Cognitive Computation
https://doi.org/10.1007/s12559-024-10272-6

An eXplainable Artificial Intelligence Methodology on Big Data Architecture

Valerio La Gatta[1] · **Vincenzo Moscato**[1] · **Marco Postiglione**[1] · **Giancarlo Sperlì**[1]

Received: 11 September 2023 / Accepted: 3 March 2024

Abstract
Although artificial intelligence has become part of everyone's real life, a trust crisis against such systems is occurring, thus increasing the need to explain black-box predictions, especially in the military, medical, and financial domains. Modern eXplainable Artificial Intelligence (XAI) techniques focus on benchmark datasets, but the cognitive applicability of such solutions under big data settings is still unclear due to memory or computation constraints. In this paper, we extend a model-agnostic XAI methodology, named *Cluster-Aided Space Transformation for Local Explanation* (CASTLE), to be able to deal with high-volume datasets. CASTLE aims to explain the black-box behavior of predictive models by combining both *local* (i.e., based on the input sample) and *global* (i.e., based on the whole scope for action of the model) information. In particular, the local explanation provides a rule-based explanation for the prediction of a target instance as well as the directions to update the likelihood of the predicted class. Our extension leverages modern big data technologies (e.g., Apache Spark) to handle the high volume, variety, and velocity of huge datasets. We have evaluated the framework on five datasets, in terms of temporal efficiency, explanation quality, and model significance. Our results indicate that the proposed approach retains the high-quality explanations associated with CASTLE while efficiently handling large datasets. Importantly, it exhibits a sub-linear, rather than exponential, dependence on dataset size, making it a scalable solution for massive datasets or in any big data scenario.

Keywords eXplainable Artificial Intelligence · Clustering · Artificial intelligence · Big data

Introduction

The Artificial Intelligence (AI) advent has radically changed our daily lives, strongly influencing the business logic of companies and offering a decisive competitive advantage to those who first explored this new road. International Data Corporation (IDC)[1] reports that 37.5 billion have been invested on AI in 2019, more than 44% with respect to 2018, and forecasts 97.9 billion spent in 2023. All these investments have led to what is called AI-driven development [1–3], whose purpose is to define methods and best practices for embedding AI into applications; not only does this enhance AI-powered solutions to be adopted in a wide range of business domains, but it also enables the democratization of AI [4, 5].

Consequently, AI is having a significant impact on society especially when it accesses to our private data and makes decisions for ourselves [6–10]. Indeed, AI has already become ubiquitous, and we got accustomed to it without even realizing: AI makes decisions for us in our daily lives, from product and movie recommendations on Amazon and Netflix, to friends suggestions on Facebook, or tailored adver-

✉ Giancarlo Sperlì
giancarlo.sperli@unina.it

Valerio La Gatta
valerio.lagatta@unina.it

Vincenzo Moscato
vincenzo.moscato@unina.it

Marco Postiglione
marco.postiglione@unina.it

[1] Department of Electrical Engineering and Information Technology (DIETI), University of Naples Federico II, Via Claudio 21, Naples 80125, Italy

[1] https://www.idc.com/getdoc.jsp?containerId=prUS45481219

Published online: 11 April 2024

tisements on Google search result pages. However, as long as we refer to movie recommendations or tailored advertisements, companies can rely on classic machine learning techniques, which learn from the data and provide opaque predictions. Nevertheless, in life-changing situations, such as disease diagnosis or military operation, it is important to know the reasons behind such a critical decision [11, 12].

Regrettably, a significant obstacle hindering the widespread adoption of AI-based systems pertains to their inherent opacity. The black-box characteristic prevalent in these systems endows them with formidable predictive capabilities but renders them inscrutable in terms of direct explication. Consequently, this predicament has catalyzed a burgeoning discourse surrounding eXplainable Artificial Intelligence (XAI), an emerging research domain replete with substantial potential to enhance the trustworthiness and transparency of AI-based systems [13–16]. XAI is unequivocally regarded as the indispensable prerequisite for AI to sustain its uninterrupted trajectory of progressive development [17]. The main goal is to find a transparent model, which explains the logic behind the prediction process, without having to forgo accuracy. Unfortunately, in many cases, the most accurate models are also the most complex ones (like complex neural networks, with many hidden layers), while the more intuitive and transparent models (like decision trees) do not always provide such high performance.

