in this case, m) and d is the dimensionality of the space. The iterative method is used specifically to avoid computing distances for all candidate points when only n_{closest} will be retained in the final selection.

3.1.2 Generating dense sampling

Unlike the standard sampling used in the LIME model, which aims to interpret local behavior, the goal of PrescrX is to find a direction along which to move in the feature space. Once the target points are identified, the next step is to generate a densely populated region between the starting point *sp* and the target points. This ensures that, when the linear model is fitted to separate the starting and target classes, there are enough points to enable a consistent and reliable fitting.

This step is also carried out through an iterative process. In each iteration, n target points are randomly sampled. For each sampled point, a new point is generated according to the following formula:

$$\text{new\_point} = \text{starting\_point} + (\text{sampled\_point} - \text{starting\_point}) \cdot \mathcal{U}(\text{low}, 1) \quad (3.1)$$

Here, the new point is obtained by perturbing *sp* uniformly along the direction of the sampled point. It is important to note that the uniform perturbation $\mathcal{U}$ is applied independently to each dimension of the point. While the upper bound of the uniform distribution is fixed at 1, the lower bound varies across sampled points and can take on values in {0, 0.25, 0.75}, distributed equally among the n sampled points. This variation helps ensure a higher density of generated points near the target points, as the final prescription will likely be close to them.

To provide a visual representation of this procedure, Fig. 3.1, 3.2, and 3.3 illustrate the density of generated points in a two-dimensional space, where the intensity of color reflects the population density. Figs. 3.1 and 3.2 show the density along the x_1 and x_2 axes, respectively, while Fig. 3.3 shows their combined effect.

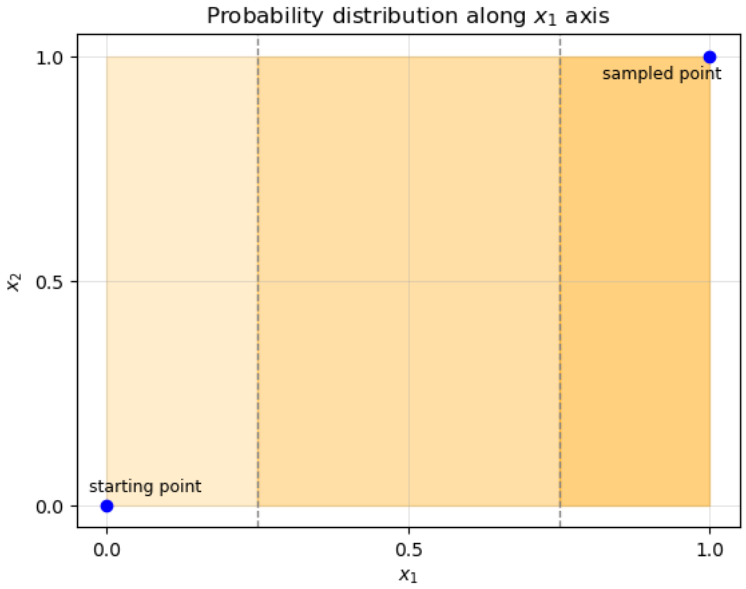

Figure 3.1: Density of points along x_1 axis

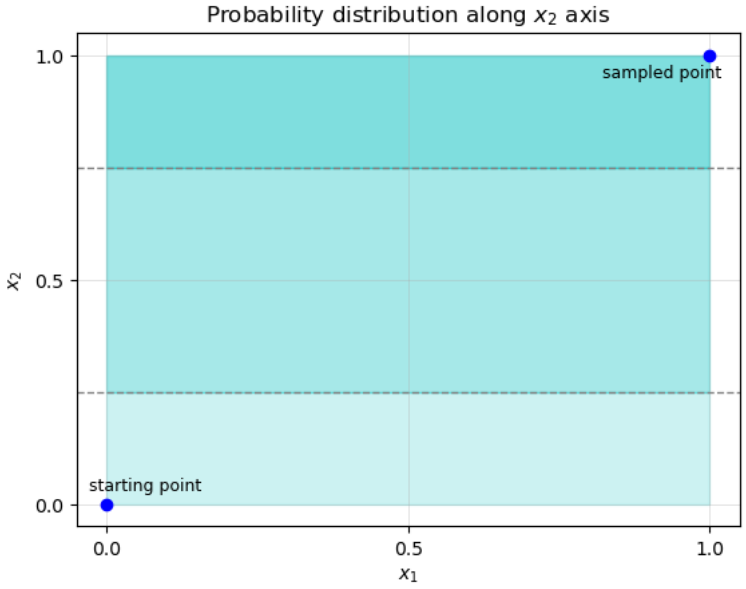

Figure 3.2: Density of points along x_2 axis

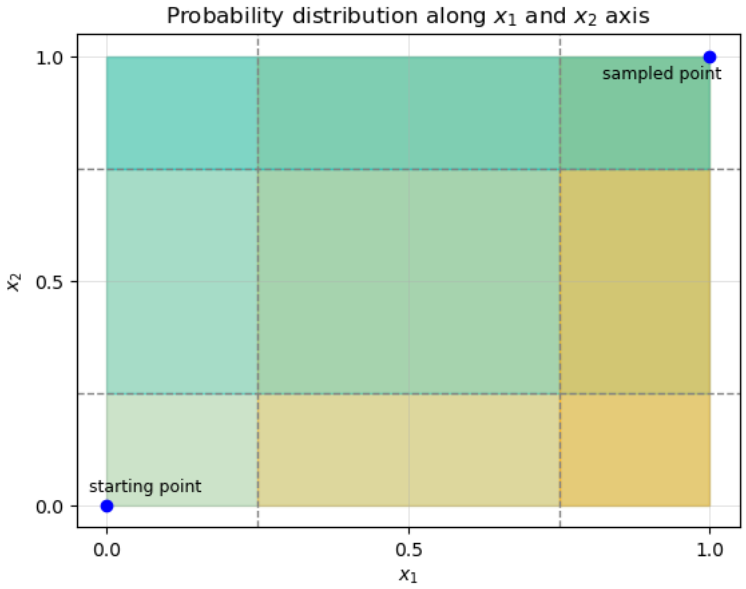

Figure 3.3: Combined density of points between x_1 and x_2 axes

If any of the newly generated points belongs to the same class as *sp*, it will become the new *sp* for the next iteration. The iterative process continues until the total number of generated points is at least four times the number of dimensions of the starting point, and at least half of the points belong to the target class *tc*.

This biased generation of points addresses two main challenges. First, in high-dimensional spaces, exploring the full neighborhood around *sp* would be computationally expensive and time-consuming. Second, it ensures a sufficient number of points between *sp* and the target points, which is essential for building a reliable linear separator that approximates the true (and complex) decision boundary of the classifier.

3.1.3 Fitting a local linear model

Now that the local region of space has been densely populated, a Support Vector Classifier (SVC) can be fitted. This fitting provides a separating hyperplane that will act as the decision boundary for the linear approximation of the model's complex behavior. A graphical representation of this process is shown in Fig. 3.4.

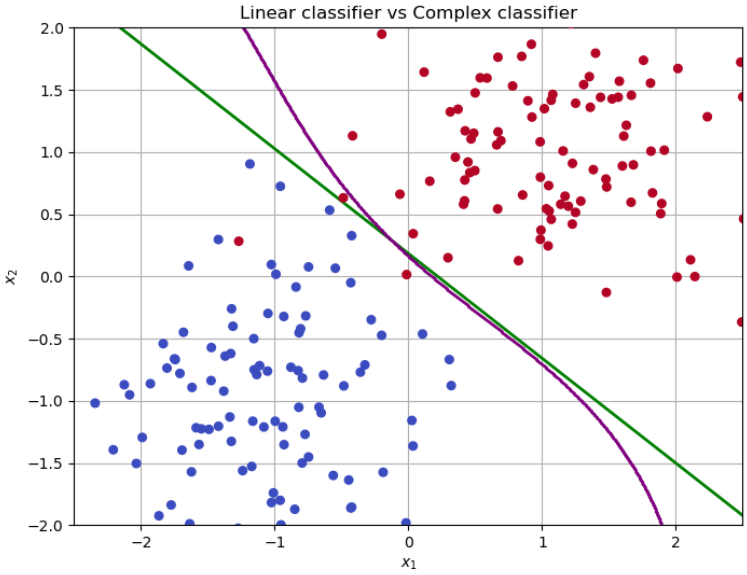

Figure 3.4: Comparison between the linear approximation and the real classifier.

In the figure, the data points are colored red or blue according to their class. The purple line represents the true classifier, while the green line is its local linear approximation. As discussed in Section 3.1, this linear model can be customized using a regularization parameter α, which helps prevent underfitting or overfitting during approximation.

3.1.4 Quadratic programming formulation

Once the linear approximation has been obtained, all the elements required to compute the prescription are available.

Recalling the primary objective of PrescrX—making the smallest possible changes to move the point *sp* into the target class *tc*—the next task is to find the shortest path to reach that class. This is formulated as a Quadratic Programming (QP) problem, defined as follows:

- **Objective Function**: Minimize the squared Euclidean distance from the starting point, expressed as $\|x - \text{starting\_point}\|^2$.
- **Constraint**: An equality constraint ensuring that the point x lies on the hyperplane defined by the linear model.

The QP is written in standard form:

$$\begin{aligned} &\text{Minimize} \quad \frac{1}{2}x^T Px + q^T x \\ &\text{subject to} \quad Ax = b \end{aligned} \tag{3.2}$$

with the parameters defined as:

- $P = 2I$, where I is the identity matrix, ensuring positive semi-definiteness.
- $q = -2 \cdot \text{starting\_point}$.
- A: the coefficients of the linear model.
- b: the intercept of the linear model, defining the separating hyperplane.

Geometrically, this formulation corresponds to projecting the starting point *sp* onto the hyperplane defined by the linear model. The line connecting these two points is orthogonal to the separating plane.

It is important to note that this process does not guarantee that the projected (i.e., prescribed) point will actually belong to the target class *tc*, since the linear model is only an approximation of the real classifier. Therefore, two outcomes are possible:

- **Case 1**: The prescribed point belongs to the target class *tc*, and—if a probability is required—it does so with a confidence greater than $trust_t$. In this case, the process is complete.
- **Case 2**: The prescribed point either does not belong to *tc*, or it does but with a confidence below $trust_t$. In this case, the process described in Section 3.1.2 is repeated. The closest point to the current prescribed point is considered, and the one that satisfies all required conditions is selected as the final prescribed point.

3.2 Current constraints and their expansion

As previously mentioned, PrescrX can operate under specific restrictions dictated by the problem at hand. These include *Locked Indices*, *Bounds*, and *Discrete Indices*, each integrated into the prescriptive process in distinct ways. Below is a brief explanation of how each constraint is managed:

- **Locked Indices**: These constraints ensure that certain dimensions or features remain unchanged throughout the process. That is, specific dimensions of the starting point *sp* must retain their original values. During Dense Sampling

(Section 3.1.2), the perturbation applied to *sp* avoids modifying the locked dimensions, focusing only on the free ones. Additionally, after the QP step (Section 3.1.4), the locked features are explicitly restored to their original values, as the optimization might not preserve them automatically.

- **Integer-Value Constraints**: Some dimensions of the starting point are required to take only integer values, typically to ensure compatibility with discrete domains. This constraint affects both the Dense Sampling and the QP optimization steps. During Dense Sampling, values for these dimensions are sampled as integers. After the QP step, the affected features are rounded to the nearest valid integer values.
- **Bounded Ranges**: Certain features are limited to lie within specific bounds, defined by lower and upper limits. These bounds may be finite or extend to infinity (e.g., $\pm\infty$). During Dense Sampling, generated values are restricted to these bounds. In the QP step, the bounds are directly incorporated into the optimization formulation. The updated version of the QP problem becomes:

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} \quad & Ax = b \\ & lb \leq x \leq ub \end{aligned} \tag{3.3}$$

  where lb and ub represent the lower and upper bounds for each feature.

All of these constraints can be applied simultaneously, allowing PrescrX to handle complex problem formulations. If the QP step fails to produce a point belonging to the target class tc, an additional Dense Sampling step is triggered—still respecting all defined constraints—to explore further solutions.

3.2.1 Cost function constraint

To enhance PrescrX's flexibility, a new type of constraint has been introduced: a *cost function constraint*. This addresses scenarios where each point $x \in X_{\text{background}}$ is associated with a cost, expressed through a generic (not necessarily linear) function $c(x)$. The constraint consists in enforcing a maximum allowable cost $cost_t$, such that the prescribed point must satisfy:

$$c(x_{\text{prescribed}}) \leq cost_t$$

This constraint modifies the Target Point Selection step (Section 3.1.1) slightly. In addition to meeting the consistency condition defined by parameter m, candidate target points must also satisfy the cost condition:

$$c(x_{\text{target}}) \leq cost_t$$

This effectively narrows the selection to a subregion of the space where both the class consistency and cost conditions are met.

As with the previous constraints, the cost function constraint can be combined with all others. Fig. 3.5 provides a visual representation of how this constraint, in combination with the classifier's decision boundary, restricts the solution space. The shaded region in the bottom-left corner represents the intersection of the classifier region and the region under the cost threshold $cost_t$, indicated by the orange boundary.

It is important to note that these regions might not intersect, depending on the problem's formulation. In such cases, PrescrX will terminate without producing a prescription, as no feasible solution exists.

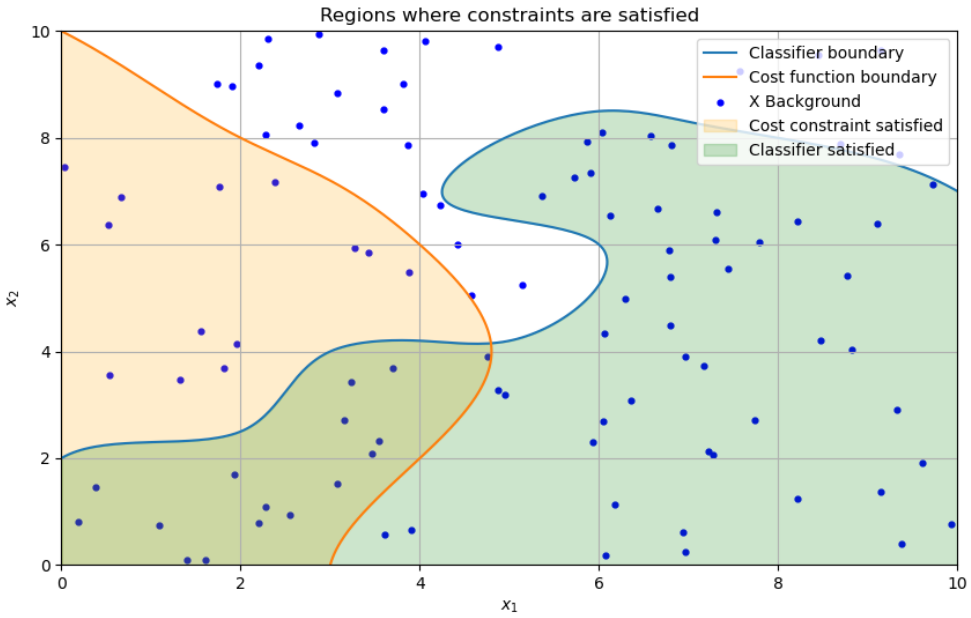

Figure 3.5: Intersection between the classifier region and the cost-constrained region.

3.3 Exploring parameters

To investigate the behavior of PrescrX, two types of case studies are proposed.

The first case study involves the MNIST dataset. MNIST (Modified National Institute of Standards and Technology database) is a large dataset of handwritten digits commonly used for training and evaluating image processing systems [33].

It contains 60,000 training images and 10,000 test images, each representing a digit from 0 to 9. In this study, each image is flattened into a linear vector of 784 dimensions (originally a 28×28 pixel image). A convolutional neural network is trained to classify these digits according to their corresponding labels. The structure of the neural network is as follows:

- **Input Layer**: shape = 784
- **Dense Layer**: 256 units, activation = ReLU
- **Dropout Layer**: rate = 0.5
- **Output Layer**: `num_classes` units, activation = Softmax

The choice of MNIST is motivated not only by its popularity and high dimensionality, which make it a standard benchmark in the machine learning community, but also by the specific nature of the prescriptions generated in this context. Since prescriptions correspond to modified images of handwritten digits, their validity can be directly assessed by human supervision. In other words, it is possible to visually inspect whether the transformed image plausibly resembles the intended digit, thereby providing an additional, intuitive form of validation beyond numerical metrics.

The second case study is based on a real-world industrial scenario. In this setting, an industrial machine cleans mechanical components using seven operational parameters. These seven parameters form the features used to train a classifier for this task. At the end of the cleaning process, each component is classified as either dirty or clean—defining a binary classification problem.

All cleaned components are identical in structure, allowing the cleaning quality to be assessed solely based on the values of the seven parameters. The dataset includes a total of 143 samples. Each sample consists of the parameters used in the cleaning process and a class label: 0 for clean and 1 for dirty. A decision tree classifier is trained on this dataset. While the internal structure of the model is not disclosed due to legal restrictions, the classifier is treated as a black-box model—used purely as a ground truth oracle for evaluating PrescrX. The internal reasoning behind its predictions is not a focus of this study.

These two case studies were chosen to explore, respectively, the behaviors in a widely studied and data-rich dataset that remains a benchmark (MNIST), and in a scenario where, despite limited available data, the context is realistic (industrial case).

3.3.1 Exploration and time complexity with n_{closest} and m

To analyze how the parameters n_{closest} and m influence the prescription process, experiments were conducted using the MNIST dataset. In this setup, the starting

point always belongs to class 1, and the target class is fixed at 8. All other parameters required for the prescription remain constant across experiments: the regularization parameter for the linear model is set to $\alpha = 0$, indicating no regularization, and the trust threshold is fixed at $trust_t = 97.5\%$, to ensure a stable and high level of confidence.

As previously discussed, n_{closest} and m define the extent of the space explored during the prescription process. Increasing their values generally leads to greater computational demands. To study this phenomenon, 100 prescriptions were executed, with both parameters ranging from 10 to 100 in steps of 10. The results are shown in Figure 3.6, where each point corresponds to a prescription with n_{closest} and m marked along the axis and warmer colors correspond to longer execution times (measured in seconds). The trend is primarily driven by n_{closest}, although the influence of m could be more thoroughly analyzed with a larger number of trials. Notably, the plot highlights four specific points: the two slowest and two fastest prescriptions, which appear on opposite sides of the chart.

