
**UNIVERSITY OF PISA**

DEPARTMENT OF COMPUTER SCIENCE

Master Degree in Data Science and Business Informatics

DEGREE THESIS

# MANAGING DISAGREEMENT IN INTERACTIVE LEARNING : A COUNTERFACTUAL EXPLANATION APPROACH

SUPERVISORS:

Prof.Roberto PELLUNGRINI

Prof.Riccardo GUIDOTTI

Prof.ssa Anna MONREALE

CANDIDATE:

Alessandro MASTRORILLI

ACADEMIC YEAR 2024-25

# Contents

# List of Figures

# List of Tables

# Chapter 1

# INTRODUCTION

In recent years, advancements in Artificial Intelligence (AI) and the increasing availability of big data have revolutionized many sectors, making automation and prediction central tools in decision-making processes. For instance , in healthcare, AI is used for diagnostic support and personalized treatment plans; in finance, for fraud detection and algorithmic trading; in the legal domain, for document classification and predictive justice; and in education, for personalized learning and automatic grading. In all these fields, automation and prediction have become central tools in decision-making processes. However, the predictive effectiveness of many machine learning (ML) models is often hindered by a fundamental limitation: their opacity. These systems are frequently perceived as "black boxes" that do not provide clear and comprehensible explanations for the decisions they produce. This issue is particularly significant in **human-in-the-loop**[48] contexts, where the role of the human is central not only as a passive user but as an active agent who can intervene, correct, or validate the system's decisions. In such scenarios, the interaction between model and user must be transparent, bidirectional, and explainable, in order to establish an effective and trustworthy dialogue between the two parties. By bidirectional interaction refers to a dynamic form of interaction where the user does not merely receive predictions, but can also provide feedback, contest or correct outputs, and influence the model's learning process over time. This perspective contrasts with traditional AI systems, which operate in a unidirectional way—providing decisions without considering human feedback. In bidirectional systems, the human is an integral part of the decision loop, contributing actively to the refinement of model behavior. Within this framework lies the **FRANK** model[43], based on the **Skeptical Learning** paradigm, which introduces the concept of *skepticality*, a metric that quantifies the degree of user skepticism toward the model's predictions. This mechanism enables the monitoring

of the alignment between human and machine decisions, granting the user corrective power. However, FRANK has a significant limitation: although it detects misalignments, it does not provide comprehensive explanations to justify such discrepancies. This shortcoming reduces the effectiveness of interaction and undermines user trust in the system. To address this gap, the field of Explainable Artificial Intelligence (XAI) [5] offers promising solutions aimed at making AI models interpretable and understandable. Among the various XAI methodologies, counterfactual explanations[26] stand out for their intuitiveness and communicative power. A counterfactual explanation takes the form: "*If X had been different, then the prediction would have changed to Y*" . This type of reasoning is deeply rooted in cognitive and decision psychology: numerous studies have shown that counterfactual thinking is a natural and frequent mental process in humans, used to reflect on missed alternatives and to learn from past situations[14]. In other words, counterfactuals not only explain *"what"* determined a decision, but also provide a deep understanding of *"how"* it could be changed. Moreover, counterfactual explanations are **model-agnostic**, making them readily applicable in dynamic contexts such as continual learning[47], which underpins the functioning of FRANK. As such, these explanations are well-suited for integration into evolving and interactive scenarios, where the system's decisions are updated over time alongside the user's knowledge. This thesis aims to extend the FRANK framework by integrating counterfactual methods(and factual/counterfactual rule) as explainable tool for the user , with two main objectives:

1. To close the human–machine interaction loop by providing interpretable explanations in cases where the user expresses skepticism toward the model's predictions.

2. To empirically evaluate the effectiveness of counterfactual explanations in online and dynamic scenarios, considering key metrics such as sparsity, temporal validity, proximity, and plausibility.

To achieve these objectives, various scenarios will be artificially simulated and tested, each characterized by the presence of users with varying degrees of experience and credibility within the decision-making domain—restricted to medium or high levels—in order to replicate realistic interactions in high-expertise contexts. The analysis will be structured around two overarching research questions, which will be addressed through graphical representations and empirical evidence:

1. What is the quality of the explanations provided at different moments of user skepticism?

2. Is the approach proposed in FRANK effective in the context of continual learning and human inter-

action?

To answer the first question, the explanations generated by different counterfactual methods will be compared at various points of detected skepticism within the decision-making cycle. Specifically, the analysis will explore whether, as the interaction progresses, it becomes possible to produce counterfactual explanations that are increasingly closer to the original instance (or to other previously processed instances), while modifying the smallest possible number of features and within the shortest amount of time. In this regard, the aim is to identify which of the tested methods prove to be the most reliable and consistent within the evolving FRANK framework, evaluating their ability to preserve the original class throughout the entire iterative cycle. This aspect is closely linked to the decision boundary learned by the model: it is hypothesized that the quality of counterfactual instances is influenced by the configuration of the decision frontier, thereby suggesting a direct relationship between the model's predictive performance and the plausibility of the generated explanations. Accordingly, each analysis will be accompanied by performance metrics of the best-performing model in terms of accuracy and F1-score, in order to provide an integrated view that connects predictive reliability with explainability. Finally, to answer the second research question concerning the efficiency of the proposed approach, performance will be evaluated across different user scenarios (by varying the credibility/expertise profile), with particular attention to the frequency and distribution of skepticism episodes over time. This will allow for an assessment of the system's capacity to adapt and self-correct in response to user behavior and feedback, thereby verifying whether the proposed extension of FRANK effectively enhances human–machine interaction in online and dynamic contexts. To complement the previously described analysis, an additional experiment was conducted with the aim of enhancing the system's adaptive capacity in the presence of disagreement between the user and the model. Specifically, the LORE algorithm (described in detail in Chapter 2, dedicated section) was employed to generate a set of synthetic instances $Z$ that are similar to a given input instance$x$. When the user expresses disagreement with the model's decision, a corrective procedure is triggered that uses **LORE** to generate a local neighborhood around an instance $x'$,which is similar to $x$ but belongs to the class indicated by the user, rather than the class predicted by the model. In other words, the system searches for a plausible variation of $x$ that reflects the human judgment while remaining close in the feature space. The set of synthetic instances generated around $x'$ is then used to update the model through a re-training phase, with the aim of temporarily aligning its decisions with the expectations expressed by the user. However, this corrective

mechanism also raises broader questions: does this form of adaptation genuinely improve the system's behavior in the long term? Or, conversely, does it risk introducing noise or instability into the continual learning process? These questions represent an additional point of reflection, which will be explored in the following chapters through an empirical analysis of the results obtained. The analysis focuses on the following key aspects:

- **Balanced and imbalanced datasets**, in order to assess the robustness of the proposed counterfactual methods across varying data distributions.
- **Continual learning models**, to evaluate how the evolving nature of the model affects the stability, plausibility, and effectiveness of the generated explanations over time.

The experimental results highlight both the opportunities and the challenges associated with the integration of counterfactual explanations into interactive and dynamic learning frameworks. On one hand, counterfactuals demonstrate potential as powerful tools to improve user trust, foster interpretability, and guide model correction. On the other hand, several limitations emerge — such as a degradation in explanation validity in the presence of imbalanced data, or inconsistent performance depending on the configuration of the decision boundary. These findings offer valuable insights for future developments, especially in the design of adaptive and user-centered AI systems. The remainder of this thesis is organized as follows:

- **Chapter 2** reviews the relevant literature(including the theoretical foundation for the proposed framework) detailing the models under consideration and the techniques used for counterfactual and factual explanation generation.
- **Chapter 3** provides a formal definition of the problem, setting and presents the extended version of the FRANK framework, incorporating the proposed explanatory and corrective mechanisms.
- **Chapter 4** describes the experimental setup, the simulated scenarios, and the results obtained, with a critical discussion of their implications.

# Chapter 2

# STATE OF THE ART

The evolution of hybrid human-machine systems requires careful consideration of three fundamental pillars:

- **Incremental learning models in dynamic scenarios**
- **Transparent interaction mechanisms between user and system**
- **XAI in decision-making**

This chapter explores the state of the art in these areas, highlighting how previous work has influenced the design of Frank and its proposed extension, with particular focus on the integration of counterfactuals and models used .

## 2.1 Related Work

This section reviews previous work that is conceptually or thematically related to the present study. It focuses on three core areas: continual learning in dynamic scenarios, the Human-in-the-Loop (HITL) paradigm, and skeptical learning. Additionally, since this thesis explores the integration of Explainable AI (XAI)—specifically counterfactual explanations—as a means to support human decision-making during instances of disagreement , a conceptual overview of post-hoc explanation methods and their key properties is provided. The theoretical underpinnings and motivations are introduced here to position this work within the broader research landscape.

### 2.1.1 Incremental Learning Models in Dynamic Scenarios

Lifelong learning, or continual learning (CL), is a machine learning paradigm in which a model incrementally learns from non-stationary data streams, acquiring new knowledge over time without forgetting previously learned information. As highlighted in [51], the human ability to accumulate knowledge throughout life is essential for autonomous agents operating in dynamic environments. In contrast, conventional machine learning models are typically trained on stationary data batches and suffer from catastrophic forgetting when exposed to changing data distributions. From this perspective, continual learning aims to strike a balance between **stability** (preserving past knowledge) and **plasticity** (incorporating new concepts) in the presence of data distribution shifts [30, 39]. CL is thus critical in real-world scenarios—such as robotics, recommender systems, or autonomous vehicles—where data is continuously generated and new contexts or tasks emerge. In the context of Continual Learning (CL), data arrives in the form of a data stream—a potentially infinite sequence of examples (or observations) that arrive over time [47]. Each element in the stream contains a set of features and, in supervised settings, an associated label. Data streams are characterized by:

- elements arriving either one at a time or in small batches;
- the learning system having no access to all past data (i.e., only a single pass over the stream is allowed);
- the underlying data distribution potentially changing over time.

In CL systems, the model processes each example as it arrives, incrementally updating its knowledge and adapting to new data without revisiting entire past batches [39, 47]. This stands in contrast to traditional learning setups, where models are trained offline on fixed datasets using batch learning. In data stream management, a common distinction is made between reactive and proactive streams. In reactive streams, data "comes at us" from external events that the model cannot control—for instance, a user's visit to a website or a sensor activation independent of the learning system [47]. In such cases, the system must simply react to incoming data.

In contrast, proactive streams allow the system to control data acquisition: for example, by reading from a file or a data source at will, deciding both the order and speed of reading [47]. In practice, reactive streams are often recorded offline and converted into proactive datasets for batch training purposes.

The challenge for Continual Learning (CL) lies in ensuring that a model trained offline on proactive data performs effectively when deployed in production and exposed to real-world, reactive data streams [47].
Online processing is the predominant approach in Continual Learning (CL) systems. It refers to a learning model that updates its parameters by observing and processing one example at a time [47]. Unlike conventional (batch) machine learning—where the model is trained on an entire static dataset—here, data arrives in a stream, and the model evolves dynamically.
An online model maintains an updated internal state and does not require reprocessing past data. This enables real-time updates and reduced memory usage, but also introduces greater sensitivity to shifts in the data distribution [39, 47]. The online approach is therefore essential for CL, as it allows the system to continuously learn from emerging data.
In real-world data streams, the target data distribution can change over time due to various factors, such as emerging trends, environmental variations, sensor malfunctions, and more. These changes are referred to as concept drift [32]. Formally, concept drift denotes variations in the generative distribution of the data. These may involve changes in the conditional distribution
$P(X \mid Y)$ — known as real drift — or in the marginal distribution of $X$ — known as virtual drift — [32].
Concept drift typically arises in supervised data stream settings, where a classifier trained on past data must adapt as the underlying concept to be predicted evolves. The ability to detect and respond to such changes is essential for maintaining reliable models in production environments [32].
In [32, 23] are described several types of concept drift . Below are the main categories, each illustrated with practical examples:

- **Abrupt drift**: The data distribution changes suddenly at a specific point in time (a change point). Examples include the immediate replacement of a sensor with another calibrated differently, or a sudden shift in the focus of interest—e.g., an analyst switching from monitoring meat prices to public transportation data [23].

- **Gradual drift**: The concept transitions more slowly over time. During this period, some examples follow the old concept while others reflect the new one, with the latter gradually dominating. Examples include a sensor slowly degrading or weather data evolving with the seasons [23].

- **Incremental drift**: Similar to gradual drift but with small, continuous changes at each step, without clearly separable stages. For instance, user preferences may change subtly over time, with each new

data block slightly different from the previous one.

- **Recurring drift**: Concepts that disappear and later re-emerge, often in seasonal contexts. For example, the class "summer holiday" may reappear every year. These recurrences imply that, although the system must learn new concepts, it should also retain knowledge of previous ones [32, 23].

To address concept drift, most methodologies involve two main phases:

1. Detecting significant changes in the data distribution
2. Adapting the model to maintain performance

Drift detection techniques commonly rely on real-time monitoring of data stream statistics or model prediction errors. For example:

- **Error monitoring**: The model continuously tracks its prediction error on incoming data. A sudden increase in error may indicate a drift. Classical methods include the **Drift Detection Method** (DDM) and the **Page-Hinkley Test**, which treat the error rate as a *Bernoulli* process and trigger alarms when statistical thresholds are exceeded [30].
- **Statistical tests on data streams**: These compare recent data windows with historical ones using tests such as the Kolmogorov–Smirnov test or other divergence measures. For instance, **ADaptive WINdowing (ADWIN)** maintains a dynamically sized window and detects drift by computing differences between segments [30].
- **Statistical Process Control (SPC) methods**: Analogous to control charts in industrial quality monitoring, these methods track statistics like the mean error within dynamic confidence limits [23]. Both SPC and distributional change tests aim to promptly identify shifts in the data stream.
- **Adaptive ensembles**: Some approaches use ensembles of models specialized for different temporal segments. For example, Adaptive Random Forest (ARF) maintains a pool of decision trees, replacing underperforming trees when a drift is detected [30]. In general, ensembles with built-in drift detectors are highly effective at adapting to various evolving concepts.

Once a drift is detected, the model can adapt in several ways:

- Retraining from scratch on recent data

- Using a sliding window to train only on the most recent examples, discarding outdated ones
- Using a sliding window to train only on the most recent examples, discarding outdated ones
- Adjusting update weights dynamically
- Using memory replay techniques to revisit relevant past examples and retain prior knowledge [39]

In short, adaptation strategies aim to update the model to reflect the new concept without entirely forgetting previously learned information. The effectiveness of each strategy depends on the type and rate of drift: for gradual drift, continuous incremental updates may suffice, whereas abrupt drifts may require resetting the model or introducing a new one alongside the old.

The challenge of concept drift is closely related to Continual Learning (CL). In a CL system, the data stream can be seen as a sequence of potentially shifting concepts. In [39] even interpret catastrophic forgetting as a direct consequence of distributional drift. In this sense, each new learning phase in CL can be viewed as a response to a drift event: the model must extend its knowledge to represent the new concept while avoiding the complete overwriting of previously acquired information.

As emphasized in recent literature, CL and streaming learning with drift share common goals [30, 32]. Classic streaming learning focuses on adapting to drift to ensure current accuracy, whereas CL imposes the additional constraint of preserving long-term knowledge . Thus, CL algorithms can be understood as advanced drift adaptation mechanisms, incorporating memory-based strategies (e.g., experience replay, knowledge distillation, modular architectures) to mitigate selective forgetting.

For example, in the presence of recurring drift, it is beneficial to retain examples of past concepts so they can be reactivated when those concepts reappear. In conclusion, managing concept drift is a core aspect of Continual Learning: timely drift detection and effective adaptive or memory-based responses are essential to achieving robust, long-term learning.

CL can therefore be regarded as a learning paradigm inherently designed to operate under distributional drift, with responsiveness to new data and model stability as its key requirements.

### 2.1.2 The Human-in-the-Loop (HITL) paradigm

In traditional Machine Learning (ML), models are typically designed and trained in a monolithic manner, with no further human intervention beyond the initial data labeling phase. The **Human-in-the-Loop**

**(HITL)** paradigm breaks this convention by incorporating human expertise directly into the learning cycle, enabling continuous interaction between users and the model.

In HITL settings, humans can intervene at multiple stages of the data pipeline—including data selection, correction of predictions, parameter tuning, and more—with the overarching goal of enhancing system performance and improving reliability. This approach addresses the limitations of purely automated systems, such as noisy or sparse data, and transparency or interpretability requirements. As such, HITL methods are particularly valuable in high-stakes domains, including medical decision support, computer vision, and autonomous driving.

In recent years, HITL has attracted considerable attention in the research community. For instance, in [48] there is a comprehensive definition of the paradigm, emphasizing the diversity of HITL variants based on who directs the learning process—human or machine. Depending on the nature of this control, several related approaches can be identified,such as active learning, interactive machine learning, and machine teaching, along with related methodologies like curriculum learning and explainable AI (XAI) techniques.

The Human-in-the-Loop (HITL) paradigm is based on the idea of enriching traditional training procedures with an iterative cycle in which the model and the human operator collaborate. In [48], HITL is defined as a set of interactions in which, in general, the system can autonomously select data or pose queries, and the user provides labels, feedback, or corrections . Depending on the degree of control, different approaches can be identified:

- **Active Learning (AL)**: The model selects informative instances to be labeled by a human (oracle). In [54] AL as a strategy where the algorithm achieves high accuracy using fewer labeled data, because it can actively choose which examples to query. At each iteration, the model formulates a query (e.g., "label this uncertain instance") and the human provides the answer, enabling a much more efficient training process .

- **Interactive Machine Learning (IML)**: The user is actively involved in every stage of training. They do not just provide labels, but can modify model parameters, add targeted examples, correct errors on the fly, and more. Unlike AL, where the system drives the queries, IML involves shared or even user-driven control. This approach requires user-friendly interfaces and integrates human-computer interaction (HCI) considerations. As noted by researchers, evaluating an IML system must account for both model accuracy and user experience[48].

- **Machine Teaching (MT)**: An expert strategically arranges the training data (e.g., by selecting key examples or assigning weights) to teach the concept to the model as efficiently as possible. Machine teaching reverses the traditional paradigm: instead of the algorithm selecting instances, the human designs the learning path [48].
- **Explainability and Curriculum Learning**: Human involvement can also extend to explanation processes (XAI) or dataset organization (curriculum learning). For instance, an expert may reorder examples from easiest to hardest to facilitate learning (curriculum learning), or derive interpretable rules from the model's decisions (explainable AI)[48].

In all these cases, *human–machine collaboration* can enhance both qualitative aspects (such as transparency and trust) and quantitative outcomes (such as accuracy), compared to fully automated systems. However, integrating humans into the loop also introduces new complexities: it becomes essential to design interfaces and algorithms that maximize human contribution without overloading the operator. As highlighted in [3], poorly managed interaction can lead to user frustration, for example when users are repeatedly asked to respond to multiple queries without perceiving any control over the process.

The various modes of collaboration envisaged by Human-in-the-Loop paradigms are outlined below:

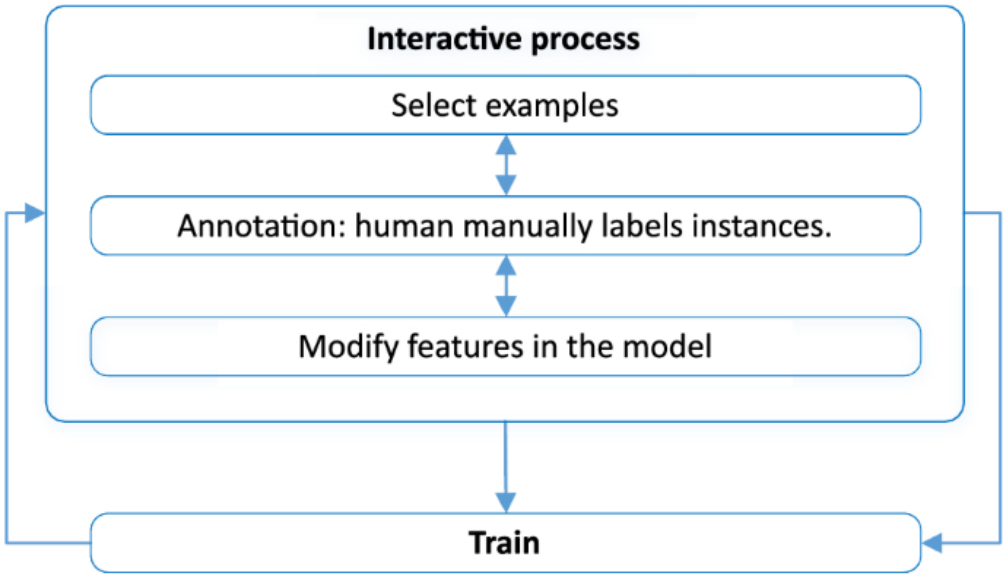


Figure 2.1: Simplified diagram of an Interactive Machine Learning (IML) process. The system selects examples to be labeled by a human, updates the model with these annotations, and repeats the cycle[48]

In a typical Interactive Machine Learning (IML) approach, illustrated in 2.1, the model identifies new examples to be evaluated, and the human operator manually provides labels. The model is then iteratively

retrained using this human feedback

This process highlights how the human expert becomes an active part of the training loop, offering corrections and targeted input to guide the system toward improved performance. However, as noted in [3], continuous interaction can lead to user frustration if annotation requests are too frequent or intrusive. This section will explore the prevoius areas, except Explainable AI (XAI) that will be addressed separately in a dedicated section.

**HITL and Active Learning**

Active learning is perhaps the most widely adopted instance of the Human-in-the-Loop (HITL) paradigm and therefore warrants dedicated attention. As previously mentioned, in Active Learning (AL), the model actively queries a human operator (oracle) for labels on the most informative examples to improve its learning process[54]. The typical AL cycle involves the following steps:

1. a model is trained on a small labeled dataset;
2. the model then evaluates the uncertainty or informativeness of each unlabeled instance and selects the one deemed most useful
3. the selected instance is submitted to the human for annotation;
4. the model is updated with the new label, and the cycle repeats.

The primary goal is to achieve high accuracy using a minimal number of labeled examples, thereby significantly reducing annotation costs. A critical aspect of Active Learning (AL) is how to assess the informativeness of an unlabeled instance. The literature proposes numerous selection strategies, which can be grouped into major families. For example:

- **Uncertainty Sampling**: the algorithm selects examples on which it is least confident (e.g., those with the lowest classification confidence or highest entropy). This approach encourages the model to refine its decision boundary. For instance, in a binary classifier, the instance with a predicted probability closest to 0.5 would be chosen.
- **Query by Committee**: a "committee" of diverse models is maintained, and instances that produce the greatest disagreement among models are selected. The intuition is that disagreement signals areas of high uncertainty that deserve human annotation.

- **Expected Error/Model Change**: each potential query is evaluated based on how much it would change the model's performance (or its parameters) if it were labeled. Instances expected to yield the greatest improvement are selected.

- **Representativeness-based Strategies**: these approaches consider not only uncertainty but also data density and diversity. For instance, preference may be given to instances that are uncertain but also representative of larger data clusters. In [1] is propose a taxonomy that distinguishes informative strategies (based purely on uncertainty) from representative strategies (which also account for dataset structure) . This latter group helps prevent overfitting to outliers and ensures better domain coverage.

Each time an instance is selected, it is manually labeled by a human (e.g., assigning a class or target value) and added to the training set (minds.wisconsin.edu). The model then updates its parameters to incorporate the human-provided information. Empirical studies show that, thanks to these targeted strategies, AL accelerates model convergence. For instance, in [54] is demonstrated that using uncertainty sampling, a logistic regression model can reach 0.90 accuracy after labeling only 30 examples, compared to 0.70 when using random sampling—representing a 67% reduction in error with less than 10% of the data. In other words, AL can achieve performance comparable to traditional supervised training with a significantly reduced number of labels. The ultimate goal of AL is to improve training efficiency. As mentioned, models can learn accurately with far fewer labeled examples [54]. In general, the literature reports that—although exceptions exist—AL reduces the number of annotations needed to reach a given level of accuracy[54]. However, this is not guaranteed in all settings: both theoretical and empirical studies have shown cases where AL, particularly in batch mode, may require more data than passive learning or may even underperform due to selection bias. Nevertheless, these outcomes are generally considered to result from unfavorable conditions or publication bias; most studies report tangible improvements[54]. As the number of labels increases, performance gains tend to saturate (approaching that of fully supervised training). Therefore, AL is particularly valuable when annotation budgets are tight and early efficiency is essential. Finally, evaluation of AL systems should consider not only final accuracy but also human effort. For example, it is useful to track how many examples per class are selected, in order to avoid strategies that systematically neglect minority classes[1].

**Interactive Machine Learning, IML**

Interactive Machine Learning (IML) extends the Human-in-the-Loop (HITL) paradigm by pushing interaction beyond simple annotation tasks. In [3] IML is defined as the design of algorithms and user interfaces that “engage users in the process of building machine learning models”. In this framework, the user is not merely a passive oracle, but an active participant in shaping the model. Typically, every user action—such as labeling an instance, adjusting parameters, or introducing constraints—results in an immediate and incremental update to the model. The key characteristics of IML are fast, focused, and incremental learning cycles : after each user interaction, the system slightly updates the model, allowing the user to iteratively explore the solution space via trial and error, and guide the system toward the desired behavior [3].

This iterative process enables even non-expert users to effectively “steer” the model at low cost, thereby democratizing the development of intelligent systems[3]. Unlike Active Learning (AL), where the machine remains “essentially a black box” offering little insight into its internal state, IML encourages users to develop an active mental model of the learning process [56].

For example, interactive clustering systems [3, 18]allow users to impose affinity constraints and iterate quickly over multiple solutions. Similarly, users can correct misclassifications or add training examples in real time . These strategies reduce system opacity by allowing users to observe the effects of their actions and continuously tailor their feedback[56].

In practice, IML encompasses any environment in which the model responds immediately to human input—be it labels, rules, or weights—and updates incrementally[3]. Numerous case studies (e.g., recommendation systems guided by user feedback) highlight how sustained user-system interaction leads to stronger synergy and models that better match real-world needs [3, 56].

In short, Interactive Machine Learning fosters fast, focused learning loops where the user “teaches through interaction,” making the model more interpretable and adaptable to human goals [3, 56].

**Machine Teaching , MT**

Machine Teaching represents a complementary approach to Active Learning (AL): instead of having the machine query the user, it is the human expert who programs the learning process. In MT, the user—referred to as the "teacher"—curates and provides the model with a set of carefully selected examples to effectively transfer their domain knowledge.

In [48] , in MT, the expert controls the learning process “by delimiting the knowledge to be transferred to the machine learning model”. In other words, the user pre-selects the most informative examples (and often determines their order) so that the model can quickly grasp a specific concept.

This paradigm has been formalized as the inverse problem of Active Learning : instead of the model selecting which examples to query, the teacher designs an optimal training set. As a result, Machine Teaching requires strong human involvement in the instructional design phase—the teacher builds the model’s learning curriculum from scratch, using domain expertise to minimize the number of examples needed.

MT therefore differs from AL not only in strategy but also in the nature of human involvement: whereas in AL the human passively responds to system queries, in MT the human takes an active, anticipatory role in shaping the learning process [48].

**Curriculum Learning**

Curriculum Learning (CL) draws inspiration from the way humans typically learn: training data is presented in a meaningful order, usually progressing from simpler to more complex examples. In [7] , is demonstrated that, for instance in computer vision tasks, models converge faster when initially trained on "easy" examples.

In the context of Human-in-the-Loop (HITL) learning, the curriculum often reflects human oversight in dataset design: the user—or dataset curator—manually defines the order or difficulty level of training instances. In [48] CL is an approach that “focuses on imposing a structure on the training set to accelerate and improve learning”.

Unlike Active Learning or Interactive Machine Learning, the human contribution in Curriculum Learning occurs primarily during the data preparation phase, rather than through real-time interaction during training. In this sense, CL bridges human expertise and HITL paradigms by leveraging human understanding of example complexity to guide model learning. However, once the curriculum is defined, the training proceeds autonomously according to the pre-established sequence, without further interaction loops with the user.

**Open Challenges**

Despite their advantages, Human-in-the-Loop (HITL) systems face several unresolved practical and theoretical challenges. Key issues include:

- **Cost and Scalability**:Engaging human experts is time-consuming and resource-intensive. To remain feasible in real-world settings, systems must minimize human intervention—e.g., through batch querying policies or effective stopping criteria[1].
- **Reliability of the Human Oracle**:Many Active Learning models assume a flawless oracle, yet in practice, human annotators are prone to errors and cognitive biases. Recent studies [52] show that when users rely on simplified heuristics, model performance may degrade significantly—sometimes falling below random baselines. This highlights the need for robust active algorithms that can handle noisy labels, for instance by explicitly modeling the annotator's error patterns[52].
- **Interaction and Usability**:User experience is critical. Amershi et al. have emphasized that continuous and unidirectional interactions can lead to user frustration[48]. Therefore, thoughtful interface design is essential: interactive AI systems should clearly communicate the purpose of each query and offer users greater control over the process[48].
- **election Bias and Domain Coverage**:Aggressive querying strategies may focus only on specific regions of the data space, neglecting others. This issue is especially problematic in multiclass or imbalanced settings. Mechanisms are needed to ensure sufficient exploration—for example, by tracking coverage metrics such as the number of failures in representing each class [1].
- **Variants and Generalizations**:Many theoretical guarantees in Active Learning rely on ideal assumptions (e.g., data separability, low noise). Extending these guarantees to more general settings—such as non-i.i.d. data, online learning, or structured outputs—remains an active area of research[1].
- **Ethical and Regulatory Aspects**:Incorporating humans into the learning loop raises concerns about accountability and transparency. Recent guidelines on trustworthy AI point out that human involvement may impact privacy and decision-making bias [52]. As such, it is crucial to develop

standards and frameworks (e.g., those proposed by Amershi et al. or EU regulations) to govern human-AI interaction in sensitive or high-stakes applications [52].

### 2.1.3 Skeptical Learning

**Skeptical learning** is an interactive machine learning paradigm in which the system exhibits skepticism toward user-provided labels. In this approach, the machine integrates multiple sources of knowledge—such as sensory data, user responses, and prior information—to assess the consistency of human annotations. In cases of conflict, it proactively queries the user to confirm or revise the provided label [67].

Rather than passively accepting labeled data, the model actively engages with the user: if it is sufficiently confident in its own prediction and suspects an annotation error, it prompts the user to re-evaluate their response[67, 13].

Infact, Skeptical Learning falls within the broader framework of human-in-the-loop machine learning: the human is not merely a passive data provider but an active collaborator in the learning process. The system autonomously formulates questions or requests confirmations based on its internal analysis; the user responds and may revise their previous input [13, 66]. At each interaction, the user retains final authority over the label but is encouraged to reflect on prior annotations and align future responses with the evolving knowledge of the model [67]. In effect, the system challenges the user when contradictions arise, initiating a rational dialogue aimed at refining shared understanding.

This dynamic supports collaborative learning: the model benefits from human-labeled data, while the user receives feedback on potential labeling mistakes. For instance, in a mobile context-recognition application, students were periodically asked to confirm their current location. If the model—relying on sensor data and campus knowledge—found the response implausible, it would rephrase the question [67]. Such interaction transcends the traditional passive-supervision paradigm by introducing an active verification phase, wherein the machine engages in meaningful dialogue with the user.

Compared to less interactive paradigms (e.g., standard human supervision), Skeptical Learning requires infrastructure capable of issuing real-time queries and sustaining adaptive dialogues. This, in turn, demands models that account for human behavior—such as response times and annotation consistency—as well as mechanisms to effectively handle erroneous or inconsistent input [67, 66]. Core Principles of Skeptical Learning are :

- **Active label verification**: The model does not take human annotations for granted. At each interaction, the predictor produces its own classification and compares it to the user-provided label. If the two diverge in a suspicious manner, the model initiates a clarification with the user [67, 66]. Rather than blindly accepting supervision, the system actively challenges the user on "dubious" examples, aiming to recover the ground truth by correcting potential labeling errors (mislabeled data) [67, 66].

- **Separate confidence measures**: The system maintains distinct confidence estimates for its own predictions and for the reliability of the human annotator. For instance, the model can track the user's past corrections or annotation mistakes [66]. When the model's confidence in its prediction is high and its trust in the current human-provided label is low, it prompts the user to re-confirm their answer [67, 66]. This dual trust mechanism (model vs. human) guides the decision to issue a follow-up query.

- **Prior knowledge and context integration**: Skeptical learning can incorporate external knowledge sources (e.g., rule bases, ontologies, logical constraints) to detect inconsistencies. For example, in personal context recognition, the system uses domain knowledge (e.g., known locations, plausible activities) alongside smartphone sensor data and user input[67]. When these three sources—data, user response, and prior knowledge—are misaligned, the system formulates a new query to resolve the inconsistency. In this setting, the model may employ logical reasoning (e.g., Description Logic) to assess the plausibility of labels[67].

- **Iterative and coherent interaction**: Learning unfolds over multiple rounds. After each user response, the system updates both its internal model and its confidence scores. If a conflict emerges (e.g., the model becomes more confident than the user), it generates a follow-up query on the same example [66]. This "*query – response – verification*" loop continues until the inconsistency is resolved or the learning process concludes. As a result, the user is actively involved in the training cycle, providing corrective feedback as needed.

These principles position Skeptical Learning as a genuinely interactive learning paradigm with advanced human supervision [67, 66]. The system learns in parallel with human decisions, challenging them when doubt arises—an approach aligned with recent trends in collaborative AI[13]. Here are the advantages of this approach:

- **Improved Annotation Quality** : Thanks to its iterative verification mechanism, Skeptical Learning actively corrects mislabeled data by filtering out inconsistent annotations. Experimental studies have shown that in real-world applications (e.g., crowdsensing via students' smartphones), the rate of labeling errors is often quite high, and a skeptical approach can significantly reduce them [67]. In other words, the final accuracy of the predictive model improves because the input data is less noisy.

- **Handling Human Uncertainty** : This approach acknowledges that users are not infallible: non-expert individuals may introduce bias or make mistakes in their responses [67, 66]. Rather than ignoring this issue, Skeptical Learning leverages human presence to address it, allowing the system to learn from subjective or partial data. The model "learns" to place greater trust in reliable users and to be more cautious with inconsistent ones.