XAI is an emerging topic whose meaning is sometimes considered related to the following different concepts, as shown in [18]: (i) *Explainability*, concerning the ability of an AI system to explain its decisions in intelligible terms to humans (e.g., [17, 19]), and (ii) *Interpretability*, corresponding to the identification of features set that has contributed to make a decision [20]. According to the Defence Advanced Research Project Agency *DARPA*, the aims of XAI are essentially two [21]: to create models that are more transparent while sustaining a strong level of learning performance (e.g., predictive accuracy) and empowering human users to comprehend, place trust in.

In conclusion, explainability constitutes a valuable instrument for rationalizing decisions made by AI systems. Its utility extends to several crucial aspects, including the validation of predictions, refinement of models, and the acquisition of fresh insights pertaining to the specific problem under consideration. This results in AI systems that can be trusted more readily, and opens up to many more different applications. XAI has the potential to deliver substantial advantages across a broad spectrum of domains that depend on AI systems [22].

In this paper, we address the problem of explaining why a specific decision has been made by the AI system. Usually, methods that deal with this problem (i.e., *local explainers*, in contrast to *global explainers* which try to explain the overall workings of the system) assume that the complex decision function which guides the model's behavior is approximable with an interpretable model in the neighborhood of the target instance we are trying to explain (e.g., [18], [23], [24]). While the above-mentioned works specifically target the local explanation problem, our firm conviction is that explanations should not only furnish insights into the model's behavior near a target instance but also empower users to comprehend the broader functioning of the model, encompassing scenarios with unknown inputs. In other words, local and global explanations should be merged to provide a deeper understanding of the model's prediction with respect to the knowledge the model has acquired during its training process.

In addition, most previous approaches are evaluated considering benchmark datasets, but their scalability to big data is often overlooked penalizing the real-world applicability of such solutions.

In this paper, we extend a recently introduced XAI methodology known as "Cluster-Aided Space Transformation for Local Explanations (CASTLE)" by the same authors, as presented in [25]. This extension seeks to amalgamate both global and local information to cater to the demands of handling massive datasets. The interpretation of the AI system's global operations is achieved through a *clustering phase* which is designed to identify regions within the instance space where data points not only accurately represent the underlying data distribution but are also consistently classified by the decision model. To enhance interpretability, we furnish the human user with an axis-aligned hyper-rectangle encompassing the instances within a cluster, presented as a logical rule, to facilitate comprehensibility. In addition, the local behavior of the model is leveraged to assist the user in comprehending the factors that bolster or attenuate a prediction. Specifically, we propose a transformation of the instance space that enables us to infer how alterations in feature values would impact the prediction probability.

In marked contrast to the original iteration of CASTLE, the formidable challenges posed by high-volume, diverse, and rapidly evolving datasets prompted the development of a specialized framework, rooted in cutting-edge big data technologies, specifically tailored for the deployment of CASTLE. Our experiments on five massive-scale datasets prove the effectiveness of the proposed framework in terms of temporal efficiency, cluster quality, and model significance.

To sum up, our contributions are as follows:

- We extend CASTLE [25] ("Cluster-Aided Space Transformation for Local Explanation"), a recently introduced explainable AI (XAI) methodology, to handle huge datasets.
- In contrast with CASTLE's original proposal, our solution leverages big data technologies (e.g., Apache Spark) and distributed computing (Hadoop file system) to approximate and make scalable the computation of the explanations.

- We conduct an extensive evaluation on five datasets with different big data characteristics (i.e., volume, variety, velocity) aiming at proving the effectiveness of the proposed extension.

The paper is organized as follows. Related works about eXplainable Artificial Intelligence (XAI) are summarized in the "Related Work" section while the "Cluster-Aided Space Transformation for Local Explanations" section describes the proposed methodology over big data architecture. Finally, the "Evaluation" section analyzes experimental results using different real-world datasets while in the "Conclusion and Future Work" section, we discuss several conclusions and future research directions.

Related Work

eXplainable Artificial Intelligence (XAI) aims to develop intelligent systems capable of elucidating their decision-making processes. Recent years have seen the emergence of various comprehensive surveys on XAI (e.g., [19, 26, 27]), focusing on the technical and comparative analysis of multiple methodologies. These studies contribute to the construction of AI/ML systems that transition from being opaque to transparent, employing a local-to-global framework. The growing reliance on decision-support systems that operate as "black boxes," particularly in critical sectors such as healthcare, security, and defense has underscored the need for XAI techniques. These techniques enhance user trust by improving the transparency and predictability of such systems [17].