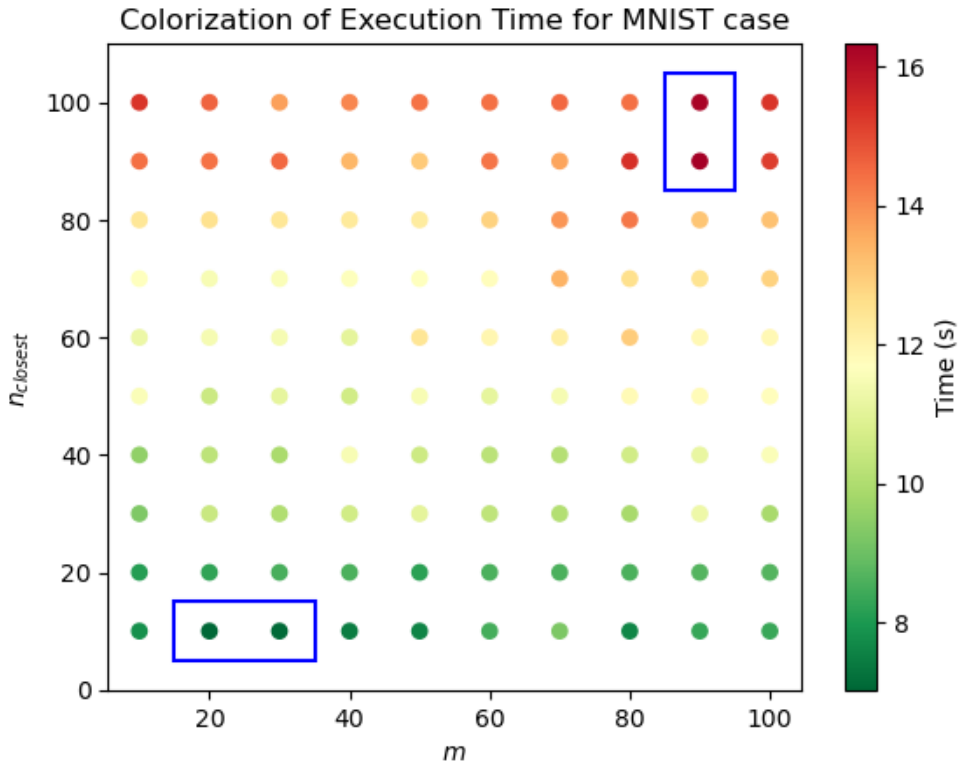

Figure 3.6: Colorization of execution time required to make the prescription in the MNIST case.

In the industrial case study, the prescriptions are performed using the same

parameters, with the exception of $trust_t$, which is not needed since the classifier is a decision tree. The starting point sp is a sample classified as dirty (class 1), and the goal is to prescribe changes that make it clean (class 0). In this case, n_{closest} and m range from 2 to 20 in steps of 2.

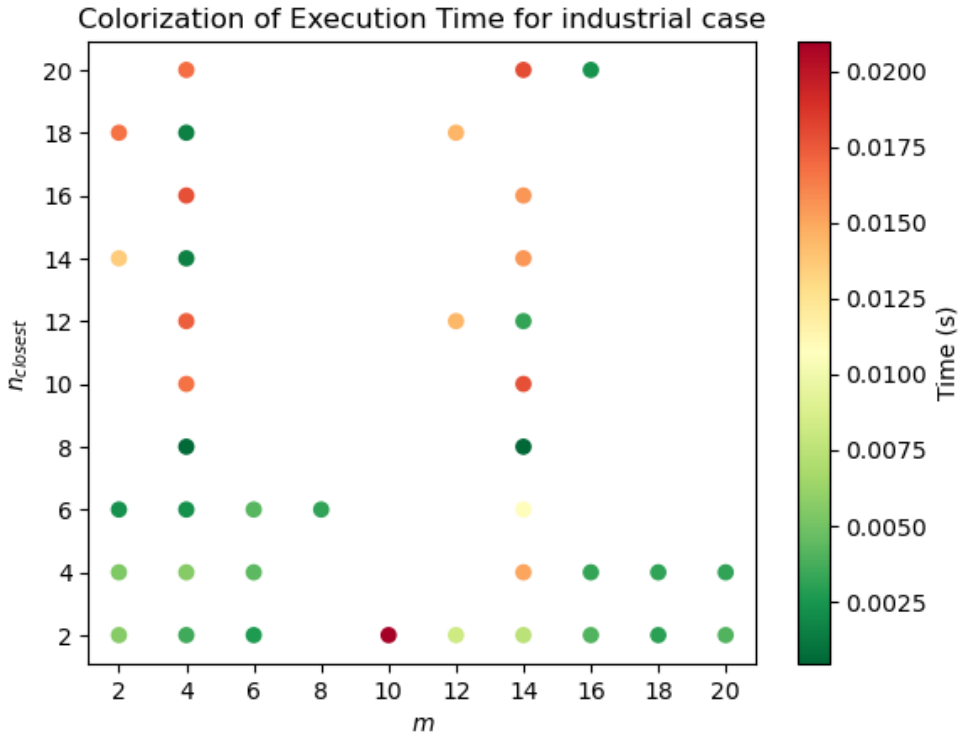

Figure 3.7: Colorization of execution time required to make the prescription in the industrial case.

What stands out in Figure 3.7 is the absence of many points, which correspond to failed prescriptions. These failures are likely due to the limited size and coverage of the dataset, which restricts the explorable space. Nonetheless, a general trend is still visible: execution time increases as n_{closest} and m grow. Additionally, comparing the legends of both figures reveals a clear difference in the time required for prescriptions. The MNIST case demands significantly more time due to the larger values of the two parameters and the much higher dimensionality of the explored space.

3.3.2 Quality of prescriptions with trust_t (trust threshold) and alpha (regularization parameter)

The trust threshold $trust_t$ can significantly affect the quality of a prescription, as it defines the minimum confidence required for a point to be considered as belonging to a given class. As previously discussed, the quality of a prescription is directly related to the reliability of the underlying classification. Since the classifier used in the industrial case is a decision tree that produces deterministic outputs, this parameter is evaluated only in the MNIST case.

Currently, there is no established standard in the literature for quantitatively measuring the quality of a prescription. However, in this context, the result can be visually assessed by inspecting the image generated at the end of the PrescrX process.

Using the same experimental setup as in the analysis of n_{closest} and m, with both parameters fixed at a value of 50, we observe how varying the $trust_t$ value affects the outcome. The tested values are $\{95\%, 99.9\%, 99.99\%\}$, and the corresponding prescribed digits are shown in Figures3.8, 3.9, and 3.10.

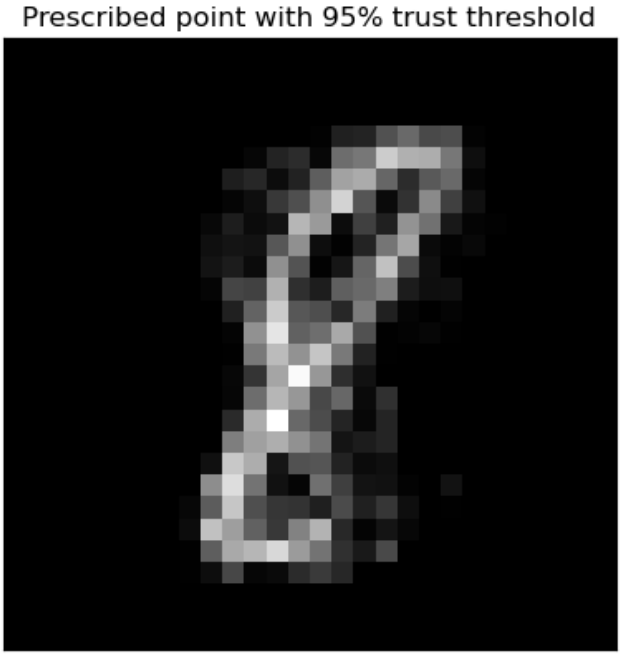

Figure 3.8: Prescribed point obtained with $trust_t = 95.0\%$.

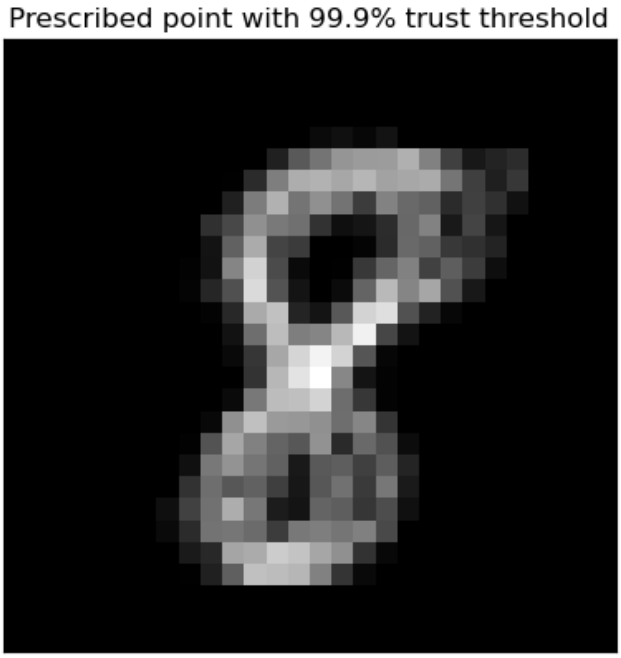

Figure 3.9: Prescribed point obtained with $trust_t = 99.9\%$.

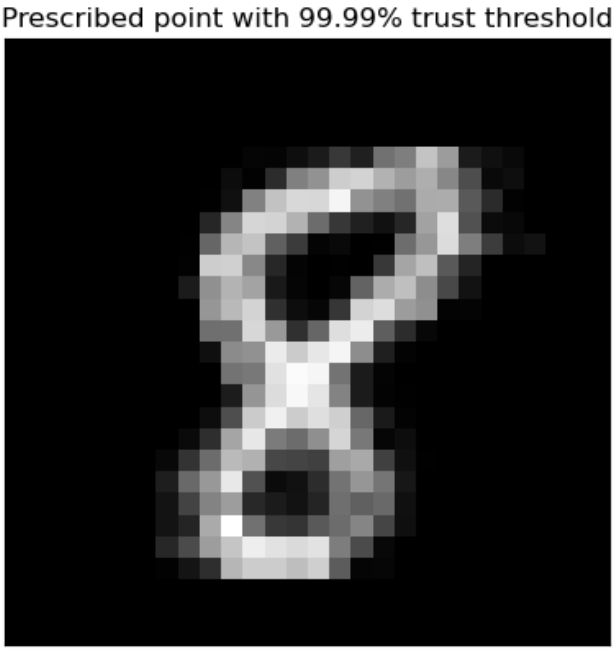

Figure 3.10: Prescribed point obtained with $trust_t = 99.99\%$.

The quality of a prescription can also be visually assessed by examining the impact of the regularization parameter α. As previously mentioned, α controls the regularization applied during the construction of the linear model. This corresponds to adding a penalty term to the objective function, which helps prevent overfitting [34].

From a visual perspective, it is evident in the following figures that higher values of α (i.e., stronger regularization) result in more conservative changes to the starting point *sp*. In other words, strong regularization encourages minimal movement in the feature space, potentially at the expense of precision in reaching the target class.

In this experiment, all parameters other than α remain unchanged from previous examples. The tested values of α are $\{0, 1, 10\}$, with the resulting prescribed points shown in order in Figures 3.11, 3.12, and 3.13.

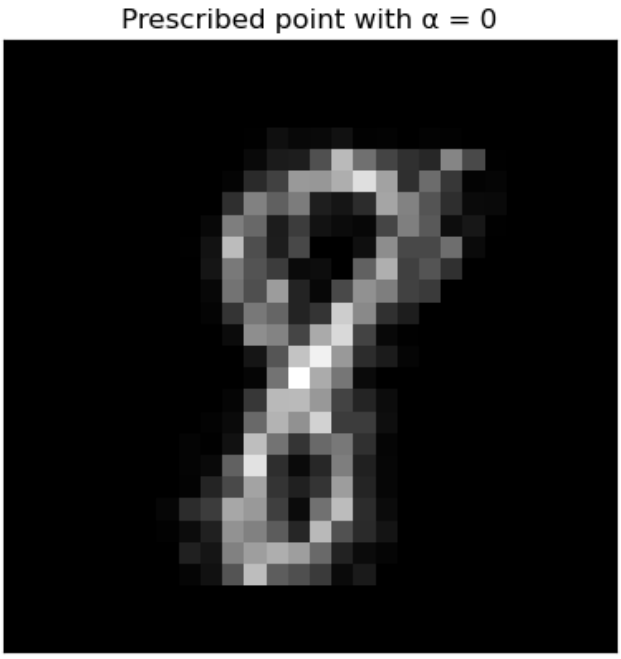

Figure 3.11: Prescribed point obtained with $\alpha = 0$.

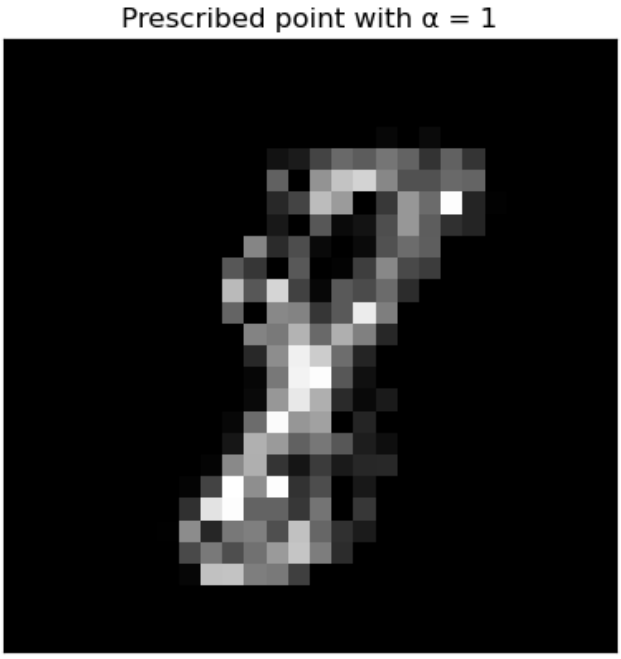

Figure 3.12: Prescribed point obtained with $\alpha = 1$.

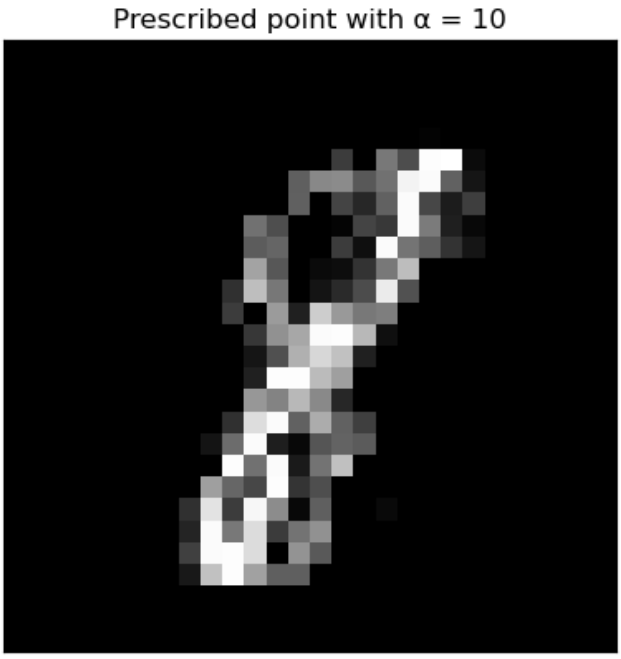

Figure 3.13: Prescribed point obtained with $\alpha = 10$.

3.3.3 Influence of cost function constraint in the prescription

The Cost Function Constraint can sensibly influence the final result of the PrescrX process. Before exploring how this constraint works in the MNIST and industrial case, it is possible to imagine some scenarios, based on Figure 3.5, where the region of space, which satisfies the Cost Function Constraint and the target class *tc* requirements, can be a location of non-dense points and it could be a region further than a region who satisfy only the target class *tc*. However, to better understand his mechanisms, it is necessary to define a cost function for both case studies. For the MNIST case, considering that this function is invented just for an exploratory purpose, without any technical aim, it follows that is possible to consider a ink function, composed as the sum of the normalized values, between 0 and 1, of the respective points where, referring to the images showed above in the Sections 3.3.1 3.3.2, values are represented with a grayscale of colors with black pixels equal to 0 and withe pixels equal to 1. The formulation of this cost function is really simple:

$$ink\ function(x) = \sum_{i=1}^{d} x_i \tag{3.4}$$

Where, d is the dimensions of the x point (784 for the MNIST case), considering the values already normalized. For the industrial case there is no need to invent a cost function, considering that is already provided. There is no interest in understanding how the following function is created; it is sufficient to know that defines the climate impact, that will be considered an adimensional value, that is the quantity of "resources" used to clean a single mechanical component:

$$\begin{aligned} climate\ impact(x) = {} & 0.0179 \cdot x_1 + 0.0179 \cdot x_2 - 0.3565 \\ & + \frac{0.8286 \cdot (0.0152 \cdot x_7 - 0.0554)}{1.0014 - 0.0004 \cdot x_7} \\ & + \frac{0.8368 \cdot (0.0342 \cdot x_6 - 0.0514)}{1.0013 - 0.0009 \cdot x_6} \end{aligned} \tag{3.5}$$

It is worth noting that in this case not all dimensions are involved in the cost function, which means that these dimensions can freely change without affecting this constraint. To understand how this constraint influences the prescribed result, starting from the same starting point *sp* several prescriptions will be executed, varying only the maximum admissible value $cost_t$, which defines the boundary of the space that satisfies the Cost Function Constraint as explained in Section 3.1.1. For the MNIST case 4 prescriptions have been executed, each one with the same parameters used in sections above; the first one has been executed without applying a Cost Function Constraint, the other three have been executed setting

the respective $cost_t$: $\{120, 100, 80\}$. These values have been decided computing the average value of *ink function* for all points who belong to the target class tc (120) and decreased respecting the scale of values available under these conditions.

From the industrial case, there is no an image to associate to the prescribed point to appreciate the quality of the prescription. Anyway, it is possible to visualize how the point moves to satisfy the Cost Function Constraint.