- **Adaptability to Complex Domains** : By integrating prior knowledge and logical reasoning, Skeptical Learning can operate in domains with semantic constraints (e.g., contextual rules, relationships among attributes). This is particularly useful in pervasive or IoT applications, where raw sensor data can be semantically interpreted. For instance, knowing a campus map and students' potential activities can help detect contradictions between GPS data and user responses [67]. This facilitates model training by incorporating commonsense heuristics.

- **Active User Engagement** : In collaborative scenarios, Skeptical Learning can involve users directly in the decision-making process, thereby enhancing trust in the final system. As users see the model "questioning itself" and seeking improvement, the learning process becomes more transparent. Some studies suggest that this approach can improve user acceptance, as the system is perceived as more cooperative and aware [67].

Here are the disadvantages of this approach:

- **Increased User Burden** : Repeatedly requesting confirmations or corrections from the user can be burdensome or frustrating. If the system generates too many verification queries, users may become fatigued or respond carelessly. Additionally, situations of cognitive overload may arise: if the model is overly skeptical, it may continue to challenge even legitimate responses, potentially alienating the user [66]. Conversely, a model that is too optimistic may halt verification prematurely, missing

opportunities to correct errors (false negative issue). Striking the right balance requires sophisticated strategies and carefully calibrated threshold parameters.

- **Complex Implementation and Tuning** : Unlike passive approaches, Skeptical Learning introduces mechanisms for conflict detection and error handling that increase system complexity. It requires the definition of confidence thresholds, reconciliation strategies (e.g., merging conflicting labels), and, in some cases, frequent retraining of the model with new data. Moreover, some classical models (e.g., Random Forests) tend to become overconfident in regions far from known data, leading to pathological behaviors (e.g., never or overly frequent queries) [13]. These aspects demand careful tuning and continuous monitoring.

To avoid confusion, it is useful to draw the following distinctions with respect to other similar paradigms:

- **Active Learning**: In traditional active learning, the model selects uncertain, unlabeled examples and queries the user to provide a new label. Skeptical Learning, by contrast, assumes that a label has already been provided by the user and decides whether to challenge it arxiv.org. In other words, active learning focuses on acquiring unknown labels, while skeptical learning concentrates on verifying potentially incorrect labels. As summarized in [13], "active learning is about acquiring unknown labels, whereas skeptical learning deals with noisy labels that may require relabeling" [13]. Operationally, the two approaches are complementary: an active system queries for new data labels, while a skeptical system checks those already provided.

- **Generic Human-in-the-Loop**: Many HITL systems involve the user supervising or passively correcting the model's predictions—for example, confirming or modifying the final outputs. Skeptical Learning, instead, adopts a more proactive stance: the machine does not wait for the user to spot and correct obvious errors but actively prompts verification during training [66] In other words, instead of relegating human intervention to a post-processing phase, Skeptical Learning embeds the user's involvement throughout the learning loop.

In summary, Skeptical Learning falls within the broader spectrum of collaborative AI as an approach focused on annotation quality through joint validation. Compared to active learning, it helps mitigate the bias introduced by mislabels; compared to traditional human-in-the-loop frameworks, it enables the model

to play a more active role in generating feedback. This makes it a hybrid paradigm aimed both at managing data noise and fostering intelligent human involvement[13, 55].

### XAI in decision-making

The rapid advancements in Artificial Intelligence (AI) and Machine Learning (ML) have propelled these technologies into a wide array of application domains, ranging from autonomous vehicles to precision medicine[53]. While modern AI systems now achieve unprecedented levels of performance in complex computational tasks, their increasing sophistication often comes at the cost of transparency and interpretability [5, 40].

The empirical success of modern AI, particularly Deep Neural Networks (DNNs), stems from efficient learning algorithms and vast parametric spaces—often comprising hundreds of layers and millions of parameters—making them "black-box" models of considerable complexity[15]. This opacity stands in stark contrast to earlier AI systems, such as expert systems, which were more readily interpretable. The term black-box denotes a lack of transparency and understanding regarding how an AI model arrives at its predictions or decisions. This opacity poses a significant challenge to the adoption of AI in high-stakes applications where decisions directly impact human lives, such as healthcare, law, or defense [24].

There exists an inverse relationship between model complexity—often associated with higher performance—and transparency. This suggests a fundamental design tension: exclusively optimizing for performance frequently leads to opaque systems. However, this is not an absolute rule; simpler models can still be highly accurate when working with well-structured data. The key implication is that Explainable AI (XAI) is not merely about fostering understanding, but about mitigating this trade-off by aiming to achieve both high performance and interpretability .

Humans are generally reluctant to adopt techniques that are not directly interpretable, manageable, or trustworthy [24]. This reluctance is amplified by the growing demand for ethical AI. Transparency is increasingly demanded by various AI stakeholders, driven by the risk of producing and relying on unjustifiable or illegitimate decisions. For instance, in precision medicine, domain experts require more than a binary prediction—they need detailed explanations to support diagnostic processes. Hence, the need for transparency is not merely a technical preference, but a social and regulatory imperative. Regulations such as the European Union's General Data Protection Regulation (GDPR)[60], which includes the "right to explanation," and the U.S. Algorithmic Accountability Act are direct responses to the ethical and practical

concerns raised by black-box AI systems .

XAI advocates the development of a suite of ML techniques[31] that :

1. yield more explainable models while maintaining high learning performance
2. enable humans to understand, appropriately trust, and effectively manage emerging AI systems.

XAI draws inspiration from the social sciences, particularly the psychology of explanation [46], and is widely recognized as a critical component for the practical deployment of AI models.

The explicit reference to social sciences and psychology underscores that XAI is fundamentally a human-centered endeavor. It is not solely about technical transparency, but about bridging the gap between complex algorithms and human understanding, trust, and decision-making. Consequently, successful XAI solutions must be evaluated not only through technical metrics, but also in terms of their effectiveness in human-centered studies [38].

Below are the main objectives of XAI:

- **Interpretability** refers to a passive characteristic of a model, indicating the extent to which the model's behavior makes sense to a human observer. This feature is often used interchangeably with transparency [5] .
- **Explainability** is considered an active characteristic of a model. It denotes any action or procedure undertaken by a model with the intent to clarify or elaborate on its internal functions [29].
- **Comprehensibility** (or intelligibility) refers to a model's ability to allow a human to understand its behavior—how it works—without requiring insight into its internal structure or the algorithmic processes it uses to process data. In other words, comprehensibility indicates the model's ability to represent the learned knowledge in a human-understandable form [29].
- **Transparency** denotes the inherent clarity of a model; a model is transparent if it is, by itself, understandable to a human [40].

Some models are inherently interpretable, such as linear regression or rule-based models, and are thus naturally closer to the notion of interpretability. Others, particularly complex or "black-box" models like deep neural networks, require the application of active explainability techniques—such as LIME or SHAP—to make their decisions understandable. This distinction implies that the choice of a specific XAI strategy

should be guided by both the intrinsic transparency of the model and the desired level of understanding for the target audience [5].

Transparency refers to a model's inherent comprehensibility, whereas comprehensibility measures the extent to which a human can understand a decision made by the model.

In addition to interpretability and transparency, XAI aims to satisfy a broader range of objectives, including:

- **Reliability**: A primary goal of XAI, even though not all reliable models are necessarily explainable .
- **Causality**: Aims to identify causal relationships that go beyond the statistical correlations typically uncovered by ML models.
- **Transferability**: Clarifies the boundaries of a model to enhance its reusability in different contexts or tasks.
- **Informativeness**: Supports decision-making processes by providing meaningful insights into the addressed problem .
- **Fairness**: Tackles bias and promotes the ethical use of algorithms.
- **Accessibility**: Ensures that non-technical users can engage with and understand the model's behavior.
- **Interactivity**: Enables a dialog between users and the model, fostering dynamic exploration and validation of decisions.
- **Privacy Awareness**: Evaluates potential privacy violations or trade- offs arising from the model's representations and outputs.

### Taxonomy of XAI Approaches

Transparent models inherently convey a certain degree of interpretability. These are often referred to as intrinsically interpretable methods or ante-hoc XAI techniques [5]. Such models are intentionally designed to be understandable during their development and training phases, making their internal mechanisms explicit and accessible.

To navigate the diverse landscape of XAI, it is essential to establish a clear categorization of methodologies. The concept of levels of transparency—including simulatability, decomposability, and algorithmic transparency [5]—reveals that even "transparent" models are not uniformly interpretable. For instance, a small decision tree may be fully simulative, while a large one may only offer algorithmic transparency.

- **Linear and Logistic Regression**: These models assume a linear relationship between input features and the output variable. Their predictions are based on a weighted sum of the inputs, and the weights directly reflect feature importance. While their simplicity ensures high interpretability, their performance can be limited in the presence of complex or non-linear data structures.

- **Decision Trees**: Decision trees represent hierarchical structures where each path from the root to a leaf node corresponds to a logical sequence of decisions. They are inherently transparent and support both non-linear relationships and feature interactions. However, as tree complexity increases, interpretability diminishes, often requiring post-hoc explanation methods.

- **K-Nearest Neighbors**: The KNN algorithm classifies instances based on the classes of the nearest neighbors, relying on similarity in feature space. While intuitive and aligned with human analogical reasoning, the model's transparency depends on the comprehensibility of the distance metric and the dimensionality of the feature space.

- **Rule-Based Learners**: Explicit Knowledge Representation These systems generate rule sets (e.g., "if-then" conditions) to explain model decisions. Fuzzy rule-based systems improve intelligibility through natural language representations. Nonetheless, there is a trade-off between rule complexity and user comprehensibility, especially as the number or length of rules increases.

- **Generalized Additive Models (GAMs)**: Interpretable Yet Flexible Structures GAMs extend linear models by allowing non-linear functions for each variable while maintaining additive structure. This balances model flexibility with interpretability, enabling users to visualize the effect of each variable independently.

- **Bayesian Models**: Probabilistic Relationships and Graphical Transparency Bayesian networks represent conditional dependencies through probabilistic graphical models. These models provide an interpretable structure for understanding relationships among variables, and they can be simulative or decomposable depending on their complexity.

### 2.1.4 Post-hoc Explainability Techniques for Black-Box Models

When machine learning models do not meet the criteria for transparency, it becomes necessary to develop and apply separate methods to explain their decisions. This is the purpose of post-hoc explainability techniques. These techniques are employed after AI models have been trained and deployed, focusing on communicating understandable information regarding how a pre-existing model produces its predictions[5]. Model-agnostic techniques are designed to be applicable to any type of model, enabling the extraction of information from its prediction procedure, regardless of its internal processing or representations.[1] These techniques offer several advantages, including:

- Compatibility with various ML models.
- Different forms of explanation.
- Flexible representations, such as using words in a text classifier to support the explanation.

Model-specific approaches, on the other hand, provide explanations based on the internal operational structure and design of the model, and may not be applicable to other models with different architectures. Although less transferable, they can offer more comprehensive insights into how a particular model functions. It is also important to make the following distinction: Local explanations focus on the predictions for individual instances, providing insights useful for examining the model's behavior with respect to specific outcomes. In contrast, global explanations offer an overview or a comprehensive description of the model's overall behavior, requiring knowledge of the input data, the algorithm, and the trained model. Global methods illustrate the model's general behavior (e.g., Partial Dependence Plots, Accumulated Local Effects), whereas local methods explain specific instances (e.g., LIME, SHAP). Here are some of the most well-known post-hoc techniques [5]:

- **Perturbation-Based Techniques** (e.g., LIME, SHAP) [41]:Perturbation-based methods operate by modifying the input data (e.g., removing, masking, or adding noise) and observing the resulting changes in the output to infer the importance of features. Prominent examples include LIME and SHAP. LIME constructs locally linear models around individual predictions, providing explanations for single instances. SHAP employs Shapley values from game theory to fairly distribute feature importance, offering both local and global explanations.Both LIME and SHAP, as perturbation-based

methods, are computationally expensive because they require multiple model inferences on perturbed inputs.

- **Gradient-Based Techniques** (e.g., Saliency Maps, Integrated Gradients) :Gradient-based methods compute the gradients of the model's prediction with respect to its input features, reflecting the sensitivity of the output to changes in the input. Common examples include saliency maps, Layer-wise Relevance Propagation (LRP), Class Activation Mapping (CAM), and integrated gradients. Saliency maps highlight important regions in an image. LRP propagates relevance scores backward through the network layers. CAM identifies the input regions on which convolutional neural networks (CNNs) focus. Integrated gradients assign importance based on the path integrals of the gradients.

- **Feature Relevance Estimation**:These methods clarify the model's functioning by calculating a relevance score for each variable, quantifying their influence or sensitivity on the output. SHAP is a key example of this approach. Feature relevance estimation is widely used and focuses on identifying the most influential factors in a model's decision-making process.

- **Visual Explanation Techniques**:Visual explanation techniques aim to visualize the model's behavior, often combined with dimensionality reduction methods. Examples include Partial Dependence Plots (PDP), Individual Conditional Expectation (ICE) plots, and Accumulated Local Effects (ALE) plots.

- **Hybrid Transparent and Black-Box Methods**: Combining Interpretability and Performance:Hybrid approaches integrate symbolic representations (e.g., logical statements, knowledge bases) with subsymbolic ones (e.g., neural networks). This integration can enhance both explainability and performance while increasing robustness. Examples include neural-symbolic systems, knowledge base-augmented systems, deep formulations of classical ML models (e.g., deep Kalman filters), relational reasoning, and case-based reasoning systems coupled with deep neural networks (DNNs).

### 2.1.5 Counterfactuals

Among the multitude of explanation techniques, counterfactual explanations (CFEs)[26] offer a unique and intuitively appealing approach to understanding AI decisions. Unlike methods that focus on why a decision was made—typically in terms of feature importance—CFEs highlight what needs to change in

the input to alter the outcome, thereby providing actionable insights. A counterfactual, or counterfactual explanation, is a hypothetical instance that outlines the minimal changes required in the input space of a model to alter its prediction toward a desired outcome [2, 61]. For example, if an AI system denies a loan to a customer, a counterfactual explanation might state: "You would have received the loan if your income had been $10,000 higher and you had no other outstanding debts with the bank" [26]. This type of explanation provides a direct and intuitively understandable "*what-if*" scenario, enabling users to comprehend which specific factors would have needed to differ in order to achieve an alternative outcome.

A crucial aspect that distinguishes counterfactuals from other forms of explanation in XAI is their inherent faithfulness to the underlying model. While certain explanation methods—such as LIME (Local Interpretable Model-agnostic Explanations)—rely on surrogate or proxy models to approximate the behavior of the black-box model, thereby introducing a degree of approximation and potential "misrepresentation" of the model's true behavior, counterfactual explanations operate differently [26]. They provide direct outputs from the algorithm itself, deriving the necessary modifications through direct queries to the black-box model. This approach ensures that counterfactual explanations remain faithful to the actual behavior of the model, as they do not depend on intermediate interpretations that may distort the decision logic. For applications in high-stakes domains, where precision and fidelity to the model's behavior are critical, counterfactuals can be considered a more trustworthy and preferable form of explanation compared to methods that trade off fidelity for broader interpretability. The primary purposes of counterfactual explanations can be summarized as follows:

- **Building trust in AI systems**. In high-stakes domains such as finance, healthcare, or criminal justice, the opacity of machine learning models can lead to a lack of public trust. By providing clear and actionable feedback, counterfactuals enhance transparency and foster confidence in algorithmic decisions, enabling individuals to understand not only why a particular decision was made, but also how a different outcome might be achieved .

- **Actionable recourse**. Counterfactual explanations offer concrete and practical suggestions for individuals seeking to change their circumstances. They point to specific, feasible modifications in input features that would result in a different (often more desirable) prediction.

- **Facilitating model understanding**. By identifying the minimal changes required to alter a prediction, counterfactuals reveal which features are most influential for a given instance. This capability

is essential for model debugging, detecting algorithmic bias, and ensuring fairness in outcomes.

It is also important to highlight several key advantages of counterfactual explanations:

- **Model-agnosticism**. Many counterfactual approaches are designed to be applicable across various types of machine learning models—whether differentiable (e.g., neural networks) or non-differentiable (e.g., decision trees)—without requiring access to internal gradients. This broad applicability significantly extends their utility for explaining a wide range of heterogeneous black-box systems.
- **Ability to incorporate feature constraints and real-world feasibility**. Real-world datasets often contain immutable features (e.g., gender or race) or attributes with constrained value ranges (e.g., age, which can only increase). Counterfactual methods can integrate such constraints, ensuring that the generated explanations are both realistic and actionable within the bounds of the real world. This helps avoid producing unrealistic or infeasible suggestions.

**Formal definitions**

This section presents the formal definitions of counterfactuals to facilitate a deeper understanding based on the formulations provided in [26]

1. (Counterfactual explanation) : Given a classifier $b$ that outputs the decision for an instance $x$, a counterfactual explanation consists of an instance $x'$ such that the decision for $b$ on $x'$ is different from $y$, i.e. $b(x') \neq y$, and such that the difference between $x$ and $x'$is minimal. Instances $x$ and $x'$ consist of a set $\{x_1, x_2, ..., x_m\}$ of $m$ attribute-value pairs $x_i = (a_i, v_i)$, where $a_i$is a feature (or attribute) and $v_1$ is a value from the domain of $a_i$
2. (Counterfactual explainer): A counterfactual explainer is a function $f_k$ that takes as input a classifier $b$, a set $X$ of known instances, and a given instance of interest $x$, and with its application $C = f_k(x, b, X)$ returns a set $C = \{x'_1, ..x'_h\}$ of $h <= k$ of valid counterfactual examples where $k$ is the number of counterfactuals required.

Most of the counterfactual explainers in the literature are designed as $f_1$ function to return a single valid counterfactual. If $C = \emptyset$, i.e. $h = 0$, it means that the explainer was not able to find any valid counterfactual.

**Proprieties of counterfactual explanations**

This section outlines the desirable properties for counterfactual explanations based on the formulations provided in [26] :

- **Validity**: A counterfactual $x'$ is valid iff it actually changes the classification outcome with respect to the original one i.e $b(x') \neq b(x)$
- **Sparsity:** This property refers to the requirement that the counterfactual $x'$ involves the smallest possible number of changes relative to the original instance $x$. Minimality is often quantified by the number of differing attribute–value pairs between $x$ and $x'$.

  A "sparse" counterfactual is easier to understand and potentially simpler for the user to act upon.
- **Similarity** : A counterfactual $x'$ should be as similar as possible to the original instance $x$. This similarity is typically measured using a distance function $d(x, x')$, with the aim of keeping $d(x, x') < \epsilon$, where $\epsilon$ is a predefined threshold. Proximity ensures that the counterfactual remains "close" to the original instance in the feature space, making it more intuitive and relevant
- **Plausibility**: A counterfactual $x'$ is considered plausible if its feature values are consistent with those observed in a reference population $X$. This implies that $x'$ should not be an outlier and should respect the existing correlations among features in real-world data. Plausibility is crucial for increasing trust in the explanation; an unrealistic or inconsistent counterfactual would be difficult to accept.
- **Discriminative Power**: A counterfactual $x'$ should exhibit high discriminative power, helping users understand the reasons behind the change in decision outcome. The idea is that, by observing $x$ and $x'$, a human should be able to infer why $b(x) = y$ and $b(x') = y'$, based on the differences $\delta_{x,x'}$.
- **Actionability**: All differences between $x$ and $x'$ must pertain exclusively to features that can be changed by the user (e.g., income, but not age, gender, or race). A counterfactual is actionable if all its modifications involve only mutable features. This property is fundamental for providing meaningful and practical recourse.
- **Causality**: A counterfactual $x'$ should respect known causal relationships among features. For example, if increasing loan duration typically leads to a higher interest rate, the counterfactual should

reflect this dependency. This property is closely related to plausibility and ensures that suggested changes are consistent with the causal structure of the domain.

- **Diversity**: When a set of counterfactuals $C = \{x'_1, \ldots, x'_k\}$ is provided, it is desirable for them to be diverse. While each $x'_i \in C$ should be minimal and similar to $x$, the differences among all counterfactuals in $C$ should be maximized. This offers users multiple paths and options to achieve a different outcome, increasing the likelihood of identifying a feasible solution.

**Properties of counterfactual explainers**

In this section, the desirable properties that counterfactual explainers should satisfy are presented [12, 27, 26]:

- **Efficiency**:An explainer $f$ should return the set $C$ of counterfactuals fast enough to ensure that they can be used in real life applications.
- **Stability** : Given two similar instances $x$and $x'$ obtaining the same classification from the classifier $b$, i.e. $y = b(x_1) = b(x_2)$, then an explainer $f$ should return two similar set $C_1, C_2$ of counterfactuals
- **Fairness**: A counterfactual explainer is fair if, given a record $x$, any counterfactual explanation $x'$ for $x$ is valid both in the "actual world" and in the "counterfactual world" when in $x'$ can also be applied changes leading it to belong to a different demographic group. For instance, suppose that gets the loan accepted by reducing its duration to 10 years from 15 years. The explainer is fair if has the loan accepted, i.e., is still a valid counterfactual, also changing, e.g., the ethnicity

**The Contribution of Cognitive Psychology to the Study of Counterfactuals and Their Relevance for XAI**

Cognitive psychology has played a pivotal role in the study of counterfactuals[14], offering a rich and nuanced experimental foundation on how individuals understand, construct, and reason with hypothetical scenarios. This body of knowledge is particularly relevant in an era where transparency in artificial intelligence—especially within the field of XAI —has become a key priority. Integrating insights from cognitive psychology into AI systems can make them not only more interpretable, but also more aligned with human modes of reasoning. One of the main contributions of cognitive psychology lies in the understanding of

the structure of counterfactuals generated by humans. Individuals tend to construct two main types of counterfactuals:

- **Additive counterfactuals** introduce new information into the mental simulation, encouraging creativity and innovative problem-solving. For example: "*If the car had detected the pedestrian earlier, the passenger would not have been injured.*"
- **Subtractive counterfactuals** remove elements from the observed reality, fostering more logical and deductive reasoning: "*If the car had not swerved, the passenger would not have been injured.*"

Beyond structure, individuals tend to generate "upward" counterfactuals—imagining better outcomes—more frequently than downward ones. This tendency is driven by a desire for improvement and learning, though it may also intensify negative emotions such as regret. Conversely, downward counterfactuals—imagining worse outcomes—may have a consolatory effect, evoking emotions such as relief or gratitude, but risk diminishing the motivation to learn from mistakes. Another crucial area where counterfactuals play a central role is **causal explanation**. Cognitive psychology shows that counterfactuals strengthen causal judgments: if one believes an outcome would have been different given an alternative action, the perceived causal link between the events becomes stronger. Conversely, in the case of semi-factuals—when the outcome would have been the same regardless—the event is perceived as inevitable, reducing the attribution of causality.

This dynamic is also reflected in the assignment of blame. Counterfactuals serve as a tool for assessing responsibility, especially in legal or moral contexts[37]. However, there is a general reluctance to construct counterfactuals that violate shared social norms: for instance, if a decision aligns with an ethical code, people are less likely to mentally "undo" it[45]. This suggests that counterfactual reasoning is deeply influenced by cultural and social contexts, rather than governed solely by abstract logical principles.

People do not modify reality arbitrarily when constructing counterfactual scenarios. Instead, they focus on critical points—called "fault lines"[34] —which are decisive junctions in the causal chain of events. Common criteria for selecting what to change include:

- **Exceptional events**: atypical behaviors are more likely to be altered to more typical ones.
- **Controllability**: changes tend to be made to actions under human control.
- **Actions rather than omissions**: people more often modify actions rather than inactions.

- **Recency**: more recent events in the temporal sequence are more likely to be changed.
- **Plausibility**: counterfactuals must appear realistic, even if not based on statistical probabilities.

These criteria suggest that counterfactuals are not mere fantasies, but refined cognitive tools oriented toward utility.

A key feature of counterfactuals is that they give rise to numerous and profound inferences. When people interpret a counterfactual like "*If the car had continued straight, the pedestrian would have been killed,*" they naturally derive both the expressed conjecture and the implicit facts (that the car did not continue straight, and the pedestrian is safe)[58].

This reasoning process is grounded in the construction of multiple mental models: one representing reality as it occurred, and another representing the hypothetical alternative. This dual simulation is cognitively demanding but enables much richer reasoning, such as facilitating more frequent modus tollens inferences compared to factual conditionals.

As a result, counterfactuals serve as powerful tools for explaining, justifying, and planning. Precisely because they activate alternative mental models, counterfactuals can be strategically leveraged in the domain of XAI. If an AI system can generate counterfactual explanations that resemble those humans naturally produce, users are more likely to understand and trust its decisions.

In summary, cognitive psychology has shown that counterfactuals are not merely imaginative exercises, but sophisticated tools for reasoning, explanation, and learning. Their structure, content, causal function, and inferential potential reflect deeply human cognitive processes. Integrating these insights into AI systems—especially in the context of XAI—not only enhances their interpretability, but also brings artificial intelligence closer to human reasoning, fostering more natural and effective human-machine interaction.

## 2.2 Background

This section introduces the learning setting and the principal tools adopted in this work. The task is framed as a *continual learning* problem in a supervised setting, where data is presented over time in the form of sequential tasks. Formally, let:

- $\mathcal{X} \subseteq \mathbb{R}^d$: input space, with $d$ the dimensionality of the features.
- $\mathcal{Y} = \{1, 2, \ldots, C\}$: label space, with $C$ total classes.

- A task $\mathcal{T}_t = \{(x_i, y_i)\}_{i=1}^{n_t}$, where $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}_t$, is a collection of labeled samples received at time step $t \in \{1, \ldots, T\}$, and $\mathcal{Y}_t \subseteq \mathcal{Y}$ represents the (possibly disjoint) subset of labels seen in task $\mathcal{T}_t$.
- The learner incrementally receives tasks $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_T$, without access to previous task data during training on $\mathcal{T}_t$ (non-i.i.d. setting).

The objective is to train a predictive model $f_\theta$ capable of integrating new information from each $\mathcal{T}_t$, while preserving performance on previously seen tasks, in the absence of their original data.
This section then provides an overview of the tools and technologies employed in the development of this work. In particular, the online learning models used in the experiments are highlighted, including:

- the Extremely Fast Decision Tree (EFDT),
- the Hoeffding Adaptive Tree (HAT),
- ensemble methods such as Online Bagging, AdaBoost, and ADWIN Boosting.

Following the description of these models, the FRANK algorithm is introduced. Finally, various frameworks for counterfactual explanation generation are discussed Such as DiCE, Growing Spheres, LORE and a prototype-guided counterfactual generation method implemented using the `alibi` library.
Subsequent subsections will delve into each of these tools in more detail, focusing on their relevance to continual learning and explainability in human–machine interaction contexts.

### 2.2.1 HoeffdingAdaptiveTreeClassifier

Among the most innovative models in this domain is the **Hoeffding Adaptive Tree (HAT)** [10], an incremental decision tree that combines statistical rigor with dynamic adaptation mechanisms. Before delving into the specifics of this model, it is necessary to introduce the Hoeffding Trees, also known as Very Fast Decision Trees (VFDT).
The latter, introduced in [20], is the first algorithm designed to build incremental decision trees on infinite data streams. Its main features are as follows:

- A statistical inequality that determines how many examples are necessary to confidently select the optimal attribute for a split [20]:

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$$

. where : $R$is the range of the variable (e.g., maximum difference between values);$\delta$ is the error probability;$n$ is the number of observed examples.

- Incremental counters: each node maintains statistics $n_{ijk}$ (counts for attribute i, value j, class k) updated in real time.

In particular , a split is executed when the observed difference between the best and second-best attribute's heuristic measure $\Delta G$ exceeds this $\epsilon$ value. However,standard Hoeffding Trees exhibit several limitations that may hinder their performance and practical utility in dynamic or long-term learning scenarios[4]:

1. They rely on the assumption that the underlying data distribution remains unchanged over time, which may not hold in many real-world applications.

2. Due to their incremental nature, these models tend to grow indefinitely, often introducing multiple splits on the same feature. This uncontrolled expansion increases structural complexity without necessarily improving predictive accuracy.

3. The resulting complexity can reduce model transparency and lead to overfitting, undermining the clarity and interpretability typically associated with these models.

Several variants of the proposed algorithm exist; one such variant is ADWIN. It maintains a window $W$ of the most recent data, which is dynamically split into two sub-windows: $W_0$ (older data) and $W_1$ (newer data). A statistical test is then applied to assess whether the difference between their means $\mu_{W_0}$ and $\mu_{W_1}$ is statistically significant: :

$$|\mu_{W_0} - \mu_{W_1}| > 2\sqrt{\frac{1}{2m} \ln \frac{4|W|}{\delta}}, \quad m = \frac{2}{\frac{1}{|W_0|} + \frac{1}{|W_1|}}$$

To identify changes in the data distribution, ADWIN continuously divides its current window into two sub-windows, denoted as $W_0$ and $W_1$, and performs a statistical comparison of their respective means. If the absolute difference between these means exceeds a predefined threshold—calculated based on a user-defined significance level $\delta$—this is interpreted as evidence that the distribution has changed, thereby indicating the occurrence of concept drift. When such a change is detected, the old

This approach has been shown to produce a low rate of false positives and false negatives[10].

This mechanism allows the algorithm to guarantee the following properties[10]:

- Detect changes in the accuracy of the tree.
- When a change occurs, HWT initializes an alternative tree rooted at the affected node. After a stabilization period, the alternative tree replaces the original one if it is more accurate.
- No fixed window size is needed. The window size adapts automatically to the data's rate of change, using a compressed version of the exponential histogram technique [17].
- The only required parameter is a confidence bound $(\delta)$, indicating how confident the user want to be in the algorithm's output.

This means the algorithm maintains a window of length $W$ using only O($\log W$) memory and O($\log W$) computational complexity.

At this point, is possible to introduce the Hoeffding Adaptive Trees (HAT): These use adaptive mechanisms to handle concept drift without requiring fixed parameters such as the temporal window size. Specifically, static counters are replaced: each node uses adaptive estimators to track the statistics $n_{i,j,k}$.

There are three main variants depending on the estimator used:

- **HAT-INC** : employs a linear incremental estimator
- **HAT-EWMA** : employs an exponential weighted moving average
- **HAT - ADWIN** : uses an ADWIN estimator. Since this is a change detector, any variation in attribute-class statistics at the node is automatically flagged.

The main advantages over traditional models are as follows:

- No need to define parameters like temporal window size
- Only essential statistics are maintained, significantly reducing memory usage compared to fixed-window models
- Thanks to ADWIN integration, HAT offers guarantees on error reduction following a concept change

### 2.2.2 Extremely Fast Decision Tree

The **Extremely Fast Decision Tree (EFDT)** [42] model introduces a novel incremental learning algorithm for decision trees, named Hoeffding Anytime Tree (HATT), which significantly improves statistical

efficiency compared to the Hoeffding Tree (HT) algorithm[20], widely regarded as the state-of-the-art approach for learning decision trees from data streams.

The implementation of HATT has been shown to achieve superior prequential accuracy (a metric for evaluating predictive performance over data streams) compared to the Very Fast Decision Tree (VFDT) implementation across many of the largest classification datasets in the UCI repository.[42]

As previously discussed, HT employs the Hoeffding Bound, a probabilistic inequality, to determine when a split at a tree node is statistically justified, based on the difference in Information Gain between the two best attributes. However, HT adopts a conservative approach: it waits until it is confident it has identified the best possible split before performing it, and never revisits that decision. While computationally efficient, this can result in slow convergence to the optimal decision tree, particularly when data are complex or when many attributes have similar information gain values.

HATT proposes a different strategy: instead of waiting until the best possible split is identified, it performs a split as soon as there is sufficient evidence that a split would be useful, and later re-evaluates that decision if another attribute proves to offer greater information gain. This approach allows faster learning from a stationary distribution and convergence toward the asymptotic decision tree (i.e., the tree that would be built by a batch algorithm on an infinite dataset).

In practice, the Hoeffding Bound [20] is used to assess whether the merit of a split based on the best attribute exceeds that of either no split or the current split. If no split exists at a node, HATT performs a split as soon as the information gain of the best candidate attribute is significantly greater than zero with high confidence[42]. Subsequently, HATT may revise this decision if evidence arises suggesting that a different attribute would yield better predictive performance. HATT (Hoeffding Anytime Tree) maintains an adaptive decision tree structure by continuously monitoring and re-evaluating the quality of splits at each internal node. At every step, it calculates the information gain for all available attributes at a given node and compares the best candidate against the current splitting attribute. If the candidate attribute demonstrates a statistically significant improvement, the node is updated accordingly. Additionally, the algorithm integrates a pruning strategy based on statistical insignificance, helping to mitigate overfitting. This dynamic re-evaluation process allows HATT to progressively refine its structure as new data becomes available.

The main steps of the algorithm are as follows:

1. Calculate the information gain $G(X)$ for all available attributes at the node.

2. Identify the attribute $X_a$ with the highest current information gain.

3. Compute the difference $G(X_a) - G(X_{\text{current}})$, where $X_{\text{current}}$ is the attribute currently used for splitting at the node.

4. If $G(X_a) - G(X_{\text{current}}) > \epsilon$ (the Hoeffding bound) and $X_a \neq X_{\text{current}}$, replace the internal node with a new one that splits on $X_a$.

5. If $X_a$ corresponds to the null split—indicating that no attribute offers a statistically significant advantage over predicting the majority class—replace the internal node with a leaf, effectively pruning the subtree.

6. Repeat the above steps continuously as new instances arrive, allowing the tree to adapt over time and converge toward a more optimal structure.

### 2.2.3 SGT Classifier

"**Stochastic Gradient Trees**"[25] propose an innovative method for the incremental learning of decision trees based on stochastic gradient information. While traditional decision trees (e.g., CART, Hoeffding Trees) require task-specific algorithms and do not leverage gradient-based optimization [20], SGTs combine the advantages of decision trees (interpretability, efficiency)[22] with the flexibility of deep learning methods (adaptability to various loss functions).

The key idea is to use the gradient and Hessian of the loss function to guide both the split selection and the update of leaf nodes, in a manner analogous to Stochastic Gradient Descent (SGD). This approach eliminates the need to design ad hoc heuristics for each task, making SGTs adaptable to different learning problems simply by changing the loss function.