In the field of healthcare, [28] examines XAI methods aimed at boosting accountability, transparency, and reliability in medical applications. Lamy et al. [29] proposed a visual explanation method for breast cancer recognition, which hinges on the quantitative and qualitative alignment of user queries with retrieved cases. Viswan et al. [16] study the application of XAI methods to Alzheimer's disease (AD) detection. Similarly, in the financial sector, Moscato et al. [30] discuss the integration of XAI techniques with Information and Communication Technologies (ICTs) to simultaneously mitigate risks and enhance efficiency. Furthermore, Rong et al. [31] present a comprehensive survey on XAI applications in social sciences, highlighting key findings and ongoing challenges.

Recent advancements in XAI have concentrated on elucidating the inner workings of black-box models, primarily through *feature importance* and *rule-based* methodologies. Feature importance methods aim to demystify model behavior within specific instances. These methods typically employ intrinsically interpretable models, such as linear models or decision trees, to assign significance to each feature. Prominent among these is the *Local Interpretable Model-agnostic Explanations* (LIME) technique [18], which provides local model-agnostic explanations, meaning it generates insights for specific predictions irrespective of the underlying ML model used. This is achieved by perturbing initial data to create new samples that help in forming explanations. *SHapley Additive exPlanations* (SHAP) [32] is another notable method in this category. It explains individual predictions using Shapley values, derived from game theory. Shapley values assess the impact of each feature on the prediction by considering the presence or absence of features in a *coalition* that contributes to the prediction and evaluating how the addition of a feature alters the prediction outcome. Furthermore, *Pattern Aided Local Explanation* (PALEX) [33] provides instance-level explanations by examining local behavior and utilizing frequent pattern sets from training data to identify locally discriminative features near a test instance. On the other hand, rule-based methodologies offer explicit rules that enhance user understanding of decision boundaries. *Interpretable Decision Sets* (IDS) [34] exemplify this approach by generating a set of independent "if-then" rules to optimize their coverage, precision, and complexity. Ribeiro et al. [24] introduced a model-agnostic method evolving from LIME, which combines local explanations with rule-based interpretability, forming *anchors* that anchor predictions locally. Additionally, other approaches like those in Guidotti et al. [35] and Grover et al. [36] employ global information to generate local explanations, including both supporting and counterfactual rules. However, these methods face challenges due to the localized nature of their explanations. For example, LIME's linear model explanation is only valid within a narrow vicinity of the target instance, beyond which its accuracy diminishes. This limitation is due to the absence of information about the neighborhood's size where the explanation remains valid. To overcome these challenges, CASTLE [25] integrates "global information" from the model into the local explanation process. This is achieved through a clustering phase to create pure clusters, characterized by high inter-cluster variance and low intra-cluster variance, ensuring homogeneous classification. These enhanced rule-based explanations include feature importance values and directional information supporting the prediction. This approach not only provides rule-based explanations but also defines supporting and opposing directions for the prediction, offering a comprehensive understanding of model behavior in various scenarios. Table 1 provides a comparative summary of the discussed related work.

In contemporary research on eXplainable Artificial Intelligence (XAI), a notable gap persists in the scalability and adaptability of prevalent explanation techniques within big data frameworks. Prevailing methodologies, while robust in standard settings, encounter significant challenges when applied to extensive, complex datasets characteristic of big data environments. This limitation poses a critical constraint,

Table 1 Related work

Method	Ref.	Approach	Use case	Advantages	Limitations
LIME	[18]	Approximates local predictions with an interpretable model	Individual predictions	Model-agnostic, versatile	Local focus, no global insights
SHAP	[32]	Uses Shapley values for explanations	Individual and global explanations	Consistent, accurate, includes feature importance	High computational cost
Anchors	[24]	Provides high-precision, rule-based local explanations	Rule-based explanations	Easy to interpret, model-agnostic	Simplistic for complex models, local focus
IDS	[34]	Generates interpretable if-then rules	Tabular data	Clear explanations, suitable for tabular data	Not ideal for complex of unstructured data
PALEX	[33]	Leverages frequent patterns in training data to provide explanations	Individual predictions	Leverages frequent patterns and provides multiple explanations	Assumes the availability of a set of frequent patterns that summarizes the possible local classification regions of the training data
LORE	[35]	Generates local, rule-based explanations with counterfactuals	Explanations for individual decisions with alternative scenarios	Provides factual and counterfactual reasoning	Focused on local explanations, may not represent global model behavior
BEEF	[36]	Generates natural language explanations balancing different aspects of forecasts	Forecasting models, where clear explanation in plain English is required	Accessible, user-friendly explanations in English	Specific to forecasting models, may lack technical depth
CASTLE	[25]	Enhances local explanations through cluster-aided space transformations	Complex models where clustering can aid in interpretation	Leverages clustering for more nuanced local explanations	High computational cost

particularly given the exponential growth and intricacy of contemporary datasets across diverse domains. Addressing this gap, our work contributes to the evolution of XAI by implementing and experimenting with the CASTLE approach in a big data context. CASTLE, with its unique approach of leveraging clustering mechanisms for explanation generation, inherently aligns with the complexities of large-scale data. This alignment is pivotal, as clustering can effectively segment vast datasets into manageable subsets, enabling more precise and computationally efficient explanations.