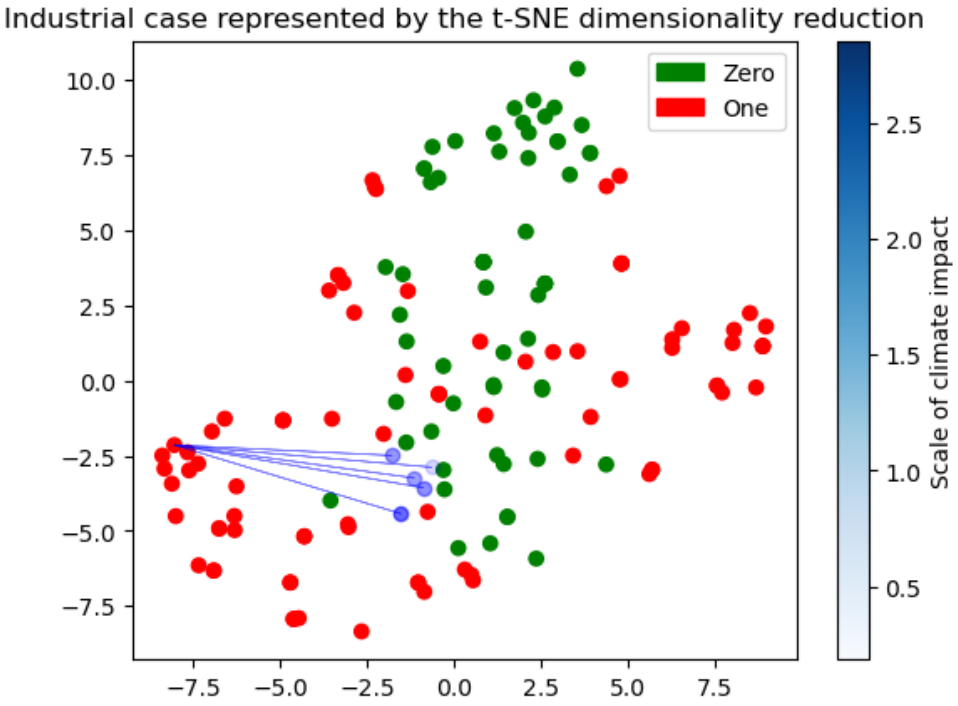

Figure 3.14: Industrial case dataset represented by the t-SNE dimensionality reduction.

In Figure 3.14 the industrial case data set is represented by the t-SNE dimensionality reduction [35]. Where green points belong to class 0 (clean), red points belong to class 1 (dirty) and blue points are the prescribed points which intensity of color represent the value of climate impact. The blue lines are a link between the starting point *sp*, in common for each prescription, and the prescribed points. From this graph it is possible to notice how the prescription need to move in different spaces to address the Cost Function Constraint. Furthermore, it is possible to appreciate the fact that all prescribed points move towards the green points; this is a base behavior of PrescrX, but from this graphical representation is possible to appreciate the result.

Chapter 4

Optimizers vs PrescrX

This chapter analyzes how PrescrX performs in comparison to other prescriptive tools. Continuing the analysis of the case studies presented earlier in Section 3.3, an optimizer is proposed for each case. The main difference between these optimizers and PrescrX lies in the objective function: while PrescrX aims to reach the target class tc with the fewest possible changes, these new optimizers instead use as objective function the probability to belong to the target class tc.

Before delving into the formulation of these optimizers, it is important to emphasize that these two optimizers are addressing a different question than PrescrX. In fact, PrescrX answers the question: "What are the minimal changes required to achieve the desired class?", whereas the new optimizers answer: "What is the point belonging to the target class with highest probability?". As a result, the outcomes are expected to differ; however, what will be particularly interesting is to observe and analyze the nature of these differences.

4.1 Optimizer for the MNIST case

Before comparing the prescriptions generated by PrescrX and the optimizer, it is essential to provide a detailed explanation of the mathematical formulation of the latter. This will allow us to understand the underlying dynamics that lead to certain types of outcomes.

Once the mathematical formulation is clearly defined, an analysis will be carried out from both a statistical perspective—comparing different prescriptions using statistical and distributional approaches—and through specific case-by-case comparisons to illustrate representative examples of the observed phenomena.

4.1.1 Mathematical formulation

The goal of the optimization process is to identify an input $\mathbf{x}^*$ that maximizes the probability assigned by a trained classification model to a desired target class $tc \in \{0, 1, \ldots, 9\}$.

Let $\mathbf{x} \in \mathbb{R}^{784}$ represent a flattened grayscale image (28×28 pixels) from the MNIST dataset, and let the classifier be a function.

$$f : \mathbb{R}^{784} \to [0,1]^{10}, \tag{4.1}$$

where $f(\mathbf{x})$ is a probability vector over the ten digit classes. The component $f_{tc}(\mathbf{x})$ denotes the predicted probability for the target class tc.

The objective is to find an input $\mathbf{x}^*$ that maximizes this probability. Formally, this is expressed as the following constrained optimization problem:

$$\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{B}} f_{tc}(\mathbf{x}), \tag{4.2}$$

where $\mathcal{B} \subset \mathbb{R}^{784}$ represents a set of box constraints applied to each component of $\mathbf{x}$.

Rather than directly maximizing the raw probability $f_{tc}(\mathbf{x})$, which may result in gradients that are too small and therefore ineffective for guiding the optimization (especially in flat regions of the probability space), the objective is reformulated as the minimization of the negative log-likelihood:

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathcal{B}} \mathcal{L}(\mathbf{x}) = -\log f_{tc}(\mathbf{x}). \tag{4.3}$$

This reformulation is common in probabilistic models, as minimizing the negative log-likelihood is equivalent to maximizing the model's confidence in the target class, while ensuring more stable and informative gradients.

The loss function to be minimized is therefore:

$$\mathcal{L}(\mathbf{x}) = -\log f_{tc}(\mathbf{x}). \tag{4.4}$$

The gradient of the loss with respect to the input $\mathbf{x}$ is computed at each iteration:

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}) = -\nabla_{\mathbf{x}} \log f_{tc}(\mathbf{x}), \tag{4.5}$$

which guides the local update of the input $\mathbf{x}$ in the optimization process.

To solve the optimization problem in Equation 4.3, the L-BFGS-B algorithm is employed. This is a quasi-Newton optimization method that:

- Is based on gradient information, ensuring efficient convergence,
- Is well-suited for high-dimensional problems,

- Supports box constraints on the variables.

The optimization iteratively updates the input $\mathbf{x}$ in the direction of the negative gradient (Equation 4.5), while ensuring that $\mathbf{x}$ remains within the constraint set $\mathcal{B}$.

Definition of box constraints

To ensure that the optimized input $\mathbf{x}^*$ remains realistic and similar to real data, each component x_i of the input is bounded by a lower and upper limit derived from the empirical distribution of training samples belonging to the target class tc. Specifically:

$$x_i \in [a_i, b_i] \quad \text{for } i = 1, \ldots, 784, \tag{4.6}$$

where

$$a_i = \min_j \left(x_j^{(i)} \mid y_j = tc \right), \quad b_i = \max_j \left(x_j^{(i)} \mid y_j = tc \right) \tag{4.7}$$

Here, $x_j^{(i)}$ denotes the i-th component (pixel) of the j-th training sample in the original dataset, and y_j is its corresponding label. These bounds ensure that the solution remains within the pixel intensity ranges observed in the training set for class tc, thus reducing the likelihood of generating implausible or out-of-distribution samples.

4.1.2 Comparison with PrescrX

Now that the structure and functioning of the optimizer have been explained, we can analyze its results and compare them with those produced by PrescrX.

To conduct this comparison, two statistical samples are considered. The first consists of 500 cases, where for each case a random point from the dataset is selected as a starting point. From each of these, two prescriptions are generated: one using the optimizer and the other using PrescrX, both targeting the same class tc.

The second sample follows the same setup, except that all starting points belong to class 1 and the target class is fixed to 8. In total, each case study comprises 1500 points: 500 starting points, 500 prescriptions generated by the optimizer, and 500 prescriptions generated by PrescrX.

Before evaluating the quality of the prescriptions using quantitative metrics, it is useful to observe their spatial distribution. To facilitate interpretation, two popular dimensionality reduction techniques—t-SNE and UMAP—are applied to project the high-dimensional data into two dimensions for visualization purposes [36].

To maintain a clear and interpretable visualization, dimensionality reduction is applied only to the second case, where all starting points belong to class 1.

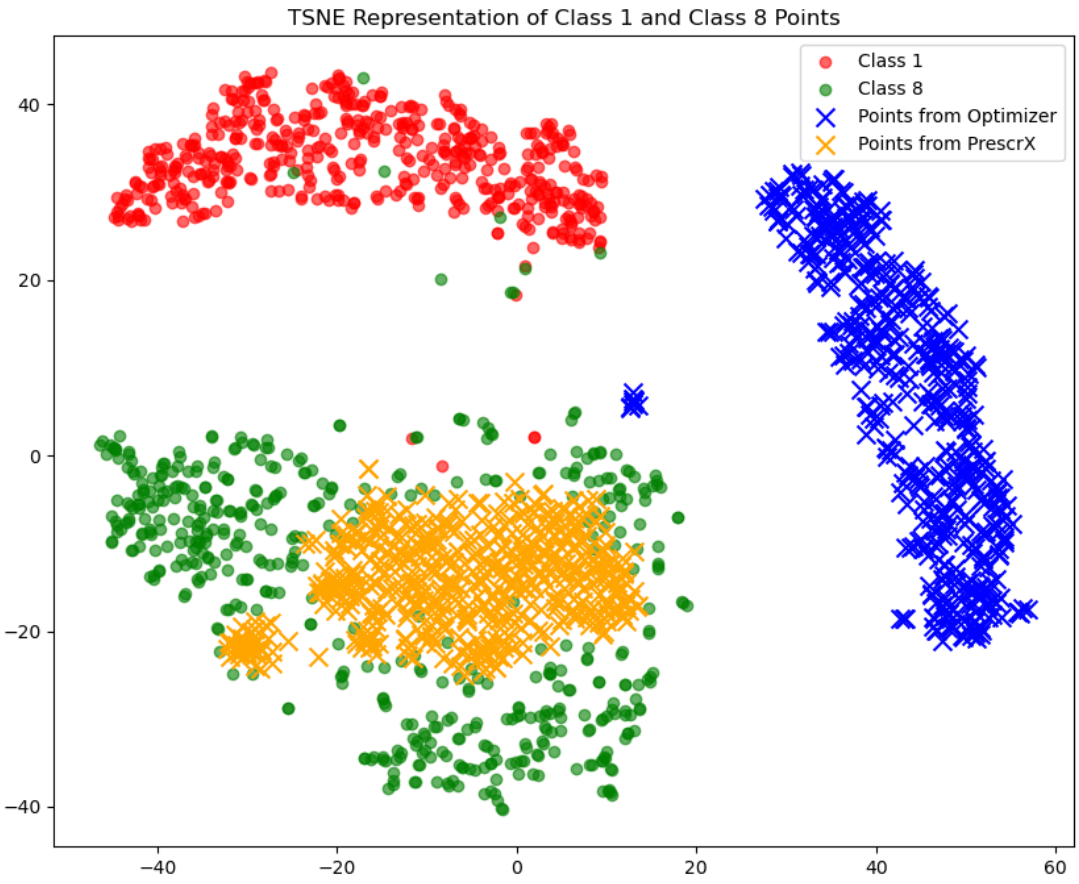

Figure 4.1: Point distribution using t-SNE dimensionality reduction for the class 1 to class 8 transformation.

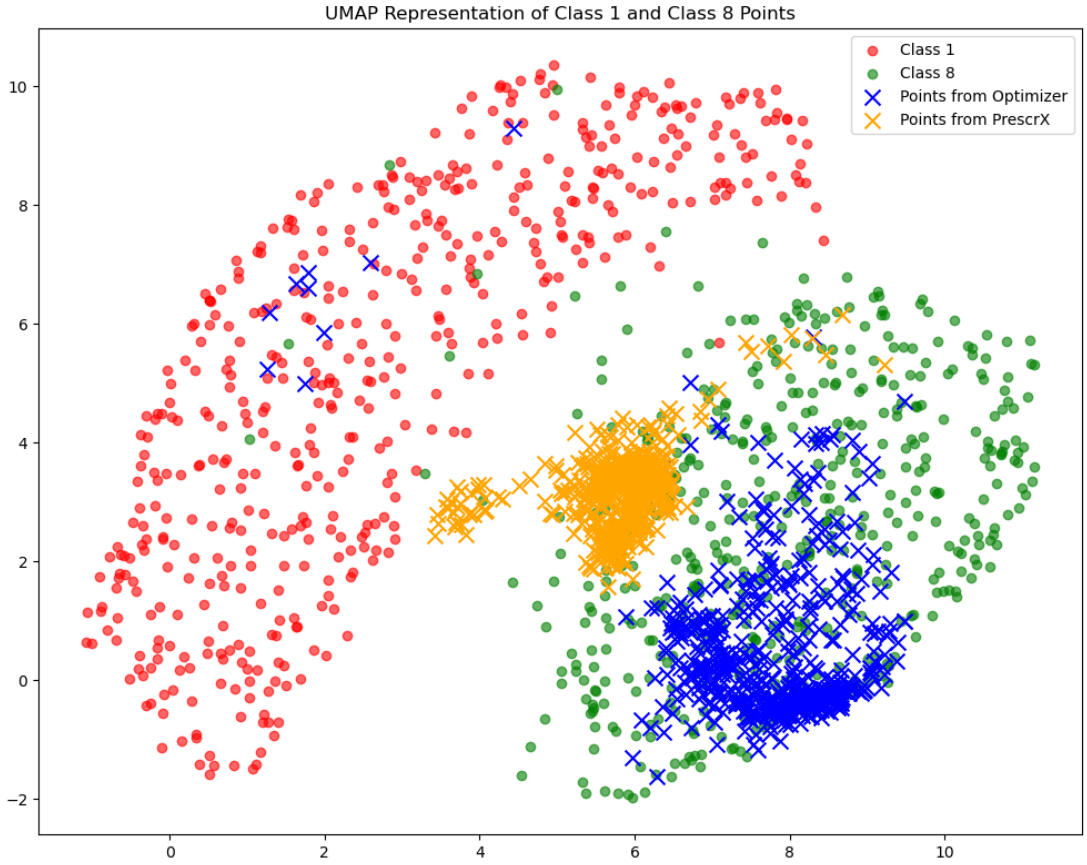

Figure 4.2: Point distribution using UMAP dimensionality reduction for the class 1 to class 8 transformation.

Figures 4.1 and 4.2 show the same dataset and use the same legend. Red points represent the 500 starting samples from class 1. All other points are classified as class 8. Green points are 500 samples drawn from the original dataset, blue crosses (`x`) represent the optimizer's prescriptions, and orange crosses (`x`) indicate prescriptions generated by PrescrX.

From the first figure (t-SNE), we can draw two key observations:

- The prescriptions generated by PrescrX (orange `x`) form a dense cluster within the cloud of original dataset samples.
  This suggests that PrescrX tends to produce prescriptions that are similar to existing data points, a behavior consistent with its strategy of navigating the data manifold.

- The prescriptions from the optimizer (blue `x`) appear as a distinct cluster, separated from the rest of the dataset.

This separation raises concerns about the plausibility of the optimizer's prescriptions, as they lie far from known data regions, potentially indicating low reliability.

The second visualization, obtained using UMAP, differs from t-SNE in its methodology: the model fitting was performed only on the original dataset (i.e., excluding any points generated by the prescriptive models), and the transformation was then applied to all data points.

Two main insights emerge from this UMAP plot:

- The PrescrX-generated points (orange `x`) are more tightly clustered, indicating that the model tends to generate prescriptions within a compact region of the data space.

- Some of the optimizer-generated points (blue `x`) diverge from the green cloud and even encroach upon the region occupied by the red points (class 1). However, it's worth noting that certain green points (i.e., real samples from the dataset) show a similar behavior, suggesting that this may reflect some natural overlap in the data space.

Adversarial samples

Given that it is possible to visually assess the quality of the generated prescriptions by inspecting the resulting images, we can further investigate the behavior observed in Figure 4.1—specifically, the separation of the optimizer's cluster from the rest of the dataset.

In Figure 4.3, we show, in order: the image corresponding to the starting point, the image produced by the optimizer, and finally the image generated by PrescrX.

As clearly visible, the image produced by the optimizer does not resemble the digit 8, despite the classifier assigning it a probability close to 1 for this class.

This scenario is a typical example of what is commonly referred to as an *adversarial sample*—a result that is highly dependent on the classifier's internal structure and does not align with human perception or empirical data [37]. In the specific case of MNIST, the pixel intensities of these optimized images are not consistent with any real samples present in the dataset.

This phenomenon consistently appears in all samples generated by the optimizer. Although these points may not offer meaningful human interpretation, they are still valuable from an analytical standpoint: they highlight edge cases and potential failure modes in the prescriptive process, enriching our understanding of the model's behavior.

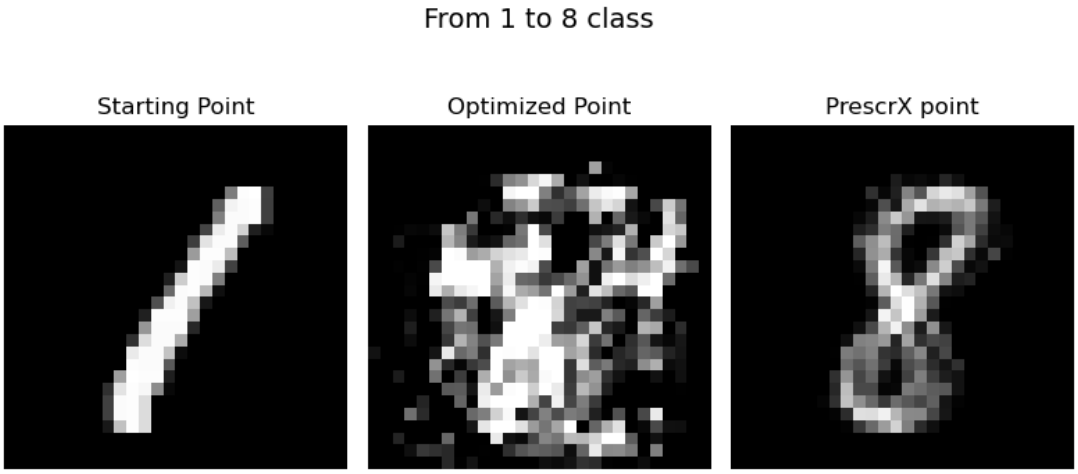

Figure 4.3: Comparison of starting point and generated prescriptions

4.1.3 Improved optimizer

As previously discussed, PrescrX generates prescriptions by relying on both the classifier and the background dataset, whereas the baseline optimizer operates solely on the classifier. This asymmetry implies that the comparison between the two approaches is not entirely fair: PrescrX benefits from additional information derived from the background data, while the optimizer does not.