Let $(x_t, y_t)$ be the feature–label pair observed at time t. The algorithm maintains a tree $f_t$that assigns a score value to each instance x. At each time step t, the goal is to minimize the expected loss conditioned on the most recent observations. A twice-differentiable loss function is defined and it measures the error between the true label $\ell(y, \hat{y})$ y and the prediction $\hat{y} = \sigma(\text{f\_t(x)})$ where $\sigma$ is a link function.

The ideal objective is to minimize the expected value $\mathbb{E}[\ell(y, \hat{y})]$

In an incremental learning setting, assuming the tree was last updated at time step $r$, the expectation is estimated using the empirical average over the observations from time $r+1$ to t. Specifically, the expected loss is approximated as follows:

$$\mathbb{E}[\ell(y, \hat{y})] \approx \frac{1}{t-r} \sum_{i=r+1}^{t} \ell(y_i, \hat{y}_i)$$

where $\hat{y}_i$ is the current prediction $f_t(x_i)$ for $x_i$. To minimize this quantity incrementally, at time $t$ is possible to consider a modification of the tree represented by a function $u : \mathcal{X} \to \mathbb{R}$. The update consists in moving from $f_t$ to:

$$f_{t+1} = f_t + u^*$$

where $u^*$ is obtained as the solution to the following optimization problem:

$$f_{t+1} = f_t + \arg\min_u \left[L_t(u) + \Omega(u)\right]$$

where : $L_t(u)$: is the loss accumulated on the current data, taking into account the modification $u$; $\Omega(u)$ is a regularization term that penalizes the complexity of the change. In detail :

$$L_t(u) = \sum_{i=r+1}^{t} \ell(y_i, f_t(x_i) + u(x_i))$$

$$\Omega(u) = \gamma|Q_u| + \frac{\lambda}{2} \sum_{j \in Q_u} v_u(j)^2$$

Here:

- $Q_u$ is the set of new leaf nodes introduced by $u$
- $v_u(j)$ is the difference between the new prediction of node $j$ and the prediction of its parent.
- $\gamma|Q_u|$ penalizes the number of newly added nodes.
- $\frac{\lambda}{2} \sum_{j \in Q_u} v_u(j)^2$ forces the predicted values of the new nodes to remain small (L2 regularization).

To simplify optimization, a second-order Taylor [22] expansion of the loss is used. The expression The

expression

$$\ell\big(y_i, f_t(x_i) + u(x_i)\big)$$

is expanded as a Taylor series up to the second order around $u = 0$:

$$\ell\big(y_i, f_t(x_i) + u(x_i)\big) \approx \ell\big(y_i, f_t(x_i)\big) + g_i u(x_i) + \frac{1}{2} h_i u(x_i)^2,$$

where :

$$g_i = \frac{\partial}{\partial f_t(x_i)} \ell\big(y_i, f_t(x_i)\big), \quad h_i = \frac{\partial^2}{\partial f_t(x_i)^2} \ell\big(y_i, f_t(x_i)\big),$$

are respectively the first and second derivatives of the loss function with respect to the current score $f_t(x_i)$, this yields:.

$$L_t(u) \approx \sum_{i=r+1}^{t} \left[ \ell(y_i, f_t(x_i)) + g_i u(x_i) + \frac{1}{2} h_i u(x_i)^2 \right]$$

Discarding the constant term:

$$\Delta L_t(u) = \sum_{i=r+1}^{t} \left[ g_i u(x_i) + \frac{1}{2} h_i u(x_i)^2 \right]$$

This expression (obtained through a second-order approximation) describes how the loss changes as a function of the update $u$. For each potential split of a leaf node $q$, the function $u$ is defined in such a way that it modifies only the examples that would fall into the newly created child nodes. Formally, $u$ is associated with a mapping $q_u : X \to Q_u \subseteq \mathbb{N}$

Each potential split is expressed as:

$$u(x) = \begin{cases} v_u(q_u(x)), & \text{if } x \in \text{Domain}(q_u) \\ 0, & \text{otherwise} \end{cases}$$

Here,$v_u(j)$ represents the prediction increment for the new node with identifier j and $\text{Domain}(q_u)$ enotes the set of instances that would fall into the newly created child nodes. With $I_j^u = \{i : q_u(x_i) = j\}$, is possible get:

$$\Delta L_t(u) = \sum_{j \in Q_u} \left[ \left( \sum_{i \in I_j^u} g_i \right) v_u(j) + \frac{1}{2} \left( \sum_{i \in I_j^u} h_i \right) v_u(j)^2 \right]$$

To find the best value of $v_u^*(j)$, the following steps must be follwed :

- Adding the regularizer:

$$\left(\sum_{i \in I_j^u} g_i\right) v_u(j) + \frac{1}{2}\left(\sum_{i \in I_j^u} h_i + \lambda\right) v_u(j)^2$$

- Setting the derivative to zero:

$$0 = \sum_{i \in I_j^u} g_i + \left(\lambda + \sum_{i \in I_j^u} h_i\right) v_u(j)$$

From which $(v_u(j))$can be immediately solved as:

$$v_u^*(j) = -\frac{\sum_{i \in I_j^u} g_i}{\lambda + \sum_{i \in I_j^u} h_i}$$

This expression provides the new optimal value that leaf $j$ should predict in order to minimize the loss (taking regularization into account).

**Splitting on Numeric Attributes**

When an attribute is nominal, each distinct value leads to the creation of a separate branch (multi-way split). For numerical attributes, however, the authors adopt an equal-width binning discretization approach. Specifically, an initial sample of data is collected to estimate the minimum and maximum of each numerical attribute. Then, a fixed number of bins is chosen to divide the interval into equally sized subintervals. Future values that fall outside the estimated range are "clipped" to the known extreme values. Once the attribute has been discretized, all possible binary splits at bin boundaries are considered, treating the attribute as ordered. In other words, each candidate split for a numerical attribute is of the form "attribute <= threshold", where the threshold corresponds to a bin boundary This approach allows for efficient handling of numerical data streams with minimal overhead, by reducing the problem of continuous values to a discrete space of possible thresholds [21].

**Determining when to Split**

When evaluating a candidate split, the method must decide whether there is sufficient statistical evidence to perform it. In traditional Hoeffding Trees, is used the Hoeffding concentration bound: it derives a threshold $\epsilon$ such that, with probability $1-\delta$, the true expected value of a given split-quality index exceeds its empirical mean by at most $\epsilon$.Concretely, for a sample mean $\bar{X}$ of i.i.d. variables bounded in the interval $[0, R]$, is possible to obtain :

$$\mathbb{E}[X] > \bar{X} - \varepsilon \quad \text{where } \varepsilon = R^2\sqrt{\frac{\ln(1/\delta)}{2n}}$$

However, applying **Hoeffding**'s bound requires knowing a range $R$ on the quantities $\Delta\ell_i$ (or on their gradients/Hessians), and such bounds are not generally finite for many continuous losses. To overcome this limitation, SGT employs instead a Student's t-test on the mean loss change. Specifically, let

$$\Delta\ell_i = \ell\big(y_i, f_t(x_i) + u(x_i)\big) - \ell\big(y_i, f_t(x_i)\big).$$

be the change in loss under the candidate split $u$. Over the $n$ examples routed to the current node, is possible to compute the sample mean

$$\bar{L} = \frac{1}{n}\sum_{i=1}^{n}\Delta\ell_i$$

and, under the null hypothesis of no loss reduction, assume

$$\mathbb{E}[\bar{L}] = 0.$$

. Then form the t-statistic

$$t = \frac{\bar{L} - \mathbb{E}[\bar{L}]}{s/\sqrt{n}} \;=\; \frac{\bar{L}}{s/\sqrt{n}}.$$

where $s$ is the sample standard deviation of the $\Delta\ell_i$. A $p$-value is obtained from the Student's t-distribution with $n-1$ degrees of freedom. If the p-value is below $\delta$ (the chosen confidence level), is possible to conclude that the mean reduction $\bar{L}$ is statistically significant and performs the split. By the central limit theorem, $\bar{L}$ tends toward normality even if the individual $\Delta\ell_i$ are not themselves normally distributed. To carry out

the t-test, one must estimate the sample variance $s^2$ Is possibile derive the variance of

$$\Delta\ell_i = G_i\, v_u(j) \;+\; \tfrac{1}{2}\, H_i \left(v_u(j)\right)^2$$

where $G_i$ and $H_i$ are the random gradient and Hessian components on the samples. Neglecting the dependence on

$v_u(j)$ (which is computed a posteriori)and using standard variance properties, one obtains :

$$\mathrm{Var}(\Delta\ell_i) = v_u(j)^2\, \mathrm{Var}(G_i) \;+\; \frac{v_u(j)^4}{4}\, \mathrm{Var}(H_i) \;+\; v_u(j)^3\, \mathrm{Cov}(G_i, H_i).$$

which can be estimated incrementally using, e.g. Welford's or Bennett et al.'s algorithms [62, 8]. For efficiency, the t-test is not performed after every new example but only periodically (e.g. every 200 observations), as is common in incremental-tree methods. In essence, SGT obtains a split criterion that directly reflects the expected loss reduction without requiring a priori bounds on the gradients. The use of the Student

t-test allows one to decide, in an adaptive and statistically principled way, when a split is justified, while preserving the generality of the method for any twice-differentiable loss function.

### 2.2.4 Online Bagging and Boosting

In the context of supervised learning, ensemble algorithms such as Bagging (Bootstrap Aggregating) and Boosting are among the most effective techniques for improving classification performance by combining multiple weak learners. However, both were originally designed to operate in batch mode, assuming that the entire dataset is available in memory. This assumption becomes a limitation in scenarios where data arrive as a stream or cannot be stored entirely—such as in real-time applications or big data environments. To overcome these limitations, Oza and Russell [50] proposed online versions of Bagging and Boosting, which process input data one instance at a time and update the model incrementally, without requiring access to past examples. These algorithms rely on probabilistic adaptations of the original mechanisms, while preserving the fundamental theoretical properties of their batch counterparts.

**Online Bagging**

In the batch version of Bagging, each model in the ensemble is trained on a subset of the dataset obtained through sampling with replacement. Formally, given a dataset $T$ of $N$ examples and a base learning algorithm $L_b$ , the algorithm generates $M$ models, each trained on a bootstrap sample of $T$.

For each example in the dataset, the number of times it appears in a bootstrap sample follows a binomial distribution, which—as the dataset size $N$ increases—converges to a Poisson(1) distribution. This statistical property is exploited to define Online Bagging.

In the online version, each new instance $d = (x, y)$ is processed only once upon arrival. For each base model $h_m$, a random integer $k \sim Poisson(1)$ is drawn, and the model is updated $k$ times using that instance. This procedure preserves the diversity of the training sets as in batch bagging, without requiring storage of the entire dataset. Under certain conditions—such as the stability of the base learner and the ability to generalize from equivalent data distributions—the data distributions seen by the online models converge to those of the batch bagging process. This, in turn, implies that the classifiers produced also converge, thereby providing theoretical validity to the online approach.

### 2.2.5 Online Boosting

Traditional boosting (in particular, AdaBoost.M1) constructs an ensemble sequentially, where each new model is trained on a weighted dataset that emphasizes examples previously misclassified. In particular , each successive *base learner* in the boosting sequence is trained to compensate for the weaknesses or misclassifications of its predecessor. This iterative process ensures that the algorithm increasingly focuses on the data instances that were previously misclassified by earlier models.

In the online version, each example $(x, y)$ is associated with a weight $\lambda$, initially set to 1. This weight is used to determine how many times the example is used to train each base model $h_m$ , similarly to online bagging but drawing $k$ from a Poisson($\lambda$) distribution instead of Poisson(1). The algorithm dynamically updates $\lambda$ according to the correctness of the classification of the example at each stage. This mechanism reproduces the idea of giving more weight to errors, but in an incremental manner. The authors also demonstrate that, under suitable assumptions, the procedure converges to the batch version given a sufficiently large number of examples and models.

### ADWIN Bagging and Boosting

**ADWIN (ADaptive WINdowing)** is a drift detection algorithm that maintains a variable-length sliding window of the most recent observed data[9]. The core idea is to statistically compare two contiguous sub-windows, $W_0$ and $W_1$, within the entire window $W$. If the difference between the means (or other statistics) of the two parts exceeds a dynamically computed threshold, based on the Hoeffding bound, it is concluded that a change in the data distribution has occurred.

When this happens, the older portion $W_0$ is discarded, and $W_1$ becomes the new current window. In the absence of drift, the window keeps expanding with new examples. This mechanism allows ADWIN to automatically adapt the window size to the current rate of change in the data, ensuring that the window always contains only examples from a stationary distribution.

The algorithm requires only one parameter: a confidence value $\delta \in (0, 1)$, which controls the sensitivity of the statistical test. Notably, it does not require specifying a fixed window length. From a computational perspective, ADWIN does not explicitly store the full window, but instead compresses it using an exponential histogram structure, achieving a memory and per-element time complexity of $\mathcal{O}(\log |W|)$ instead of $\mathcal{O}(|W|)$ In the ensemble methods proposed in [11], namely **ADWIN Bagging** and **ADWIN Boosting**, the ADWIN algorithm is employed as a drift detector to monitor the performance of the base classifiers in the ensemble. In both approaches, a similar mechanism is adopted: whenever ADWIN detects a change in the data distribution, the least performant classifier in the ensemble is discarded and replaced with a newly initialized one, ready to learn from future data.

In ADWIN Bagging, this means that, upon drift detection, the classifier with the highest estimated error (as tracked by ADWIN) is removed and substituted with a fresh instance of the base learner.

Likewise, in ADWIN Boosting, the classifier showing the worst performance is eliminated and retrained from scratch using the upcoming data.

In practice, drift detection causes the ensemble to "forget" outdated data by removing the corresponding outdated classifiers. This allows the ensemble to adapt quickly to newly emerging concepts. As a result, the ensemble consistently maintains only classifiers trained on the current data distribution, ensuring rapid recovery in performance following concept drift.

### 2.2.6 A Frank System for Co-Evolutionary Hybrid Decision-Making

FRANK [43] is a human-in-the-loop system for co-evolutionary hybrid decision-making, based on the Skeptical Learning paradigm. In FRANK, a human user sequentially labels unlabeled records from a dataset, while an interpretable machine learning (ML) model—implemented as an incremental decision tree—learns in parallel from the user's decisions.

The ML model in FRANK is continuously updated in an online fashion, record by record, using the labels provided by the user. The goal is to support the user in decision-making: whenever the model becomes skeptical of a proposed user decision (e.g., because it contradicts prior examples), the system may challenge the user, who nonetheless retains the final veto power over each decision.

#### Operating Principles

Given a machine learning (ML) model—specifically, an Extremely Fast Decision Tree (EFDT)[42], denoted as $f$—and a dataset $X = \{x_1, .., x_n\}$ for which the values of the target variable $Y$ are initially unknown, a human user is tasked with assigning a label $y_i$ to each record $x_i \in X$ At the outset, the user provides an initial label $\hat{y}_i$ for each instance $x_i$ based on their own knowledge, intuition, or domain expertise. Independently, the predictive function $f$ produces its own label $\tilde{y} = f(x_i)$ for the same instance. The function $f$ may represent a model that has been pre-trained on a limited amount of labeled data, or it may start from an untrained (blank) state, in which case it may be unable to provide predictions for the earliest examples in the dataset.

A comparison is then made between the user-provided label $\hat{y}_i$ and the model-generated label $\tilde{y}_i$ If the two labels differ (i.e., $\hat{y}_i \neq \tilde{y}_i$) and if the model is considered skeptical—according to a formalized notion of **skepticism**that depends on the history of prior decisions made by both the user and the model—then the system prompts the user with a choice: whether to accept the model's suggestion $\tilde{y}_i$ as final label $y_i$

If the user agrees, then $y_i = \tilde{y}_i$; otherwise, $\hat{y}_i = y_i$ In either case, the pair $(x_i, y_i)$ is subsequently used to incrementally update and train the model $f$, allowing it to refine its predictions over time.

The **skeptical score** is calculated in this way , given a record $x_i$,a prediction $\tilde{y}_i$ and a user decision $\hat{y}_i$:

$$\text{skpt}(x_i, \tilde{y}_i, \hat{y}_i, Y, \tilde{Y}, \hat{Y}) = \text{conf}(f, x_i, \tilde{y}_i) \cdot \text{ea}(\tilde{y}_i, Y, \tilde{Y}) \cdot \text{conf}(f, x_i, \hat{y}_i) \cdot \text{ea}(\hat{y}_i, Y, \hat{Y})$$

where :

- conf$(f, x_i, \tilde{y}_i)$ and conf$(f, x_i, \hat{y}_i)$ are are the model confidence score towards $\tilde{y}_i$ and $\hat{y}_i$[43]. Confidence always applies to the prediction probability.

- The function $ea$ computes the empirical accuracy of either the model or the user with respect to their corresponding label. Specifically, the empirical accuracy is defined as the cardinality of the intersection between the subset of all their past decisions labeled either $\hat{y}_i$ or $\tilde{y}_i$, and the corresponding subset in the ground truth label set $Y$.

The main objective of SL is to correct user's inconsistencies assuming a well-trained model reliably approximates the user's past behavior.

Another core function of this framework is the treatment of *fairness*. It is possible to distinguish between two principal notions of fairness: individual fairness and group fairness. The former requires that similar individuals are treated similarly, while the latter ensures that different demographic groups receive equitable treatment [35]. The attribute on which potential discrimination may occur—such as race or gender—is typically referred to as a sensitive attribute or protected attribute.

In order to handle fairness , FRANK incorporates an interactive variant of Preferential Sampling (PS), an algorithm grounded in the group-based notion of fairness [35]. The original formulation of PS assumes a binary classification task, where the label set $L = \{+, -\}$ distinguishes between favorable (+) and unfavorable (-) decisions. Among the set of features $A$, a binary sensitive attribute $sa \in A$ is identified—for example, gender or race. The attribute $sa$can take on two values $\{v, \bar{v}\}$ , representing a discriminated group $v$ and a privileged group $\bar{v}$ (e.g., female and male for the attribute gender).

Assuming that the user possesses a priori knowledge of the discriminated group, the discrimination score of a dataset is computed as:

$\text{disc}(X, sa) = \frac{|PP|}{|PP \cup PN|} - \frac{|DP|}{|DP \cup DN|}$

where:

- $DP = \{x_i \in X \mid x_i[sa] = v \wedge y_i = +\}$ : discriminated group with positive outcome;

- $DN = \{x_i \in X \mid x_i[sa] = v \wedge y_i = -\}$ : discriminated group with negative outcome;

- $PP = \{x_i \in X \mid x_i[sa] = \bar{v} \wedge y_i = +\}$:privileged group with positive outcome;

- $PN = \{x_i \in X \mid x_i[sa] = \bar{v} \wedge y_i = -\}$:privileged group with negative outcome.

Here, $|PP|$ (respectively, $|PN|$) denotes the number of records from the privileged group with a positive (respectively, negative) label, while $|DP|$ (respectively, $|DN|$) refers to those from the discriminated group. The goal of Preferential Sampling is to adjust the dataset such that $disc(X, sa) \approx 0$ thereby mitigating bias. This is achieved by:

- removing selected instances from $PP$ and $DN$
- replicating (i.e., adding copies of) instances from $DP$ and $DN$

based on their ranking, which is computed via a classifier trained on the original dataset $D$. The selection prioritizes records closest to the decision boundary, The selection prioritizes records closest to the decision boundary, i.e., those with a predicted probability of the positive class closest to 0.5, reflecting the lowest classifier confidence.

**FRANK algorithm**

Here is the pseudocode of the algorithm[43]:

**Algorithm 1** FRANK

**Input:** $X$ - records to label, $R$ - supervisor rule set, $sa$ - sensitive attribute, $s$ - skepticism threshold, $k$ - number of iterations for GFC, $stp$ - stopping condition
$X', Y', \hat{Y}, \ddot{Y}, f \leftarrow$ *initialize()*
**while** $stp \neq$ True **do**
  $x_i \leftarrow$ RECEIVE_RECORD$(X)$
  $\hat{y}_i \leftarrow$ USER_DECISION$(x_i)$
  $\ddot{y}_i \leftarrow f(x_i)$
  **while** $\hat{y}_i \notin L$ **do**
    $\hat{y}_i \leftarrow$ USER_DECISION$(x_i)$
  **end while**
  **if** IDEAL_RULE$(x_i, R)$ **then**
    $\bar{y}_i \leftarrow$ RULE_LABEL$(x_i, R)$
    $y_i \leftarrow \bar{y}_i$
  **else if** INDIVIDUAL_FAIRNESS$_{sa}(x_i, X')$ **then**
    $y'_p \leftarrow$ GET_SIMILAR_PAST_LABEL$(x_i, X', Y')$
    **if** $\hat{y}_i \neq y'_p$ **then**
      $(y_i, Y') \leftarrow$ SOLVE_CONFLICT$(x_i, y'_p, \hat{y}_i, Y')$
    **else**
      $y_i \leftarrow \hat{y}_i$
    **end if**
  **else if** $\hat{y}_i \neq \ddot{y}_i \wedge$ SKEPT$_s(f, x_i, \ddot{y}_i, \hat{y}_i, \hat{Y}, \ddot{Y})$ **then**
    **if** IS_EXPL_DESIRED$(x_i, \ddot{y}_i)$ **then**
      $e_i \leftarrow$ GET_EXPLANATION$(x_i, \ddot{y}_i, f, X')$
      **if** ACCEPT_LABEL_CHANGE$(x_i, e_i, \ddot{y}_i)$ **then**
        $y_i \leftarrow \ddot{y}_i$
      **else**
        $y_i \leftarrow \hat{y}_i$
      **end if**
    **else if** ACCEPT_LABEL_CHANGE$(x_i, \ddot{y}_i)$ **then**
      $y_i \leftarrow \ddot{y}_i$
    **else**
      $y_i \leftarrow \hat{y}_i$
    **end if**
  **else**
    $y_i \leftarrow \hat{y}_i$
  **end if**
  $X' \leftarrow X' \cup \{x_i\}$
  $Y' \leftarrow Y' \cup \{y_i\}$
  $\hat{Y} \leftarrow \hat{Y} \cup \{\hat{y}_i\}$
  $\ddot{Y} \leftarrow \ddot{Y} \cup \{\ddot{y}_i\}$
  $f \leftarrow$ UPDATE$(f, x_i, y_i)$
  **if** $|Y'| \bmod k = 0$ **then**
    $Y' \leftarrow$ GROUP_FAIRNESS_CHECK$_{sa}(X', Y')$
  **end if**
**end while**

In accordance with [33], in addition to FRANK and the user, a third agent is required to configure FRANK's parameters and define certain aspects of its functionality.
FRANK requires access to:

- $X$ : a set of records for labeling the dataset which are received sequentially;
- $R$ :a set of rules provided by the user's supervisor;
- $sa$ : a sensitive attribute ;
- $s$ : a skepticality threshold ;
- $k$: a predefined number of iterations after which a group-level fairness check is conducted on the accumulated records and decisions;
- a stopping condition $stp$;
- $X'$ : a set of record analyzed so far;
- $Y'$: a set of final hybrid decisions taken by Frank and the user on the records in $X'$.
- $f$ : FRANK's ML model. It might be completely untrained, pre-trained non-interactively on some records, or pre-trained during a precedent process run of the hybrid learning;
- $\tilde{Y}$ : a set of decisions predicted by Frank's ML model $f$ alone;
- $\hat{Y}$ : a set of decisions proposed by the human user alone;
- $\ddot{Y}$ : a set to store the decisions taken by Frank and the user without possible re-labelling due to fairness corrections;

The following list provides a detailed explanation of the sequential steps executed by the algorithm implemented in the FRANK framework:

1. At the beginning of each iteration, the algorithm verifies whether the stopping condition stp has been satisfied. If this condition is met, the execution is stopped.
2. If the stopping condition is not met, FRANK receives a new record $x_i$ from the dataset $X$.

3. Consistent with the Skeptical Learning framework [65], the human user assigns a label $\hat{y}_i$ to the record $x_i$, and the machine learning model $f$ independently assigns a label $\tilde{y}_i$.

4. FRANK examines whether $x_i$ is covered by a rule in the supervisor-provided rule set $R$. If a rule applies, a decision $\bar{y}_i$ is derived and set as the final label $y_i$. This step is referred to as the *Ideal Rule Check* (IRC) phase.

5. If no rule is applicable, FRANK evaluates whether the user's decision $\hat{y}_i$ aligns with the labels of previously seen "similar" records, to ensure individual fairness. This step is referred to as the *Individual Fairness Check* (IFC) phase.

6. If there is a disagreement between the user's label $\hat{y}_i$ and the model's label $\tilde{y}_i$, FRANK assesses its skepticality towards $\hat{y}_i$. If skeptical, FRANK asks the user whether they would like to receive an explanation for its prediction $\tilde{y}_i$. This step is referred to as the *Skeptical Learning Check* (SLC) phase and it proceeds as follows:

   (a) If the user accepts, an explanation $e_i$ is presented, after which the user is asked to confirm whether they accept $\tilde{y}_i$ as the final label.

   (b) If the user declines the explanation, FRANK asks whether they wish to revise their original decision $\hat{y}_i$.

   At any point, the user retains the option to reject the change and preserve their original label.

7. If there is agreement between $\hat{y}_i$ and $\tilde{y}_i$, or if the model is not skeptical, the user's label $\hat{y}_i$ is automatically accepted as the final decision $y_i$.

8. At the end of each iteration, a definitive label $y_i$ is always determined. This can result from the user's decision alone, from the rule-based supervisor, or through interaction with FRANK.

9. Finally, every $k$ records, FRANK performs Group Fairness Check (GFC), asking the user if they want to change the label of some past records to reduce the dataset's discrimination as computed by PS [35].

A summary of the operation of FRANK is shown in 2.2

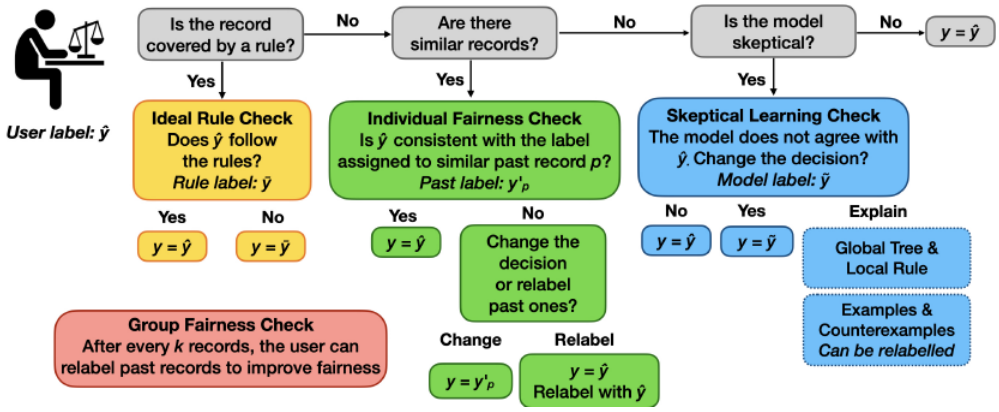


Figure 2.2: FRANK workflow [43]

IRC, IFC, and GFC serve as safeguard mechanisms designed to prevent potential bias or bad faith behavior on the part of the user—issues that may not be identifiable by the trained model alone.

To maintain consistency and prevent logical contradictions, once a final label $y_i$ is assigned, no subsequent function with lower priority is allowed to override it. Accordingly, any record that has been labeled as a result of IRC or IFC cannot be relabeled during the application of the Group Fairness Check (GFC).

In the following, a comprehensive explanation of the four checking functions integrated into the FRANK framework is provided :

- **Ideal Rule Check** : Frank is equipped with a set of decision rules $R = \{r_1, \ldots, r_m\}$ provided by the user's supervisor. Each rule $r \in R$ comprises a set of conditions $c_1, c_2, \ldots, c_k$ and an associated label $\bar{y}$, defining an *if-then* structure applied to individual records. For example, a rule may be expressed as:

  **IF** Age $< 30 \wedge$ Years of Experience $> 5$ **THEN** $\bar{y} = +$.

  Frank applies the *Ideal Rule Check* (IRC) to verify whether a given record $x_i$ satisfies the conditions of any rule in $R$. In the event of a divergence between the user's assigned label $\hat{y}_i$ and the supervisor's prescribed label $\bar{y}_i$, i.e., $\hat{y}_i \neq \bar{y}_i$, the user is informed that their decision is not aligned with the established rule set.

  In the case where multiple rules are triggered simultaneously, the current implementation of Frank resolves such conflicts by prioritizing the first rule appearing in the set $R$. To prevent interference with fairness-related functionalities, it is further required that rules must not include conditions

based on sensitive attributes, such as gender or race.

- **Individual Fairness Check**: The Individual Fairness Check (IFC) is designed to reduce the number of pairs of records that violate the condition of individual fairness—namely, that similar individuals should receive similar treatment. This check verifies whether records similar to the current input $x_i$ have previously been assigned a different label than the current user-assigned label $\hat{y}_i$.

  Concretely, Frank invokes the individual fairness function to assess if there exists at least one record $x_p \in X$ such that $x_p$ is "similar" to $x_i$, except for the value of a binary sensitive attribute $sa \in A$. Formally, Frank considers two records $x_i$ and $x_p$ to be similar if:

  $$x_i[a_j] = x_p[a_j] \quad \forall a_j \in A \setminus \{sa\}$$

  That is, the records must have identical values for all attributes except for the sensitive attribute $sa$. For example, a male and a female applicant with the exact same résumé would be considered similar.

  If such a similar record $x_p$ exists and a disagreement is observed between the current user-assigned label $\hat{y}_i$ and the previously stored label for $x_p$, Frank invokes the conflict resolution function. In this case, the user is prompted to either:

  - revise their current decision by changing $\hat{y}_i$ to ensure consistency with similar past decisions; or
  - maintain their current decision $\hat{y}_i$, which leads to a re-labelling of the affected past records in $Y$ to match $\hat{y}_i$.

  In the latter case, the ML model $f$ is also retrained to reflect the updated label of those records. Alternatively, if the user chooses to retain the previous labels and discard the current decision, Frank assigns the previous label $y_i$ and stores the decision in $\ldots Y$, indicating that no relabelling occurred. If no similar records are found, the process proceeds directly to the Skeptical Learning Check (SLC).

  If the user's label $\hat{y}_i$ is consistent with all previously labeled similar records, it is accepted without modification and stored as the final label $y_i$, confirming that no violation of the individual fairness condition has occurred.

- **Skeptical Learning Check**:

If a disagreement arises between the label provided by the user and the label predicted by Frank's machine learning model $f$, i.e., $\hat{y}_i \neq \tilde{y}_i$, Frank evaluates its own skepticality through the SKEPT function (line 16). In particular, we define Frank as *skeptical* of the user's decision $\hat{y}_i$ if:

$$\text{skpt}(x_i, \tilde{y}_i, \hat{y}_i, Y, \tilde{Y}, \hat{Y}) > s$$

where $s$ is the skepticality threshold, and the skepticality score is computed according to Formula 1. It is important to emphasize that the SKEPT function does not take $Y'$—the set of final decisions—as input. Instead, it uses $\hat{Y}$, the set of user decisions prior to any relabelling triggered by fairness mechanisms. The two sets coincide until the user modifies a label in response to a fairness-related warning.

In accordance with [64], at the beginning of the process, for each possible label $\ell \in L$, the empirical accuracy values for the user are initialized to 1, while those for Frank's model are set to 0. This ensures that Frank is not skeptical during the early stages of decision-making.

If Frank is skeptical, it proposes its own label $\tilde{y}_i$ as the final decision $y_i$. Initially, Frank offers to explain why it believes $\tilde{y}_i$ is preferable to $\hat{y}_i$ (line 17). If the user refuses to view an explanation, Frank simply asks whether the user is willing to accept its proposed label $\tilde{y}_i$ (line 21). If the user declines both the explanation and the proposed label, then the user's original label $\hat{y}_i$ is accepted as the final decision $y_i$ (line 22). It should be noted that the user always holds full veto power over Frank's suggestions.

Conversely, if the user agrees to view an explanation, Frank invokes the get explanation function . Given that Frank's ML model $f$ is implemented as an Extremely Fast Decision Tree (EFDT), it can generate two types of explanations:

- **Logic-based Explanations**:Frank can provide a visual representation of the entire decision tree, offering a global explanation of the model's behavior. Additionally, Frank can display the specific decision rule applied for classifying the input $x_i$, thereby offering factual and local justification for the predicted label $\tilde{y}_i$.
- **Instance-based Explanations**: Frank can retrieve and present real-world examples and counterexamples:

* Examples: Real records that fall within the same leaf node as $x_i$ and are labeled with $\tilde{y}_i$, highlighting the similarities that justify the model's decision.
* Counterexamples: Records located in a sibling leaf node of $x_i$ and labeled with $\hat{y}_i$, emphasizing the distinctions that discredit the user's label.
* Synthetic Counterexamples: Artificially generated records similar to $x_i$ but assigned the label $\hat{y}_i$, to reinforce contrastive reasoning.

These explanations are intended to support the user in understanding the rationale behind the model's decision and to inform their acceptance or rejection of the proposed label $\tilde{y}_i$.

- **Group Fairness Check**: Unlike the previously described checks that operate on individual records, the Group Fairness Check (GFC) monitors the trend of all assigned labels $Y'$. GFC assesses whether one of the values of a binary sensitive attribute $sa \in A$ is being systematically disadvantaged compared to the other, following the same rationale as Preferential Sampling (PS).

  It is important to highlight that, with respect to the current implementation, GFC is the only function in Frank that explicitly requires a binary classification setting, i.e., $L = \{+, -\}$, and a binary sensitive attribute $sa$. Nevertheless, alternative methodologies supporting non-binary sensitive attributes, such as those described in [44] , could potentially replace PS to enhance generalization.

  GFC is designed to operate independently from the other checking functions (e.g., IRC, IFC, SLC), and it is systematically triggered every $k$ labels assigned, i.e., after every $k$ additions to the set $Y'$ (see lines 26–27).