To mitigate this imbalance, an improved version of the optimizer is introduced. The idea is to extend the prescriptive process of the optimizer by incorporating the background dataset, thus enriching its search space with information about the empirical data distribution. In the literature, the most common strategy to achieve this integration is to include a similarity factor within the objective function. Such a factor is designed to penalize solutions that deviate excessively from the original data manifold, thereby promoting prescriptions that remain close to real and plausible instances.

The notion of similarity can be defined in multiple ways, depending on the problem at hand. From a geometric perspective, similarity may be quantified using distance measures such as Euclidean or Mahalanobis distance, while in other cases it may involve structural or semantic criteria more closely aligned with the domain. By embedding this similarity term into the objective function, the optimizer is encouraged to balance the pursuit of high classification confidence with the need for prescriptions that are realistic and interpretable.

PrescrX explores the data space by selecting target points based on notions of proximity. Analogously, a concept of similarity is introduced into the optimizer. Before explaining how this similarity is incorporated, it is important to clarify the

type of similarity being used and its purpose.

The similarity metric adopted in this case is the *Structural Similarity Index* (SSIM) [38], a widely used measure for assessing the perceptual similarity between two images. Unlike pixel-wise differences, SSIM provides a more human-aligned evaluation by comparing three local components between two image patches: luminance (l), contrast (c), and structure (s). These components are computed as follows:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{4.8}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{4.9}$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \quad \text{with } C_3 = \frac{C_2}{2} \text{ typically} \tag{4.10}$$

The SSIM index for a given window is then computed as the product of these three terms:

$$SSIM = l \cdot c \cdot s \tag{4.11}$$

By setting $\alpha = \beta = \gamma = 1$, we obtain the commonly used simplified formula:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{4.12}$$

where μ_x, σ_x^2 represent the local mean and variance of image x, and σ_{xy} is the local covariance between x and y.

The resulting SSIM score lies within the interval $[-1,1]$ (often normalized to $[0,1]$), with 1 indicating identical images.

We incorporate SSIM into the optimizer by modifying the original loss function in Equation 4.4, yielding:

$$\mathcal{L}(\mathbf{x}) = -\left(\log f_{tc}(\mathbf{x}) + \lambda \log(SSIM(\mathbf{x}, y))\right) \tag{4.13}$$

Here, y is the image belonging to the target class tc that has the highest SSIM score with respect to the current candidate $\mathbf{x}$. Importantly, y is reselected at each iteration, since $\mathbf{x}$ is updated dynamically.

The parameter λ controls the relative weight of the similarity term. Since the primary goal remains achieving class tc, SSIM should play a secondary—yet meaningful—role in the optimization. While the choice of λ influences the final outcome, in this case study a value of 0.1 proved effective. However, λ can generally range within $(0, 1]$ (excluding 0, as that would ignore the similarity component altogether).

The introduction of this parameter aims to generate prescriptions that are not only valid in terms of classification, but also visually coherent and interpretable.

When reapplying t-SNE dimensionality reduction using this modified loss function, we observe in Figure 4.4 that the optimizer's outputs no longer form a separate, isolated cluster. Instead, they appear closer to the distribution of original dataset samples.

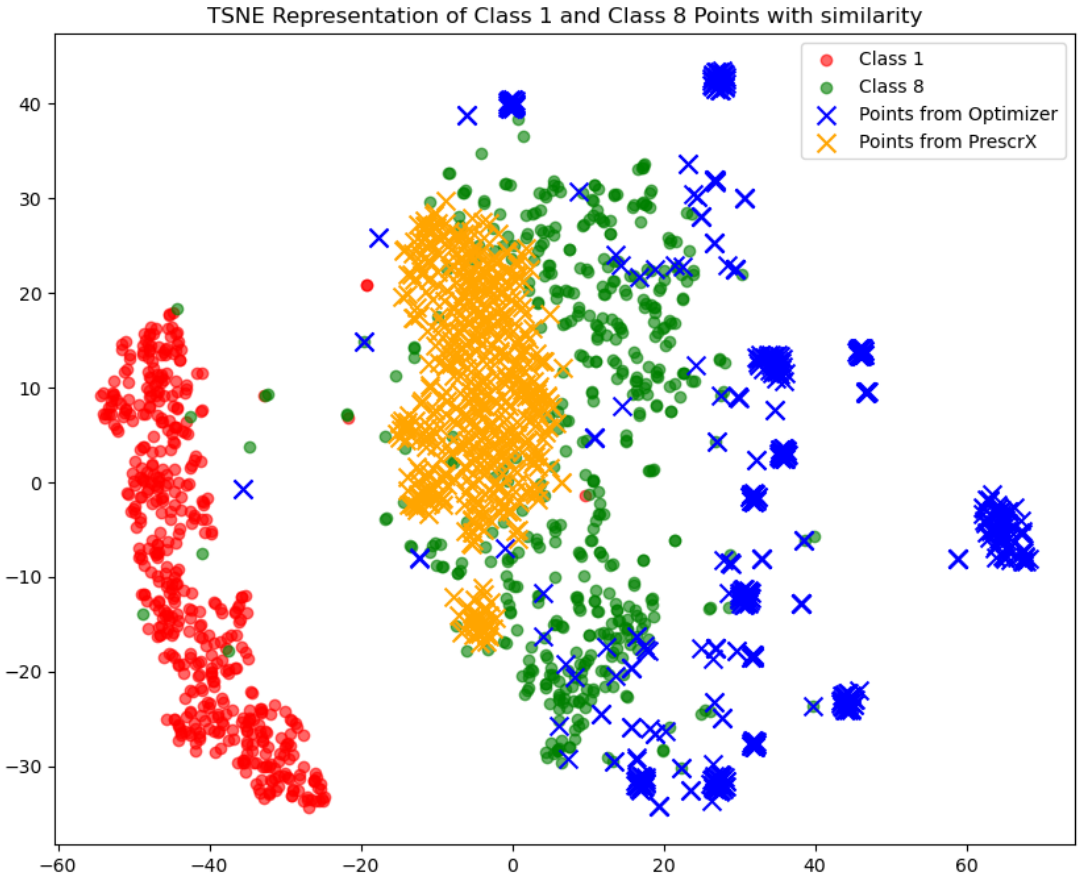

Figure 4.4: Point distribution using t-SNE for class 1 to class 8 transformation with SSIM incorporated.

Additionally, to visually confirm the effectiveness of integrating SSIM into the optimization process, Figure 4.5 compares prescriptions generated with and without the similarity constraint.

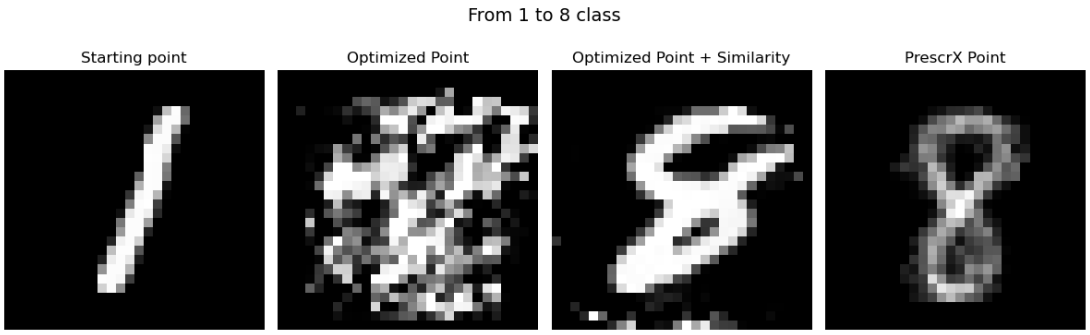

Figure 4.5: Comparison of starting point and prescriptions with and without SSIM regularization.

Both the starting point and the corresponding prescriptions from each model are shown. By examining the central images, the difference between the two optimizer variants—one with and one without the similarity term—is clearly noticeable.

4.2 Optimizer in the industrial case

As with the MNIST case, before analyzing and comparing the outputs of the optimizer and PrescrX in the industrial setting, it is essential to present a proper mathematical formulation of the optimizer. In this scenario, the technical challenges differ significantly from those in MNIST, which results in distinct optimization behaviors and outcomes.

4.2.1 Mathematical formulation and limitations

The classifier used in this case study does not provide gradient information that can be used during optimization. This is not an isolated situation—many machine learning models either do not expose gradients or are not designed to compute them at all. As such, a gradient-free optimization approach is required. Several well-known methods exist in this category, including Powell's method, Nelder-Mead simplex, and Bayesian Optimization [39].

The first method attempted was Powell's algorithm, due to its simplicity and intuitive behavior. Powell's method performs sequential one-dimensional searches along each coordinate direction, and then combines them into new directions for successive iterations until convergence [40].

Unfortunately, Powell's method—and other derivative-free methods tested—proved unsuccessful. Specifically, none of the applied techniques produced points with a predicted probability of belonging to the target class tc higher than approximately 30%.

To understand the cause of this behavior, an analysis of the underlying dataset was performed to examine how the data points are distributed in space. It was observed that the two classes are strongly separated, leaving a wide unpopulated region between the clusters.

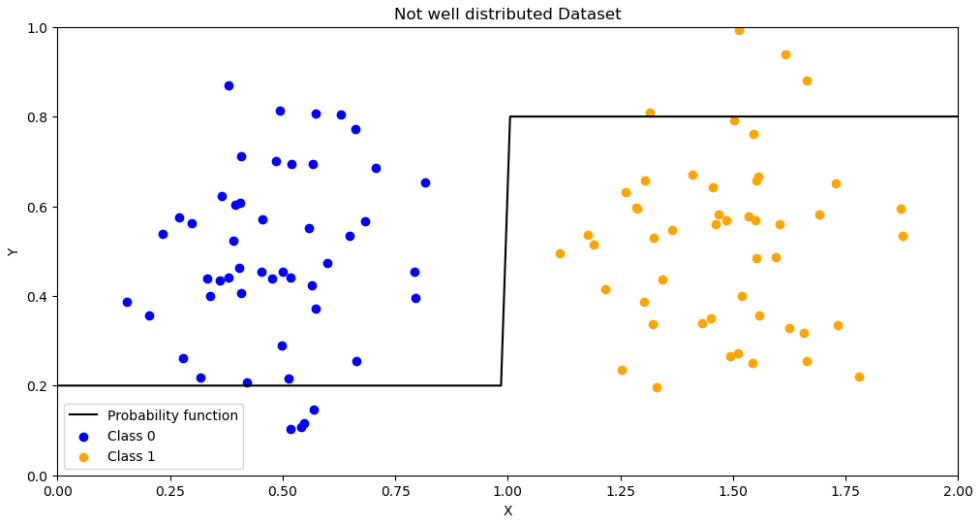

Figure 4.6: Example of poorly distributed dataset with a large gap between clusters

As illustrated in Figure 4.6, the empty space between the two clusters results in a steep jump between them, and the probability function becomes flat near the clusters. This flatness prevents the optimizer from identifying a meaningful direction to move toward higher probability regions. This issue has been studied previously by Moustapha Maliki and Sudret Bruno [41], who proposed a solution to handle such discontinuities. However, that method lies beyond the scope of this work and will not be implemented here.

Given this phenomenon, local optimization based solely on the classifier's output becomes impractical.

Nonetheless, introducing a similarity constraint—like in the MNIST case—also proves beneficial here. However, due to the limited size of the dataset, using pointwise similarity (as done with SSIM) would result in repeatedly selecting the same few data points, failing to explore the broader space.

Therefore, in this context, we propose a **statistical or distributional similarity** approach. The goal is to guide optimization towards results that are not only class-consistent but also statistically aligned with the target distribution. To achieve this, we introduce the **Mahalanobis Distance** [42].

The Mahalanobis distance between two vectors $\mathbf{x}$ and $\mathbf{y}$, with respect to a covariance matrix $\mathbf{\Sigma}$, is defined as:

$$d_M(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \mathbf{y})} \tag{4.14}$$

In particular, the Mahalanobis distance of a point $\mathbf{x}$ from a distribution with mean μ and covariance $\boldsymbol{\Sigma}$ is given by:

$$d_M(\mathbf{x}, \mu) = \sqrt{(\mathbf{x} - \mu)^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \mu)} \tag{4.15}$$

To visualize the difference between Euclidean and Mahalanobis distances, consider the 2D example in Figure 4.7. Here, both the red and blue points are equidistant from the distribution's mean under Euclidean distance. However, the red point lies within the 50th percentile (first Gaussian contour), while the blue point is beyond the 90th percentile—thus, the Mahalanobis distance correctly identifies the red point as statistically closer.

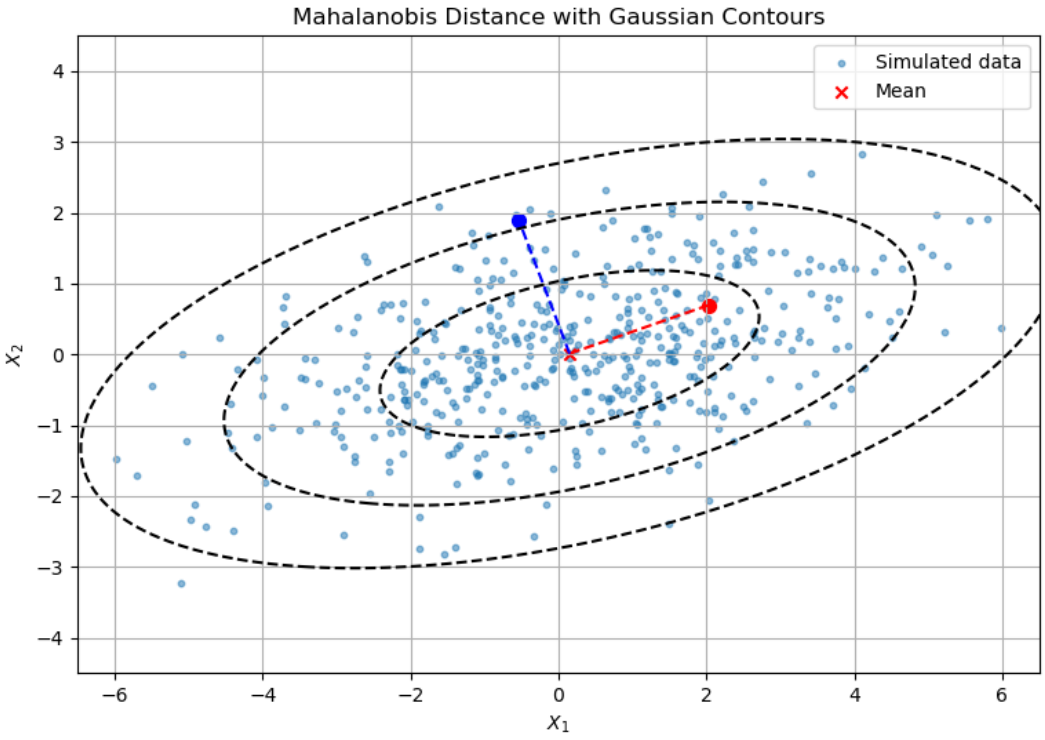

Figure 4.7: Mahalanobis vs. Euclidean distance in a 2D example

By introducing Mahalanobis distance, the objective function is reformulated as:

$$\mathcal{L}(\mathbf{x}) = -f_{tc}(\mathbf{x}) + \lambda d_M(\mathbf{x}, \mu_{\mathbf{tc}}) \tag{4.16}$$

Here, $f_{tc}(\mathbf{x})$ is the probability that point $\mathbf{x}$ belongs to the target class tc. In this industrial case, we have $\mathbf{x} \in \mathbb{R}^7$ and the classifier is $f : \mathbb{R}^7 \to [0,1]^2$. The

Mahalanobis term $d_M(\mathbf{x}, \mu_{\mathbf{tc}})$ measures how close the candidate point is to the statistical center of the target class.

This formulation mirrors the logic used in the MNIST case (Section 4.1.1): the first term encourages classification success, and the second encourages statistical similarity. However, unlike in MNIST, only the classification term is negated, since the Mahalanobis distance is naturally minimized, and no logarithmic transformation is applied—due to the smaller, less complex search space.

The parameter λ again controls the relative importance of the similarity term. In this study, we fix $\lambda = 0.1$, although it can vary in the range (0,1], with 0 excluded to avoid ignoring similarity altogether.

In the industrial case, introducing statistical similarity not only prevents the generation of adversarial examples but also makes the search space more navigable for the optimizer. This enables the optimization process to successfully converge to valid prescriptions.

To visualize this benefit, consider Figure 4.8, where we extend the earlier step-function probability landscape (Figure 4.6) by adding the similarity component $(-d_M)$. The result is a smoother, more informative search landscape, avoiding flat or abrupt regions that obstruct optimization.