  When invoked, Frank computes the discrimination score $\text{disc}(X', sa)$ of the subset $X'$ of analyzed records with respect to the assigned labels $Y'$. Following the assumptions in [44], Frank initially assumes one of the two values of $sa$ to correspond to the discriminated group. If the resulting discrimination score is positive, this assumption is maintained; otherwise, the alternative group is considered to be discriminated against.

  Subsequently, Frank partitions the records in $X'$ into the four groups: DP,DN,PP,PN

  Frank then determines how many instances should be removed from the $DN$ and $PP$ groups to approximate fairness (i.e., to reach $\text{disc} \approx 0$). Records in these two groups are sorted based on

the confidence scores output by Frank's machine learning model $f$, i.e., the predicted probability associated with the assigned class.

Finally, the records with the highest confidence are presented to the user, who is given the opportunity to revise the label of some (or all) of them. This interactive relabeling process enables the user to actively participate in reducing group discrimination and improving the overall fairness of the dataset.

### 2.2.7 DICE: Diverse Counterfactual Explanations

**DICE (Diverse Counterfactual Explanations)**[49] is based on the idea that effective counterfactual explanations must satisfy two crucial properties:

- **Feasibility** ensures that the suggested counterfactual actions are realistic and actionable within the user's context, respecting the causal laws of the real world (e.g., one cannot reduce their level of education or change their race).
- **Diversity** aims to maximize the variation among the $k$ generated counterfactuals $\{c_1, ...c_k\}$[36]. This is a distinctive aspect of DICE, which acknowledges the importance of providing the user with a diverse range of alternatives.

DICE formulates counterfactual generation as an optimization problem for any differentiable machine learning classifier, with the goal of producing a set of $k$ counterfactual examples $\{c_1, \ldots, c_k\}$.
The combined loss function is defined as follows [49]:

$$C(x) = \arg\min_{c_1,\ldots,c_k} \left[ \frac{1}{k} \sum_{i=1}^{k} \text{yloss}(f(c_i), y) + \frac{\lambda_1}{k} \sum_{i=1}^{k} \text{dist}(c_i, x) - \lambda_2 \cdot \text{dpp\_diversity}(c_1, \ldots, c_k) \right]$$

Where:

- $\text{yloss}(f(c_i), y)$: A term that encourages the counterfactual $c_i$ to achieve the desired prediction $y$.
- $\text{dist}(c_i, x)$: Measures the proximity between the counterfactual $c_i$ and the original input $x$.

- dpp_diversity$(c_1, \ldots, c_k)$: A diversity metric based on **Determinantal Point Processes (DPP)**, where the kernel is typically defined as:

$$K_{i,j} = \frac{1}{1 + \text{dist}(c_i, c_j)}$$

  DPPs are commonly used in subset selection tasks to promote diversity.

- $\lambda_1$ and $\lambda_2$: Hyperparameters that balance the influence of proximity, prediction loss, and diversity in the overall objective.

The optimization is performed using gradient descent, for a maximum of 5,000 steps or until convergence and validity of the generated counterfactual are achieved. The counterfactual instances are initialized randomly. With regard to distance functions, these are crucial for evaluating both the proximity between the counterfactual instances and the original input, as well as the diversity among multiple counterfactuals:

- **Continuous Features:** The distance for continuous features is defined as:

$$\text{dist}_{\text{cont}}(c, x) = \frac{1}{d_{\text{cont}}} \sum_{p=1}^{d_{\text{cont}}} \frac{|c_p - x_p|}{\text{MAD}_p}$$

  where:

  - $d_{\text{cont}}$ is the number of continuous features
  - $c_p$ and $x_p$ denote the values of the $p$-th feature for the counterfactual $c$ and the original instance $x$, respectively
  - $\text{MAD}_p$ is the Median Absolute Deviation of the $p$-th feature in the training set. This normalization accounts for varying feature ranges and reflects the relative frequency of observing a feature at a given value.

- **Categorical Features:** For categorical features, the distance is defined as:

$$\text{dist}_{\text{cat}}(c, x) = \frac{1}{d_{\text{cat}}} \sum_{p=1}^{d_{\text{cat}}} \mathbb{I}(c_p \neq x_p)$$

  where : $d_{\text{cat}}$ is the number of categorical features and $\mathbb{I}(c_p \neq x_p)$ is an indicator function that equals 1 if the $p$-th categorical feature differs between $c$ and $x$, and 0 otherwise. This approach acknowledges

the difficulty in defining a gradient of "changeability" for categorical variables (e.g., education vs. race).

DICE,moreover, incorporates several practical considerations to enhance the applicability and utility of counterfactual explanations:

- **Choice of *yloss*:** DICE adopts a hinge-loss function defined as $\max(0, 1 - z \cdot \text{logit}(f(c)))$, where $z = -1$ for a desired class $y = 0$, and $z = 1$ for $y = 1$. This design choice is critical: unlike $L_1$ or $L_2$ loss functions, hinge-loss imposes no penalty as long as $f(c)$ exceeds a fixed threshold (e.g., $0.5$ for the desired class 1). This prevents the optimization process from pushing $f(c)$ excessively close to 0 or 1, which could lead to overly drastic and less feasible modifications of the input $x$.
- **Relative Feature Scaling:** To ensure that features contribute appropriately to the objective function, all features are transformed onto a common scale. Continuous features are scaled accordingly, while categorical features are one-hot encoded and treated as continuous variables within the $[0, 1]$ range. A regularization term is added for categorical features to enforce the one-hot encoding constraint during optimization.
- **Hyperparameter Selection:** To maintain consistency and avoid user-specific tuning for each input, DICE employs fixed hyperparameters, with $\lambda_1 = 0.5$ and $\lambda_2 = 1$, determined through grid search.

To assess the quality of a set of counterfactual examples, DICE proposes the following quantitative metrics[49]:

- **Validity:** The fraction of unique counterfactual examples that yield the desired outcome. Formally:

$$\%\text{Valid-CFs} = \frac{|\{c \in C : f(c) > 0.5\}|}{k}$$

  where $k$ is the number of generated counterfactuals.

- **Proximity:** Measures how close the generated counterfactuals are to the original input, computed as the average distance across both continuous and categorical features. Lower distances imply better proximity. Is possible to distinguish 2 types of proxitmity:
  - **Continuous Proximity:** Measures the average normalized distance across continuous fea-

tures between the counterfactual examples and the original input. Formally:

$$\text{Continuous-Proximity} = -\frac{1}{k}\sum_{i=1}^{k} \text{dist}_{\text{cont}}(c_i, x)$$

A higher (less negative) value indicates better proximity.

- **Categorical Proximity:** Quantifies similarity in categorical features by penalizing mismatches. It is defined as:

$$\text{Categorical-Proximity} = 1 - \frac{1}{k}\sum_{i=1}^{k} \text{dist}_{\text{cat}}(c_i, x)$$

where $\text{dist}_{\text{cat}}$ returns 1 for a categorical mismatch and 0 otherwise. Higher values indicate closer categorical alignment.

- **Sparsity:** Captures the number of features that differ between the original input and a counterfactual. Formally:

$$\text{Sparsity} = 1 - \frac{1}{kd}\sum_{i=1}^{k}\sum_{l=1}^{d} \mathbf{1}_{[}c_i^l \neq x^l]$$

Higher values (closer to 1) indicate fewer changes and are thus more desirable.

- **Diversity:** Assesses the dissimilarity between the generated counterfactuals. It is computed as the average pairwise distance:

$$\frac{1}{C_k^2}\sum_{i=1}^{k-1}\sum_{j=i+1}^{k} \text{dist}(c_i, c_j)$$

*Count-Diversity* also measures the fraction of differing features between each pair of counterfactuals:

$$\frac{1}{C_k^2}\sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\sum_{l=1}^{d} \mathbf{1}_{[}c_i^l \neq c_j^l]$$

- **Approximation of the Local Decision Boundary:** This metric evaluates how well the counterfactual examples help a user understand the local decision boundary of the ML model. A 1-Nearest Neighbor (1-NN) classifier is trained on the generated counterfactuals and the original input. Its performance (F1 score, precision, recall) is then compared to that of the original ML model on synthetic test samples sampled at increasing distances from the original input. This serves as a proxy for how well a user could "guess" the model's local decision boundary.

### 2.2.8 Growing Spheres

Growing Spheres [57]is characterized as a post-hoc interpretability approach designed to explain individual predictions of a classifier. A key strength of the method lies in its model-agnostic and data-agnostic nature: it requires no access to the internal structure of the model or to its training data. The only prerequisites are :

1. access to the feature representation of the input;
2. an understanding of the semantic meaning of the attributes;
3. the ability to query the classifier to obtain predictions. These minimal assumptions make Growing Spheres particularly well-suited for true black-box scenarios.

The method operates within an instance-based paradigm, providing explanations through comparison with relevant neighboring instances. This approach is motivated by cognitive science findings, which suggest that humans often learn and reason effectively by means of examples. At its core, the method applies the principle of inverse classification, aiming to identify the smallest changes required to alter a model's decision. Its primary goal is to elucidate the decision-making process of the classifier, rather than to faithfully reconstruct the underlying data-generating process that the model approximates.

Growing Spheres adopts a nearest-opposite search strategy, seeking the closest instance with a different classification by minimizing both the $\ell_2$ norm (for proximity) and the $\ell_0$ norm (for sparsity of changes). This optimization implicitly assumes that the decision boundary in the local neighborhood of the input instance is relatively simple, i.e., "clear and smooth." As a result, the method can yield intuitive and actionable counterfactual explanations that reveal how the classifier could have reached a different decision.

The explanation in the Growing Spheres framework is derived from a *nearby point classified differently*—referred to as the "enemy" $e$—relative to the original observation $x$. The final explanation is expressed as the displacement vector $e - x$.

The goal is formulated as a minimization problem defined over the input space $\mathcal{X}$ using a cost function $c(x, e)$:

$$e^* = \arg\min_{e \in \mathcal{X}} \{c(x, e) \mid f(e) \neq f(x)\}$$

where the cost function $c(x, e)$ balances proximity and sparsity:

$$c(x, e) = \|x - e\|_2 + \gamma \|x - e\|_0$$

- $\|x - e\|_2$ (Euclidean norm or $\ell_2$ norm): Measures the proximity between the original instance $x$ and the candidate counterfactual $e$. This term ensures that the counterfactual remains close to the input.
- $\|x - e\|_0$ ($\ell_0$ norm): Measures the sparsity of the difference vector $e - x$, defined as the number of non-zero coordinates (i.e., the number of features that have been altered,

$$\|x - e\|_0 = \sum_{i=1}^{d} \mathbb{1}_{x_i \neq e_i}$$

  ). This component promotes interpretability by producing explanations that are simple and easy to understand.
- $\gamma \in \mathbb{R}^+$: A non-negative hyperparameter that controls the trade-off between proximity (via the $\ell_2$ norm) and sparsity (via the $\ell_0$ norm).

Due to the discontinuous nature of the cost function and the black-box assumption of the classifier, *Growing Spheres* employs a two-phase heuristic to approximate the solution:

1. **Generation**:

   This phase focuses on minimizing the $L_2$ component of the cost function. It systematically explores the input space by generating instances within spherical $L_2$ layers around the original observation $x$ until an "enemy" (i.e., an instance classified differently from $x$) is found. The process begins by generating $n$ observations uniformly within an $L_2$ sphere of radius $\eta$. If enemies are found in this initial sphere, $\eta$ is halved and the process is repeated. This ensures that the algorithm identifies the closest decision boundary. If no enemies are initially found, the search expands iteratively into concentric spherical layers with increasing radii until the first enemy is discovered. The algorithm returns the enemy $e$ that is closest to $x$ in terms of Euclidean ($L_2$) distance.

2. **Feature Selection**: This phase aims to minimize the $L_0$ component of the cost function, thereby rendering the difference vector $e - x$ sparse and more interpretable. Starting from the enemy $e$ found in the generation phase, the algorithm iteratively attempts to align as many coordinates of $e$ with $x$ as possible as long as the predicted class does not change[57].

At each iteration, the algorithm greedily identifies the feature (coordinate) in $e - x$ with the smallest absolute difference that is not yet aligned with $x$. It then sets this coordinate in $e$ to match the corresponding coordinate in $x$ (effectively removing that feature from the set of changes).

This iterative process continues as long as the predicted class of the modified enemy $e'$ remains different from the original observation $x$. The final output $e^*$ is the sparsest enemy found that still results in a different classification.

Growing Spheres explicitly states that its goal is to "provide information about the classifier, not about the reality it is approximating." This is vividly illustrated by the MNIST handwritten digits database[63, 57], where the generated "enemies" are not perfect digits of the target class, but rather "noisy versions of the original digits."

This highlights a crucial distinction: the explanation is faithful to the model's decision boundary, but does not guarantee that the counterfactual instance is realistic or causally plausible in the real world. As a result, while Growing Spheres provides accurate insights into the model's behavior, the resulting counterfactuals may require further human interpretation or domain-specific filtering to be truly actionable or intuitively understandable in real-world contexts.

### 2.2.9 LORE : Local Rule-based Explanations

LORE (Local Rule-based Explanations)[28] has been proposed as a solution to the problem of explaining the outcome of a black-box model, particularly suitable for the analysis of relational and tabular data. The method is structured as a two-stage architecture, specifically designed to provide interpretable and faithful explanations.

Given a binary black-box predictor $b$ and a specific instance $x$ for which $b$ returns the outcome $y$, LORE proceeds to construct a simple and interpretable predictor $c$. Once the local interpretable predictor $c$ is obtained, a meaningful local explanation is derived from it. This explanation aims to make the decision logic of the black-box model transparent within the specific context of instance $x$. The explanation produced by LORE is defined as a structured pair: $e = \langle r, \Phi \rangle$, where:

- $r$ is a **decision rule** ($r = p \rightarrow y$) that concisely describes the specific reason for the decision value $y$ assigned to the instance $x$ by the local predictor $c(x)$. The premise $p$ of this rule is a Boolean condition over the feature values, typically expressed as a conjunction of split conditions that the

instance $x$ satisfies. This rule directly addresses the question: *Why was $x$ labeled with outcome $y$?* The rule $r$ is constructed such that it is consistent with the behavior of $c$ and satisfied by the instance $x$, ensuring that the explanation is faithful to the local behavior of the interpretable model.

- $\Phi$ is a set of **counterfactual rules**. These rules suggest the minimal set of specific changes to the feature values of $x$ that, if applied, would flip the decision from $y$ to a different outcome $\hat{y} \neq y$. The counterfactual rules provide *actionable recourse*, offering practical guidance on what must be modified in the instance to achieve a different result.

The strategic choice of decision trees as the interpretable predictor $c$ is fundamental to LORE [28]. This decision is motivated by their intrinsic ability to naturally derive clear logical rules (i.e., root-to-leaf paths) and to support the symbolic reasoning required for counterfactual extraction. As inherently transparent models, decision trees facilitate a relatively smooth transition from complex black-box behavior to human-understandable explanations, particularly at the local level.

**LORE algorithm**

This section presents and details the complete LORE algorithm [28], explaining each of its components and operational steps in depth:

- **Synthetic neighborhood generation phase**: This phase is a fundamental pillar of the LORE approach. Its objective is to create a set of instances $Z$ that are both close to the instance $x$ to be explained and representative of the local decision behavior of the black-box model $b$. It is crucial that this neighborhood is balanced, including instances that the black box classifies with the same outcome as $b(x)$ (the subset $Z_{=}$) and instances classified with a different outcome (the subset $Z_{\neq}$).

  LORE adopts an approach based on Genetic Algorithms (GAs) for generating these instances $z \in Z = Z_{=} \cup Z_{\neq}$. GAs, inspired by the biological metaphor of evolution, rely on three distinct components:

  - Potential solutions to the problem are encoded as representations, called chromosomes, which in this context correspond to instances in the feature space $\mathcal{X}^m$. These representations are designed to support variation and selection operations.

- A **fitness function** evaluates the "quality" of each chromosome, determining which are the "fittest" individuals with respect to the desired outcome. Individuals with higher fitness scores are favored for survival and reproduction, thereby guiding the evolution of the population towards optimal solutions.
- **CrossOver** and **Mutation** operators generate a new generation of chromosomes by recombining the characteristics of their "parents." Crossover combines the genetic material of two individuals, while mutation introduces random variations.

Synthetic neighborhood generation phase implements an evolutionary approach to generate the neighborhoods $Z_=$ and $Z_{\neq}$. The process is structured by maximizing the following fitness functions:

$$\text{fitness}_=(z) = \mathbb{I}_{b(x)=b(z)} + (1 - d(x, z)) - \mathbb{I}_{x=z}$$

$$\text{fitness}_{\neq}(z) = \mathbb{I}_{b(x)\neq b(z)} + (1 - d(x, z)) - \mathbb{I}_{x=z}$$

The function $\text{fitness}_=(z)$ rewards instances $z$ that are similar to $x$ (quantified by the term $1-d(x, z)$), are not identical to $x$ (penalized by $-\mathbb{I}_{x=z}$), and for which the black box $b$ produces the same outcome as $x$ (indicated by $\mathbb{I}_{b(x)=b(z)}$). Conversely, $\text{fitness}_{\neq}(z)$ guides the generation of instances $z$ that are similar but not equal to $x$ and for which $b$ returns a different decision.

These fitness functions are crucial as they steer the genetic algorithm to explore instances close to $x$ that lie on both sides of the black box's decision boundary.

During the phase for evolutionary mechanisms , there are 3 main functions:

1. **Selection (select**$(P_i)$**):** The population $P_{i+1}$ is selected from the current population $P_i$ favoring individuals with higher fitness scores, ensuring that the "fittest" individuals are more likely to contribute to the next generation.
2. **Crossover (crossover**$(P_{i+1}, p_c)$**):** The crossover operator is applied to a proportion of $P_{i+1}$ according to the probability $p_c$. LORE employs a two-point crossover where two "parents" and two random crossover points (features) are selected. The feature values between these two points are exchanged between the parents to produce two new "offspring." This mechanism

allows combining successful traits of different individuals, exploring new combinations in the instance space.

3. **Mutation (mutate$(P'_{i+1}, p_m)$):** Subsequently, a proportion of the offspring resulting from crossover is mutated, controlled by the mutation probability $p_m$. Mutation consists of randomly replacing feature values based on their empirical distribution. This operator introduces variability in the population, helping to prevent premature convergence and to explore regions of the solution space that might not be reachable through crossover alone.
4. **Evaluation and Update:** The individuals in the new population are re-evaluated according to the fitness function, and the population is updated for the next generation.

This porcedure is executed twice: once using fitness$_=$ to derive instances in the neighborhood $Z_=$ sharing the same decision as $x$, and once using fitness$_\neq$ to derive instances in $Z_\neq$ with a different decision from $x$. Finally, $Z$ is set as the union of $Z_=$ and $Z_\neq$.

The genetic neighborhood generation process produces a neighborhood that is significantly denser in the predictor's decision boundary region compared to uniform random generation. This density of generated instances is a key factor for extracting high-quality interpretable local predictors.

The targeted exploration, guided by the fitness functions, actively seeks instances near $x$ that define the decision boundary, resulting in a relevant and balanced neighborhood. The distance function $d(x, z)$ is directly incorporated into the fitness functions. This interaction ensures that the generated neighborhood is not only geometrically close to $x$ but also semantically relevant to the decision context of $x$.

This tight coupling enables the genetic algorithm to effectively explore the decision boundary and generate high-quality training data for the local decision tree. The distance function $d(x, z)$ is a key component in defining the fitness functions used by the genetic algorithm. It is designed to handle the presence of mixed feature types — both categorical and continuous — within tabular data. Formally, $d(x, z)$ is computed as a weighted sum of the Simple Matching coefficient for categorical features and the normalized Euclidean distance for continuous features:

$$d(x, z) = \frac{h}{m} \cdot \text{SimpleMatch}(x, z) + \frac{m - h}{m} \cdot \text{NormEuclid}(x, z)$$

where $h$ represents the number of categorical features and $m$ is the total number of features. Its significance lies in ensuring that the synthetic instances generated by the genetic algorithm are truly "close" to the original instance $x$.

Once the neighborhood $Z$ around the instance $x$ has been generated (via the genetic algorithm), the next step in the LORE framework consists of constructing an interpretable predictor $c$. This predictor is trained on the instances $z \in Z$, which have been labeled with the black box decision $b(z)$. The primary objective of $c$ is to faithfully mimic the behavior of the black box $b$ within this specific neighborhood $Z$.

- **Extracting Rules** : The process of extracting the decision rule $r = p \rightarrow y$ from the decision tree $c$ is straightforward and proceeds as follows:

  1. Initially, the interpretable predictor $c$ (the decision tree) is constructed on the balanced set of neighborhood instances $Z$.
  2. LORE identifies the unique path within the decision tree $c$ starting from the root and leading to the leaf node satisfied by the specific instance $x$ to be explained.
  3. The consequence $y$ of the rule is simply the decision $c(x)$, i.e., the class assigned by the decision tree $c$ to the instance $x$.

- **Counterfactual rules extraction**: Building on these premises, the generation of counterfactual rules is undertaken. Their objective is to provide the user with guidance on the minimal changes to the features of $x$ that would lead to a different outcome from the black box. This is achieved by analyzing the structure of a surrogate decision tree trained locally around the instance of interest. Specifically, the algorithm explores decision paths that lead to a different prediction than the one originally assigned, and selects those that require the smallest number of modifications to the input features. The resulting rules take the form of conditional statements that indicate which feature changes would be sufficient to obtain the alternative outcome.

### 2.2.10 Counterfactual guided by Prototypes using alibi explain

Most counterfactual generation methods, such as those proposed in [61, 57] have primarily focused on generating sparse counterfactuals by minimizing objective functions that account for prediction variation

and the magnitude of perturbations. However, there are significant limitations of these approaches :

- their inability to consider local and class-specific interpretability. This implies that, although a counterfactual may be sparse, it can still fall out of distribution with respect to the target class's data distribution, leading to outcomes that are difficult to interpret. A vivid example of this issue is observed in the MNIST dataset, where a counterfactual '3' generated from an original '5', despite being sparse, both visually and through reconstruction (using a general autoencoder or one trained specifically on '5's), resembled a '5' more than a '3'[59]. In [19] attempted to address this challenge by introducing an autoencoder-based loss term $(L_{A_E})$. to penalize out-of-distribution counterfactuals. However, this term does not take into account the data distribution for each prediction class $i$.
- the excessive computational cost. For istance, methods based on the numerical evaluation of the gradient, the computational complexity increases proportionally with the dimensionality of the feature space.

Prototype-guided counterfactual generation is based on the optimization of an augmented objective function, which incorporates a prototype-based class loss term. This approach is designed to overcome the limitations of previous methods in terms of interpretability and computational efficiency.

**Methodology of Prototype-Guided Counterfactual Generation**

The search for a counterfactual instance is an optimization problem that aims to minimize an objective function designed to encode the desirable properties of the counterfactual. $\mathbf{x}_{cf} = \mathbf{x}_0 + \boldsymbol{\delta}$ ,where: $\mathbf{x}_0$ is the original instance and $\boldsymbol{\delta}$ is the perturbation. In particular is introducted a prototype-based loss term, resulting in the full objective function:

$$L = c \cdot L_{\text{pred}} + \beta \cdot L_1 + L_2 + L_{\text{AE}} + L_{\text{proto}}.$$

Each term contributes to a specific aspect of the counterfactual's quality. In particular :

- $$L_{\text{pred}} := f_\kappa(\mathbf{x}_0, \boldsymbol{\delta}) = \max\left(\left[f_{\text{pred}}(\mathbf{x}_0 + \boldsymbol{\delta})\right]_{t_0} - \max_{i \neq t_0}\left[f_{\text{pred}}(\mathbf{x}_0 + \boldsymbol{\delta})\right]_i,\ -\kappa\right)$$

  where : $[f_{\text{pred}}(\mathbf{x}_0 + \boldsymbol{\delta})]_i$ denotes the predicted probability of the $i$-th class and $\kappa \geq 0$ constrains the divergence.

It plays the primary role of encouraging the perturbed instance $\mathbf{x}_{cf}$ to be classified into a class different from the original prediction $t_0$.

- The distance loss term,

$$f_{\text{dist}}(\boldsymbol{\delta}) = \beta \cdot \|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\delta}\|_2^2 = \beta \cdot L_1 + L_2,$$

is an elastic net regularizer[68] Its role is to minimize the distance between the original instance $\mathbf{x}_0$ and the counterfactual $\mathbf{x}_{cf}$.

- The autoencoder loss term,

$$L_{\text{AE}} = \gamma \cdot \|\mathbf{x}_0 + \boldsymbol{\delta} - \text{AE}(\mathbf{x}_0 + \boldsymbol{\delta})\|_2^2,$$

where AE is an autoencoder trained on the entire training set, penalizes counterfactual instances that deviate significantly from the overall data manifold of the training set.

- The prototype loss term can be defined as

$$L_{\text{proto}} = \theta \cdot \|\text{ENC}(\mathbf{x}_0 + \boldsymbol{\delta}) - \text{proto}_j\|_2^2$$

or using k-d trees as

$$L_{\text{proto}} = \theta \cdot \|\mathbf{x}_0 + \boldsymbol{\delta} - \text{proto}_j\|_2^2.$$

Its role is twofold:

1. it explicitly guides the perturbations $\boldsymbol{\delta}$ towards an interpretable counterfactual $\mathbf{x}_{cf}$ that falls within the distribution of the target counterfactual class $j$
2. it significantly accelerates the search process by directing $\mathbf{x}_{cf}$ towards a representative point of the target class.

The explanation provided above is summarized and detailed through these two algorithms, depending on the availability of a trained autoencoder[19]:

1. **Counterfactual Search with Encoded Prototypes** This approach is employed when an encoder is available to project data into a latent space. The encoder $\text{ENC}(\mathbf{x})$ maps an instance $\mathbf{x}$ from the

$D$-dimensional feature space $\mathcal{X}$ to an $E$-dimensional latent space $\mathbb{R}^E$. Prototypes are defined within this latent space.

For each class $i$, its prototype $\text{proto}_i$ is computed as the mean encoding of the $K$ nearest instances in the latent space belonging to that class. It is crucial to note that these $K$ instances are ordered according to their increasing $L_2$ distance from $\text{ENC}(\mathbf{x}_0)$, the encoding of the original instance to be explained. This ensures that the prototype is relevant to the proximity of the original instance in the latent space.

2. **Counterfactual Search with k-d Trees**:

   This method is employed when a trained encoder is not available, operating directly in the original feature space. Instead of using encodings, each class $i$ is represented by a separate k-d tree, constructed using the training instances belonging to that class ($X_i$).

   The prototype ($\text{proto}_j$) for the target counterfactual class $j$ is defined as the closest element (the $k$-th nearest neighbor, $x_{j,k}$) in the k-d tree of class $j$ to the original instance $\mathbf{x}_0$. The closest $x_{j,k}$ among all classes $j \neq t_0$ becomes the class prototype. This directly selects a representative instance from the target class data.

A crucial aspect is the handling of categorical features, as they lack inherent numerical order or distance. This methodology proposes two approaches to address this challenge:

1. **Modified Value Difference Metric (MVDM):** This metric[16] infers the similarity between two categories ($v_1$, $v_2$) of a feature based on how similarly they are distributed across the prediction classes. If two categories consistently lead to similar prediction probabilities, they are considered to be closer. The formula is given by:

$$d(v_1, v_2) = \sum_{i=1}^{n} \left| \frac{c_{i1}}{c_1} - \frac{c_{i2}}{c_2} \right|^{\alpha}, \quad \text{with } \alpha = 1,$$

   where $c_{ij}$ is the number of instances of category $v_j$ that belong to class $i$, and $c_j$ is the total number of instances of category $v_j$.

2. **Association-Based Distance Metric (ABDM):** This metric[68] defines the distance between two categories of a feature based on the dissimilarity of their conditional probability distributions with

respect to other features in the dataset. If two categories exhibit similar associations with other features, they are considered to be close.

Once pairwise distances between categorical values have been inferred, *Multidimensional Scaling* (MDS) is applied. MDS projects these distances into a two-dimensional Euclidean space. The norms of these 2D embeddings are then used as one-dimensional numerical representations for the categorical values. The data point with the largest Frobenius norm in the embedded space is set as the origin. Finally, these numerical values are scaled (using standard or min-max normalization) to match the range of other pre-processed numerical features. This enables meaningful perturbations in the resulting numeric space.The key challenge with categorical variables is that standard numerical perturbations (e.g., adding a small $\epsilon$) are not meaningful. By inferring distances based on model predictions (MVDM) or on relationships with other features (ABDM), and subsequently embedding them via MDS, this approach ensures that perturbations in the numeric space correspond to semantically meaningful changes in the categorical space. For example, modifying *"High School"* to *"Bachelor's Degree"* is a more interpretable and realistic perturbation than changing it to an arbitrary category. It is important to note that the objective function is optimized using the *Fast Iterative Shrinkage-Thresholding Algorithm* (FISTA)[6]. It operates by iteratively updating the perturbation $\delta$ using a momentum-based approach. A key feature of FISTA is its treatment of the $L_1$ regularization term: instead of including the term $\beta \cdot L_1$ directly in the objective function, it effectively "removes" this term and applies a soft-thresholding operation that forces perturbations with $|\delta_k| < \beta$ (for feature $k$) to zero. This mechanism directly promotes sparsity in the resulting counterfactual.The optimal counterfactual $x_{\text{cf}}$ is selected from the sequence of iterations. It is defined as $x_0 + \delta_{n^*}$, where $n^*$ is the iteration that minimizes the elastic net regularization term $\beta \cdot \|\delta^n\|_1 + \|\delta^n\|_2^2$.

# Chapter 3

# PROBLEM STATEMENT AND EXPLANATION OF THE PROPOSED SOLUTION

This chapter formalizes the learning problem addressed in this work and introduces the core components of the proposed solution. As discussed in Chapter 2, the FRANK framework operates within the context of *human-in-the-loop* systems and the *Skeptical Learning* paradigm. Specifically, the system models an iterative scenario in which a user interacts with a continuously updated predictive model. FRANK's process manages skepticism as follows:

1. An initial predictive model is trained on a preliminary subset of data.
2. A simulated user is introduced, characterized by two key parameters:
    - **Expertise**: the degree of decision-making competence;
    - **Believability**: the user's prior trust in the model's predictions.
3. An iterative training phase begins. In each iteration, a new instance $x$ is presented to both the model and the user, which produce respective predictions $\hat{y}_M$ and $\hat{y}_U$.
4. A historical **log** is continuously updated and contains all the instances seen by the model—both from the initial training set and those encountered during the interactive training phase.

5. If the predictions match ($\hat{y}_M = \hat{y}_U$), the instance is added to the log and the model is updated accordingly.

6. If the predictions diverge ($\hat{y}_M \neq \hat{y}_U$), a skepticism management procedure is triggered. Specifically, a skepticism score is computed (as defined in Chapter 2), and if it exceeds a predefined threshold $\tau$, the system considers the disagreement significant and proceeds as follows:

    - The system offers the possibility to provide an explanation to the user, which the user can choose to accept or decline.
    - If the user accepts: the system generates and displays the explanation. If the user confirms the system's explanation—thus acknowledging the misclassification—the model is updated using the predicted label ($y \leftarrow \hat{y}_M$),and logs the instance.
    - If the user refuses the explanation or chooses not to view one, the system proceeds with updating the model using the label provided by the user ($y \leftarrow \hat{y}_U$), and logs the instance.

The main contributions of this work with respect to the original FRANK framework can be summarized as follows:

- **Integration of counterfactual explanations:** This work extends the FRANK framework by incorporating model-agnostic explanation methods based on counterfactual reasoning. These explanations are designed to support user decision-making in skeptical situations, enhancing model transparency and user trust.
- **Model correction via user-aligned counterfactuals:** A novel mechanism is introduced to update the predictive model based on user disagreement. By leveraging the LORE algorithm to generate semantically consistent counterfactual neighborhoods, the system adapts the model to better reflect user reasoning over time.

These contributions are explored in detail in the following sections.

## 3.1 Integration of counterfactual explanations

In the original work in [43], the FRANK framework was evaluated using a single predictive model: the Extremely Fast Decision Tree (EFDT). While this model offers high transparency and is particularly well-

suited for generating local explanations, its exclusive use limits the generalizability of the obtained results.One of the main objectives of this study is therefore to extend the evaluation of the framework to a broader range of models, in order to analyze how they perform within the interactive loop with the user and in scenarios involving skepticism. In addition to the introduction of multiple predictive models, the framework has been extended through the integration of explanation techniques based on counterfactuals, as well as factual and counterfactual rules. This enhancement was made possible by the adoption of **eXplainable Artificial Intelligence (XAI)** tools which, being model-agnostic, allow for the generation of consistent and applicable explanations across different types of models.As anticipated in Chapter 2, this work adopts counterfactual explanations as the primary method to support the user's decision-making process. This choice is motivated not only by their formal properties—such as interpretability, proximity to the original instance, and adherence to real-world plausibility—but also by their well-documented psychological relevance.Indeed, numerous studies in cognitive science have shown that counterfactual reasoning is a natural and intuitive mechanism through which humans understand decisions, evaluate alternatives, and attribute responsibility [14]. These characteristics make counterfactuals particularly suitable for enhancing trust and transparency in human-in-the-loop systems. Their explanatory power lies in answering questions of the type: "What minimal change would have led to a different outcome?"—thus aligning with how users naturally seek justification in uncertain or conflicting scenarios.

The experiments conducted with the newly extended FRANK framework proposed in this work focus exclusively on the functionality of Skeptical Learning, deliberately leaving aside other advanced features present in the ESO system—such as the **Ideal Rule Check (IRC)**, the **Individual Fairness Check (IFC)**, and the **Group Fairness Check (GFC)**—which have been extensively discussed in Chapter 2. This choice was made to enable a targeted and effective evaluation of the new methodology introduced, which combines iterative learning with counterfactual explanations.

The main objective is twofold: on the one hand, to analyze the quality of the explanations generated in skeptical scenarios; on the other, to assess the overall effectiveness of the framework within a realistic interactive loop involving a simulated user. To achieve this goal, for each counterfactual instance generated by the various explainers employed , the primary quality metrics defined in the literature will be computed:

- **Proximity:** the similarity between the original instance and the counterfactual;
- **Sparsity:** the minimum number of attributes that are modified;

- **Plausibility:** the degree to which the counterfactual is consistent with the data distribution(the minimum distance between counterfactual instance and all istances in the log data);
- **Temporal validity:** the persistence of the counterfactual effect during every moment of skepticism;
- **Execution time:** the computational efficiency of the explanation method.

To support this analysis, each experiment will also include an evaluation of the predictive performance metrics of the most effective model, in terms of accuracy and F1-score. This aims to offer an integrated perspective on model performance and the quality of the generated explanations. Such an approach seeks to highlight possible trade-offs between accuracy and transparency, and to identify the most promising configurations for practical applications in real-world scenarios.

## 3.2 Model correction via user-aligned counterfactuals

An additional experiment was conducted with the aim of adapting the model based on the user's decisions during skeptical interactions. The objective of this experiment is twofold: first, to progressively align the model's behavior with the user's reasoning; and second, to reduce the level of skepticism over successive iterations, thereby improving the effectiveness of human-model interaction. To achieve this goal, a synthetic instance generator based on the LORE algorithm—previously described in Chapter 2—was employed. LORE enables the construction of a synthetic neighborhood around a given instance, while preserving semantic consistency and plausibility with respect to the observed data.
The rationale behind the proposed approach is as follows: During each moment of skepticism, whenever a user expresses disagreement with the explanation provided by the FRANK method (i.e., rejects the counterfactual proposed), an attempt is made to adjust the model accordingly. The correction is performed through the following steps:

1. Given a counterfactual instance returned by FRANK, the user—by expressing disagreement—implicitly suggests that the instance does not align with their reasoning.
2. LORE is then used to generate new synthetic instances that are similar to the rejected counterfactual.
3. These new instances are labeled with the opposite class to that of the original counterfactual, reflecting the user's viewpoint.

4. The model is updated using these newly labeled instances.

The rationale behind this approach is that, if a counterfactual explanation is rejected by the user, it implies that the model's reasoning is not aligned with human intuition. By retraining the model on user-aligned data, this method aims to reduce skepticism over time and improve trust in the system's decisions.
This newly generated dataset is then used to selectively update the model, reinforcing its generalization toward the user's preferences. In this way, an adaptive learning mechanism is implemented, allowing the system to evolve over time by progressively minimizing decision misalignments and making the interaction increasingly smooth and coherent. The pseudocode corresponding to this methodology is presented below:

**Algorithm 2** Model Correction Using LORE in Skeptical Situations

**Require:** Model $M$, instance $x$, user decision $y_u$, counterfactuals $\mathcal{CF} = \{cf_1, cf_2, \ldots, cf_n\}$, LORE generator LORE_Gen

**Ensure:** Updated model $M$

$\hat{y} \leftarrow M.\text{PREDICT}(x)$
**if** $\hat{y} \neq y_u$ **then**
  // Skeptical situation
  **if** User **does not** accept model prediction **then**
    Select one or more counterfactuals $cf \in \mathcal{CF}$ coherent with $y_u$
    $\mathcal{N} \leftarrow \text{LOREGen}(cf)$ {Generate synthetic neighborhood}
    **for** $z \in \mathcal{N}$ **do**
      Assign label $\neq y_u$ to $z$
    **end for**
    $M \leftarrow \text{UpdateModel}(M, \mathcal{N})$
  **else**
    $M \leftarrow \text{UpdateModel}(M, \{(x, \hat{y})\})$
  **end if**
**else**
  $M \leftarrow \text{UpdateModel}(M, \{(x, \hat{y})\})$
**end if**

**return** $M$

# Chapter 4

# EXPERIMENTS

Following the conceptual and algorithmic description of the proposed framework, this chapter presents the experimental protocol designed to assess its effectiveness. The evaluation focuses on three main aspects:

- the quality and diversity of counterfactual explanations generated;
- the predictive performance of the model during and after the iterative learning phase;
- the framework's ability to adapt to user feedback within a human-in-the-loop setting.

To this end, a series of experiments were conducted using multiple datasets, predictive models, and simulated user profiles. The overall design aims to replicate realistic deployment conditions while maintaining control over key experimental variables. Both qualitative and quantitative analyses are provided to support the conclusions drawn.

## 4.1 Experimental setting

After defining the problem and introducing the proposed solution, this section outlines the tools and methodology employed to experimentally evaluate the framework. The objective is to produce reliable results concerning the quality of counterfactual explanations, the model's effectiveness in user interaction, and its adaptive behavior throughout the iterative cycle.

To simulate a realistic human-centered usage context, four user profiles were defined, differentiated by their levels of *believability* (trust in the model) and *expertise* (domain competence):

- User A: believability = 0.8, expertise = 0.8

- User B: believability = 0.5, expertise = 0.8
- User C: believability = 0.8, expertise = 0.5
- User D: believability = 0.5, expertise = 0.5

Subsequently, each dataset $\mathcal{D}$ was partitioned into three subsets:

$$\mathcal{D} = \mathcal{D}_{\text{init}} \cup \mathcal{D}_{\text{iter}} \cup \mathcal{D}_{\text{test}},$$

where:

- $\mathcal{D}_{\text{init}}$ is the **initial training set**, used to train the model and establish a first predictive baseline;
- $\mathcal{D}_{\text{iter}}$ is the **iterative training set**, used to simulate real-time arrival of data and user interaction;
- $\mathcal{D}_{\text{test}}$ is the **final test set**, reserved for performance evaluation.

The data splitting percentages were adapted to the size of the original dataset:

- For **large-scale datasets**, most instances were allocated to the initial training and test sets, while a smaller (but significant) portion was used in the iterative phase to manage computational costs;
- For **smaller datasets**, the split was more balanced to ensure adequate representativeness in each phase.

To ensure a realistic experimental context, both balanced and imbalanced datasets were used. In the latter case, a historical log-guided *under-sampling* strategy was adopted to preserve class distribution consistency over time. This choice was motivated by the need to avoid *oversampling* techniques, which could generate artificial and implausible explanations as demonstrated in [69].

**Predictive Models**

The models selected for comparison are all online and suitable for incremental learning:

- Extremely Fast Decision Tree (EFDT)
- Hoeffding Adaptive Tree (HAT)

- Self-Generating Tree Classifier (SGT)
- AdaBoost
- Bagging
- ADWINBoost

All models were implemented using the RIVER library in Python. For each decision tree, the parameter GRACE_PERIOD was set to 10 in order to enable frequent updates, which are particularly useful in scenarios affected by concept drift.

For imbalanced datasets, only ensemble models were employed, as they demonstrated greater robustness during preliminary analyses. In particular, the SGT classifier was selected as the base learner due to its superior dynamic adaptability compared to more static tree models. For the balanced case, five base models were employed, in order to capture a broader variety of model behaviors and potential decision boundaries. Conversely, for the imbalanced case — given the need to handle larger data volumes and ensure computational feasibility — the number of base models was reduced to three.

During testing, it was observed that models such as EFDT and HAT, even when used as base learners within ensemble methods, fail to correctly update their structure in the presence of imbalanced distributions: the number of splits, the features selected for splitting, and the corresponding thresholds remained unchanged over time. This behavior highlights a limitation of the RIVER library when applied in imbalanced learning contexts. To evaluate model performance, both during and after the iterative interaction with the user, two standard classification metrics were employed: **accuracy** and **F1-score**. These were computed on a test set.

**Counterfactual Explanation Generation**

The following algorithms were adopted for the generation of counterfactual explanations:

- DICE
- GrowingSpheres
- Alibi with prototype-based explanations
- LORE

For prototype-based counterfactuals, the experiments were conducted using k-d trees, as this approach resulted in lower execution times compared to training an encoder-decoder architecture.
LORE deserves special attention for its dual functionality:

- In a separate experiment, during an event of disagreement between the model and the user, it generates a synthetic neighborhood around a counterfactual instance $x'$, which aligns with the user's preference. This set is then used to locally update the model.
- Furthermore, LORE provides both factual and counterfactual rules, which are not only displayed to the user as part of the explanation, but also employed to filter instances in either the historical log or the generated neighborhood. The nearest istance filtered by a counterfactual rule in the generated neighborhood is treated as counterfactual.

The following pseudocode formally describes the decision-making process in the event of a disagreement by using counterfactuals explanations:

**Algorithm 3** Skepticism Management in FRANK with Counterfactual Explanations

**Require:** Instance $x$, model prediction $\hat{y}_M$, user prediction $\hat{y}_U$, skepticism threshold $\tau$
**Ensure:** Final label $y$, updated log (i.e., set of all data $(x, y)$ seen), updated model

**if** $\hat{y}_M = \hat{y}_U$ **then**
  $y \leftarrow \hat{y}_M$ {No skepticism; agreement reached}
**else**
  $score \leftarrow$ ComputeSkepticismScore$(x, \hat{y}_M, \hat{y}_U)$
  **if** $score \geq \tau$ **and** UserRequiresExplanation() **then**
    $cf\_dice \leftarrow$ GenerateDiceExplanation$(x, \hat{y}_M, model)$
    $cf\_gs \leftarrow$ GenerateGSExplanation$(x)$
    $cf\_proto \leftarrow$ GenerateProtoExplanation$(x, model)$
    $rule \leftarrow$ GenerateLoreRule$(x, model)$
    $counter\_rule \leftarrow$ GenerateCounterLoreRule$(x, model)$
    $neighborhood \leftarrow$ GenerateNeighborhood$(x, model)$
    $filtered\_log\_rule \leftarrow$ FilterByRule(log_data, $rule$)
    $filtered\_log\_crule \leftarrow$ FilterByCRule(log_data, $counter\_rule$)
    $filtered\_nb\_rule \leftarrow$ FilterByRule$(neighborhood, rule)$
    $filtered\_nb\_crule \leftarrow$ FilterByCRule$(neighborhood, counter_rule)$
    $cf\_lore \leftarrow$ SelectMostSimilarInstance$(x, filtered_nb_crule)$
    $cf\_set \leftarrow$ CollectAllCounterfactuals()
    **for all** $cf \in cf\_set$ **do**
      CalculateSparsity($x$, $cf$)
      CalculateProximity($x$, $cf$)
      CalculatePlausibility(log_data, $cf$)
      CalculateTemporalValidity($cf$, $cf\_set$)
    **end for**
    **if** UserAccepts($\hat{y}_M$) **then**
      $y \leftarrow \hat{y}_M$
    **else**
      $y \leftarrow \hat{y}_U$
    **end if**
  **else**
    $y \leftarrow \hat{y}_U$ {Strong skepticism not overcome}
  **end if**
**end if**
UpdateModel$(x, y)$
UpdateLog$(x, y)$

**Datasets**

The experimental analysis was conducted using two categories of datasets, representative of both balanced and imbalanced scenarios:

1. Balanced :
    - Phishing : a dataset comprising 1,250 instances and 9 features, in which the classes (phishing vs. non-phishing) are approximately balanced[1].
    - Elec2: a data stream dataset containing 45,312 examples and 8 features, frequently used in studies on concept drift and incremental learning[2].
2. Unbalanced :
    - CreditCard : a fraud detection dataset comprising 284,807 transactions, of which only 492 (0.172%) are fraudulent[3].
    - HTTP : an HTTP traffic stream consisting of 567,498 observations, characterized by a strong predominance of the negative class (0.4%) positive labels.[4].

## 4.2 Experimental results

This section presents and analyzes the results obtained through the experimental evaluation of the proposed framework. The assessment is structured in progressive stages, reflecting the main goals of the study: to evaluate model performance, understand the impact of user interaction on learning, and assess the quality of the counterfactual explanations generated during the process. The structure of the section is organized as follows:

1. Before introducing user simulation and interaction, all datasets were split into three subsets,as described in the previous section:$\mathcal{D}_{\text{init}}$,$\mathcal{D}_{\text{iter}}$ and $\mathcal{D}_{\text{test}}$. In this preliminary phase, where no user interaction or skepticism mechanism is involved, each predictive model is trained incrementally on the available data as follows:

[1] `https://riverml.xyz/0.14.0/api/datasets/Phishing`
[2] `https://riverml.xyz/dev/api/datasets/Elec2`
[3] `https://riverml.xyz/dev/api/datasets/CreditCard`
[4] `https://riverml.xyz/dev/api/datasets/HTTP`

- The model is initially trained on $\mathcal{D}_{\text{init}}$.
- Then, for each incoming instance $x_i \in \mathcal{D}_{\text{iter}}$, the model produces a prediction $\hat{y}_i = M(x_i)$, which is treated as if it were the true label.
- The model is then updated using the pair $(x_i, \hat{y}_i)$ — effectively simulating a scenario where the model "trusts itself" in absence of user feedback.
- At the end of this phase, the accuracy and F1-score were computed on $\mathcal{D}_{\text{test}}$ to determine the best-performing model per dataset. These results serve as a baseline for choosing the model to be used in the subsequent interactive experiments.

The partitioning ratios were adapted to the nature of each dataset to ensure both representativeness and computational feasibility. Specifically:

- **Phishing:**

$$\text{Split: } 30\%\ \mathcal{D}_{\text{init}}, \quad 40\%\ \mathcal{D}_{\text{iter}}, \quad 30\%\ \mathcal{D}_{\text{test}}$$

- **Elec2:**

$$\text{Split: } 69.2\%\ \mathcal{D}_{\text{init}}, \quad 0.08\%\ \mathcal{D}_{\text{iter}}, \quad 30\%\ \mathcal{D}_{\text{test}}$$

- **HTTP & CreditCard (imbalanced datasets):** A custom strategy was adopted to ensure both representativeness and computational feasibility:

$$\text{Split: } 20\%\ \mathcal{D}_{\text{init}}, \quad 40\%\ \mathcal{D}_{\text{iter}}, \quad 40\%\ \mathcal{D}_{\text{test}}$$

  Within $\mathcal{D}_{\text{iter}}$, a balanced sample of classes was enforced:

  - 900 instances from the majority class;
  - 100 instances from the minority class;

  The remaining minority class samples were added to $\mathcal{D}_{\text{test}}$ to preserve diversity and enable a reliable generalization assessment.

This partitioning scheme was consistently applied throughout the evaluation, both in the *preliminary streaming phase* (without user interaction) and in the *skeptical learning phase* (with active user feedback and model correction).

2. After choosing the best models ,the quality of generated counterfactuals is examined throughout the skepticism iterations using key metrics: sparsity, proximity, and plausibility. These are illustrated through line plots, highlighting how explanations evolve across interactions and user profiles. To ensure comparability across different methods and generation rounds, proximity and plausibility metrics were scaled to a normalized range. This allows for the relative performance trends to be compared, regardless of the original feature scales. In order to generate a sufficiently large and meaningful number of counterfactuals — while also accounting for the user's level of criticality — a skepticism threshold greater than 50% was adopted across all datasets.

3. Additional results include a comparative table of execution times for each explanation metmethod a plot showing the evolution of the skepticism score across rounds. These contribute to evaluating both system responsiveness and user-model dynamics.

4. At the end accuracy and F1-scores are reported , highlighting the model's ability to adapt over time in response to user feedback.

The following sections of this chapter present the actual results of the experiments conducted using the previously described methodology. Specifically, the analysis will include:

- The **initial performance** (accuracy and F1-score) of the models obtained **without any user involvement**, reported in tabular form for each dataset $\mathcal{D}$.
- For each simulated user profile and dataset, dedicated subsections will illustrate:
  - Line plots showing the evolution of counterfactual instances in terms of **proximity**, **sparsity**, and **plausibility**, for each counterfactual generation method across the different stages of skepticism;
  - The **evolution of the skepticism score** throughout the interactions, to analyze the dynamics of user trust;
  - A comparative table reporting the **execution times** of each counterfactual explanation method, including total generation time.

  Each analysis concludes with the **final performance scores** (accuracy and F1-score) obtained by the model after incorporating user feedback.

- In the subsection dedicated to the final experiment—focused on model correction through the LORE tool—the discussion will center on the evolution of skepticism and the final model performance. This experiment will consider a simulated expert user with low trust in the system, allowing for a comparative analysis with the same scenario conducted **without** the aid of the tool.

This structure aims to guide the reader through the interpretation of the many plots and tables, highlighting the most relevant insights and supporting a comprehensive comparison of the different methods in terms of explanation quality, user interaction dynamics, and computational efficiency.

### 4.2.1 Static model performance

Before analyzing the impact of user interaction, the predictive models are evaluated independently on each dataset. The purpose of this evaluation is twofold:

1. To compare model effectiveness in standard conditions and identify the most promising candidates for the interactive phase;

2. To establish a performance baseline for assessing the benefit of human-in-the-loop updates.

The tables below report the accuracy and F1-score for each model, evaluated on a fixed test set:

| Model | F1 | Accuracy |
|---|---|---|
| **HoeffdingAdaptiveTreeClassifier** | 0.8556739210944818 | 0.8586666666666667 |
| **SGTClassifier** | 0.8635132774283718 | 0.8666666666666667 |
| **AdaBoostClassifier(SGTClassifier)** | 0.8635132774283718 | 0.8666666666666667 |
| **BaggingClassifier(SGTClassifier)** | 0.8608979627461106 | 0.864444 |
| **ADWINBoostingClassifier(SGTClassifier)** | 0.8556739210944818 | 0.8586666666666667 |
| **ExtremlyFastDecisionTreeClassifier** | 0.8607976480370613 | 0.8666666666666667 |

Table 4.1: Predictive performance of models on the Phishing dataset (Accuracy and F1-score)

| Model | F1 | Accuracy |
|---|---|---|
| **HoeffdingAdaptiveTreeClassifier** | 0.6015073503836388 | 0.6068182542972861 |
| **SGTClassifier** | 0.6153035039691874 | 0.6926683037280352 |
| **AdaBoostClassifier(SGTClassifier)** | 0.667713413909252 | 0.7190738909972255 |
| **BaggingClassifier(SGTClassifier)** | 0.5788972234909957 | 0.6748732340466244 |
| **ADWINBoostingClassifier(SGTClassifier)** | 0.4587050279602993 | 0.5671142009758586 |
| **ExtremelyFastDecisionTreeClassifier.** | 0.5603452036696217 | 0.5664167721416286 |

Table 4.2: Predictive performance of models on Elec2 dataset (Accuracy and F1-score)

| Model | F1 | Accuracy |
|---|---|---|
| AdaBoostClassifier(SGTClassifier) | 0.9964083559858418 | 0.9998419888338776 |
| BaggingClassifier(SGTClassifier) | 0.4971839192076131 | 0.9887987640015445 |
| ADWINBoostingClassifier(SGTClassifier) | 0.4971839192076131 | 0.9887987640015445 |

Table 4.3: Predictive performance of models on the HTTP dataset (Accuracy and F1-score)

| Model | F1 | Accuracy |
|---|---|---|
| AdaBoostClassifier(SGTClassifier) | 0.5141477622168725 | 0.9965970285393534 |
| BaggingClassifier(SGTClassifier) | 0.7367682572774206 | 0.9974390008595134 |
| ADWINBoostingClassifier(SGTClassifier) | 0.4991346060920217 | 0.9965444052693434 |

Table 4.4: Predictive performance of models on the CreditCard dataset (Accuracy and F1-score)

Based on the test performance the experiments were conducted using the following models for each dataset:

- Phishing: AdaBoost
- Elec2: AdaBoost
- HTTP: AdaBoost
- Credit Card: Bagging

### 4.2.2 User A Phishing

In this user scenario, LORE consistently achieves the best performance in terms of proximity across the majority of rounds(see Figure 4.1), producing counterfactuals that remain significantly closer to the original instances compared to the other methods. GrowingSphere and DICE exhibit moderate and relatively stable proximity values throughout the rounds. Conversely, prototype-based methods frequently display much higher proximity values, with notable peaks that set them apart from the other techniques.

When considering plausibility, the prototype-based methods tend to generate counterfactuals with very low plausibility scores, particularly during the initial and final phases of the interaction. This result may reflect a trade-off with their typically higher proximity values. In contrast, LORE and DICE exhibit more stable and balanced behavior, with DICE generally achieving the highest plausibility across the rounds. GrowingSphere, however, often shows the lowest plausibility scores among the compared methods (see Figure 4.2). Sparsity results further confirm the distinct characteristics of each method. DICE achieves the highest sparsity, consistently modifying only a single feature, with rare instances where two features are

changed. LORE also maintains low sparsity, typically introducing only minimal changes, though occasionally modifying five or six features. Prototype-based methods, as well as GrowingSphere, produce counterfactuals that involve a substantially higher number of feature modifications. In particular, GrowingSphere systematically alters nearly all features in each instance, consistent with its design(see Figure 4.3).

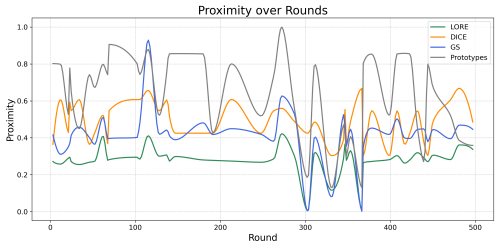


Figure 4.1: Evolution of proximity – *Phishing*, User A

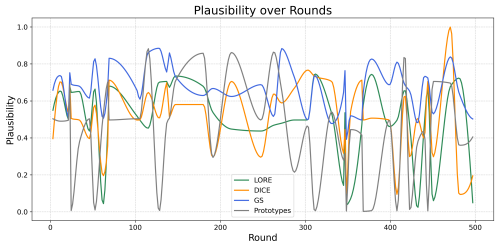


Figure 4.2: Evolution of plausibility – *Phishing*, User A

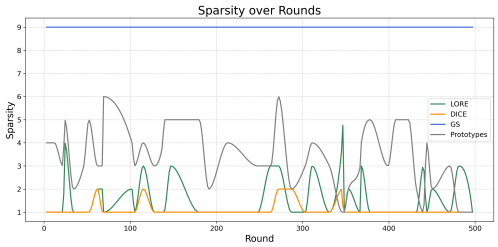


Figure 4.3: Evolution of sparsity – *Phishing*, User A

With regard to execution times(see Table 4.5), the methods show markedly different computational profiles. As evidenced by the data, GrowingSpheres and DICE demonstrate very low execution times, making them suitable for interactive scenarios where rapid response is essential. Conversely, LORE and Prototypes require substantially longer computation times, with Prototypes being the most time-consuming approach. On the other hand, the evolution of user skepticism during the rounds presents a non-monotonic trend(see Figure 4.4), characterized by fluctuations rather than a consistent decline. Starting from an initially high value (approximately 0.85), skepticism decreases during several phases—reaching levels around 0.60— rising again significantly during several rounds, even exceeding 0.90. This behavior can be attributed to the properties of the simulated user, who is defined by both a high level of expertise (80%) and credibility assigned to the model (80%). As a result, the user tends to challenge the model's decisions frequently, due to their expertise, yet still partially trusts the system, which prevents skepticism from stabilizing at lower levels. With respect to performance metrics, the accuracy starts from an initial value of 86.66% and

| | GrowingSphere | DICE | LORE | Prototypes |
|---|---|---|---|---|
| **0** | 0.0327651500701904 | 0.0914449071810737 | 2.911627054214477 | 2.180316686630249 |
| **1** | 0.0317411422729492 | 0.071340852759385 | 2.4817705154418945 | 2.806678533554077 |
| **2** | 0.0349659919738769 | 0.0962249075639647 | 2.749589681625366 | 3.218672513961792 |
| **3** | 0.0350587368011474 | 0.1239390912262467 | 5.616375207901001 | 4.257542133331299 |
| **4** | 0.0384204387664794 | 0.0683049085624754 | 5.381465435028076 | 4.81611967086792 |
| **...** | ... | ... | ... | ... |
| **42** | 0.1187727451324462 | 0.0720568985864827 | 2.738556146621704 | 27.43708848953247 |
| **43** | 0.1231820583343505 | 0.066185194286545 | 2.7542378902435303 | 28.307119131088257 |
| **44** | 0.1273412704467773 | 0.0289069507187539 | 2.8533709049224854 | 9.738186120986938 |
| **45** | 0.1250913143157959 | 0.0529274498661531 | 2.4637486934661865 | 9.940947532653809 |
| **Sum** | 3.483048200607298 | 3.1521667618403204 | 170.89005136489868 | 588.8097553253174 |

Table 4.5: Execution times on *Phishing* (User A)

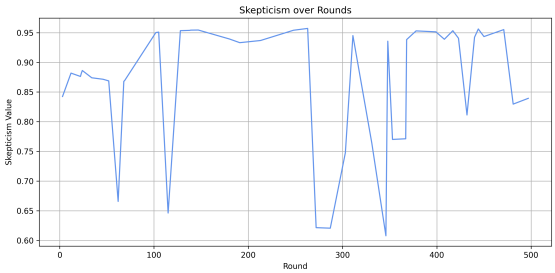


Figure 4.4: Evolution of skepticism – *Phishing*, User A

gradually improves over the iterations, reaching a final accuracy of 90%. Similarly, the F1 score starts from 86% and improves to 89% by the end of the process. This highlights a clear enhancement of the initial performance, achieved thanks to the user's interaction, compared to the baseline scenario without any user involvement.

### 4.2.3 User B Phishing

As shown in Figure 4.5 , the LORE method consistently achieves significantly lower proximity values across most rounds.The GrowingSphere method starts with higher proximity values during the initial rounds but progressively decreases over time, reaching low values in certain cases. In contrast, the DICE method maintains relatively stable proximity values across rounds, generally remaining within a mid-range. Counterfactuals generated using prototype-based methods exhibit consistently higher proximity values. Regarding plausibility(see Figure 4.6), GrowingSphere achieves both the highest and lowest scores across rounds, with higher values typically observed in early rounds and lower values towards the end. On average, plausibility scores are relatively comparable across the different methods, although prototype-based methods tend to achieve slightly higher values. An interesting observation is that, in the final rounds, most methods display a marked decrease in both proximity and plausibility scores. This may indicate that, through successive iterations, the system has progressively improved its ability to generate counterfactuals that are both closer to the original instances and more plausible. In terms of sparsity(see Figure 4.7), both the DICE and LORE methods consistently achieve low values, typically modifying only a small number of features (in the case of DICE, often between one and two features). In contrast, counterfactuals generated using prototype-based methods rarely achieve low sparsity, generally requiring changes to a larger number of features. As expected, the GrowingSphere method systematically modifies nearly all features, due to its

generation strategy.

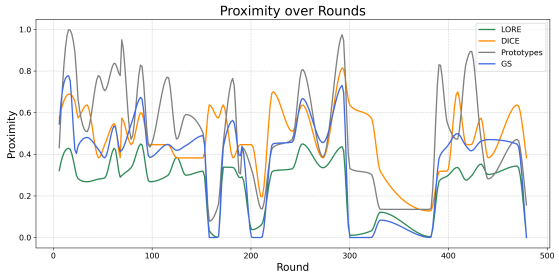


Figure 4.5: Evolution of proximity – *Phishing*, User B

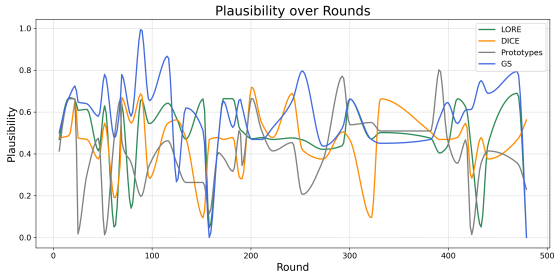


Figure 4.6: Evolution of plausibility – *Phishing*, User B

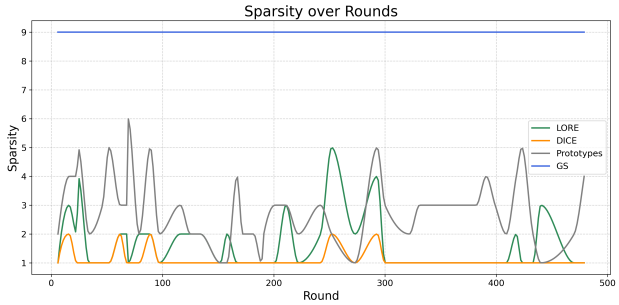


Figure 4.7: Evolution of sparsity – *Phishing*, User B

The evolution of skepticism across rounds is characterized by significant fluctuations during the initial interactions(see Figure 4.8). Starting from values slightly above 80%, skepticism decreases to around 60%, rises again toward 90%, and subsequently oscillates in a similar manner. After approximately 100 iterations, skepticism gradually stabilizes, maintaining values between 70% and 80%. This observed trend suggests that a high level of expertise (0.8) tends to drive the user toward higher skepticism levels (typically between 70% and 80%), as an experienced user is more capable of critically evaluating the model's outputs. At the same time, the medium believability score (0.5) prevents the user from either fully trusting or completely rejecting the counterfactuals and explanations provided by the system. This balanced perspective likely contributes to the stabilization of skepticism over time, avoiding extreme oscillations and leading to a more consistent evaluation behavior. In terms of execution time (see Table 4.6), the GrowingSphere and DICE methods are the fastest, requiring approximately 8 to 9 seconds in total to resolve each moment of skepticism. The LORE method is noticeably slower, with a total execution time of around 158 seconds. Prototype-based methods exhibit the longest execution times, exceeding 1000 seconds in total.

| | GrowingSphere | DICE | LORE | Prototypes |
|---|---|---|---|---|
| 0 | 0.1295332908630371 | 0.2366292037803533 | 2.3896279335021973 | 9.790966033935549 |
| 1 | 0.1313531398773193 | 0.2052081279874932 | 5.514675617218018 | 29.51377868652344 |
| 2 | 0.1476912498474121 | 0.2440999731669844 | 5.682497024536133 | 31.95142889022827 |
| 3 | 0.1417028903961181 | 0.2874150899650347 | 5.844188928604126 | 33.66842794418335 |
| 4 | 0.1556451320648193 | 0.2004631829772429 | 2.5817174911499023 | 11.544197797775269 |
| ... | ... | ... | ... | ... |
| 39 | 0.2535595893859863 | 0.2217913755451204 | 3.013669729232788 | 17.225842475891113 |
| 40 | 0.2580933570861816 | 0.2485920054807013 | 5.471231937408447 | 17.281646251678467 |
| 41 | 0.2575232982635498 | 0.2205298901263923 | 3.0762314796447754 | 17.356639862060547 |
| 42 | 0.3207976818084717 | 0.2063272520075776 | 2.4153406620025635 | 17.127222537994385 |
| Sum | 9.1181480884552 | 8.724689084437049 | 158.14203763008118 | 1105.3431687355042 |

Table 4.6: Execution times on *Phishing* (User B)

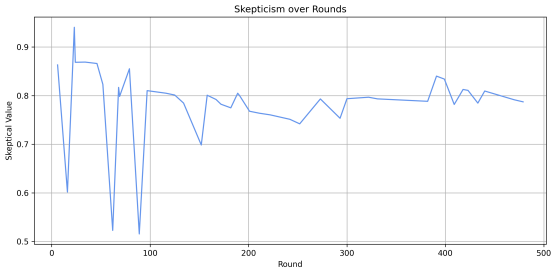


Figure 4.8: Evolution of skepticism– *Phishing*, User B

With respect to performance metrics, the accuracy starts from an initial value of 86.66%, increases in some instances up to 90%, and eventually stabilizes around 88%. Similarly, the F1 score begins at 86%, reaches 88%, briefly returns to 86%, and then stabilizes again at 88%. This indicates an overall improvement in performance compared to the initial scenario without user interaction.

### 4.2.4 User C Phishing

In this user scenario, LORE once again demonstrates superior performance in terms of proximity(see Figure 4.9), consistently achieving the lowest values and clearly distinguishing itself from all other methods. DICE and GrowingSphere display similar behaviors, with fluctuating proximity trends across iterations, while prototype-based methods continue to generate counterfactuals with comparatively high proximity. In terms of plausibility (see Figure 4.10), prototype-based methods reach the lowest scores overall, with LORE maintaining consistently low values as well. DICE tends to produce plausibility values around the average, whereas GrowingSphere exhibits the highest plausibility scores. Regarding sparsity(see Figure 4.11), DICE shows excellent performance, typically modifying only very few features ( one or two), and LORE remains similarly efficient. In contrast, prototype-based methods require modifications to a significantly larger number of features, while GrowingSphere, as expected, alters nearly all available features due to its inherent nature.