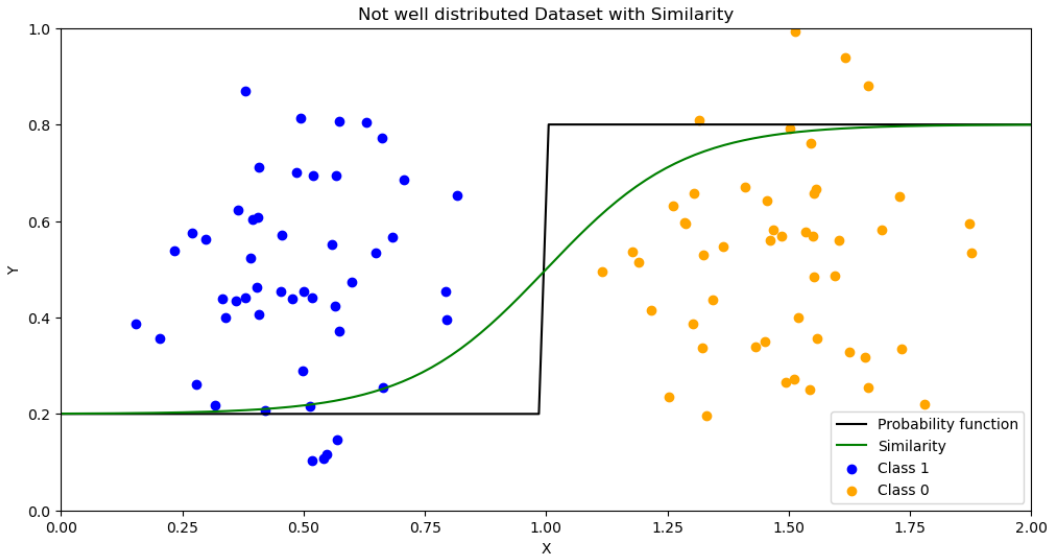

Figure 4.8: Visualization of poorly distributed data with added similarity term to enable optimization

Now that prescriptions can be generated both from PrescrX and the optimizer, we can proceed with a first comparison by analyzing how their outputs aggregate and distribute across the space.

4.2.2 Comparison with PrescrX

For each point in the original dataset, two prescriptions are generated with the goal of reaching the class to which the point does not currently belong: one using the optimizer and one using PrescrX. As previously discussed, the aim of this analysis is to explore how different prescriptions can arise from the same starting point, depending on the specific priorities of the method used.

To visualize the differences between the outcomes, a t-SNE representation is shown below.

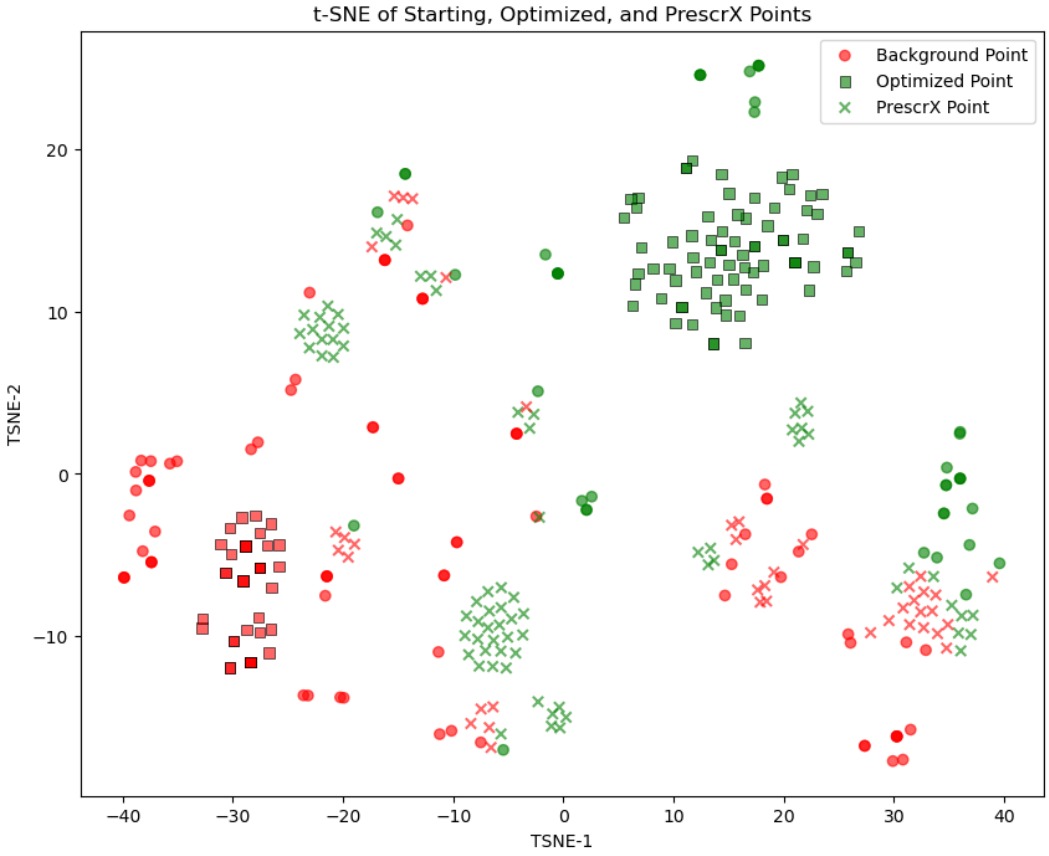

Figure 4.9: Distribution of prescriptions in space depending on the model used

In Figure 4.9, each background dataset point has undergone a prescription, including transitions from class 0 ("clean") to class 1 ("dirty") and vice versa. Colors are consistent with those used in Figure 3.14, where green indicates class 0 and red indicates class 1.

This visualization reveals that the two prescriptive models generate outcomes that are distributed very differently across the space. Specifically, the points

obtained using the optimizer (represented as squares) tend to form compact, well-defined clusters—one per target class. By contrast, the points generated by PrescrX (plotted as crosses) do not form isolated clusters but are rather spread more closely around the original background data points. This is consistent with the design of PrescrX, which searches for solutions in the proximity of real data.

Given the simplicity of the dataset in this industrial case, it is also feasible to inspect individual prescriptions from specific starting points. Figure 4.10 presents several representative cases by randomly selecting background points. The index for each example is shown in the plot.

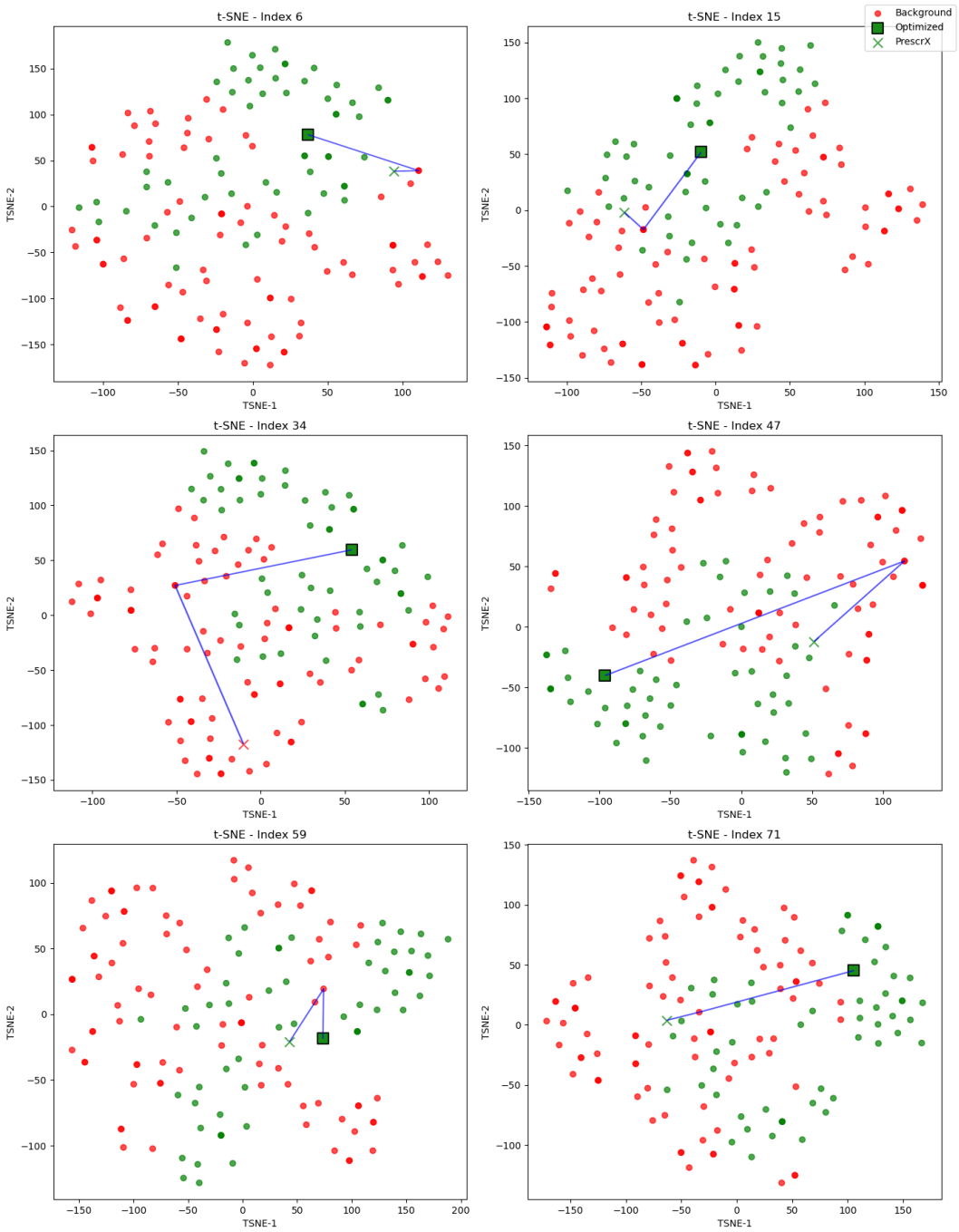

Figure 4.10: Example cases showing the prescriptions from both methods

From the plots in Figure 4.10, it is evident that prescriptions generated by the optimizer tend to occupy regions densely populated by points of the target class *tc*

in the background dataset. Additionally, in the plot with index 34, we observe that the point generated by PrescrX still belongs to class 1. This highlights a known limitation of PrescrX: it does not always succeed in completing a prescription that transitions the point to the target class. Nonetheless, for this industrial case study, only 6 points in the entire dataset failed to produce a successful prescription using PrescrX.

Now that a general understanding of the prescriptive behavior of both tools has been established across the two case studies, the next chapter will focus on evaluating the generated prescriptions analytically through the introduction of evaluation metrics.

This final analysis serves two main purposes:

- To identify the strengths and weaknesses of each prescriptive model, and
- To assess the reliability of individual prescriptions.

This is particularly relevant because, in many prescriptive use cases, it is not necessary to generate prescriptions for the entire dataset. Instead, the typical requirement is to generate a valid prescription for a single, specific point under analysis.

Chapter 5

Metrics

In order to quantitatively assess the quality of a prescription, it is necessary to introduce metrics or parameters capable of analyzing one or more aspects of the obtained result. As previously mentioned, the primary focus of this study is to define metrics that evaluate a *single prescription*. Nevertheless, these metrics will also be reported at a *statistical and distributional level*, in order to observe and consolidate the different behaviors of PrescrX and the optimizers.

The following section introduces a set of metrics, together with their mathematical formulation and operational details.

5.1 Mathematical formulation of metrics

In this section, we present the metrics designed to evaluate the quality of a prescription. It is important to note that, in this context of analysis, it is essential to assess whether the obtained prescription is coherent and reliable with respect to the analyzed setting.

5.1.1 Normalized distance by mean

This metric has been designed to measure how far the starting point must move in order to obtain the prescription. In many contexts, it is important not to deviate excessively from the initial point. As already mentioned, in a clinical setting—where a point in the space represents the dosages of the different drugs taken by a patient—a large deviation, and therefore a large variation in dosages, could prove either ineffective or even harmful.

Below we present the mathematical formulation, including all relevant steps.

Let:

- $\mathcal{D} = \{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$ be a dataset of n points in $\mathbb{R}^d$,

- $\mathbf{y} = \{y_i \in \mathcal{C}\}_{i=1}^{n}$ be the associated class labels,
- $\mathbf{x}_s \in \mathbb{R}^d$ be a starting point,
- $\mathbf{x}_p \in \mathbb{R}^d$ be a prescribed point (target point from a different class),
- $f : \mathbb{R}^d \to \mathcal{C}$ be a classifier used to predict class labels.

If class labels are not provided, they are inferred using the classifier:

$$y_i = f(\mathbf{x}_i), \quad \forall i \in \{1, \ldots, n\} \tag{5.1}$$

The class of the starting point $\mathbf{x}_s$ is determined as:

$$c_s = \begin{cases} y_i & \text{if } \exists i \text{ such that } \mathbf{x}_i = \mathbf{x}_s, \\ f(\mathbf{x}_s) & \text{otherwise.} \end{cases} \tag{5.2}$$

The class of the prescribed point $\mathbf{x}_p$ is:

$$c_p = \begin{cases} y_j & \text{if } \exists j \text{ such that } \mathbf{x}_j = \mathbf{x}_p, \\ f(\mathbf{x}_p) & \text{otherwise.} \end{cases} \tag{5.3}$$

It is required that the two points belong to different classes:

$$c_s \neq c_p \tag{5.4}$$

We define the subsets of the dataset belonging to the respective classes as:

$$\mathcal{D}_{c_s} = \{\mathbf{x}_i \in \mathcal{D} \mid y_i = c_s\}, \tag{5.5}$$

$$\mathcal{D}_{c_p} = \{\mathbf{x}_j \in \mathcal{D} \mid y_j = c_p \text{ and } \mathbf{x}_j \neq \mathbf{x}_p\}. \tag{5.6}$$

The direct Euclidean distance between $\mathbf{x}_s$ and $\mathbf{x}_p$ is computed as:

$$d_{\text{direct}} = \|\mathbf{x}_s - \mathbf{x}_p\|_2 \tag{5.7}$$

The mean inter-class distance between $\mathcal{D}_{c_s}$ and $\mathcal{D}_{c_p}$ is:

$$\bar{d}_{\text{mean}} = \frac{1}{|\mathcal{D}_{c_s}| \cdot |\mathcal{D}_{c_p}|} \sum_{\mathbf{x}_i \in \mathcal{D}_{c_s}} \sum_{\mathbf{x}_j \in \mathcal{D}_{c_p}} \|\mathbf{x}_i - \mathbf{x}_j\|_2 \tag{5.8}$$

Finally, the normalized distance is defined as:

$$\text{NormalizedDistance} = \begin{cases} 1.0 & \text{if } \bar{d}_{\text{mean}} = 0, \\ \dfrac{d_{\text{direct}}}{\bar{d}_{\text{mean}}} & \text{otherwise.} \end{cases} \tag{5.9}$$

The value of NormalizedDistance is a positive real number in the interval $[0, \infty)$. What matters, however, is whether it is greater or smaller than 1. If NormalizedDistance is smaller than 1, this means that the traveled distance is less than the average distance between the two classes of interest. Conversely, if NormalizedDistance is greater than 1, the distance between the starting point and the prescribed point exceeds the average inter-class distance.

5.1.2 Gain of Neighbors

To obtain a first estimate of whether the prescribed point effectively moves toward a region of the space populated by the background dataset, we introduce the *Gain of Neighbors* (GoN) metric. As will be shown in the mathematical formulation, the underlying idea is to verify whether the prescribed point has a greater number of neighbors belonging to the target class tc compared to the starting point. In addition, this value provides an initial indication of whether the prescribed point could potentially represent an adversarial example. In fact, if the prescribed point has fewer neighbors belonging to the target class than the starting point, there is a risk that it lies in an anomalous region of the space.

Let:

- $\mathcal{D} = \{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$ be a dataset of n points in $\mathbb{R}^d$,
- $\mathbf{y} = \{y_i \in \mathcal{C}\}_{i=1}^n$ be the corresponding class labels,
- $\mathbf{x}_s \in \mathbb{R}^d$ be the starting point,
- $\mathbf{x}_p \in \mathbb{R}^d$ be the prescribed point,
- $f : \mathbb{R}^d \to \mathcal{C}$ be a classifier used to predict class labels.

If class labels are not given, they are predicted by the classifier:

$$y_i = f(\mathbf{x}_i), \quad \forall i \in \{1, \dots, n\} \tag{5.10}$$

The classes of the starting and prescribed points are assigned as follows:

$$c_s = \begin{cases} y_i & \text{if } \exists i \text{ such that } \mathbf{x}_i = \mathbf{x}_s, \\ f(\mathbf{x}_s) & \text{otherwise.} \end{cases} \tag{5.11}$$

$$c_p = \begin{cases} y_j & \text{if } \exists j \text{ such that } \mathbf{x}_j = \mathbf{x}_p, \\ f(\mathbf{x}_p) & \text{otherwise.} \end{cases} \tag{5.12}$$

We require that the two points belong to different classes:

$$c_s \neq c_p \tag{5.13}$$

Define the Euclidean distance between the two points as:

$$d = \|\mathbf{x}_s - \mathbf{x}_p\|_2 \tag{5.14}$$

Let the neighborhood radius be half this distance:

$$r = \frac{d}{2} \tag{5.15}$$

Define the neighborhoods of $\mathbf{x}_s$ and $\mathbf{x}_p$ as:

$$\mathcal{N}_s = \{\mathbf{x}_i \in \mathcal{D} \setminus \{\mathbf{x}_s\} \mid \|\mathbf{x}_i - \mathbf{x}_s\|_2 \leq r\}, \tag{5.16}$$

$$\mathcal{N}_p = \left\{\mathbf{x}_i \in \mathcal{D} \setminus \{\mathbf{x}_p\} \mid \|\mathbf{x}_i - \mathbf{x}_p\|_2 \leq r\right\}. \tag{5.17}$$

Let the percentage of neighbors around $\mathbf{x}_s$ and $\mathbf{x}_p$ that belong to class c_p be defined as:

$$P_s = \begin{cases} 0 & \text{if } |\mathcal{N}_s| = 0, \\ \dfrac{1}{|\mathcal{N}_s|} \displaystyle\sum_{\mathbf{x}_i \in \mathcal{N}_s} \mathbf{1}_{\{y_i = c_p\}} & \text{otherwise.} \end{cases} \tag{5.18}$$

$$P_p = \begin{cases} 0 & \text{if } |\mathcal{N}_p| = 0, \\ \dfrac{1}{|\mathcal{N}_p|} \displaystyle\sum_{\mathbf{x}_i \in \mathcal{N}_p} \mathbf{1}_{\{y_i = c_p\}} & \text{otherwise.} \end{cases} \tag{5.19}$$

The final output is the difference in percentages:

$$GoN = P_p - P_s \tag{5.20}$$

The value of GoN lies in the interval $[-1, 1]$. Values smaller than 0 indicate a loss of neighbors, suggesting low prescription quality, whereas values greater than 0 indicate good quality. It is important to note that this metric depends strongly on the number of neighbors of the starting point. For example, if the starting point is located in an anomalous region of the space—such as a low-density area with labels assigned with low reliability—the GoN value may potentially lose significance.