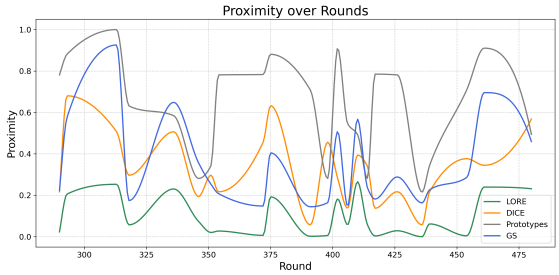


Figure 4.9: Evolution of proximity – *Phishing*, User C

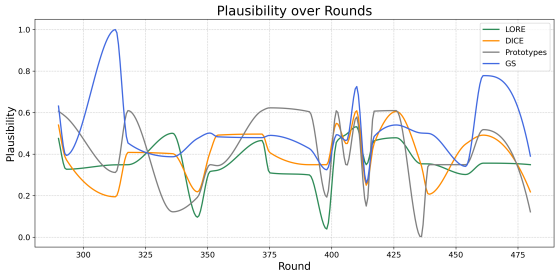


Figure 4.10: Evolution of plausibility – *Phishing*, User C

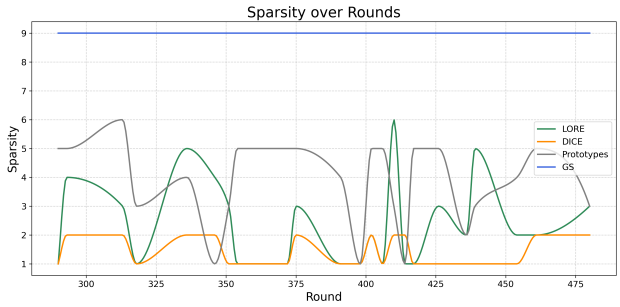


Figure 4.11: Evolution of sparsity – *Phishing*, User C

Execution times follow the patterns observed in previous analyses(see Table 4.8): DICE and GrowingSphere remain the fastest, LORE performs at a good pace, while prototype-based methods remain considerably slower. A particularly interesting observation in this scenario concerns the evolution of user skepticism(see Figure 4.16)). Here, skepticism grows gradually and incrementally — starting from around 51% and reaching approximately 59% — though remaining noticeably lower than in previous user cases. This trend can be interpreted in light of the user profile's characteristics, which are defined by a high believability toward the system and low expertise . This combination describes a relatively inexperienced user with a strong predisposition to trust the system, resulting in lower levels of skepticism and a more gradual increase as iterations progress.

| | GrowingSphere | DICE | LORE | Prototypes |
|---|---|---|---|---|
| 0 | 0.0313923358917236 | 0.057680130662419844 | 5.065478801727295 | 2.073697090148926 |
| 1 | 0.0322356224060058 | 0.049221412206311245 | 4.7870941162109375 | 2.571575403213501 |
| 2 | 0.0346863269805908 | 0.05969130097716723 | 4.8627636432647705 | 3.4362988471984863 |
| 3 | 0.0376801490783691 | 0.0713519576747149 | 2.467597484588623 | 3.864832162857056 |
| 4 | 0.0367712974548339 | 0.04794404811489638 | 4.907405614852905 | 4.369758129119873 |
| 5 | 0.0389642715454101 | 0.04794426682047346 | 5.118749141693115 | 5.075977325439453 |
| 6 | 0.0415618419647216 | 0.07210038595589888 | 2.4725193997735596 | 5.7532219886779785 |
| 7 | 0.0415565967559814 | 0.061286472135543425 | 2.5601649284362797 | 6.330151796340942 |
| 8 | 0.0446441173553466 | 0.04480927374966414 | 2.631962776184082 | 7.093559980392456 |
| 9 | 0.0443866252899169 | 0.05829085607867124 | 5.175671577453613 | 7.791061639785767 |
| 10 | 0.0502407550811767 | 0.0448899565847442 | 2.6275277137756348 | 8.407564640045166 |
| 11 | 0.0508291721343994 | 0.044859157002687854 | 2.676964521408081 | 9.33905577659607 |
| 12 | 0.0504696369171142 | 0.05428651245834479 | 5.04172945022583 | 10.112552404403688 |
| 13 | 0.0544769763946533 | 0.0255759482977927 05 | 2.5438485145568848 | 10.560664176940918 |
| 14 | 0.0563583374023437 | 0.02808517455822946 | 5.347852945327759 | 11.397639036178589 |
| 15 | 0.0570380687713623 | 0.04357288495083011 | 5.128779649734497 | 12.152721643447876 |
| 16 | 0.0595064163208007 | 0.03757107281978607 | 2.5647666454315186 | 12.42696738243103 |
| 17 | 0.0631630420684814 | 0.055249441162148194 | 2.2349629402160645 | 12.923559665679932 |
| 18 | 0.0656371116638183 | 0.03896723560051098 | 5.135103464126587 | 13.540266275405884 |
| 19 | 0.0658397674560546 | 0.03224959132483971 | 2.600393056869507 | 14.383044958114624 |
| 20 | 0.0702190399169921 | 0.07058756880363368 | 2.5384750366210938 | 14.919869661331177 |
| 21 | 0.0713238716125488 | 0.048055641369735634 | 5.140568733215332 | 15.647621870040894 |
| 22 | 0.0754737854003906 | 0.05196282942444444 | 5.382441282272339 | 16.412907123565674 |
| Sum | 1.1744551658630358 | 1.1462331187334887 | 89.01282143592834 | 210.58456897735596 |

Table 4.7: Execution times on *Phishing* (User C)

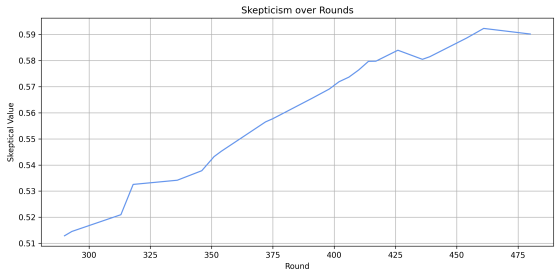


Figure 4.12: Evolution of skepticism– *Phishing*, User C

With respect to performance metrics, the accuracy starts at 86% and progressively degrades to 64%. Similarly, the F1 score begins at 85%, then decreases to 80%, subsequently drops to 70%, and eventually reaches 54%. This behavior is expected, given the low expertise of the user.

### 4.2.5 User D Phishing

In this scenario, LORE once again demonstrates the best performance in terms of proximity(see Figure 4.13), consistently achieving lower values than all other methods. Both GrowingSphere and DICE show comparable trends with similar fluctuations throughout the iterations, while prototype-based methods perform the worst in terms of proximity. When considering plausibility(see Figure 4.14), prototype-based methods again reach the lowest scores, particularly during the initial and final stages. LORE and DICE exhibit similar, stable trends, whereas GrowingSphere performs the worst in this regard, producing plausibility scores significantly lower than the other methods. Sparsity results remain consistent with previous observations(see Figure 4.15): DICE achieves optimal sparsity by modifying only a single feature across nearly all instances (with rare exceptions where two features are modified). LORE also maintains low sparsity, typically altering very few features (with occasional peaks where five or six features are changed). In contrast, prototype-based methods and GrowingSphere exhibit poor sparsity, as they tend to modify a large number of features.

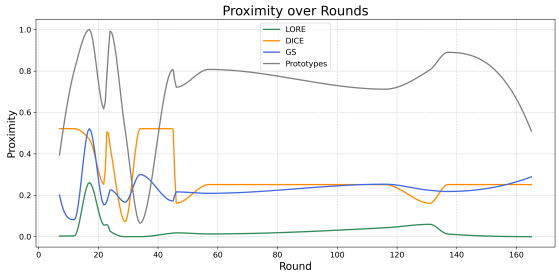


Figure 4.13: Evolution of proximity – *Phishing*, User D

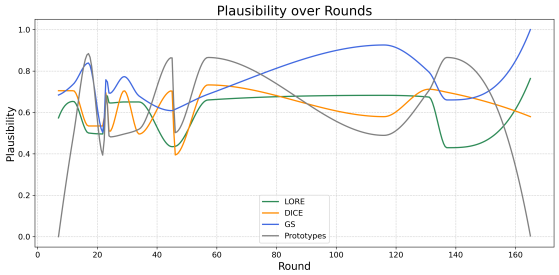


Figure 4.14: Evolution of plausibility – *Phishing*, User D

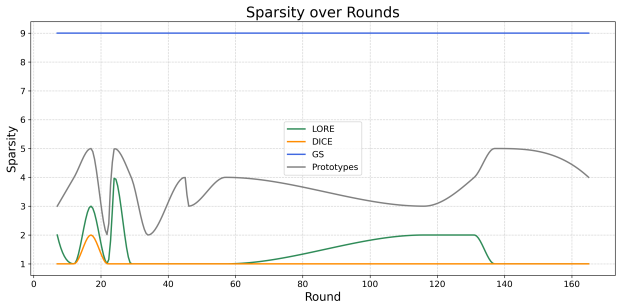


Figure 4.15: Evolution of sparsity – *Phishing*, User D

Execution times follow the same trends observed in previous cases(see Table 4.8): DICE and Growing-Sphere remain the fastest, followed by LORE with moderate execution time, while prototype-based methods continue to require significantly longer processing times. The evolution of user skepticism in this scenario displays a decreasing trend (see Figure 4.16), starting from approximately 85% and gradually stabilizing around 50%. This behavior can be explained by the characteristics of the user profile, which combines medium levels of expertise (0.5) and believability (0.5). Such a profile reflects a user who initially exhibits high skepticism but, through interaction with the system, adjusts their stance toward a more balanced and neutral level of skepticism.

| | GrowingSphere | DICE | LORE | Prototypes |
|---|---|---|---|---|
| 0 | 0.0777275562286377 | 0.10736651113682809 | 2.5636887550354004 | 17.56893539428711 |
| 1 | 0.0809240341186523 | 0.09805054000291787 | 2.1903953552246094 | 18.055743932724 |
| 2 | 0.0820996761322021 | 0.10958150442390614 | 4.806715726852417 | 18.626711606997937 |
| 3 | 0.0872118473052978 | 0.12242391568992267 | 2.5676610469818115 | 19.389503002166748 |
| 4 | 0.089876651763916 | 0.09664372086537812 | 5.3307411670684814 | 21.239134788513184 |
| 5 | 0.0950691699981689 | 0.0966439617357716 | 4.978014230728149 | 21.67958950996399 |
| 6 | 0.0979771614074707 | 0.12324819377643076 | 5.014715909957886 | 22.008766174316406 |
| 7 | 0.0998198986053466 | 0.11133833893047929 | 2.6845951080322266 | 7.533104658126831 |
| 8 | 0.1032555103302002 | 0.09319125136312915 | 4.998496770858765 | 22.954946517944336 |
| 9 | 0.1053023338317871 | 0.1080391307987858 | 2.666815280914306 | 23.80380797386169 |
| 10 | 0.1105477809906005 | 0.09328011103690931 | 2.689833641052246 | 24.747111320495605 |
| 11 | 0.1113402843475341 | 0.09324619005716961 | 5.454967737197876 | 25.24818468093872 |
| 12 | 0.1150188446044921 | 0.10362896514889604 | 4.988768100738525 | 26.54959201812744 |
| 13 | 0.1192357540130615 | 0.07200871577591626 | 5.02630352973938 | 26.69218873977661 |
| 14 | 0.125779390335083 | 0.07477224071553605 | 5.0002121925354 | 27.66003942489624 |
| Sum | 1.501185894012451 | 1.5034632914579762 | 60.96192455291748 | 323.7573597431183 |

Table 4.8: Execution times on *Phishing* (User D)

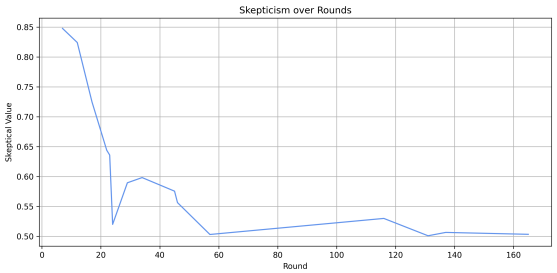


Figure 4.16: Evolution of skepticism– *Phishing*, User D

With respect to performance metrics, the accuracy starts at 86%, then drops sharply and stabilizes between 68% and 69%, with the latter representing the final value. As for the F1 score, it begins at 86% and subsequently decreases to 63%.

### 4.2.6 User A Elec2

In this new dataset scenario, prototype-based methods and DICE exhibit the highest proximity values(see Figure 4.17), consistently generating counterfactuals that remain relatively distant from the original instances. In contrast, LORE and GrowingSphere demonstrate much better proximity performance, frequently producing closer counterfactuals and, at times, displaying nearly identical proximity trends.

With respect to plausibility(see Figure 4.18), prototype-based methods and LORE show very similar and favorable trends, emerging as the best-performing approaches in this regard. GrowingSphere attempts to emulate this behavior but still exhibits occasional peaks in plausibility scores, indicating less consistent performance. DICE, interestingly, achieves the highest plausibility values overall, although these results must be interpreted in relation to its lower proximity scores.

Sparsity patterns remain consistent with prior observations(see Figure 4.19). DICE and LORE produce the most parsimonious counterfactuals, with DICE alternating between modifying one or two features per instance. Conversely, prototype-based methods and GrowingSphere generate counterfactuals that involve significantly more feature changes, resulting in lower sparsity.

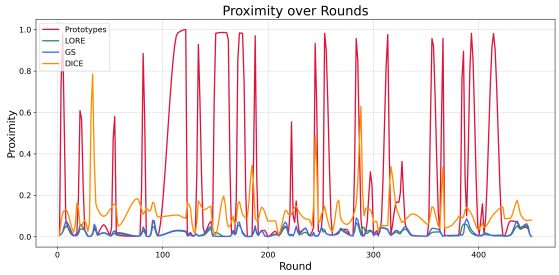


Figure 4.17: Evolution of proximity – *Elec2*, User A

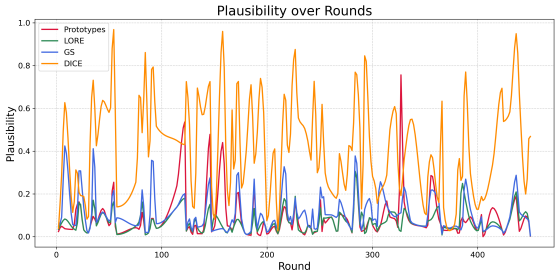


Figure 4.18: Evolution of plausibility – *Elec2*, User A

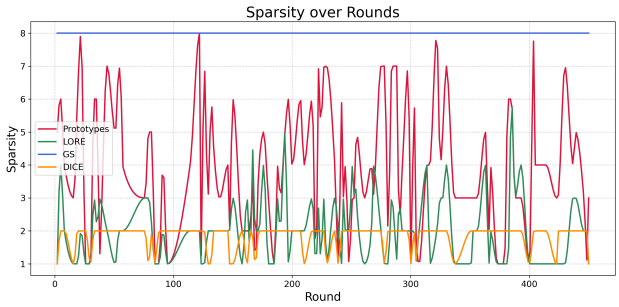


Figure 4.19: Evolution of sparsity – *Elec2*, User A

Execution times remain aligned with previous trends(see Table 4.9): GrowingSphere and DICE maintain the shortest computation times, suitable for interactive scenarios. LORE is slightly slower but still efficient, while prototype-based methods require considerably longer execution times. An interesting dynamic is observed in the evolution of user skepticism (see Figure 4.20). The skepticism level follows a highly fluctuating pattern, oscillating continuously between values above 90% and around 50% throughout the iterations. This behavior is driven by the nature of the user profile, which combines high expertise with strong believability toward the model. Consequently, the user frequently exhibits skepticism due to their expertise but, at the same time, tends to trust the model's responses, preventing skepticism from remaining consistently high or low.

| | GrowingSphere | DICE | LORE | Prototypes |
|---|---|---|---|---|
| **0** | 0.0453789234161376 | 0.4407832668291214 | 2.5588269233703613 | 99.8862509727478 |
| **1** | 0.0531113147735595 | 0.2969458214864591 | 3.448808431625366 | 79.78239297866821 |
| **2** | 0.0559210777282714 | 0.474982490000257 | 3.81825852394104 | 147.10631346702576 |
| **3** | 0.0529978275299072 | 0.6732677360568646 | 2.9126458168029785 | 75.62963271141052 |
| **4** | 0.0660617351531982 | 0.2752247075098047 | 2.955069065093994 | 106.1039412021637 |
| **...** | ... | ... | ... | ... |
| **108** | 0.7990455627441406 | 0.3866070906777486 | 5.085091829299927 | 322.51019382476807 |
| **109** | 0.800377607345581 | 0.3114021758116942 | 3.833357095718384 | 66.19044089317322 |
| **110** | 0.7615857124328613 | 0.1063805658204151 | 2.3784255981445312 | 56.56826138496399 |
| **111** | 0.8247489929199219 | 0.3222599020602624 | 2.665956735610962 | 247.36656665802 |
| **Sum** | 36.76578235626221 | 35.576326641914015 | 362.7141590118408 | 15802.44746875763 |

Table 4.9: Execution times on *Elec2* (User A)

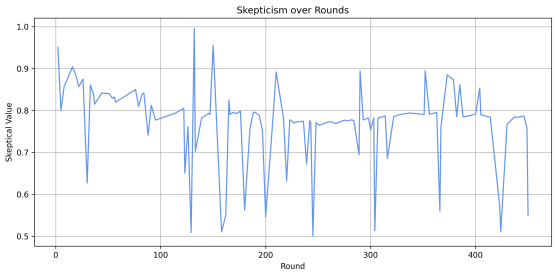


Figure 4.20: Evolution of skepticism– *Elec2*, User A

With regard to accuracy and F1 score, the former, as in the initial scenario without any user involvement, remains at 71%, while the latter remains stable at 66%.

### 4.2.7 User B Elec2

In this user configuration , the results are once again largely consistent with the previously observed trends. Regarding proximity(see Figure 4.21), both LORE and GrowingSpheres demonstrate the best performance, consistently producing counterfactuals that remain close to the original instances and exhibiting very similar trends. DICE shows higher proximity values, clearly deviating from the best-performing methods. Prototypes remains the method with the highest proximity overall, though it occasionally reaches low proximity values comparable to the best methods.

In terms of plausibility(see Figure 4.22), Prototypes and LORE appear to be the most effective methods, generating counterfactuals with high plausibility throughout the rounds. GrowingSpheres also achieves relatively good plausibility, although it exhibits occasional peaks. Conversely, DICE tends to produce counterfactuals with lower plausibility.

Regarding sparsity(see Figure 4.23), LORE and DICE remain the most efficient methods, modifying the smallest number of features. In particular, DICE alternates between modifying just one or two features. Prototypes and GrowingSpheres once again result in counterfactuals that alter a greater number of features, although Prototypes sometimes manages to produce instances with minimal changes.

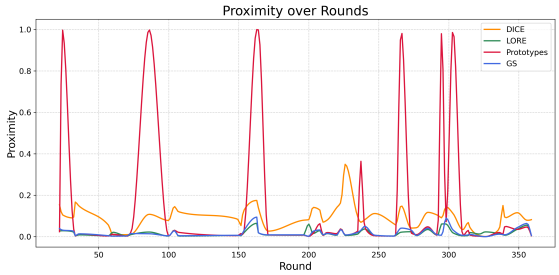


Figure 4.21: Evolution of proximity – *Elec2*, User B

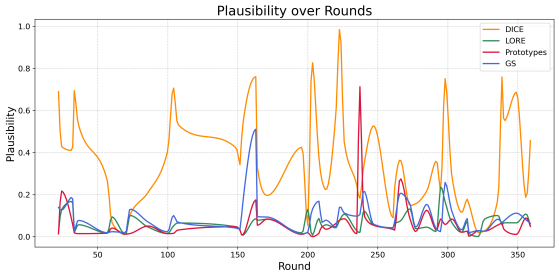


Figure 4.22: Evolution of plausibility – *Elec2*, User B

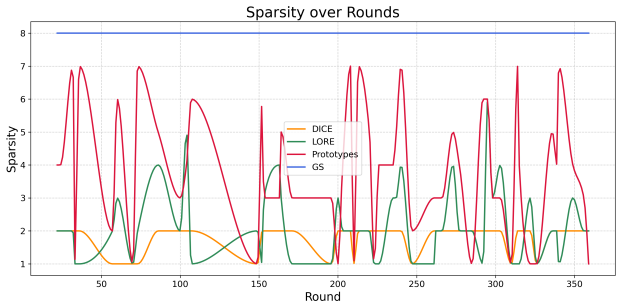


Figure 4.23: Evolution of sparsity – *Elec2*, User B

Execution times follow the same patterns as previously observed(see Table 4.10): GrowingSpheres and DICE remain the fastest methods, while LORE is slightly slower, and Prototypes continues to be the most time-consuming. Turning to the evolution of skepticism (see Figure 4.24) : an oscillating trend can be observed. Initially, skepticism rises, reaching values above 0.85, then it progressively decreases towards 0.60. After this, the pattern stabilizes with alternating increases and decreases between approximately 0.60 and 0.75. This behavior reflects the characteristics of the user: high expertise (0.8) combined with a moderate believability towards the model (0.5). In this configuration, the user, while highly capable of critically evaluating the explanations (due to expertise), maintains a balanced attitude of trust toward the system. This leads to skepticism that fluctuates rather than stabilizing at consistently low or high levels. In this case, both accuracy and F1 score remain fully consistent with the initial results observed without user involvement, confirming the stability of the model's performance

| | GrowingSphere | DICE | LORE | Prototypes |
|---|---|---|---|---|
| **0** | 0.0527036190032959 | 0.1277720063525366 | 4.507094621658325 | 83.91810059547424 |
| **1** | 0.0484721660614013 | 0.1004778171109014 | 3.2439498901367188 | 26.2456476688385 |
| **2** | 0.0525844097137451 | 0.1342615547933211 | 2.8032212257385254 | 80.6694540977478 |
| **3** | 0.0491793155670166 | 0.1718876056231327 | 2.364407539367676 | 20.190219163894653 |
| **4** | 0.0552811622619628 | 0.0963560795897325 | 2.697443962097168 | 76.09793186187744 |
| **...** | ... | ... | ... | ... |
| **49** | 0.1779248714447021 | 0.030637747188145 | 2.445678949356079 | 108.7683608531952 |
| **50** | 0.1718733310699463 | 0.1203515956765365 | 5.128350496292114 | 119.8752257823944 |
| **51** | 0.1802961826324463 | 0.089868486055049 | 4.0570762157440186 | 40.22289991378784 |
| **52** | 0.1745681762695312 | 0.0773239209750418 | 2.9273486137390137 | 34.30518627166748 |
| **Sum** | 5.640314579010008 | 5.124002086170658 | 170.71632146835327 | 4171.111896514893 |

Table 4.10: Execution times on *Elec2* (User B)

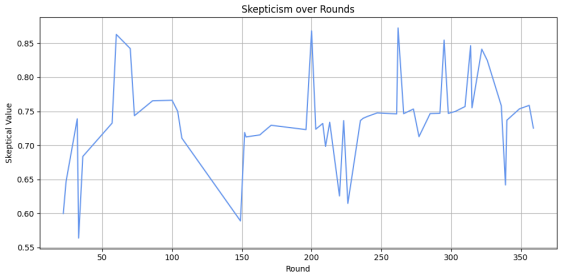


Figure 4.24: Evolution of skepticism– *Elec2*, User B

### 4.2.8 User C Elec2

In this scenario, the observations concerning proximity, plausibility, sparsity, and execution time remain consistent with the trends identified in earlier experiments on this dataset.

Specifically, in terms of proximity(see Figure 4.25), LORE and GrowingSpheres continue to be the best-performing methods, consistently generating counterfactuals close to the original instances. DICE occasionally produces higher proximity values, while Prototypes exhibits the highest peaks in proximity, though it also frequently aligns with the best methods, demonstrating moments of strong performance.

Regarding plausibility(see Figure 4.26), Prototypes and LORE again prove to be the most effective, maintaining high plausibility throughout the rounds. GrowingSpheres performs relatively well, although a few small peaks can still be observed. Conversely, DICE shows higher fluctuations and elevated peaks in plausibility, indicating less stability.

Sparsity results also follow previous trends(see Figure 4.27): DICE continues to modify the fewest features, often alternating between one and two modifications. LORE introduces slightly more feature changes but remains efficient. Prototypes and GrowingSpheres alter a significantly larger number of features, as consistently observed in earlier cases.

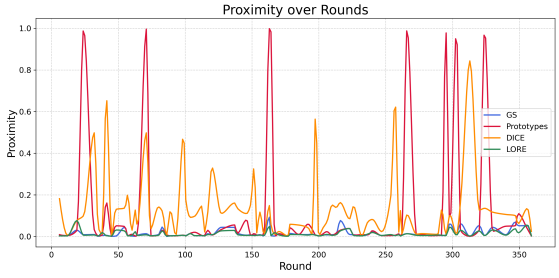


Figure 4.25: Evolution of proximity – *Elec2*, User C

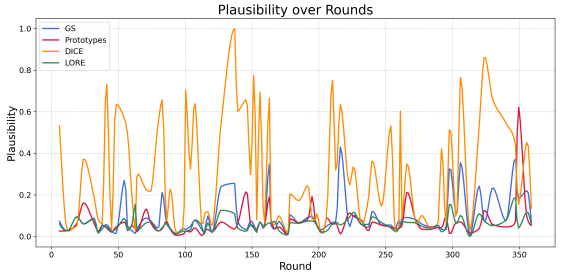


Figure 4.26: Evolution of plausibility – *Elec2*, User C

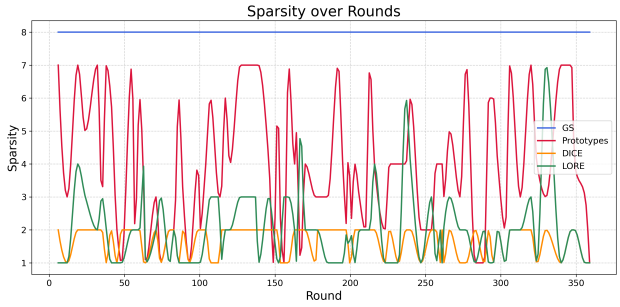


Figure 4.27: Evolution of sparsity – *Elec2*, User C

Execution times show no deviation from prior observations(see Table 4.11): GrowingSpheres and DICE remain the fastest methods, LORE is moderately slower, and Prototypes remains the most computationally expensive. The evolution of user skepticism(see Figure 4.28), in this configuration, is particularly noteworthy. Starting from an initial value above 90% (though below 100%), skepticism decreases progressively to around 60%, then oscillates between 60% and 70% — at times dropping as low as 50% — and stabilizes between 60% and 70% beyond 200 iterations. This dynamic can be explained by the nature of the user: possessing limited expertise (0.5), but a high believability towards the model (0.8). Consequently, the user initially questions the model but gradually tends to trust it more consistently as the interaction progresses, leading to a stabilization of skepticism at intermediate values.

| | GrowingSphere | DICE | LORE | Prototypes |
|---|---|---|---|---|
| **0** | 0.1871023178100586 | 0.5491370039739256 | 2.528474807739258 | 113.58986592292786 |
| **1** | 0.1903285980224609 | 0.4357229891388945 | 3.1974411010742188 | 123.27069640159608 |
| **2** | 0.1943902969360351 | 0.5761026608989189 | 2.7764546871185303 | 127.57317042350768 |
| **3** | 0.211925983428955 | 0.7324480811353407 | 3.685026168823242 | 46.2561194896698 |
| **4** | 0.2165071964263916 | 0.4185961642226639 | 3.621765375137329 | 126.00689816474916 |
| **...** | ... | ... | ... | ... |
| **82** | 0.7631044387817383 | 0.7243863373388086 | 5.036540269851685 | 375.0197722911835 |
| **83** | 0.767186164855957 | 0.3678498251980011 | 6.491708517074585 | 294.6524999141693 |
| **84** | 0.8250267505645752 | 0.3160127919381963 | 4.129286766052246 | 74.42103552818298 |
| **85** | 0.7685739994049072 | 0.3707992566737127 | 2.5339224338531494 | 61.253971576690674 |
| **Sum** | 39.59598994255066 | 37.80337003447868 | 298.86106133461 | 13255.522666454315 |

Table 4.11: Execution times on *Elec2* (User C)

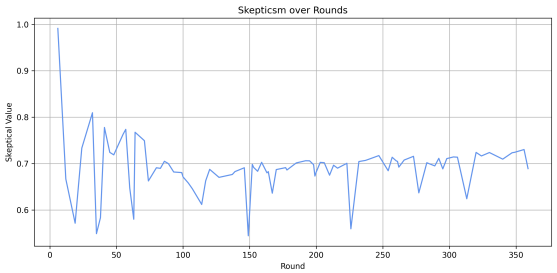


Figure 4.28: Evolution of skepticism– *Elec2*, User C

Despite the user's limited expertise, accuracy and F1 score remain unchanged compared to the initial scenario, indicating that the user interaction did not negatively impact the model's performance

### 4.2.9 User D Elec2

In this scenario, the overall trends observed remain consistent with those from previous cases on the same dataset.

Regarding proximity (see Figure 4.29), LORE and GrowingSpheres again achieve the lowest values, indicating the generation of counterfactuals close to the original instances. DICE performs moderately, with slightly higher proximity, while Prototypes consistently exhibits the highest peaks in proximity across rounds.

In terms of plausibility(see Figure 4.30), Prototypes and LORE continue to be the most effective methods. GrowingSpheres maintains relatively low plausibility values, although it slightly diverges from the best-performing methods. DICE, while still the method with the highest plausibility values in this scenario, shows some improvement compared to previous cases, with its plausibility becoming somewhat closer to the underlying data distribution.

Observations regarding sparsity and execution times remain consistent with earlier scenarios(see Figure 4.31): DICE modifies the fewest features, LORE introduces slightly more changes, and Prototypes and GrowingSpheres modify a larger number of features. GrowingSpheres and DICE remain the fastest methods, followed by LORE, with Prototypes continuing to be the most computationally expensive.

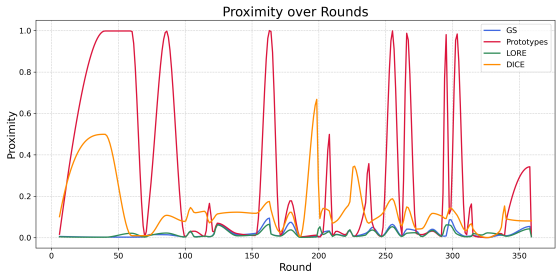


Figure 4.29: Evolution of proximity – *Elec2*, User D

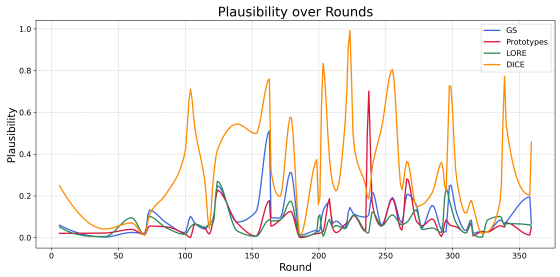


Figure 4.30: Evolution of plausibility – *Elec2*, User D

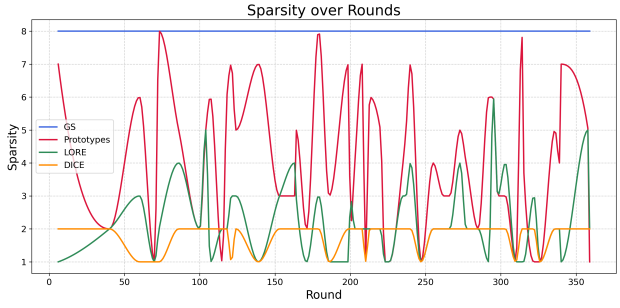


Figure 4.31: Evolution of sparsity – *Elec2*, User D

The evolution of user skepticism follows a familiar pattern(see Figure 4.32): starting from an initial value

above 90%, skepticism stabilizes between 50% and 70% in subsequent rounds. This behavior reflects the characteristics of the user, who possesses both limited expertise (0.5) and low believability in the model (0.5). Such a profile leads to an initial phase of high skepticism that gradually stabilizes at intermediate levels as interaction progresses.

| | GrowingSphere | DICE | LORE | Prototypes |
|---|---|---|---|---|
| 0 | 0.0542147159576416 | 0.166550743635025 | 2.9218883514404297 | 83.6490626335144 |
| 1 | 0.057518720626831 | 0.1276646247421308 | 2.877521514892578 | 16.212088584899902 |
| 2 | 0.0567901134490966 | 0.1757964240981983 | 2.782621622085572 | 71.57242822647095 |
| 3 | 0.0565624237060546 | 0.2294023793103148 | 2.8446640968322754 | 54.4183521270752 |
| 4 | 0.0536742210388183 | 0.1217923720851812 | 2.784111261367798 | 85.5128829479217 5 |
| ... | ... | ... | ... | ... |
| 49 | 0.2330341339111328 | 0.0281632532794613 | 2.3353283405303955 | 93.83380436897278 |
| 50 | 0.2429807186126709 | 0.1559788607618425 | 2.3615219593048096 | 156.88998174667358 |
| 51 | 0.2730824947357178 | 0.1125494767829957 | 4.682828903198242 | 170.66839575767517 |
| 52 | 0.2429604530334472 | 0.0946771947070477 | 2.3226542472839355 | 30.505219221115112 |
| Sum | 7.214992761611937 | 6.479400698235644 | 166.90523886680603 | 4351.528907299042 |

Table 4.12: Execution times on *Elec2* (User D)

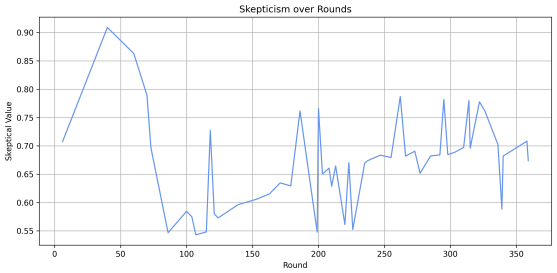


Figure 4.32: Evolution of skepticism– *Elec2*, User D

Accuracy and F1 score show no variation from the baseline, confirming the robustness of the model regardless of user input in this context

### 4.2.10 User A HTTP

In this new dataset, the behavior of the methods shows several consistent trends with previous experiments, though the new data distribution leads to some distinct patterns.

Looking at proximity (as shown in Figure 4.33), all methods fluctuate with a similar trend across the rounds. LORE and GrowingSpheres generally achieve lower proximity values compared to DICE and Prototypes, confirming their ability to generate closer counterfactuals. DICE displays more variability but still performs moderately well. Prototypes remains the method with the highest peaks, although in certain iterations it achieves proximity values comparable to the best-performing methods.

Regarding plausibility(see Figure 4.34), DICE is the method that consistently achieves the lowest values, distinguishing itself from the other techniques. Prototypes follow in second place, and LORE ranks third. In only a few cases do LORE and Prototypes reach plausibility levels similar to DICE. GrowingSpheres is the method that most frequently produces significant peaks in plausibility across iterations, indicating less plausible counterfactuals in general.

As for sparsity(see Figure 4.35), DICE and Prototypes are the most efficient methods, modifying only two features in most rounds. Occasionally, LORE modifies just one feature but typically alters two to three

features. GrowingSpheres consistently changes three features, which is an inherent characteristic of the method.

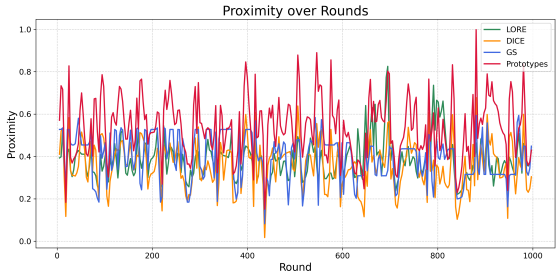


Figure 4.33: Evolution of proximity – *HTTP*, User A

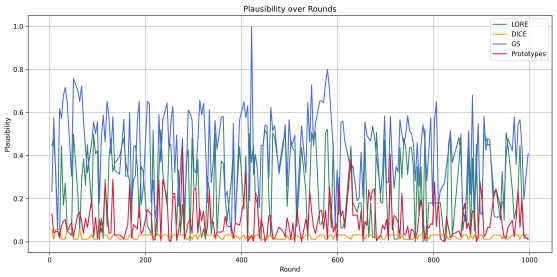


Figure 4.34: Evolution of plausibility – *HTTP*, User A

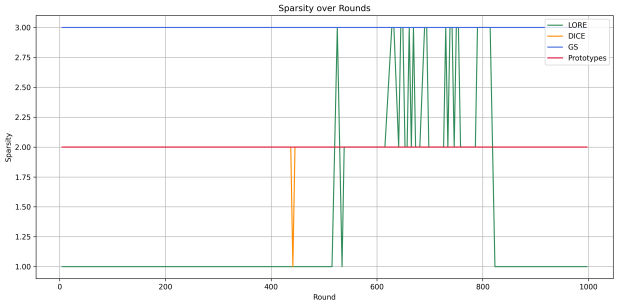


Figure 4.35: Evolution of sparsity – *HTTP*, User A

Execution times remain unchanged from previous experiments(see Table 4.13): GrowingSpheres and DICE continue to be the fastest methods, followed by LORE, while Prototypes remains the most time-consuming. In terms of skepticism, the user exhibits very high levels of skepticism throughout the interaction. Starting at nearly 99%, skepticism fluctuates but consistently remains around 90%, with some minor ups and downs. This behavior can be attributed to both the imbalanced nature of the dataset, which tends to increase skepticism, and the user profile—despite having high expertise, the user also shows a high level of trust in the model, which moderates their skepticism somewhat.

Performance metrics reflect the high quality of the initial model: both accuracy and F1-score reach 99%, consistent with the initial results obtained without user interaction.

### 4.2.11 User B HTTP

In this scenario, regarding proximity(see Figure 4.37), the best-performing methods are DICE and Growing Spheres, while LORE and Prototypes show the worst results. Notably, for the first time, LORE exhibits

| | GrowingSphere | DICE | LORE | Prototypes |
|---|---|---|---|---|
| **0** | 0.0809535980224609 | 0.1248668848103224 | 15.989806175231934 | 164.23192739486694 |
| **1** | 0.0841460227966308 | 0.1131610816162593 | 16.10595464706421 | 170.4133658409119 |
| **2** | 0.0852622985839843 | 0.1276500917483868 | 16.188448429107666 | 172.52615451812744 |
| **3** | 0.0902771949768066 | 0.1437869750272541 | 16.34280776977539 | 159.30217909812927 |
| **4** | 0.0855767726898193 | 0.1113933702615367 | 16.350345849990845 | 141.1730842590332 |
| **...** | ... | ... | ... | ... |
| **209** | 0.1486017704010009 | 0.1867349343662719 | 16.807871341705322 | 201.5148613452912 |
| **210** | 0.151064395904541 | 0.1262343230894607 | 16.739994287490845 | 194.69123816490173 |
| **211** | 0.1500897407531738 | 0.1366440872173992 | 16.609339475631714 | 191.59702324867249 |
| **212** | 0.1502633094787597 | 0.1332969641872322 | 16.578392505645752 | 198.75979351997373 |
| **Sum** | 24.64622545242309 | 24.65807537267623 | 3442.4707334041595 | 37920.12711167336 |

Table 4.13: Execution times on *HTTP* (User A)

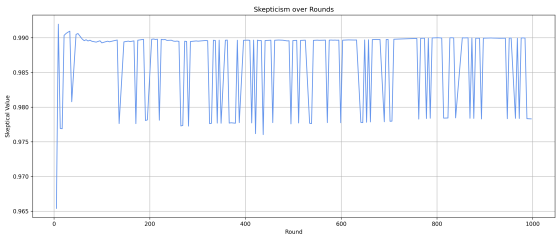


Figure 4.36: Evolution of skepticism– *HTTP*, User A

pronounced peaks in proximity values, surpassing all other methods.

As for plausibility(see Figure 4.38), DICE remains the most effective method, clearly outperforming the others. Prototypes ranks second, although it occasionally presents higher peaks, while LORE and Growing Spheres yield lower plausibility scores overall.

The observations related to sparsity are consistent with previous scenarios(see Figure 4.39): Prototypes and DICE consistently modify only two features; LORE alternates between modifying one, two, or three features; and Growing Spheres, by design, modifies all three features

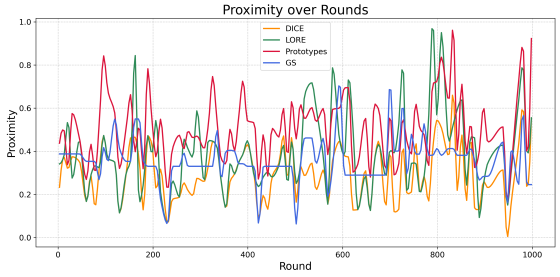


Figure 4.37: Evolution of proximity – *HTTP*, User B

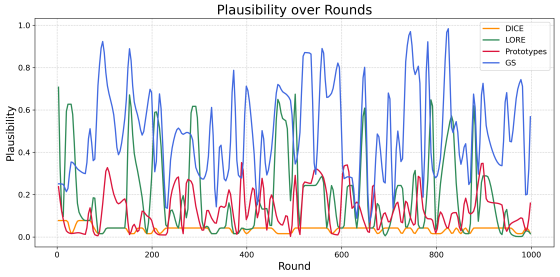


Figure 4.38: Evolution of plausibility – *HTTP*, User B

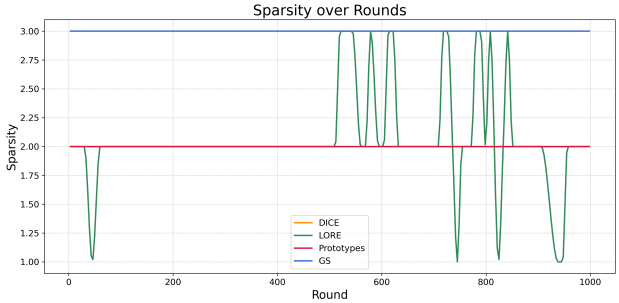


Figure 4.39: Evolution of sparsity – *HTTP*, User B

Similarly, execution time remains in line with previous observations across the tested methods(see Table 4.14).

Regarding skepticism(see Figure 4.40), the user in this scenario demonstrates high expertise but a fluctuating level of trust in the model. Initially, skepticism increases from 66% to 75%, followed by a stable phase around 70%, with minor oscillations and a gradual increase exceeding 76% towards the later iterations.

| | GrowingSphere | DICE | LORE | Prototypes |
|---|---|---|---|---|
| **0** | 0.0809667110443115 | 0.1050942441372596 | 16.291840076446533 | 158.94805765151978 |
| **1** | 0.0857534408569336 | 0.0992010887204827 | 16.292980909347534 | 169.43226671218872 |
| **2** | 0.0889191627502441 | 0.106495418475419 | 16.900047063827515 | 116.83290696144104 |
| **3** | 0.0908093452453613 | 0.1146193514963488 | 17.073153018951416 | 111.4716911315918 |
| **4** | 0.09006929397583 | 0.0983111542518539 | 17.15290355682373 | 111.5436041355133 |
| **...** | ... | ... | ... | ... |
| **93** | 0.1173610687255859 | 0.0974433110646429 | 16.851349353790283 | 189.44957828521729 |
| **94** | 0.1161146163940429 | 0.096845195758531 | 16.93014430999756 | 182.91435623168945 |
| **95** | 0.1178176403045654 | 0.0869016036319992 | 15.990328073501589 | 185.98973631858823 |
| **96** | 0.1178181171417236 | 0.1032325575259723 | 16.703119039535522 | 188.19994616508484 |
| **Sum** | 9.746977329254149 | 9.650305573708112 | 1573.0212910175323 | 15241.771156072617 |

Table 4.14: Execution times on *HTTP* (User B)

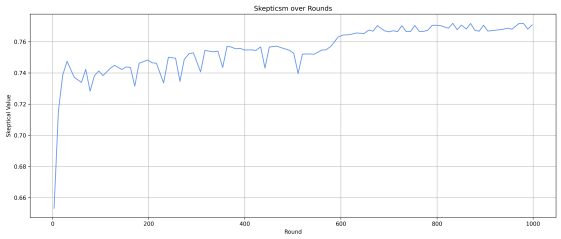


Figure 4.40: Evolution of skepticism– *HTTP*, User B

In this case, the accuracy begins at 99% and subsequently decreases slightly to 98%. More notably, the F1 score shows a significant drop, starting at 99% and declining to 49%, indicating a considerable deterioration in model performance. This decline occurs despite the user's high level of expertise, suggesting that other factors—such as user believability or the complexity of the dataset—may have had a stronger influence on the interaction outcomes.

### 4.2.12 User C HTTP

In this scenario with a new user, the following observations can be made:

Regarding proximity(see Figure 4.41), LORE exhibits both the lowest and the highest values among all methods. Growing Spheres and DICE follow similar trends, emerging as the most effective approaches (particularly DICE). Prototypes, while showing higher proximity values compared to Growing Spheres and DICE, maintains stable behavior without significant peaks.

In terms of plausibility(see Figure 4.42), DICE consistently yields the lowest values, outperforming the other methods. Prototypes ranks second, followed by LORE, which demonstrates greater variability. Growing Spheres performs the worst in this regard.

As for sparsity(see Figure 4.43), DICE modifies very few features (even generating counterfactuals with only one feature changed in certain cases). Prototypes, similar to DICE, most often modifies two features, though occasionally only one. LORE shows more variability, modifying between one and three features.

Growing Spheres, by design, modifies all features.

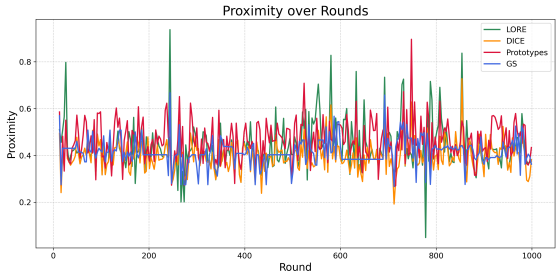


Figure 4.41: Evolution of proximity – *HTTP*, User C

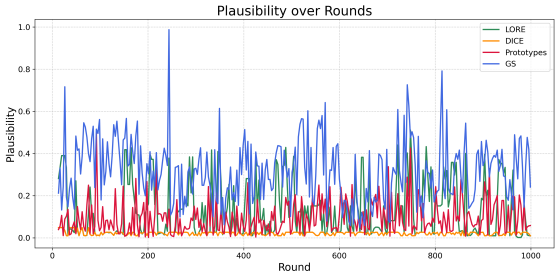


Figure 4.42: Evolution of plausibility – *HTTP*, User C

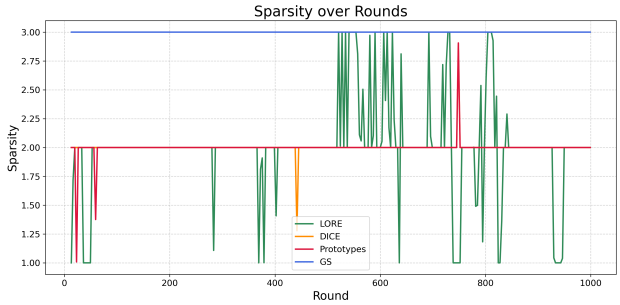


Figure 4.43: Evolution of sparsity – *HTTP*, User C

Observations regarding execution time remain consistent with previous scenarios(see Table 4.15). Concerning skepticism, the initial value is low (around 60%) but progressively increases to approximately 90%, stabilizing at this level(see Figure 4.44). This behavior is attributed both to the highly imbalanced nature of the dataset and to the user's characteristics: a medium level of expertise (50%) combined with a high degree of trust in the model. Given the strong dominance of the majority class and the user's moderate uncertainty, skepticism remains high throughout the iterations.

| | GrowingSphere | DICE | LORE | Prototypes |
|---|---|---|---|---|
| **0** | 0.1202614307403564 | 0.1864669188280457 | 17.00933265686035 | 181.89398097991943 |
| **1** | 0.1150591373443603 | 0.166854397174497 | 16.488343477249146 | 63.25091743469238 |
| **2** | 0.1096467971801757 | 0.19113005090315 | 17.210270166397095 | 179.4527530670166 |
| **3** | 0.1157839298248291 | 0.2181666384384752 | 17.275328397750854 | 190.79694056510925 |
| **4** | 0.1155400276184082 | 0.1638926800973377 | 16.25243330001831 | 185.29030108451843 |
| **...** | ... | ... | ... | ... |
| **425** | 0.2279064655303955 | 0.2133977522502513 | 18.116999864578247 | 244.1453154087067 |
| **426** | 0.2264924049377441 | 0.1413214510450302 | 16.69684910774231 | 247.2700712680817 |
| **427** | 0.2203512191772461 | 0.1923149629760828 | 17.966904640197754 | 246.52920484542847 |
| **428** | 0.2265334129333496 | 0.2038164147248969 | 16.488656044006348 | 243.53734493255607 |
| **Sum** | 73.41260528564453 | 73.89554205025982 | 7371.424669027328 | 73865.6376106739 |

Table 4.15: Execution times on *HTTP* (User C)

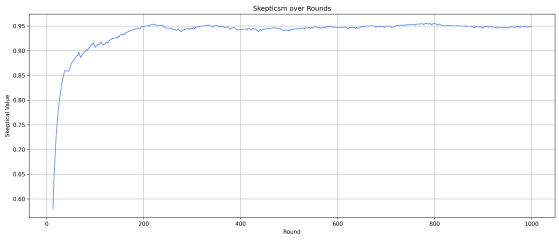


Figure 4.44: Evolution of skepticism– *HTTP*, User C

In this scenario, the accuracy remains stable at 99%, while the F1 score also holds steady at 99%, consistent with the initial results obtained without user involvement. This outcome demonstrates that even with a user of limited expertise, strong trust in the model can contribute to maintaining good overall performance.

### 4.2.13 User D HTTP

In this scenario with a new user, the following trends are observed:

Regarding proximity( see Figure 4.45), LORE once again exhibits both the highest and lowest values. However, Growing Spheres and DICE maintain a relatively consistent trend, consistently achieving lower proximity values compared to the average. Prototypes displays several high peaks, though less pronounced than in previous cases, and generally remains closer to the overall average.

As for plausibility(see Figure 4.46), previous observations still hold: DICE and Prototypes continue to perform best, while LORE shows slightly lower performance, and Growing Spheres maintains relatively high plausibility values.

In terms of sparsity(seeF igure 4.47 ), DICE typically modifies only two features, as does Prototypes (which, on one occasion, modifies just a single feature). LORE exhibits greater variability in the number of features modified, while Growing Spheres consistently alters the same number of features across iterations.

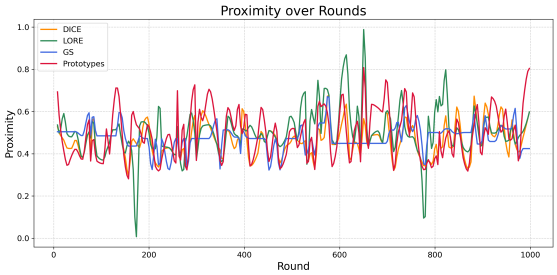


Figure 4.45: Evolution of proximity – *HTTP*, User D

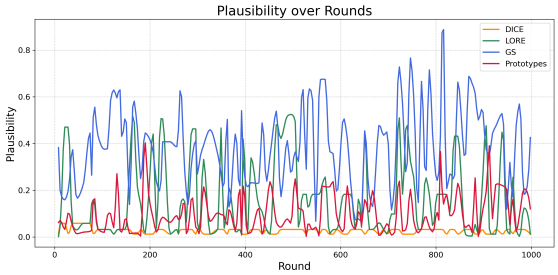


Figure 4.46: Evolution of plausibility – *HTTP*, User D

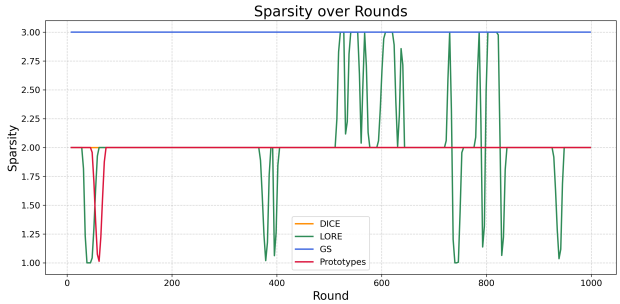


Figure 4.47: Evolution of sparsity – *HTTP*, User D

Observations regarding execution time remain unchanged from earlier scenarios (see Table 4.16).

With respect to skepticism( see Figure 4.48, the initial value (the highest) starts at 62%, then drops to 55%, subsequently rising again to 62%, before fluctuating around 60% and stabilizing at that level. These medium-to-low skepticism levels reflect the nature of the user under analysis, who is characterized by moderate expertise and moderate trust in the model (both at 50%). In this case, the accuracy reaches 98%,

| | GrowingSphere | DICE | LORE | Prototypes |
|---|---|---|---|---|
| **0** | 0.2190287113189697 | 0.2397219750534036 | 17.966047763824463 | 248.6939401626587 |
| **1** | 0.2221713066101074 | 0.2343995739931857 | 16.71584153175354 | 248.37247967720032 |
| **2** | 0.2271850109100341 | 0.2409874451208888 | 16.593268871307373 | 87.99209260940552 |
| **3** | 0.2224416732788086 | 0.2483245721209067 | 18.0058913230896 | 83.30957126617432 |
| **4** | 0.2217993736267089 | 0.2335958300182906 | 16.757105112075806 | 84.26908588409424 |
| **...** | ... | ... | ... | ... |
| **120** | 0.250324010848999 | 0.2421889522062156 | 18.100392818450928 | 271.56309938430786 |
| **121** | 0.23887300491333 | 0.2279360055628378 | 18.132932901382446 | 93.82595181465148 |
| **122** | 0.2507040500640869 | 0.2473167555714151 | 16.728947401046753 | 256.74941325187683 |
| **123** | 0.2518365383148193 | 0.2238081665597023 | 16.61987853050232 | 252.17348504066467 |
| **Sum** | 29.20925498008728 | 29.128639251623078 | 2111.7349860668182 | 21325.22906279564 |

Table 4.16: Execution times on *HTTP* (User D)

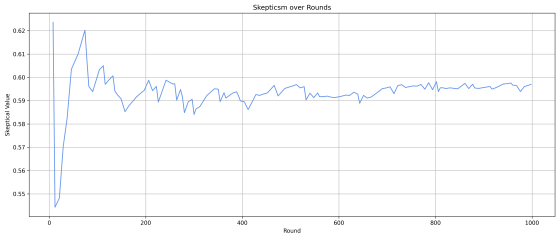


Figure 4.48: Evolution of skepticism– *HTTP*, User D

while the F1 score, starting from 99%, gradually decreases and stabilizes at 53%. This behaviour, despite involving a user with both low expertise and low believability, results in a stronger performance impact compared to users with higher expertise but low believability. This can be partially explained by the imbalanced nature of the dataset, which may have influenced shifts in the model's decision boundary, making it more sensitive to certain inputs during the interaction process.

### 4.2.14 User A CreditCard

In this scenario, clear differences emerge between the methods across the various evaluation dimensions. Regarding proximity(see Figure 4.49), Growing Spheres consistently outperforms the other approaches, maintaining significantly lower distances from the original instances. Prototypes ranks second, achieving

relatively low proximity values as well. DICE also performs well on this metric, although with slightly higher variability. In contrast, LORE, while sometimes capable of generating counterfactuals close to the original instances, frequently exhibits high peaks, resulting in much larger proximity values overall.

When considering plausibility(see Figure 4.50), DICE stands out as the best-performing method, consistently producing counterfactuals that remain close to the distribution of the training data. Growing Spheres also maintains good plausibility levels, outperforming the other methods except DICE. LORE shows highly variable results—occasionally matching the best methods, but often generating less plausible counterfactuals. In this particular scenario, Prototypes performs the worst, with plausibility scores considerably higher than those of the other approaches.As for sparsity(see Figure 4.51 ), all methods generally tend to modify most of the features in this dataset, given the complexity of the data (involving approximately 30 features). However, LORE occasionally manages to generate counterfactuals with significantly fewer changes—sometimes as few as a single feature—demonstrating greater flexibility in certain rounds.

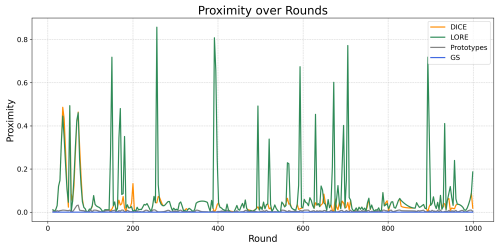


Figure 4.49: Evolution of proximity – *CreditCard*, User A

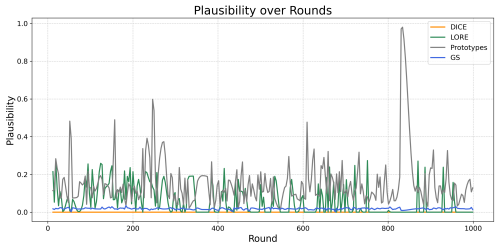


Figure 4.50: Evolution of plausibility – *CreditCard*, User A

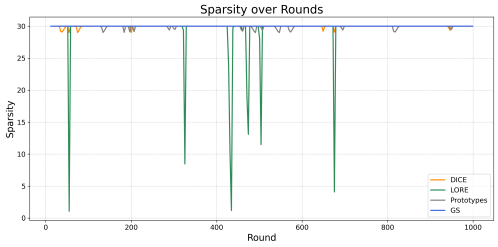


Figure 4.51: Evolution of sparsity – *CreditCard*, User A

A notable difference from previous experiments is observed in execution time(see Table 4.17). While Growing Spheres remains the fastest method, DICE shows a substantial increase in computation time due to the high number of features being modified—making it, in this case, the most computationally expensive approach. Similarly, LORE also experiences a marked increase in execution time, becoming comparable to

Prototypes. Regarding user skepticism(see Figure 4.52), the observed trend is characterized by considerable fluctuations. Starting from an initially high level (above 0.8), skepticism remains relatively elevated throughout the interaction rounds—generally oscillating between 0.7 and 0.9. This behavior can be explained by the nature of the user, who exhibits high expertise. As a result, the user remains critical of the model's decisions, although their trust in the system tempers extreme levels of skepticism.

| | GrowingSphere | DICE | LORE | Prototypes |
|---|---|---|---|---|
| **0** | 0.1055400371551513 | 18.24267530441284 | 9.385316133499146 | 2.163888692855835 |
| **1** | 0.0919568538665771 | 11.080273389816284 | 9.555785179138184 | 0.603428840637207 |
| **2** | 0.1104209423065185 | 18.20548391342163 | 9.13257360458374 | 2.3973186016082764 |
| **3** | 0.113433837890625 | 52.3952202796936 | 9.191731691360474 | 2.3182735443115234 |
| **4** | 0.1008884906768798 | 18.394290447235107 | 9.189640760421751 | 0.6930904388427734 |
| **...** | ... | ... | ... | ... |
| **255** | 0.1648359298706054 | 53.72067880630493 | 9.411983966827393 | 29.7269606590271 |
| **256** | 0.1732611656188964 | 19.63370537757873 | 10.264654636383057 | 30.096396684646606 |
| **257** | 0.1888253688812255 | 38.09709906578064 | 10.005592107772827 | 30.529496669769287 |
| **258** | 0.1547977924346923 | 36.72624611854553 | 10.38351058959961 | 31.13139533996582 |
| **Sum** | 35.83648419380187 | 7554.774345159531 | 2495.751020908356 | 2993.3571321964264 |

Table 4.17: Execution times on *CreditCard* (User A)

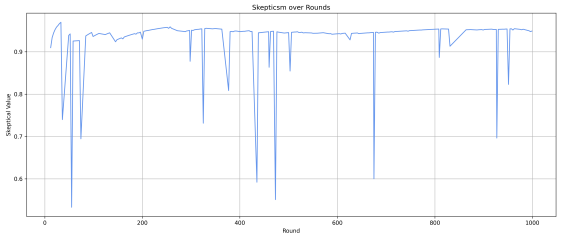


Figure 4.52: Evolution of skepticism– *CreditCard*, User A

With respect to performance metrics, the accuracy remains unchanged, consistently confirming a value of 99%. Similarly, the F1 score remains stable at 73%, thereby confirming the initial results obtained without the involvement of any user.

### 4.2.15 User B CreditCard

In this scenario, the behavior of the different methods is largely consistent with the previous case. Regarding proximity(see Figure 4.53 ), Growing Spheres and Prototypes remain the best-performing methods, achieving the lowest distances. DICE also performs well on this dimension, whereas LORE again proves to be the worst performer, exhibiting the highest proximity values.

When evaluating plausibility (see Figure 4.54 ), DICE once again achieves the best results, closely followed by Growing Spheres, with both methods outperforming the others. Although LORE and Prototypes occasionally manage to emulate the top-performing methods, they still exhibit frequent and pronounced peaks—particularly LORE, which stands out as the least stable in this respect.

Regarding sparsity (see Figure 4.55 ), the situation is similar to the previous scenario: nearly all methods tend to modify most of the available features. Only LORE demonstrates some capacity to generate counterfactuals involving fewer feature changes, though such cases are quite rare.

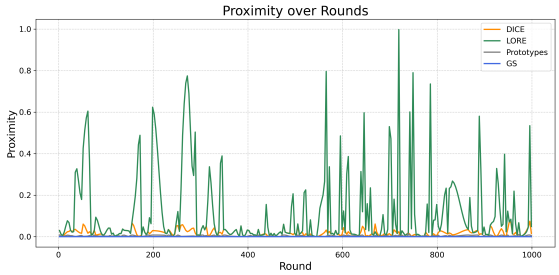


Figure 4.53: Evolution of proximity – *CreditCard*, User B

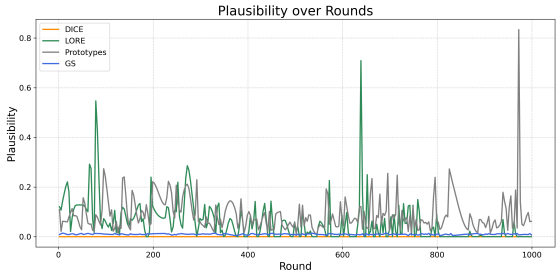


Figure 4.54: Evolution of plausibility – *CreditCard*, User B

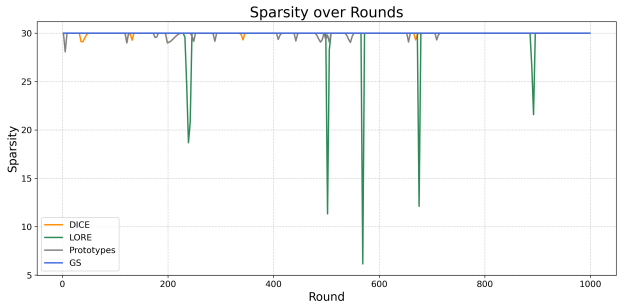


Figure 4.55: Evolution of sparsity – *CreditCard*, User B

In terms of execution time(see Table 4.18), the trend remains largely unchanged: Growing Spheres remains the fastest method, followed by LORE. However, in this instance, DICE is faster than Prototypes, although both methods still exhibit relatively high computational costs.

As for user skepticism(see Figure 4.56), the dynamics observed are as follows: the process starts from a very high level of skepticism, then temporarily drops to around 75%, before stabilizing at approximately 90%. This pattern is consistent with the characteristics of the user in this scenario—an expert user with medium levels of trust in the model (experience = 8.0, believability = 0.5). Given the imbalanced nature of the problem, this profile naturally leads to persistently high skepticism.

As previously introduced in earlier sections, an additional analysis was performed to evaluate the temporal validity of the counterfactuals generated during different skepticism phases. In previous cases, counterfactuals retained their validity over time. However, in this scenario, this property no longer holds: in fact, counterfactuals generated by LORE in three separate instances lost their validity over time.

With regard to accuracy and F1 score, it can be observed that accuracy remains constant at 99%, while the F1 score increases from an initial value of 70% to 77%, demonstrating an improvement in performance

| | GrowingSphere | DICE | LORE | Prototypes |
|---|---|---|---|---|
| **0** | 0.1673479080200195 | 19.11127877235413 | 10.208683252334597 | 31.080141305923465 |
| **1** | 0.1787199974060058 | 88.97361969947815 | 10.499401330947876 | 10.943153619766235 |
| **2** | 0.1982336044311523 | 22.762858867645264 | 10.169520139694214 | 30.441282987594604 |
| **3** | 0.1636736392974853 | 53.24799466133118 | 9.580557584762571 | 31.06731986999512 |
| **4** | 0.1692333221435547 | 18.788711309432983 | 9.59542989730835 | 10.139233827590942 |
| **...** | ... | ... | ... | ... |
| **236** | 0.2259695529937744 | 54.72469615936279 | 11.158422708511353 | 60.72156286239624 |
| **237** | 0.2246615886688232 | 20.063913583755493 | 11.263635873794556 | 60.93554782867432 |
| **238** | 0.2365400791168213 | 37.97861003875733 | 10.597909450531006 | 61.403197288513184 |
| **239** | 0.2132890224456787 | 37.24977922439575 | 9.799585342407228 | 61.89899301528931 |
| **Sum** | 47.46377635002136 | 6998.948773860931 | 2424.1539158821106 | 8103.903839826584 |

Table 4.18: Execution times on *CreditCard* (User B)

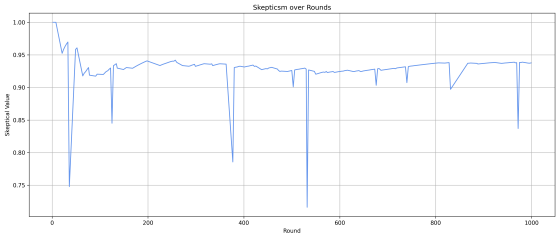


Figure 4.56: Evolution of skepticism– *CreditCard*, User B

compared to the initial situation.

### 4.2.16 User C CreditCard

In this scenario, concerning proximity (see Figure 4.57 ), Growing Spheres and Prototypes emerge as the best-performing methods, consistently achieving the lowest distance values. DICE also manages to maintain relatively low proximity values, while LORE proves to be the least effective, exhibiting the highest distances overall.

With respect to plausibility(see Figure 4.58 ), DICE and Growing Spheres again demonstrate superior performance, achieving the lowest distances from the data distribution. Both LORE and Prototypes follow a similar trend for the majority of iterations, although LORE stands out as the worst-performing method due to the presence of a pronounced peak.

Regarding sparsity(see Figure 4.59 ), as in previous scenarios, all methods tend to modify the majority of available features. However, it is noteworthy that in this case LORE manages, on several occasions, to generate counterfactuals by modifying only a few features.

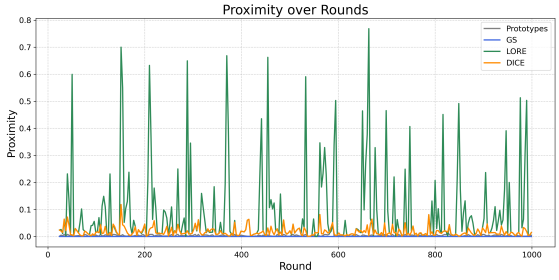


Figure 4.57: Evolution of proximity – *CreditCard*, User C

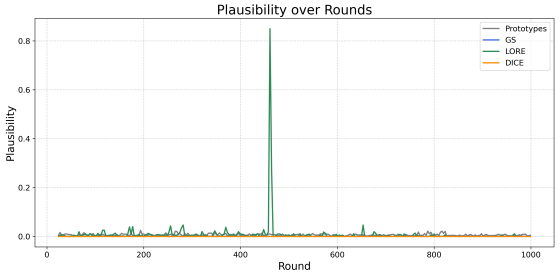


Figure 4.58: Evolution of plausibility – *CreditCard*, User C

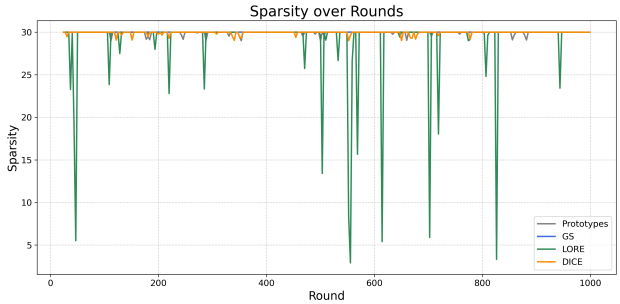


Figure 4.59: Evolution of sparsity – *CreditCard*, User C

In terms of execution time(see Table 4.19), Growing Spheres remains the fastest method, followed by LORE, then DICE, and finally Prototypes, which continues to be the most computationally expensive. As for skepticism(see Figure 4.60), the trend observed begins with relatively low levels of skepticism, around 60%, gradually increasing towards 80%, with a noticeable dip around rounds 400 to 600, where skepticism temporarily drops to approximately 50%. This behavior can be explained by the nature of the user: moderately experienced (expertise = 0.5) but generally inclined to trust the model (believability = 0.8). The relatively lower levels of skepticism, compared to previous cases, are driven by the user's limited experience, while the occasional increases are influenced by the imbalanced nature of the dataset.

| | GrowingSphere | DICE | LORE | Prototypes |
|---|---|---|---|---|
| **0** | 0.223102331161499 | 19.13886547088623 | 10.851053476333618 | 21.016706705093384 |
| **1** | 0.233492374420166 | 36.43486404418945 | 10.804463148117064 | 61.6051504611969 |
| **2** | 0.2445886135101318 | 26.23741841316223 | 9.38146448135376 | 62.21474719047546 |
| **3** | 0.1994407176971435 | 53.73727464675903 | 9.429591178894045 | 61.97120523452759 |
| **4** | 0.2387146949768066 | 35.67949843406677 | 10.641354084014893 | 61.561898946762085 |
| **...** | ... | ... | ... | ... |
| **622** | 0.3814165592193603 | 20.557482481002808 | 9.803272485733032 | 51.69369697570801 |
| **623** | 0.3576412200927734 | 20.322673082351685 | 10.20525288581848 | 54.276127576828 |
| **624** | 0.3982315063476562 | 13.54385471343994 | 10.200433731079102 | 155.2636640071869 |
| **625** | 0.3600747585296631 | 37.698708057403564 | 9.97295880317688 | 157.4044325351715 |
| **Sum** | 183.64973163604736 | 18185.760690927505 | 6444.790090799332 | 49785.825026750565 |

Table 4.19: Execution times on *CreditCard* (User C)

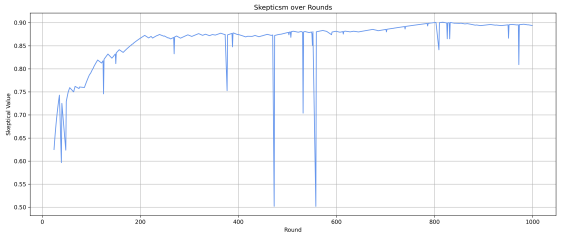


Figure 4.60: Evolution of skepticism– *CreditCard*, User C

With regard to accuracy and F1 score, it can be observed that accuracy remains constant at 99%, while the F1 score, as previous user, increases from an initial value of 70% to 77%.