5.1.3 Features preservation score

Depending on the analyzed context, it may be of interest to generate prescriptions that alter as few features as possible. For example, in an industrial scenario where features represent the operating parameters of a machine, modifying only one or two of them could entail a lower cost compared to changing all of them.

For this purpose, we introduce the *Features Preservation Score* (FPS), which measures how many features remain unchanged when moving from the starting point to the prescribed point. Naturally, this type of metric is meaningful only in contexts where the problem requires such an evaluation. The mathematical formulation is provided below.

Let:

- $\mathbf{x}_s \in \mathbb{R}^d$ be the starting point (original input),
- $\mathbf{x}_p \in \mathbb{R}^d$ be the prescribed (modified or target) point.

Assume $\mathbf{x}_s$ and $\mathbf{x}_p$ have the same dimensionality, i.e.,

$$\dim(\mathbf{x}_s) = \dim(\mathbf{x}_p) = d \tag{5.21}$$

Let m be the number of features that differ between the two points:

$$m = \sum_{j=1}^{d} \mathbf{1}_{\{x_{s,j} \neq x_{p,j}\}} \tag{5.22}$$

Then the feature preservation score is defined as:

$$\text{FeaturePreservationScore} = 1 - \frac{m}{d} \tag{5.23}$$

This score lies in the interval [0,1], where a value of 1 indicates that all features are preserved (i.e., $\mathbf{x}_s = \mathbf{x}_p$), whereas a value of 0 means that all features have been modified. Clearly, if the value is exactly 1, the prescription loses significance, since no changes have been applied with respect to the starting point.

5.1.4 Robustness

Finally, we introduce a metric to measure the robustness of the prescribed point. In this case, the calculation of the metric is strongly tied to its definition: to measure robustness, the prescribed point is perturbed, and we verify whether it continues to belong to the target class despite the perturbations.

The rationale behind measuring robustness lies in assessing the reliability of a prescription. A point that does not change class under perturbations is considered robust, and therefore more reliable, compared to a point that may change class depending on the perturbation.

Several variants of this metric can be proposed. For example, instead of simply observing whether the point changes class after perturbation, one could monitor the prediction confidence. Indeed, if a point starts with 95% confidence and drops to 90% after perturbation, it is more robust than a point that, starting again from 95%, drops to 60%. Another possible variant consists in defining an ad hoc perturbation method tailored to the problem under analysis. In the following mathematical formulation, perturbations are introduced through Gaussian noise, although alternative approaches may certainly be considered depending on the application domain.

Let:

- $\mathcal{D} = \{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$ be the dataset used to estimate per-feature statistics,
- $\mathbf{x}_p \in \mathbb{R}^d$ be the prescribed point on which robustness is evaluated,

- $f : \mathbb{R}^d \to \mathcal{C}$ be a classifier used for prediction,
- $\alpha \in \mathbb{R}_+$ be a scaling factor for feature-wise noise,
- $T \in \mathbb{N}$ be the number of perturbation trials.

Let the standard deviation of each feature over the dataset be defined as:

$$\sigma_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{i,j} - \bar{x}_j)^2}, \quad \text{for } j = 1, \ldots, d \tag{5.24}$$

where $\bar{x}_j$ is the mean of the j-th feature in $\mathcal{D}$. Define the perturbation magnitude per feature as:

$$\varepsilon_j = \alpha \cdot \sigma_j \tag{5.25}$$

Let the original predicted class of $\mathbf{x}_p$ be:

$$c_p = f(\mathbf{x}_p) \tag{5.26}$$

For each trial $t = 1, \ldots, T$, generate noise from a normal distribution:

$$\boldsymbol{\eta}^{(t)} \sim \mathcal{N}(0, \mathrm{diag}(\varepsilon_1^2, \ldots, \varepsilon_d^2)) \tag{5.27}$$

and define the perturbed point as:

$$\tilde{\mathbf{x}}_p^{(t)} = \mathbf{x}_p + \boldsymbol{\eta}^{(t)} \tag{5.28}$$

Let $N_{\text{consistent}}$ be the number of trials where the predicted class remains unchanged:

$$N_{\text{consistent}} = \sum_{t=1}^{T} \mathbf{1}_{\{f(\tilde{\mathbf{x}}_p^{(t)}) = c_p\}} \tag{5.29}$$

The robustness score is then computed as:

$$\text{RobustnessScore} = \frac{N_{\text{consistent}}}{T} \tag{5.30}$$

The robustness score takes values in the interval [0,1], where 0 indicates a point that changes class under every perturbation, and 1 indicates a point that consistently remains in the original class despite all perturbations.

5.2 Analysis with metrics

As in the previous chapters, we now extend the analysis of the prescriptive models from a quantitative perspective, in order to further expand or consolidate the understanding of their behavior. Consistently with the two case studies considered so far, the following analysis is carried out by examining one metric at a time.

Since in both case studies it is not required to minimize the number of modified features, the analysis based on the *Features Preservation Score* (FPS) will not be included.

5.2.1 Analysis through normalized distance

As previously discussed, this metric is designed to evaluate whether the distance between the starting point *sp* and the prescribed point is consistent with the distribution of the background dataset. In line with the expectations outlined earlier, PrescrX is anticipated to yield lower values for this metric compared to the optimizers.

Starting with the MNIST case, a quantitative assessment can be obtained both from a distributional and statistical perspective. From the histograms in Figures 5.1 and 5.2, as well as the summary statistics in Tables 5.1 and 5.2, it is evident—in both the random sample and the 1-to-8 case (Section 4.1.2)—that the distances generated by PrescrX are consistently smaller than those obtained with the optimizers, regardless of whether similarity is incorporated or not.

Moreover, both the mean and the median of PrescrX distances are lower than 1, meaning that the distance between starting and prescribed points is smaller than the average inter-class distance observed in the background dataset. This result reinforces the key behavioral trait of PrescrX. By contrast, the prescriptions generated by the optimizers yield mean and median distances greater than 1, indicating that the prescribed points lie further apart than the average inter-class distance. This phenomenon is particularly evident in the 1-to-8 case, where the histograms are more clearly separated. Although not explicitly shown, this behavior consistently appears across all cases—whether considering all points from the same starting class or targeting the same class *tc*.

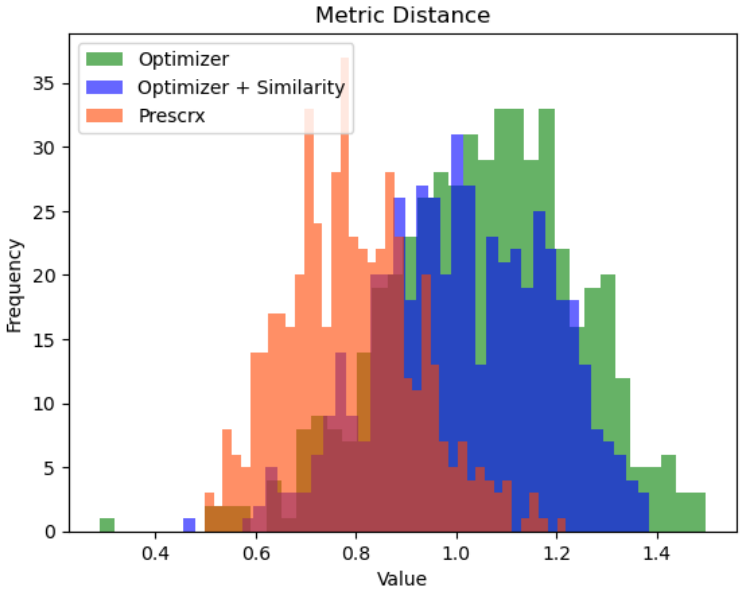

Figure 5.1: Distribution of Normalized Distance values for each prescriptive model in the MNIST case study, considering the random sample.

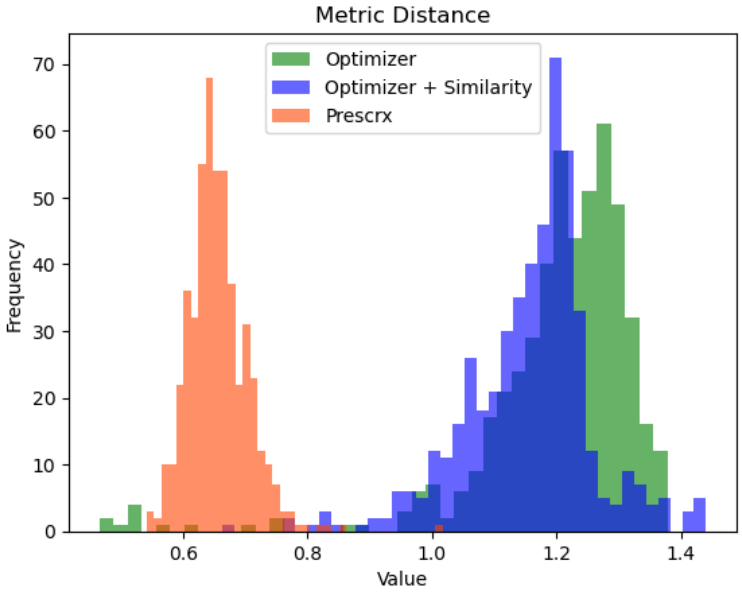

Figure 5.2: Distribution of Normalized Distance values for each prescriptive model in the MNIST case study, considering the sample where all starting points are 1 and the target class is 8.

Distance Metrics Comparison, MNIST, random

Statistic	Optimizer	Optimizer SSIM	PrescrX
Mean	1.053	1.016	0.791
Std Dev	0.187	0.169	0.132
Min	0.288	0.456	0.499
25%	0.929	0.893	0.698
Median	1.059	1.013	0.783
75%	1.179	1.151	0.876
Max	1.498	1.385	1.218

Table 5.1: Summary statistics for distance metrics in the random MNIST sample: optimizer, optimizer with SSIM, and PrescrX.

Distance Metrics Comparison, MNIST, from 1 to 8

Statistic	Optimizer	Optimizer SSIM	PrescrX
Mean	1.203	1.156	0.656
Std Dev	0.132	0.104	0.048
Min	0.465	0.663	0.541
25%	1.161	1.100	0.626
Median	1.227	1.173	0.650
75%	1.282	1.214	0.681
Max	1.379	1.441	1.017

Table 5.2: Summary statistics for distance metrics in the MNIST 1-to-8 sample: optimizer, optimizer with SSIM, and PrescrX.

Turning to the industrial case (Figures 5.3 and 5.3), the same trends observed for MNIST are further confirmed. For this analysis, only cases where prescriptions transition from the "dirty" class to the "clean" class (from 1 to 0) are considered. It is worth recalling that in the industrial setting only one optimizer is visualized, since the inclusion of similarity is necessary for prescriptions to be generated at all.

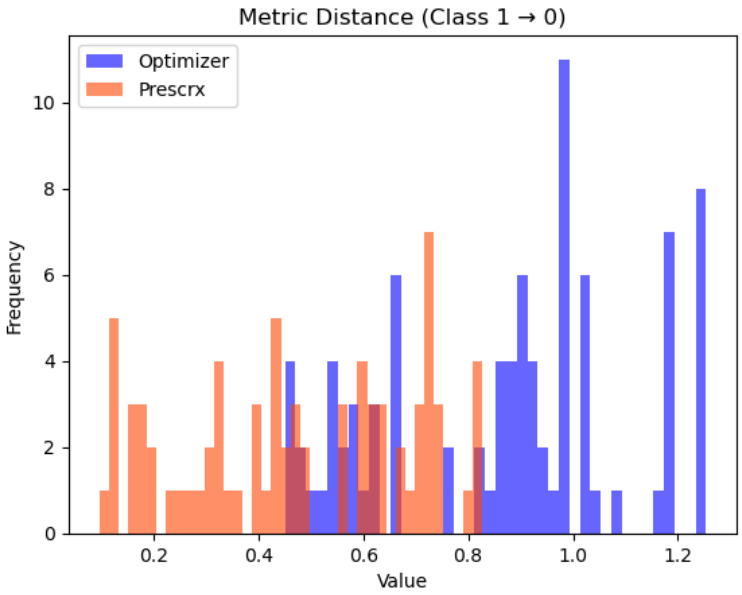

Figure 5.3: Distribution of Normalized Distance values for each prescriptive model in the industrial case study.

Distance Metrics Comparison, industrial

Statistic	Optimizer	PrescrX
Mean	0.829	0.652
Std Dev	0.272	0.689
Min	0.325	0.037
25%	0.632	0.279
Median	0.857	0.428
75%	0.960	0.707
Max	2.478	2.786

Table 5.3: Summary statistics for distance metrics in the industrial case: optimizer and PrescrX.

5.2.2 Analysis through GoN

The purpose of the GoN metric is to verify whether the prescribed point lies in a region with a higher density of points belonging to the target class *tc* compared to the starting point. If GoN has a value greater than 0, this condition is satisfied.

Starting with the MNIST case study, the three prescriptive models exhibit very different behaviors. As shown in Figures 5.4 and 5.5, which respectively

correspond to the random sample and the 1-to-8 case, the optimizer without similarity consistently reports a GoN score less than or equal to 0. This result is consistent with what was observed in the previous chapter: since this model generates adversarial samples, the prescribed points do not fall in the vicinity of the background dataset. For future studies, this metric could thus be employed as an initial check for the possible existence of adversarial samples whenever non-positive values are obtained.

The optimizer with similarity, on the other hand, displays a completely different behavior. The GoN score for points generated by this model is not only almost always strictly positive, but also very close to 1, indicating that the prescriptions are located in highly populated regions of the background dataset corresponding to the target class tc.

PrescrX also tends to provide prescriptions with non-negative GoN scores, although the values are generally lower than those obtained by the optimizer with similarity. This outcome can be explained by the nature of PrescrX, which prioritizes staying as close as possible to the starting point, rather than seeking regions of high density in the target class.

A numerical confirmation of these observations can be found in Tables 5.4 and 5.5.

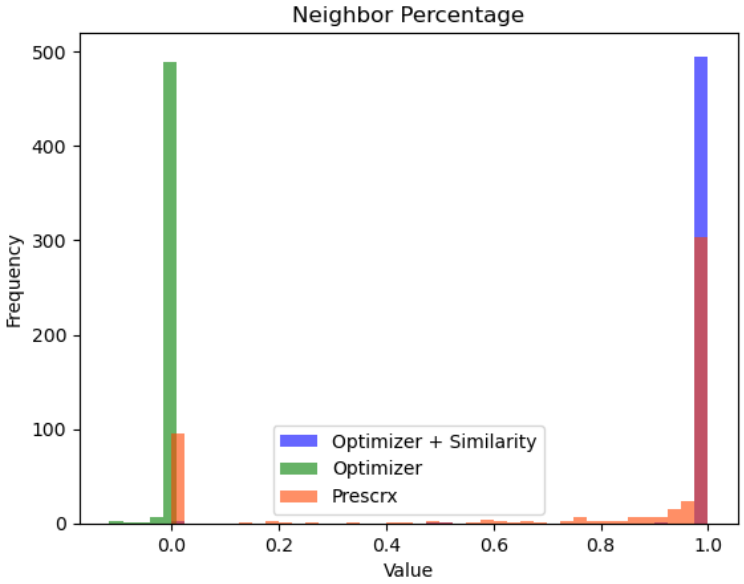

Figure 5.4: Distribution of GoN scores for each prescriptive model in the MNIST case study, considering the random sample.

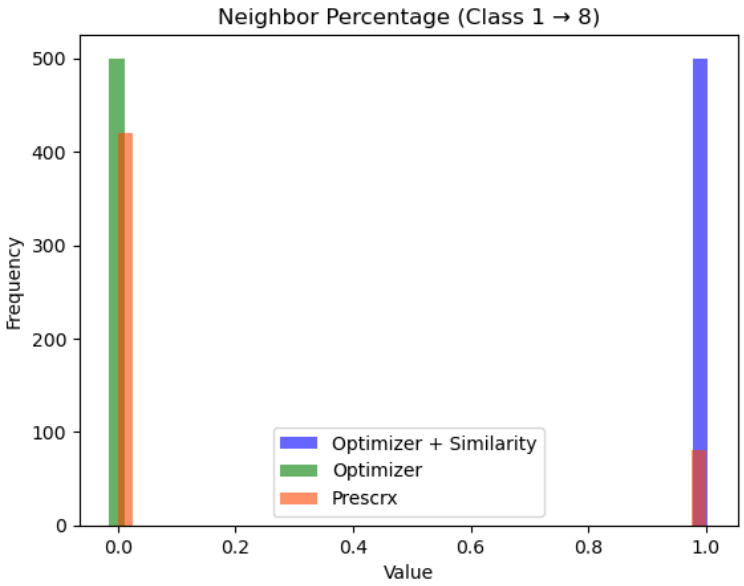

Figure 5.5: Distribution of GoN scores for each prescriptive model in the MNIST case study, considering the sample where all starting points are 1 and the target class tc is 8.

GoN Comparison, MNIST, random

Statistic	Optimizer	Optimizer SSIM	PrescrX
Mean	-0.001	0.993	0.767
Std Dev	0.008	0.071	0.393
Min	-0.116	0.000	0.000
25%	0.000	1.000	0.752
Median	0.000	1.000	1.000
75%	0.000	1.000	1.000
Max	0.000	1.000	1.000

Table 5.4: Summary statistics of GoN scores for the random MNIST sample: optimizer, optimizer with SSIM, and PrescrX.

GoN Comparison, MNIST, from 1 to 8

Statistic	Optimizer	Optimizer SSIM	PrescrX
Mean	-0.003	0.998	0.160
Std Dev	0.003	0.003	0.367
Min	-0.014	0.978	0.000
25%	-0.005	0.997	0.000
Median	-0.003	0.998	0.000
75%	-0.001	1.000	0.000
Max	0.000	1.000	1.000

Table 5.5: Summary statistics of GoN scores for the MNIST 1-to-8 sample: optimizer, optimizer with SSIM, and PrescrX.

Turning to the industrial case, the behavior of PrescrX is consistent with that observed in MNIST, apart from a few negative GoN values. These are attributable to the relatively small background dataset, which does not fully populate the feature space under analysis. Similarly, the optimizer also records some negative GoN scores for the same reason. A visual and quantitative confirmation of these findings is provided in Figure 5.6 and Table 5.6.