### 4.2.17 User D CreditCard

In this final scenario, the analysis of the proximity shows that Growing Spheres once again proves to be the best method, consistently maintaining the lowest values across iterations, followed by the Prototypes method. Dice also performs well, achieving relatively low distances, albeit slightly higher than the afore-

mentioned two methods. LORE, on the other hand, results to be the worst method in terms of proximity, often exhibiting the highest peaks(see Figure 4.61).

Regarding plausibility(see Figure 4.62 ), all methods generally demonstrate very good performance; however, LORE exhibits two prominent peaks, which negatively affect its overall results. The behaviour of sparsity is consistent with previous scenarios(see Figure 4.63): LORE tends to modify fewer features, whereas the other methods usually involve a higher number of feature changes.

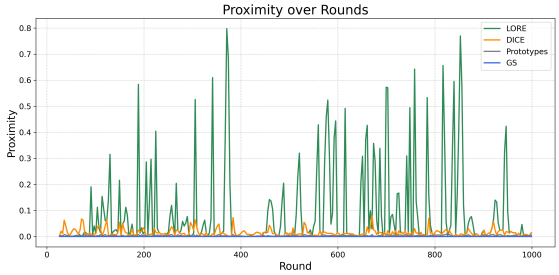


Figure 4.61: Evolution of proximity – *CreditCard*, User D

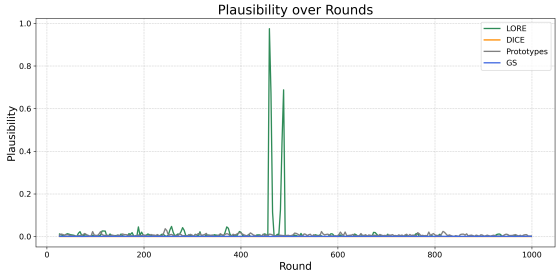


Figure 4.62: Evolution of plausibility – *CreditCard*, User D

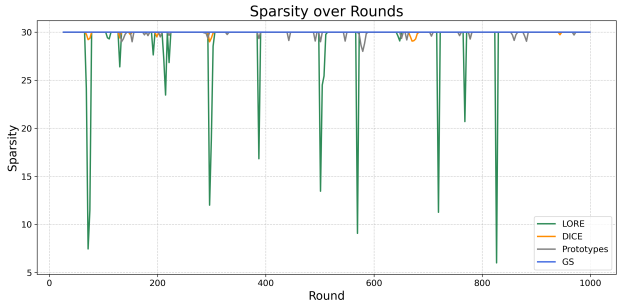


Figure 4.63: Evolution of sparsity – *CreditCard*, User D

Execution time analysis reveals that Growing Spheres remains the fastest method, followed by Dice. LORE and Prototypes exhibit very similar execution times, both significantly higher than Growing Spheres(see Table 4.20).

The scepticism graph shows generally low levels of scepticism(see Figure 4.64, oscillating around 60% for most iterations (with only one instance around 50%). This trend reflects the characteristics of the user, who demonstrates both low expertise and low believability towards the model, resulting in limited engagement and critical assessment.

| | GrowingSphere | DICE | LORE | Prototypes |
|---|---|---|---|---|
| **0** | 0.3879940509796142 | 26.855701684951782 | 9.805030345916748 | 156.72383379936218 |
| **1** | 0.3803634643554687 | 36.037841796875 | 11.837139129638672 | 158.07559394836426 |
| **2** | 0.3617341518402099 | 36.20536375045776 | 9.798059225082396 | 155.81741905212402 |
| **3** | 0.3579800128936767 | 6.922845840454102 | 12.155484676361084 | 158.12950587272644 |
| **4** | 0.3882672786712646 | 36.77466464042664 | 10.255151271820068 | 156.56972193717957 |
| **...** | ... | ... | ... | ... |
| **313** | 0.4500916004180908 | 55.65294075012207 | 10.631399869918823 | 188.0265381336212 |
| **314** | 0.4524340629577636 | 38.20973992347717 | 10.1258282661438 | 189.97475004196167 |
| **315** | 0.470304012298584 | 13.961116313934326 | 13.327064514160156 | 189.253671169281 |
| **316** | 0.4332056045532226 | 38.22078061103821 | 10.305019855499268 | 190.14132952690125 |
| **Sum** | 128.3826961517334 | 9211.569787502289 | 3393.8944928646088 | 38492.064413785934 |

Table 4.20: Execution times on *CreditCard* (User D)

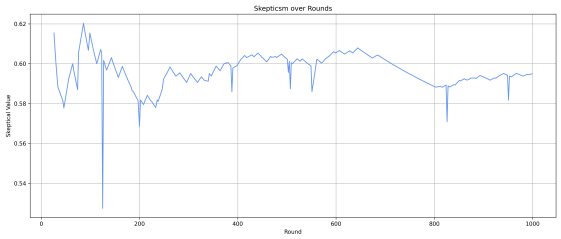


Figure 4.64: Evolution of skepticism– *CreditCard*, User D

Finally, with regard to performance metrics, accuracy remains stable at 99%, while the F1 score reaches 70%, highlighting a performance decrease .

### 4.2.18 LORE Experiment Aimed at Model Correction

After analyzing the characteristics of the counterfactuals generated by different methods and user profiles across the datasets, this section presents the results of a dedicated experiment leveraging LORE to reduce model-user disagreement during skeptical actions. The motivation and methodology for this correction strategy were introduced in Chapter 3; the procedure is briefly recalled here, along with concrete examples of its implementation and effects.

The core idea involves exploiting counterfactual explanations as a bridge between model predictions and human feedback. When disagreement arises and the user explicitly rejects the model's decision, the model is updated by generating additional training instances that reflect the user's perspective, thereby realigning the decision boundary accordingly.

Consider the following situation occurring during a skeptical event:

$$f(x) = 0 \quad \text{(the model predicts class 0 for instance } x\text{)},$$

$$h(x) = 1 \quad \text{(the user believes the correct label for } x \text{ is 1)}.$$

A counterfactual $x'$ is generated such that $f(x') = 1$. However, the user rejects this counterfactual, maintaining the original belief that $h(x) = 1$.

Given the persistent disagreement, a correction mechanism is triggered. As human feedback prevails in this context, the final decision is updated to $f(x) = 1$. Consequently, the system reverses the counterfactual decision, enforcing $f(x') = 0$.

To reinforce this correction, a neighborhood around $x'$—denoted by $X'_g$—is sampled via LORE, and label 0 is assigned to all its members, i.e.,

$$\forall x'_g \in X'_g, \quad y'_g = 0,$$

thereby injecting counterexamples that support the user's perspective. The model is then retrained using these newly added instances.

This strategy enables the model to adapt locally to user-driven corrections, thereby aiming to reduce future disagreement in similar regions of the feature space. To evaluate the effectiveness of the correction mechanism, we selected a user profile characterized by a high level of expertise (0.8) but low trust in the model. This combination was deliberately chosen to generate more meaningful moments of skepticism: the user's strong domain knowledge increases the likelihood of disagreement with the model, making their feedback especially valuable during critical decision points.

Since different counterfactual generation methods were used throughout the previous experiments, it was necessary to choose a specific approach to base the synthetic instance generation on. In this context:

- **LORE** was selected for balanced datasets, as it consistently showed strong performance in terms of *proximity* and *plausibility*.
- **DICE** was adopted for imbalanced datasets, where it demonstrated more reliable behavior across key counterfactual quality metrics.

Another important aspect was the number of synthetic instances generated for each skeptical event. To

ensure the integrity of the learned data distribution and avoid introducing significant bias, we adjusted this number based on the dataset type:

- For **balanced datasets**, *10 synthetic instances* were generated per skeptical interaction.
- For **imbalanced datasets**, this number was reduced to *5 synthetic instances* per event, primarily to avoid excessively unbalancing the model during fine-tuning.

The following section reports the evolution of skepticism levels and the impact of this correction strategy on the model's predictions and alignment with user expectations:

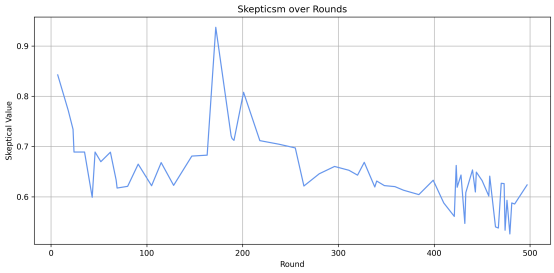


Figure 4.65: Skeptcism on Phishing dataset

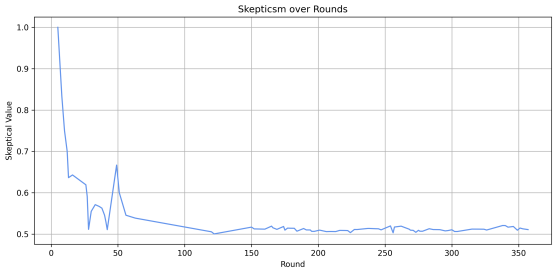


Figure 4.66: Skeptcism on Elec2 dataset

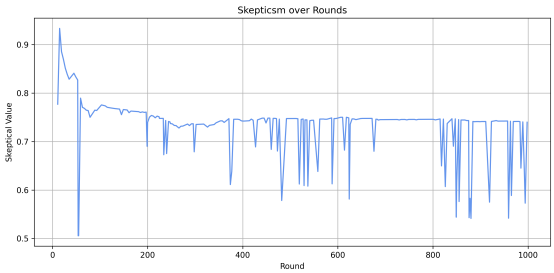


Figure 4.67: Skeptcism on CreditCard dataset

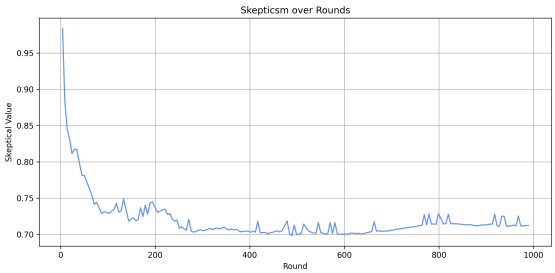


Figure 4.68: Skeptcism on HTTP dataset

The results illustrated in the figures above demonstrate how the levels of skepticism effectively decrease and tend to stabilize over the course of iterations, especially when compared to the baseline scenario where no corrective mechanism was employed. A particularly noteworthy case is the phishing dataset (see Figure 4.65), where the evolution of skepticism initially follows a decreasing trend, then stabilizes, reaches a sharp peak (over 90%), and finally settles again at a steady level.

Overall, the dynamics observed here are consistent with the patterns described in previous sections: namely, balanced datasets tend to exhibit lower average levels of skepticism compared to imbalanced ones.

Regarding the classification performance (accuracy and F1-score) after applying the correction mechanism, we observe the following outcomes:

- **Phishing dataset**: Accuracy initially starts at 86%, drops slightly to 85%, then increases to 88%, drops again to 85%, and finally stabilizes at 87%. The F1-score follows a similar pattern, starting at 86%, dropping to 84%, peaking at 88%, decreasing again to 84%, and finally settling at 86%. Although these results are marginally better than the baseline scenario without user involvement, they are slightly inferior to the version without any correction mechanism. This indicates that the example generation process introduced a mild class imbalance, which affected the model's performance.
- **Elec2 dataset**:Both accuracy and F1-score remain constant at 71% and 66%, respectively. These results show no difference compared to either the baseline model or the setup with user interaction without correction. This is likely due to the dataset's size and balance, which reduce the influence of incremental synthetic examples.
- **Credit Card dataset** :Accuracy remains constant at 99%, while the F1-score shows a notable decline. It begins at 70%, increases to 72%, then drops progressively to 66%, 52%, and finally 49%. This trend indicates a clear deterioration of performance, both compared to the initial baseline and the setting with user feedback only. The model was negatively impacted by the corrective mechanism, likely due to its sensitivity to imbalanced data and label noise.
- **HTTP dataset** : Accuracy remains stable at 98%, while the F1-score starts at 99% and eventually stabilizes at 49%. Although this drop may seem significant, the final performance remains comparable to the results obtained in the setting with user involvement but without the corrective mechanism. This suggests that the model maintains a similar decision behavior under both configurations, likely due to the strong class imbalance and the model's inherent confidence in its predictions.

## 4.3 Discussion of Experimental Results

From the experiments presented in the previous section, we can gather the following insights:

- In several scenarios, that methods producing counterfactuals by modifying only a few features tend to exhibit higher proximity or plausibility compared to methods that modify a larger number of features.

- GrowingSpheres consistently modifies all features(as observed in all sparsity plots). This behavior is consistent with the authors' stated goal: "Growing Spheres explicitly states that its goal is to provide information about the classifier, not about the reality it is approximating" [57]. In other words, the method focuses on exploring the classifier's decision boundary, without necessarily producing plausible or minimally invasive counterfactuals. As a result, it tends to generate broad modifications across the feature space.

- Prototype-based counterfactuals exhibit comparatively lower plausibility and higher proximity values than other methods(particularly in the balanced experiments). This behavior is expected: prototypes, being actual examples from the dataset , produce counterfactuals that align well with the data distribution but may be further from the original instance — resulting in higher proximity and lower plausibility when compared to more "aggressive" methods that seek to minimize distance from the starting point.

- LORE and Prototypes generally exhibit longer execution times than other methods, particularly on the balanced dataset. This can be explained as follows:

  - For LORE, it is necessary to first build a local surrogate model, generate a neighborhood, apply counterfactual rules, and subsequently filter valid instances — a complex and computationally intensive process.
  - For Prototypes, it is sometimes necessary to perform a higher number of iterations and conduct a broader exploration of hyperparameters to select the most representative prototypes suitable for generating effective counterfactuals, thereby increasing the total computation time.

After these general considerations, it is useful to summarize the most relevant results according to the different scenarios analyzed:

- **Balanced case**: In this scenario, regarding the search for counterfactuals, the best-performing method was LORE, as it was able to identify counterfactuals very close to the original instance (with particularly notable results in the phishing experiments), counterfactuals that were also well-aligned with the original data distribution, while modifying an acceptable number of features. Although its execution time was higher compared to methods such as DICE or GrowingSpheres, it

remained within acceptable limits. DICE and GrowingSpheres followed, offering good computational performance and achieving solid results overall. By contrast, Prototypes — despite achieving good plausibility — did not perform as well in terms of sparsity, proximity, or execution time.

A significant pattern was observed in relation to skepticism levels across both datasets: with expert users (regardless of their level of trust in the model), skepticism levels remained consistently high, between 80% and 90%(as seen in figures4.4, 4.20, 4.8, and 4.24 ). Conversely, with non-expert users, skepticism levels were lower (in phishing, in fact, some cases only reached 50% ,as seen in Figure 4.16).

Additionally, in the phishing dataset, expert users were able to improve model performance on the test set, while less experienced users tended to degrade performance. This behavior was not observed in Elec2, where — given the larger size of the training set — model performance remained stable regardless of the type of user.

- **Imbalance case** : As for the imbalanced case, the results, as expected, show different characteristics compared to the balanced scenario.

  With regard to counterfactual generation, DICE proved to be the most effective method, achieving medium-to-low levels of proximity, plausibility, sparsity, and execution time. Other methods displayed more varied behavior: LORE produced anomalous peaks across several metrics, Prototypes performed worse than in the balanced case, and GrowingSpheres, while maintaining good proximity, plausibility, and execution time in certain experiments (e.g., credit card), failed to replicate this performance on the HTTP dataset, where plausibility showed extremely high peaks.

  A notable case is the credit card dataset, where all methods — except LORE — were able to generate highly effective counterfactuals (as evidenced by the proximity and plausibility graphs). This is due to the fact that, as shown in the sparsity graphs, LORE does not modify all features, an action that proves crucial for this type of data. Conversely, DICE, in order to find even one valid counterfactual, needed to alter a large number of features, which led to increased execution times — in contrast with the balanced case, where DICE produced counterfactuals in much shorter time. Additionally, LORE was the only method that failed to respect validity in three separate instances, making it less reliable in dynamic or iterative skepticism settings. Validity, in this context, refers to the consistency of a counterfactual across multiple skepticism events — that is, whether a previously generated coun-

terfactual continues to lead to the desired prediction over time. This limitation further questions LORE's robustness, particularly in high-stakes or imbalanced scenarios where preserving the reliability of model corrections is essential. Regarding skepticism levels, in the imbalanced dataset they generally appear higher, as the model's predicted probabilities are more heavily skewed towards the majority class. This also affects test set performance: for both datasets involved, accuracy exceeds 90%, while F1-score is more sensitive to the user's perceived credibility of the model.

In particular, in the HTTP dataset, regardless of user experience, when users displayed high credibility toward the model, excellent performance was achieved (99% on both metrics). In the credit card dataset, with highly credible users, there was a substantial improvement compared to the initial performance (without user interaction), reaching 99% accuracy and 77% F1-score. These results suggest that in imbalanced settings, credibility plays a more decisive role, while in balanced settings, experience tends to be the more influential factor.

An additional and relevant contribution was provided by the model correction experiment based on LORE. The objective was to assess whether the introduction of synthetic examples aligned with the user's perspective could reduce skepticism levels and improve the alignment between model and user. The results confirmed a tangible reduction and stabilization of skepticism over time, supporting the effectiveness of the proposed mechanism from an interactive standpoint. However, a critical limitation also emerged: in some cases, the addition of synthetic data introduced a certain degree of imbalance in the model, leading to a degradation in classification metrics—particularly in terms of the F1-score. This suggests that, although promising, the correction strategy requires further refinement, such as example selection or weighting techniques, to avoid unintended side effects.

# Chapter 5

# CONCLUSIONS

In recent years, the need to develop machine learning systems capable of interacting directly and dynamically with humans has become increasingly central, especially in complex decision-making contexts. This thesis has focused on FRANK, a framework designed specifically for this interactive space, introducing the concept of skepticality to allow users to express dissent and intervene in the decision-making cycle.
While FRANK provided a mechanism for detecting misalignment between the user and the model, it lacked explanatory tools to justify such misalignments. This gap defines the main contribution of this work: the integration of counterfactual explanations and fact/counterfactual rules into the FRANK framework, with the aim of making the system not only reactive, but also interpretable. Specifically, the contributions of this thesis can be summarized as follows:

1. The extension of the FRANK framework with counterfactual explanation methods to support users in understanding and, if necessary, correcting model predictions. The methods adopted for generating counterfactuals include DICE, GrowingSpheres, LORE, and Prototypes;

2. An empirical analysis of the effectiveness of these explanations in dynamic scenarios, evaluating their quality, plausibility, and adaptability over time;

3. The introduction of a correction mechanism based on LORE, capable of generating synthetic examples to temporarily realign model decisions with user expectations.

The experiments conducted on counterfactual generation aimed primarily to leverage the proposed methods to generate counterfactuals in the shortest possible time, given the large volume of data and users to

analyze. Naturally, with a more in-depth exploration and greater computational resources, it would have been possible to explore a wider range of parameter combinations for the various methods, and thereby obtain counterfactuals with better proximity, sparsity, plausibility, and validity. The results obtained suggest that integrating counterfactual explanations can indeed strengthen human-computer interaction by increasing transparency and user trust. Despite the promising outcomes, this work presents some critical issues:

1. In the LORE-based correction experiment, although the skepticism threshold tends to decrease over time, the model's predictive performance (in terms of F1-score and accuracy) also shows a downward trend in some scenarios. This suggests that the local correction process may introduce unwanted noise or bias into the model's behavior.

2. Some counterfactual generation methods, particularly Prototypes in most cases, are computationally intensive, making their adoption in real-time settings or with large datasets challenging.

3. The experiments highlight that each method has strengths and weaknesses depending on the scenario. For instance, Prototypes generate counterfactuals that are highly consistent with the data distribution, but at the cost of high computational time and reduced proximity. Conversely, methods like DICE and GrowingSpheres are more efficient but often sacrifice properties such as sparsity or plausibility. This trade-off reflects a crucial challenge for interactive systems: balancing explanatory power with system responsiveness — a key factor for true usability in continuous, interactive environments.

4. In the presence of imbalanced data distributions, user skepticism tends to remain high over time. Moreover, in scenarios where users are knowledgeable but have low trust in the model, a degradation in predictive performance is observed, reducing the effectiveness of label correction on individual instances.

5. In certain tests, models such as EFDT and HAT, even when used as base learners in ensemble methods, did not correctly update their structure in the presence of imbalanced data. The number of splits, selected features, and splitting thresholds remained unchanged over time. This behavior reveals a limitation of the river library, especially when applied to imbalanced learning settings, where one would expect continuous structural adaptation of the model.

Finally, several promising directions for future work can be identified:

1. The integration of outlier detection algorithms could be employed to either reject anomalous instances — or alternatively, notify users of their presence — during skeptical interactions. This would help prevent the generation of unstable or implausible counterfactuals in cases where the original instance is not well supported by the data distribution.

2. Instead of applying a fixed skepticism threshold, a more dynamic approach could be explored by calibrating the skepticism threshold in real time, based on the specific user profile (e.g., expertise,credibility) and on the behavior of the model (e.g., uncertainty of predictions, local density of the data). Such an adaptive mechanism could enhance both user experience and the overall robustness of the system.

3. The adoption of prequential evaluation metrics (e.g., prequential accuracy and F1) to better capture the temporal evolution of the model. While this thesis used a static test set to maintain consistency with the original FRANK evaluation [43], future work could employ a more realistic setup where $\mathcal{D}_{\text{iter}}$ and $\mathcal{D}_{\text{test}}$ streams are merged. This would simulate real-time human interaction and enable online evaluation of model adaptation. Naturally, such an approach would increase computational demands, particularly in scenarios with many users or high-frequency interactions, and these challenges must be carefully considered in terms of scalability and performance.

# References


[1] Wolfram Schenck Alaa Tharwat. A survey on active learning: State-of-the-art, practical challenges and research directions. *Mathematics*, 11(4):820, 2023. doi: 10.3390/math11040820. URL https://www.mdpi.com/2227-7390/11/4/820.

[2] Maarten de Rijke Aleksandra Lucic, Hinda Haned. Why does my model fail? contrastive local explanations for retail forecasting. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT*)*, pages 90–98, Barcelona, Spain, 2020. ACM.

[3] Saleema Amershi, Maya Cakmak, W. Bradley Knox, and Todd Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120, 2014. doi: 10.1609/aimag.v35i4.2513.

[4] Jean Paul Barddal and Fabrício Enembreck. Learning regularized hoeffding trees from data streams. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19)*, page 8, Limassol, Cyprus, 2019. Association for Computing Machinery. ISBN 978-1-4503-5933-7. doi: 10.1145/3297280.3297334. URL https://doi.org/10.1145/3297280.3297334.

[5] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies,opportunities and challenges toward responsible ai. *Information Fusion*, 2019.

[6] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009. ISSN 1936-4954. doi: 10.1137/080716542. URL https://dx.doi.org/10.1137/080716542.

[7] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*, pages 41–48, New York, NY, USA, 2009. Association for Computing Machinery. doi: 10.1145/1553374.1553380.

[8] J. Bennett, R. Grout, P. Pebay, D. Roe, and D. Thompson. Numerically stable, single-pass, parallel statistics algorithms. In *2009 IEEE International Conference on Cluster Computing and Workshops*, pages 1–8. IEEE, August 2009.

[9] Albert Bifet and Ricard Gavaldà. Learning from time-changing data with adaptive windowing. In *SIAM International Conference on Data Mining (SDM)*, pages 443–448, 2007.

[10] Albert Bifet and Ricard Gavaldà. Adaptive learning from evolving data streams. In *International Symposium on Intelligent Data Analysis (IDA)*, pages 249–260, Berlin, Heidelberg, 2009. Springer.

[11] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavaldà. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2009.

[12] Francesco Bodria, Fosca Giannotti, Riccardo Guidotti, Fabio Naretto, Dino Pedreschi, and Salvatore Rinzivillo. Benchmarking and survey of explanation methods for black box models. *CoRR*, abs/2102.13076, 2021.

[13] Andrea Bontempelli, Stefano Teso, Fausto Giunchiglia, and Andrea Passerini. Learning in the wild with incremental skeptical gaussian processes. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, pages 2886–2892. IJCAI Organization, 2020. doi: 10.24963/ijcai.2020/399. URL https://arxiv.org/abs/2011.00928.

[14] Ruth M.J. Byrne. Counterfactuals in explainable artificial intelligence (xai): Evidence from human reasoning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.

[15] David Castelvecchi. Can we open the black box of ai? *Nature News*, 538(7623):20, 2016.

[16] S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10(1):57–78, 1993. ISSN 1573-0565. doi: 10.1023/A:1022664626993. URL https://doi.org/10.1023/A:1022664626993.

[17] Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, 31(6):1794–1813, 2002. doi: 10.1137/S0097539701398363.

[18] Andrew McCallum David Cohn, Rich Caruana. Semi-supervised clustering with user feedback. In *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, volume 4, pages 17–32. Chapman and Hall/CRC, 2003.

[19] Amit Dhurandhar, Tathagata Pedapati, Ankur Balakrishnan, Pin-Yu Chen, Karthikeyan Shanmugam, and Ruchir Puri. Model agnostic contrastive explanations for structured data. *arXiv preprint arXiv:1906.00117*, 2019. URL `https://arxiv.org/abs/1906.00117`.

[20] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 71–80. ACM, 2000.

[21] Eibe Frank and Ian H. Witten. Making better use of global discretization. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*, page 9, 1999.

[22] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[23] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):44, 2014. doi: 10.1145/2523813. URL `https://dl.acm.org/doi/10.1145/2523813`.

[24] Brent Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a right to explanation. *AI Magazine*, 38(3):50–57, 2017.

[25] Henry Gouk, Bernhard Pfahringer, and Eibe Frank. Stochastic gradient trees. In *Proceedings of the Asian Conference on Machine Learning (ACML)*, pages 1094–1109, October 2019.

[26] Riccardo Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, 2022.

[27] Riccardo Guidotti and Salvatore Ruggieri. On the stability of interpretable models. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, Budapest, Hungary, 2019. IEEE.

[28] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. Local rule-based explanations of black box decision systems, 2018. URL https://arxiv.org/abs/1805.10820.

[29] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys*, 51(5): 93:1–93:42, 2018.

[30] Nuwan Gunasekara, Bernhard Pfahringer, Heitor Murilo Gomes, and Albert Bifet. Survey on online streaming continual learning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)*, pages 6628–6636. International Joint Conferences on Artificial Intelligence, 2023. URL https://www.ijcai.org/proceedings/2023/0743.pdf.

[31] David Gunning. Explainable artificial intelligence (xai). Technical report, Defense Advanced Research Projects Agency (DARPA), 2017.

[32] Fabian Hinder, Valerie Vaquet, and Barbara Hammer. One or two things we know about concept drift—a survey on monitoring in evolving environments. part a: detecting concept drift. *Frontiers in Artificial Intelligence*, 7, 2024. doi: 10.3389/frai.2024.1330257. URL https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2024.1330257/full.

[33] Drew Jarrett et al. Online decision mediation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[34] Daniel Kahneman and Amos Tversky. The simulation heuristic. In Daniel Kahneman, Paul Slovic, and Amos Tversky, editors, *Judgment Under Uncertainty: Heuristics and Biases*, pages 201–208. Cambridge University Press, New York, 1982.

[35] Faisal Kamiran and Toon Calders. Classification with no discrimination by preferential sampling. In *Belgian Dutch Conference on Artificial Intelligence (BNAIC)*, volume 1. Citeseer, 2010.

[36] Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012. doi: 10.1561/2200000044.

[37] Heather C. Lench, Darren Domsky, Rachel Smallman, and Kathleen E. Darbor. Beliefs in moral luck: When and why blame hinges on luck. *British Journal of Psychology*, 106(2):272–287, 2015.

[38] Jennifer Lerman. Big data and its exclusions. *Stanford Law Review Online*, 66:55, 2013.

[39] Timothée Lesort, Massimo Caccia, and Irina Rish. Understanding continual learning settings with data distribution drift analysis, 2022. URL https://arxiv.org/abs/2104.01678.

[40] Zachary C. Lipton. The mythos of model interpretability. *Queue*, 16(3):30:31–30:57, 2018.

[41] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4765–4774, 2017.

[42] Chaitanya Manapragada, Geoffrey Webb, and Mahsa Salehi. Extremely fast decision tree. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*, pages 1953–1962, New York, NY, USA, 2018. ACM. doi: 10.1145/3219819.3220005.

[43] Federico Mazzoni, Riccardo Guidotti, and Alessio Malizia. *A Frank System for Co-Evolutionary Hybrid Decision-Making*, page 236–248. Springer Nature Switzerland, 2024. ISBN 9783031585531. doi: 10.1007/978-3-031-58553-1_19. URL http://dx.doi.org/10.1007/978-3-031-58553-1_19.

[44] Francesco Mazzoni et al. Genfair: A genetic fairness-enhancing data generation framework. In *Data Science*, volume 14276 of *Lecture Notes in Computer Science*, pages 356–371. Springer, 2023.

[45] Rachel McCloy and Ruth M. J. Byrne. Counterfactual thinking about controllable actions. *Memory & Cognition*, 28:1071–1078, 2000.

[46] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[47] Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, and Albert Bifet. River:

machine learning for streaming data in python, 2020. URL https://arxiv.org/abs/2012.04740.

[48] E. Mosqueira-Rey, E. Hernández-Pereira, D. Alonso-Ríos, et al. *Human-in-the-loop machine learning: a state of the art.* Springer, 2023. URL https://doi.org/10.1007/s10462-022-10246-w.

[49] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT* '20)*, pages 607–617, New York, NY, USA, 2020. Association for Computing Machinery. doi: 10.1145/3351095.3372850.

[50] N. C. Oza. Online bagging and boosting. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2340–2345. IEEE, October 2005.

[51] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. doi: 10.1016/j.neunet.2019.01.012. URL https://doi.org/10.1016/j.neunet.2019.01.012.

[52] Sriram Ravichandran, Nandan Sudarsanam, Balaraman Ravindran, and Konstantinos V. Katsikopoulos. Active learning with human heuristics: An algorithm robust to labeling bias. *Frontiers in Artificial Intelligence*, 7:1491932, 2024. doi: 10.3389/frai.2024.1491932. URL https://www.frontiersin.org/articles/10.3389/frai.2024.1491932/full.

[53] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education Limited, Malaysia, 3rd edition, 2016.

[54] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. URL https://minds.wisconsin.edu/bitstream/handle/1793/60660/TR1648.pdf.

[55] Stefano Teso and Antonio Vergari. Efficient and reliable probabilistic interactive learning with structured outputs. *arXiv preprint arXiv:2202.08566*, 2022. URL https://arxiv.org/abs/2202.08566. Accessed: 2025-05-25.

[56] Stefano Teso, Oznur Alkan, Wolfgang Stammer, and Elizabeth Daly. Leveraging explanations in interactive machine learning: An overview. *Frontiers in Artificial Intelligence*, 6:1066049, 2023. doi: 10.3389/frai.2023.1066049. URL https://www.frontiersin.org/articles/10.3389/frai.2023.1066049/full.

[57] Christophe Marsala Thibault Laugel, Marie-Jeanne Lesot and Marcin Detyniecki Xavier Renard. Comparison-based inverse classification for interpretability in machine learning. In *17th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2018)*, pages 100–111, Cádiz, Spain, 2018. Springer. doi: 10.1007/978-3-319-91473-2_9. URL https://hal.science/hal-01905982.

[58] Valerie Thompson and Ruth M. J. Byrne. Reasoning counterfactually: Making inferences about things that didn't happen. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 28:1154–1170, 2002.

[59] Arnaud Van Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584*, 2020. URL https://arxiv.org/abs/1907.02584.

[60] Sandra Wachter, Brent Mittelstadt, and Luciano Floridi. Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, 7(2):76–99, 2017.

[61] Sandra Wachter, Brent D. Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harvard Journal of Law & Technology*, 31:841, 2017.

[62] B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

[63] Yoshua Bengio Yann LeCun, Léon Bottou and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[64] Mattia Zeni, Wanyi Zhang, Andrea Passerini, and Fausto Giunchiglia. Fixing mislabeling by human annotators leveraging conflict resolution and prior knowledge. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)*, 3(1):32:1–32:23, 2019.

[65] Wanyi Zhang. Personal context recognition via skeptical learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*, 2019.

[66] Wanyi Zhang. Personal context recognition via skeptical learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI-19)*, pages 6482–6488. International Joint Conferences on Artificial Intelligence Organization, 2019. URL https://www.ijcai.org/proceedings/2019/0930.pdf.

[67] Wanyi Zhang, Mattia Zeni, Andrea Passerini, and Fausto Giunchiglia. Skeptical learning - an algorithm and a platform for dealing with mislabeling in personal context recognition. *OpenReview*, 2022. URL https://openreview.net/forum?id=HOOiwNICra. Published: 01 Jan 2022, Last Modified: 27 Jan 2025.

[68] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005. ISSN 13697412, 14679868. URL https://www.jstor.org/stable/3647580.

[69] Mustafa Çavuş and Jakub Kuzilek. An effect analysis of the balancing techniques on the counterfactual explanations of student success prediction models. *Journal of Measurement and Evaluation in Education and Psychology*, 15(Special Issue):302–317, 2024. doi: 10.21031/epod.1526704. URL https://doi.org/10.21031/epod.1526704.