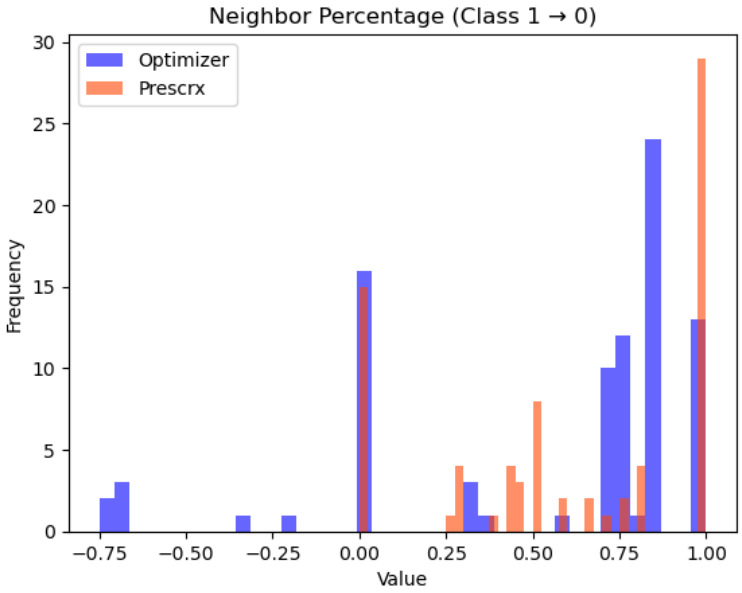

Figure 5.6: Distribution of GoN scores for each prescriptive model in the industrial case study.

GoN Comparison, industrial

Statistic	Optimizer	PrescrX
Mean	0.612	0.398
Std Dev	0.398	0.512
Min	-0.750	-0.624
25%	0.586	0.000
Median	0.704	0.500
75%	0.854	1.000
Max	1.000	1.000

Table 5.6: Summary statistics of GoN scores for the industrial case: optimizer and PrescrX.

5.2.3 Analysis through robustness

For evaluations based on this metric, several variants can be considered depending on the problem at hand.

In the MNIST case study, instead of measuring the number of times the point under analysis changes class after perturbation, we assess how much the prediction confidence of the perturbed point decreases on average. The reason for adopting this variant lies in the fact that all prescribed points exhibit very high confidence values (above 95%). Perturbing them while maintaining a humanly interpretable image (the digit of the corresponding class) essentially results in no class changes. This is a favorable sign for the robustness of all prescribed points, regardless of the model. Nevertheless, a deeper level of analysis is required.

In the following plots (Figures 5.7 and 5.8), what is reported is not the number of times the class remains the same as the target class, but rather the extent to which prediction confidence decreases after perturbation. Note that the x-axis, which indicates the percentage drop in confidence after perturbation, is displayed on a logarithmic scale. From the graphs, it is evident that PrescrX tends to produce prescriptions with greater instability, although this still amounts to only a few percentage points, which—as noted earlier—are insufficient to induce a change of class.

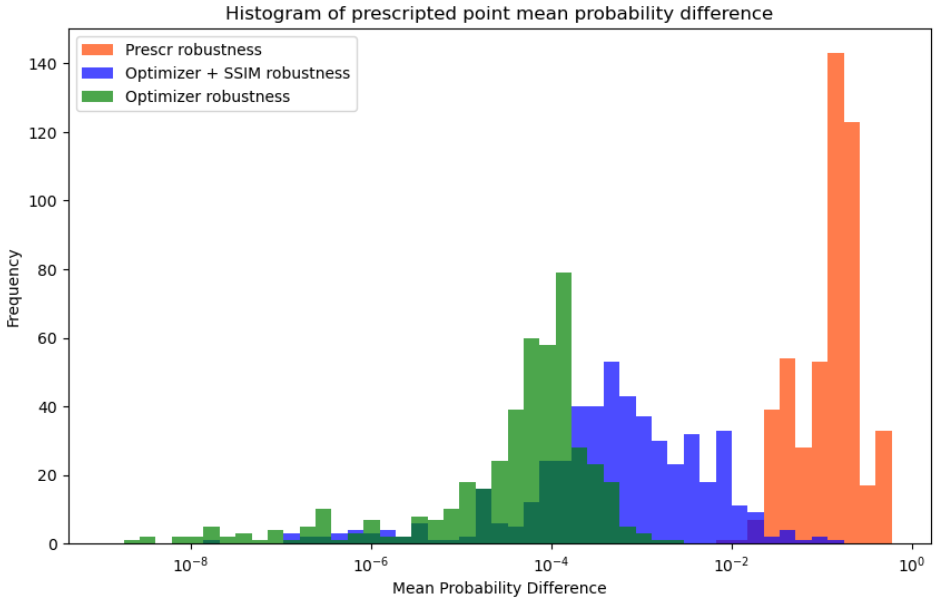

Figure 5.7: Distribution of robustness values for each prescriptive model in the MNIST case study, considering the random sample.

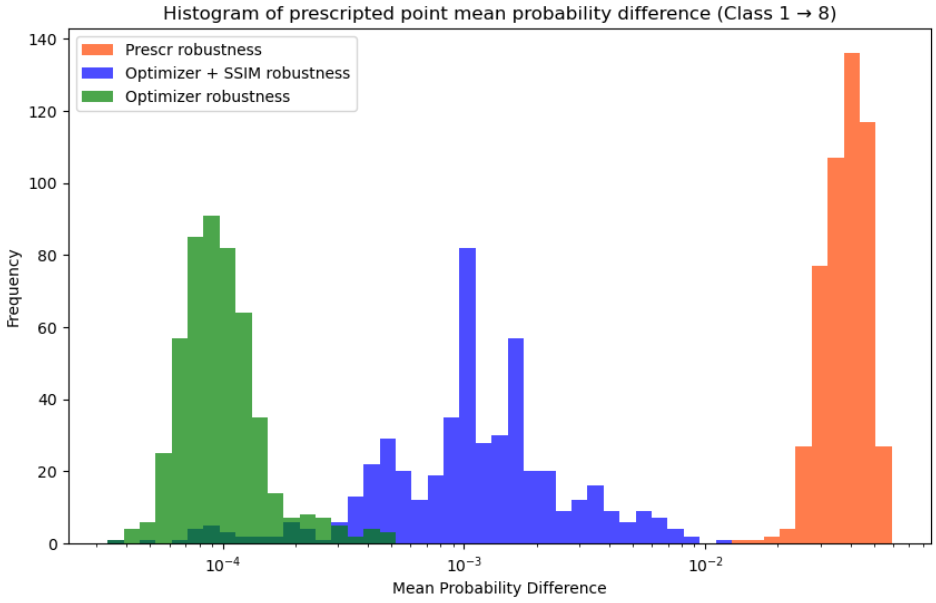

Figure 5.8: Distribution of robustness values for each prescriptive model in the MNIST case study, considering the sample where all starting points are 1 and the target class tc is 8.

For the industrial case study, robustness is evaluated using the original mathematical formulation in Equation 5.30, with results shown in Figure 5.9 and Table 5.7. Here, the optimizer provides significantly more robust prescriptions. As observed in the table, no prescription generated by the optimizer has a robustness lower than 100%. Conversely, PrescrX produces prescriptions with noticeably lower robustness, consistent with previous observations in the industrial case.

This marked difference can also be explained by the fact that the optimizer tends to produce prescriptions that are very similar to each other, concentrating within a compact region of the space. By contrast, the prescriptions generated by PrescrX do not form a single identifiable cluster, as already noted in Figure 4.1 of the previous chapter.

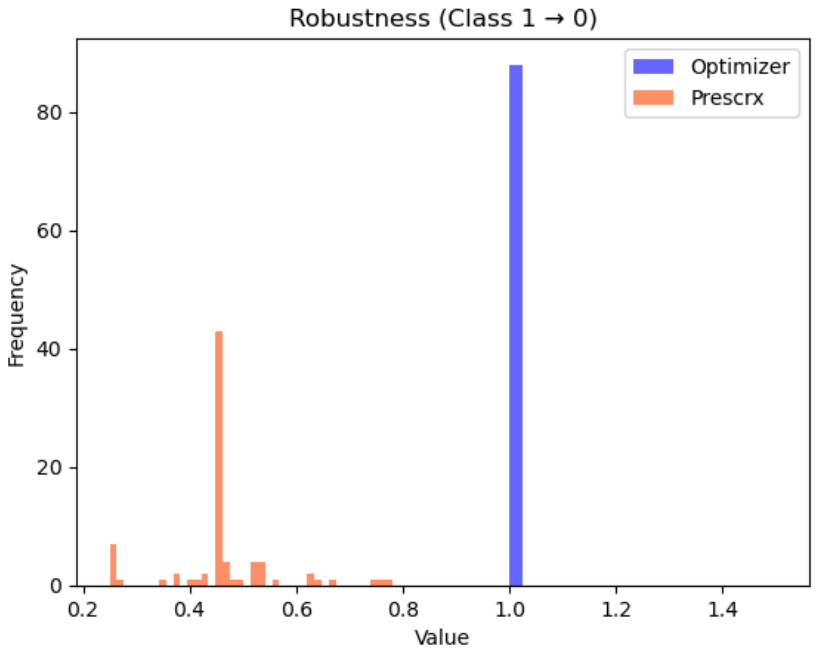

Figure 5.9: Distribution of robustness values for each prescriptive model in the industrial case study.

Robustness Comparison, industrial

Statistic	Optimizer	PrescrX
Mean	1.000	0.583
Std Dev	0.000	0.207
Min	1.000	0.250
25%	1.000	0.460
Median	1.000	0.520
75%	1.000	0.650
Max	1.000	1.000

Table 5.7: Summary statistics for robustness metrics in the industrial case: optimizer and PrescrX.

Execution time analysis

Although not a proper metric, the time required to obtain a prescription can be of fundamental importance. In certain contexts, execution time may be critical. For example, in an industrial setting where prescriptions are used to prevent the production of defective components, the process would be ineffective—or at least unsatisfactory—if prescriptions required too long to compute. Conversely, in a medical context, the time required to generate a prescription may reasonably span several minutes or even hours (except in specific urgent scenarios).

Looking at the MNIST case study, it can be observed from the logarithmic-scale plots (Figures 5.10 and 5.11) and the corresponding tables (Tables 5.8 and 5.9) that the optimizer without similarity is approximately two orders of magnitude faster than both PrescrX and the similarity-based optimizer. This consistent difference is explained by the high computational cost of space exploration. Indeed, whether performed through Euclidean-based metrics (as in PrescrX) or through similarity-based metrics such as SSIM (as in the optimizer with similarity), this step is the most time-consuming part of the process. At the same time, however, it is also what ensures that the resulting prescriptions are not adversarial examples. In certain applications, therefore, selecting the most efficient method for space exploration can be crucial if minimal execution time is a requirement of the problem.

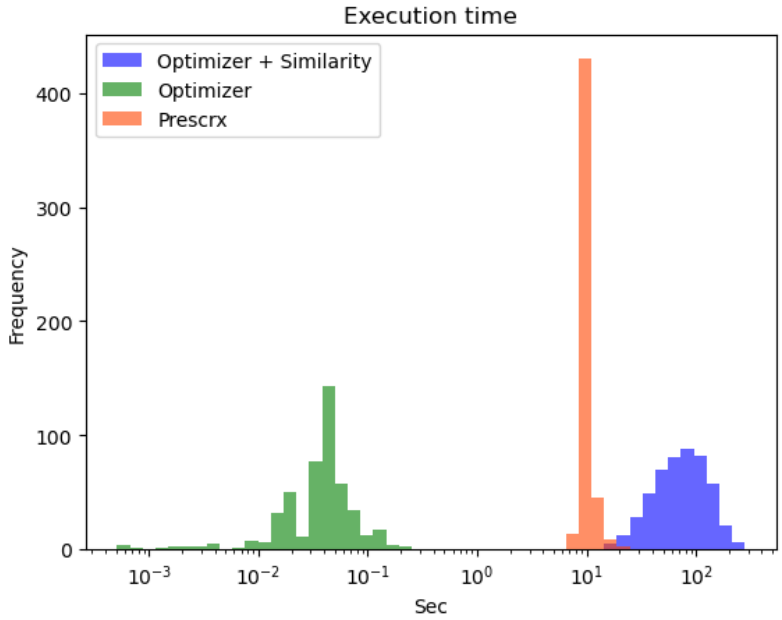

Figure 5.10: Execution time distribution for each prescriptive model in the MNIST case study, random sample.

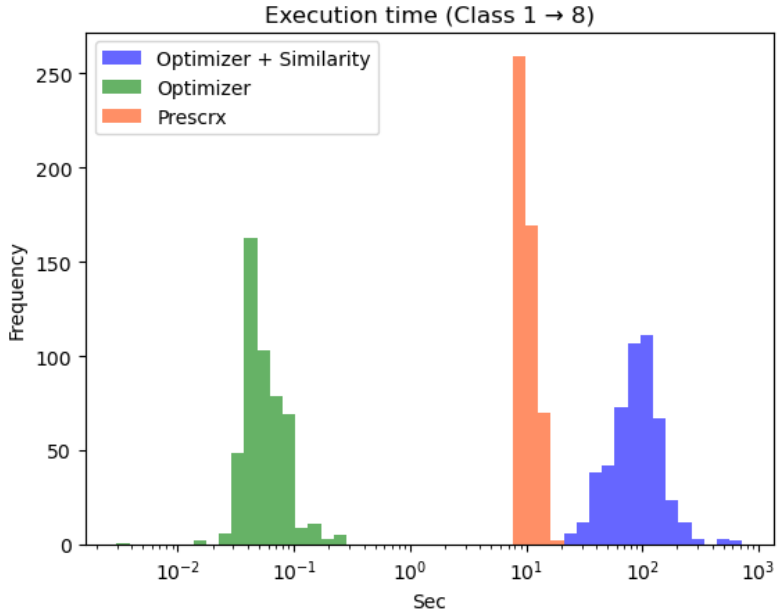

Figure 5.11: Execution time distribution for each prescriptive model in the MNIST case study, 1-to-8 sample.

Execution time, MNIST, random

Statistic	Optimizer	Optimizer SSIM	PrescrX
Mean	0.043	82.619	9.683
Std Dev	0.031	44.584	1.537
Min	0.000	16.189	8.231
25%	0.019	48.367	8.897
Median	0.043	74.588	9.213
75%	0.050	106.800	9.879
Max	0.248	280.488	21.559

Table 5.8: Summary statistics for execution times in the random MNIST case: optimizer, optimizer with SSIM, and PrescrX.

Execution time, MNIST, 1 to 8

Statistic	Optimizer	Optimizer SSIM	PrescrX
Mean	0.062	100.381	10.470
Std Dev	0.032	64.960	1.555
Min	0.003	22.745	8.747
25%	0.046	64.716	9.247
Median	0.050	88.814	9.770
75%	0.070	118.905	11.195
Max	0.267	725.888	17.860

Table 5.9: Summary statistics for execution times in the MNIST 1-to-8 case: optimizer, optimizer with SSIM, and PrescrX.

Finally, considering the industrial case (Figures 5.12 and Table 5.10), it becomes evident that PrescrX achieves significantly lower execution times compared to the similarity-based optimizer. Again, the chosen space exploration strategy proves to be decisive: PrescrX explores only the local region relevant to the prescription, whereas the optimizer performs exploration over the entire dataset, thereby making the process far more time-consuming.

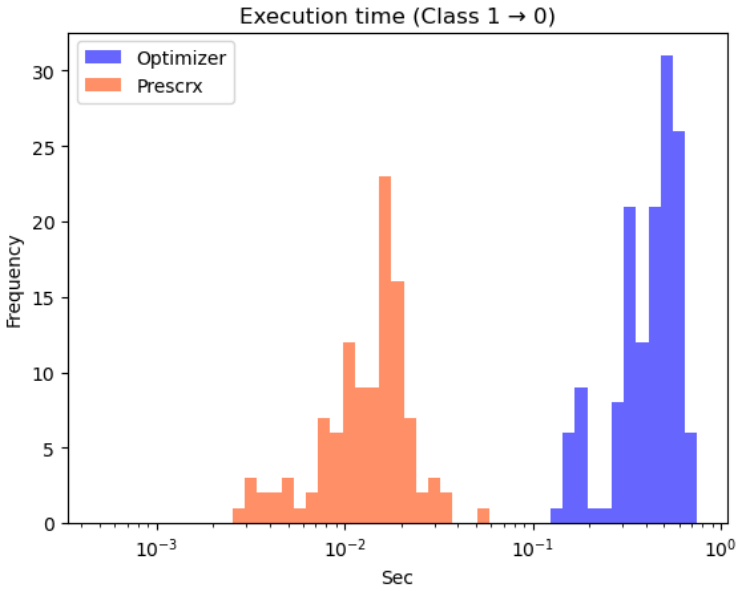

Figure 5.12: Execution time distribution for each prescriptive model in the industrial case study.

Execution time, industrial

Statistic	Optimizer	PrescrX
Mean	0.436	0.014
Std Dev	0.148	0.008
Min	0.144	0.000
25%	0.328	0.008
Median	0.448	0.014
75%	0.552	0.018
Max	0.751	0.055

Table 5.10: Summary statistics for execution times in the industrial case: optimizer and PrescrX.

Thanks to this set of proposed metrics, it is possible to conduct a quantitative evaluation of prescriptive models, as illustrated in this chapter, as well as to evaluate single prescriptions by jointly considering the entire metric set. Moreover, this should be regarded only as a first step toward quantitatively testing the quality of prescriptions. Future work may explore alternative approaches, either by developing new general-purpose metrics applicable across contexts, or by introducing ad hoc metrics specifically tailored to the problem under analysis.

Chapter 6

Conclusions and Future Work

6.1 Summary of contributions

This thesis investigated prescriptive analytics from both a methodological and an empirical perspective. First, it clarified the position of prescriptive analytics within the broader analytics spectrum, emphasizing its distinctive goal: recommending concrete actions rather than only describing or predicting outcomes. Then it introduced *PrescrX*, a local, model-agnostic prescriptive tool inspired by XAI ideas (notably LIME), and formalized its core pipeline: targeted neighborhood construction, local linear approximation of the decision boundary, and a constrained quadratic projection to obtain minimal interventions.

Beyond the tool itself, the work proposed an initial set of evaluation metrics for prescriptions—including Normalized Distance, Gain of Neighbors (GoN), Feature Preservation, and Robustness—in order to move from qualitative, visual checks to quantitative and repeatable assessments. Finally, two case studies (MNIST and an industrial cleaning process) were developed to compare PrescrX with optimizers built around different objectives, highlighting how alternative objectives yield qualitatively different prescriptions.

6.2 What the results show

6.2.1 Minimal-change prescriptions remain close to the data manifold

Across experiments, PrescrX consistently generated prescriptions that stay near empirical data regions. In the MNIST 1→8 setting, t-SNE and UMAP visualizations

showed that PrescrX points concentrated within or near the cloud of true samples from the target class, whereas standard probability-maximizing optimizers produced a distinct cluster detached from the data manifold and often corresponding to adversarial-looking digits.

This behavior is coherent with the tool's objective: PrescrX searches for the smallest feasible move that flips the class, not for the *most confident* point in the target class irrespective of plausibility. The local linearization plus projection encode this "short move" bias, and the targeted sampling steers the search along directions supported by the background data.

6.2.2 Optimizers and the role of similarity

Optimizers that simply maximize target-class probability can converge to solutions that are correct for the classifier but implausible for humans, i.e., adversarial. Introducing an explicit similarity term into the optimizer (SSIM for images; Mahalanobis distance for tabular industrial data) partially mitigated this effect: it both improved plausibility and made the optimization landscape smoother and more navigable in the industrial case.

6.2.3 Metric-based evidence

The metrics corroborated the qualitative patterns:

- **Normalized Distance**: PrescrX achieved the smallest feature-space changes in the MNIST 1→8 analysis (median ≈ 0.65) compared with the optimizer variants, aligning with its minimal-change objective.

- **Gain of Neighbors (GoN)**: The optimizer with similarity typically produced prescriptions in very dense target-class regions (GoN close to 1), while PrescrX prescriptions showed non-negative but generally smaller GoN, reflecting its preference to remain near the starting point rather than to migrate to the densest cluster. The optimizer without similarity often had GoN ≤ 0, matching its tendency to leave the data manifold.

- **Robustness**: In the industrial case, the optimizer with similarity delivered highly robust prescriptions (all at or near 100% under the adopted definition), whereas PrescrX, which disperses outputs around the background data, had lower robustness statistics. This difference is consistent with each method's objective and search behavior.

6.2.4 Constraints and practical feasibility

PrescrX natively accommodates domain constraints (locked features, bounds, integer domains) and can incorporate a cost function constraint. In the industrial case, using the domain-provided *climate impact* function, prescriptions visibly relocated in space to satisfy the cost threshold while still moving towards "clean" regions, illustrating how feasibility filters shape actionable recommendations.

6.2.5 Execution time trade-offs

A clear trade-off emerged between realism and speed. In MNIST, the baseline optimizer (no similarity) was roughly two orders of magnitude faster than PrescrX and than the similarity-based optimizer, because dense exploration or similarity computations dominate runtime. Although not a formal metric, compute time is central for deployment: the best choice depends on application latency requirements.

6.3 Guidelines for choosing a method

The case studies suggest the following guidelines for selection and tuning:

- If **minimal, interpretable change** and **staying close to observed data** are priorities (e.g., process set-point tweaks, user-facing recourses), PrescrX is well-suited. Prefer moderate n_{closest} and m to control runtime, set $\texttt{trust}_t$ to filter unstable local fits, and use α to regularize the local model when sampling is sparse.
- If the goal is to **maximize success probability** and prescriptions can deviate significantly from the starting point, use an optimizer with an explicit **similarity term**; in tabular contexts, a Mahalanobis component helps both plausibility and convergence.
- Always consider **domain constraints** early (bounds, discreteness, lockings, and cost caps). These fundamentally shape the feasible space and, therefore, the nature of the prescriptions.

6.4 Limitations

Several limitations deserve attention:

1. **Local linearity assumption**. PrescrX relies on a local linear surrogate. When class boundaries are highly curved or fragmented around the starting

point, the approximation may mislead the projection step, requiring additional sampling rounds and careful regularization.

2. **Data sparsity**. In small or poorly covered datasets, the target-point selection may fail or become brittle, as observed in parts of the industrial experiments where prescriptions could not be produced for certain parameter settings.

3. **Compute cost**. Dense local exploration and similarity estimates (when used) are the dominant contributors to runtime, which can be critical in real-time applications.

4. **Objective mismatch**. Different prescriptive objectives (minimal change vs. maximum confidence) are not directly comparable; metric selection must reflect the intended use (e.g., Normalized Distance for recourse-effort, GoN for plausibility, Robustness for stability under perturbations).

6.5 Implications

6.5.1 Scientific

The metrics proposed here operationalize aspects of prescription quality and open a path towards more standardized evaluation. In particular, GoN behaved as an early warning for adversarial prescriptions in MNIST; Robustness separated clustered, optimizer-style outputs from locally faithful, PrescrX outputs in the industrial case.

6.5.2 Industrial

For process control and quality assurance (e.g., the cleaning case), minimal, bounded changes that respect operational constraints and cost caps are often preferable to large jumps with marginal confidence gains. PrescrX provides such prescriptions and can be combined with a cost function to encode sustainability or safety policies (e.g., climate impact thresholds).

6.6 Future work

Three directions appear particularly promising:

1. **Causal prescription**. Integrate causal structure learning and identification into PrescrX so that recommended actions act on causes rather than proxies. This would directly align the tool with the decision-centric spirit of prescriptive analytics and mitigate spurious recourses.

2. **Adaptive exploration and surrogates**. Replace fixed exploration schedules with adaptive sampling schemes that estimate local curvature and uncertainty; investigate non-linear but still interpretable surrogates (e.g., locally weighted splines) while preserving the minimal-change objective.

3. **Scalability and real-time deployment**. Engineer approximate nearest-neighbor search, batched QP solves, and hardware acceleration to reduce latency by one to two orders of magnitude. Coupling PrescrX with streaming pipelines would enable on-line corrective prescriptions in IoT settings.

6.7 Closing remarks

Prescriptive analytics promises not only to predict but to *shape* outcomes. This thesis shows that the choice of objective and the way we explore the neighborhood of a decision point are decisive: minimal-change, data-manifold-aware methods such as PrescrX produce prescriptions that are plausible and interpretable, while pure probability maximization requires explicit similarity control to avoid adversarial or brittle results. The proposed metrics, together with the two case studies, provide a first step towards principled, comparable assessments of prescriptive systems and towards safer, more useful deployments of AI-driven decision support.

Bibliography

[1] Katerina Lepenioti, Alexandros Bousdekis, Dimitris Apostolou, and Gregoris Mentzas. «Prescriptive analytics: Literature review and research challenges». In: *International Journal of Information Management* 50 (2020), pp. 57–70. ISSN: 0268-4012. DOI: https://doi.org/10.1016/j.ijinfomgt.2019.04.003. URL: https://www.sciencedirect.com/science/article/pii/S0268401218309873 (cit. on pp. 3, 15).

[2] Debashish Roy, Rajeev Srivastava, Mansi Jat, and Mustafa Said Karaca. «A Complete Overview of Analytics Techniques: Descriptive, Predictive, and Prescriptive». In: *Decision Intelligence Analytics and the Implementation of Strategic Business Management*. Ed. by P. Mary Jeyanthi, Tanupriya Choudhury, Dieu Hack-Polay, T. P. Singh, and Sheikh Abujar. Cham: Springer International Publishing, 2022, pp. 15–30. ISBN: 978-3-030-82763-2. DOI: 10.1007/978-3-030-82763-2_2. URL: https://doi.org/10.1007/978-3-030-82763-2_2 (cit. on p. 3).

[3] Wayne W Eckerson. «Predictive analytics». In: *Extending the Value of Your Data Warehousing Investment. TDWI Best Practices Report* 1 (2007), pp. 1–36 (cit. on p. 3).

[4] Pedersen T.B Frazzetto D. Nielsen T.D. «Prescriptive analytics: a survey of emerging trends and technologies». In: *The VLDB Journal 28* (2019), pp. 575–595. DOI: https://doi.org/10.1007/s00778-019-00539-y (cit. on pp. 4, 15).

[5] László E. Szabó Gábor Hofer-Szabó Miklós Rédei. «On Reichenbach's Common Cause Principle and Reichenbach's Notion of Common Cause». In: *The British Journal for the Philosophy of Science* 50.3 (1999) (cit. on p. 5).

[6] Philipp Hennig, Michael A Osborne, and Mark Girolami. «Probabilistic numerics and uncertainty in computations». In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 471.2179 (2015), p. 20150142 (cit. on p. 8).

[7] Ernesto C. Martínez, Mariano D. Cristaldi, and Ricardo J. Grau. «Dynamic optimization of bioreactors using probabilistic tendency models and Bayesian active learning». In: *Computers & Chemical Engineering* 49 (2013), pp. 37–49. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2012.09.010. URL: https://www.sciencedirect.com/science/article/pii/S0098135412002888 (cit. on p. 8).

[8] Lily Geraldine Koops. «Optimized maintenance decision-making—A simulation-supported prescriptive analytics approach based on probabilistic cost-benefit analysis». In: *PHM Soc. Eur. Conf.* Vol. 5. 2020, p. 14 (cit. on p. 8).

[9] Arash Rasaizadi, Iman Farzin, and Fateme Hafizi. «Machine learning approach versus probabilistic approach to model the departure time of non-mandatory trips». In: *Physica A: Statistical Mechanics and its Applications* 586 (2022), p. 126492 (cit. on p. 8).

[10] Lele Luan, Nesar Ramachandra, Sandipp Krishnan Ravi, Anindya Bhaduri, Piyush Pandita, Prasanna Balaprakash, Mihai Anitescu, Changjie Sun, and Liping Wang. «Scalable Probabilistic Modeling and Machine Learning With Dimensionality Reduction for Expensive High-Dimensional Problems». In: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Vol. 87295. American Society of Mechanical Engineers. 2023, V002T02A011 (cit. on p. 8).

[11] Fiorentia-Zoi Anglou, Stavros Ponis, and Athanasios Spanos. «A machine learning approach to enable bulk orders of critical spare-parts in the shipping industry». In: *Journal of Industrial Engineering and Management* 14.3 (2021), pp. 604–621 (cit. on p. 9).

[12] Dana Pessach, Gonen Singer, Dan Avrahami, Hila Chalutz Ben-Gal, Erez Shmueli, and Irad Ben-Gal. «Employees recruitment: A prescriptive analytics approach via machine learning and mathematical programming». In: *Decision support systems* 134 (2020), p. 113290 (cit. on p. 9).

[13] Sharan Srinivas and A Ravi Ravindran. «Optimizing outpatient appointment system using machine learning algorithms and scheduling rules: A prescriptive analytics framework». In: *Expert Systems with Applications* 102 (2018), pp. 245–261 (cit. on p. 9).

[14] Edwin KP Chong and Stanislaw H Żak. *An introduction to optimization*. Vol. 76. John Wiley & Sons, 2013 (cit. on p. 9).

[15] Martin Moesmann and Torben Bach Pedersen. «Data-Driven Prescriptive Analytics Applications: A Comprehensive Survey». In: *arXiv preprint arXiv:2412.00034* (2024) (cit. on p. 10).

[16] Johannes Kunze Von Bischhoffshausen, Markus Paatsch, Melanie Reuter, Gerhard Satzger, and Hansjoerg Fromm. «An Information System for Sales Team Assignments Utilizing Predictive and Prescriptive Analytics». In: *2015 IEEE 17th Conference on Business Informatics*. Vol. 1. 2015, pp. 68–76. DOI: 10.1109/CBI.2015.38 (cit. on p. 10).

[17] Molham Aref, Balder ten Cate, Todd J. Green, Benny Kimelfeld, Dan Olteanu, Emir Pasalic, Todd L. Veldhuizen, and Geoffrey Washburn. «Design and Implementation of the LogicBlox System». In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. SIGMOD '15. Melbourne, Victoria, Australia: Association for Computing Machinery, 2015, pp. 1371–1382. ISBN: 9781450327589. DOI: 10.1145/2723372.2742796. URL: https://doi.org/10.1145/2723372.2742796 (cit. on p. 10).

[18] Ahmed Ghoniem, Agha Iqbal Ali, Mohammed Al-Salem, and Wael Khallouli and. «Prescriptive analytics for FIFA World Cup lodging capacity planning». In: *Journal of the Operational Research Society* 68.10 (2017), pp. 1183–1194. DOI: 10.1057/s41274-016-0143-x. eprint: https://doi.org/10.1057/s41274-016-0143-x. URL: https://doi.org/10.1057/s41274-016-0143-x (cit. on p. 11).

[19] Aanchal Goyal et al. «Asset health management using predictive and prescriptive analytics for the electric power grid». In: *IBM Journal of Research and Development* 60.1 (2016), pp. 4–1 (cit. on p. 11).

[20] Christopher Wissuchek and Patrick Zschech. «Prescriptive analytics systems revised: a systematic literature review from an information systems perspective». In: *Information Systems and e-Business Management* (2024), pp. 1–75 (cit. on p. 11).

[21] Lian Duan and Ye Xiong. «Big data analytics and business analytics». In: *Journal of Management Analytics* 2.1 (2015), pp. 1–21 (cit. on p. 12).

[22] Zhi-Hui Zhan, Lin Shi, Kay Chen Tan, and Jun Zhang. «A survey on evolutionary computation for complex continuous optimization». In: *Artificial Intelligence Review* 55.1 (2022), pp. 59–110 (cit. on p. 12).

[23] Crina Grosan and Ajith Abraham. «Hybrid evolutionary algorithms: methodologies, architectures, and reviews». In: *Hybrid evolutionary algorithms*. Springer 2007, pp. 1–17 (cit. on p. 12).

[24] Olivier Francon, Santiago Gonzalez, Babak Hodjat, Elliot Meyerson, Risto Miikkulainen, Xin Qiu, and Hormoz Shahrzad. «Effective reinforcement learning through evolutionary surrogate-assisted prescription». In: *Proceedings of the 2020 Genetic and evolutionary computation conference*. 2020, pp. 814–822 (cit. on p. 13).

[25] Risto Miikkulainen, Olivier Francon, Elliot Meyerson, Xin Qiu, Darren Sargent, Elisa Canzani, and Babak Hodjat. «From prediction to prescription: evolutionary optimization of nonpharmaceutical interventions in the COVID-19 pandemic». In: *IEEE Transactions on Evolutionary Computation* 25.2 (2021), pp. 386–401 (cit. on p. 13).

[26] Sorawee Yanta, Sotarat Thammaboosadee, Pornchai Chanyagorn, and Rojjalak Chuckpaiwong. «Course performance prediction and evolutionary optimization for undergraduate engineering program towards admission strategic planning». In: *ICIC Express Letters* 15.6 (2021), pp. 567–573 (cit. on p. 13).

[27] Hasan Davulcu, Michael Kifer, CR Ramakrishnan, and IV Ramakrishnan. «Logic based modeling and analysis of workflows». In: *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. 1998, pp. 25–33 (cit. on p. 13).

[28] Joao Marques-Silva. «Logic-based explainability: past, present and future». In: *International Symposium on Leveraging Applications of Formal Methods*. Springer. 2024, pp. 181–204 (cit. on p. 13).

[29] Cindy G. de Jesus and Mark Kristian C. Ledda. «Intervention Support Program for Students at Risk of Dropping Out Using Fuzzy Logic-Based Prescriptive Analytics». In: *2021 IEEE 17th International Colloquium on Signal Processing & Its Applications (CSPA)*. 2021, pp. 144–149. DOI: 10.1109/CSPA52141.2021.9377304 (cit. on p. 13).

[30] Henri Laude. «France's governmental big data analytics: From predictive to prescriptive using R». In: *Federal data science*. Elsevier, 2018, pp. 81–94 (cit. on p. 14).

[31] Manjula Ramannavar and Nandini S Sidnal. «A proposed contextual model for big data analysis using advanced analytics». In: *Big Data Analytics: Proceedings of CSI 2015*. Springer. 2018, pp. 329–339 (cit. on p. 14).

[32] Fan Du, Catherine Plaisant, Neil Spring, and Ben Shneiderman. «EventAction: Visual analytics for temporal event sequence recommendation». In: *2016 IEEE Conference on Visual Analytics Science and Technology (VAST)*. 2016, pp. 61–70. DOI: 10.1109/VAST.2016.7883512 (cit. on p. 14).

[33] Han Xiao, Kashif Rasul, and Roland Vollgraf. «Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms». In: *arXiv preprint arXiv:1708.07747* (2017) (cit. on p. 25).

[34] Christos Thrampoulidis, Samet Oymak, and Babak Hassibi. «Regularized linear regression: A precise analysis of the estimation error». In: *Conference on Learning Theory*. PMLR. 2015, pp. 1683–1709 (cit. on p. 31).

[35] Farzana Anowar, Samira Sadaoui, and Bassant Selim. «Conceptual and empirical comparison of dimensionality reduction algorithms (pca, kpca, lda, mds, svd, lle, isomap, le, ica, t-sne)». In: *Computer Science Review* 40 (2021), p. 100378 (cit. on p. 34).

[36] Salifu Nanga, Ahmed Tijani Bawah, Benjamin Ansah Acquaye, Mac-Issaka Billa, Francis Delali Baeta, Nii Afotey Odai, Samuel Kwaku Obeng, and Ampem Darko Nsiah. «Review of dimension reduction methods». In: *Journal of Data Analysis and Information Processing* 9.3 (2021), pp. 189–231 (cit. on p. 37).

[37] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. «Explaining and harnessing adversarial examples». In: *arXiv preprint arXiv:1412.6572* (2014) (cit. on p. 40).

[38] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. «Image quality assessment: from error visibility to structural similarity». In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612 (cit. on p. 42).

[39] Jeffrey Larson, Matt Menickelly, and Stefan M Wild. «Derivative-free optimization methods». In: *Acta Numerica* 28 (2019), pp. 287–404 (cit. on p. 44).

[40] Michael JD Powell. «A view of algorithms for optimization without derivatives». In: *Mathematics Today-Bulletin of the Institute of Mathematics and its Applications* 43.5 (2007), pp. 170–174 (cit. on p. 44).

[41] Maliki Moustapha and Bruno Sudret. «Learning non-stationary and discontinuous functions using clustering, classification and Gaussian process modelling». In: *Computers & Structures* 281 (2023), p. 107035 (cit. on p. 45).

[42] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. «The mahalanobis distance». In: *Chemometrics and intelligent laboratory systems* 50.1 (2000), pp. 1–18 (cit. on p. 45).