The application of our method in real-world industrial scenarios offers significant potential. In sectors such as finance, healthcare, and manufacturing, where decisions based on big data analytics have far-reaching consequences, our approach can provide clear, localized explanations for model predictions. This clarity is vital for compliance with regulatory standards, such as the General Data Protection Regulation (GDPR), which mandates the right to explanation for automated decisions. Moreover, in industries where predictive maintenance is crucial, such as manufacturing, our method can elucidate the reasoning behind predictions, enabling more informed and proactive decision-making processes.

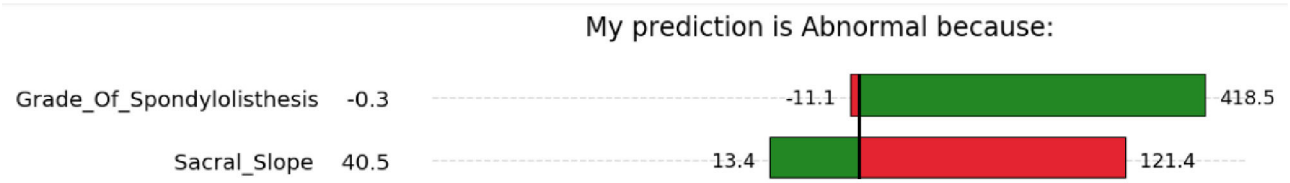
Supporting this approach, recent research has highlighted the necessity for scalable XAI solutions in industrial settings. For instance, studies by Ribeiro et al. [18, 24] and Lundberg et al. [32] have underscored the importance of model-agnostic explanations, but also their limitations in terms of scalability and computational overhead. CASTLE's innovative application in a big data context not only builds upon these foundational works but also transcends their limitations, offering a pragmatic and effective solution for real-world industrial applications. This advancement in XAI methodology is poised to significantly enhance the interpretability, trustworthiness, and applicability of machine learning models in the era of big data.

Cluster-Aided Space Transformation for Local Explanations

The key idea behind our approach is to design a novel XAI methodology, named *Cluster-Aided Space Transformation for Local Explanations* (CASTLE), capable of working on the top of a big data architecture. In particular, clusters are the pillar of this approach, and the description of the cluster incorporating the target instance is the key to obtain a comprehensive and faithful "global" explanation. Despite its real shape, the cluster provided with the explanation has to be described in a way users can easily understand. For example, when working with tabular data, a good choice for cluster representations could be the *axis-aligned hyperrectangles* which contain cluster instances. Formally, a cluster

Feature	Value
Grade_Of_Spondylolisthesis	-0.254399986
Sacral_Slope	40.47523153
Pelvic_Radius	98.67291675
Lumbar_Lordosis_Angle	39.60911701
Pelvic_Tilt	22.55258597
Pelvic_Incidence	63.027817500000005

(a) Example of target instance



(b) Example of CASTLE explanation

Fig. 1 Example of CASTLE explanation

C_j, $j \in \{1, ..., k\}$ (k being the number of clusters), can be completely characterized by the following coordinates:

$$C_j = ((l_j^1, u_j^1), (l_j^2, u_j^2), ..., (l_j^d, u_j^d)) \quad (1)$$

with d being the dimensionality of data (i.e., the number of features) and (l_j^i, u_j^i) being the lower and upper bound respectively for cluster C_j along the dimension (i.e., features) i.

To show an example of its working, let us consider again the instance from the *vertebral column dataset*[2] (Fig. 1a). For simplicity, let us just consider the top two features: *Grade of Spondylolisthesis* and *Sacral Slope*. A possible cluster could be described in the form:

$$-11.1 \leq Grade\ Of\ Spondylolisthesis \leq 418.5$$

$$13.4 \leq Sacral\ Slope \leq 121.4$$

As will be detailed later, clusters' description can be used as an explanation only if the properties of purity, coverage, and overlap are enforced. Moreover, it is worth noting that the cluster's representation as an axis-aligned hyper-rectangle is a necessary choice to favor the interpretability of explanations in the trade-off between expressive power and interpretability. For example, considering data distributed as a slanted line in a high-dimensional space, our description would be interpretable but it would include unnecessary volume; on the other hand, an elliptical description would be more accurate but not interpretable at all.

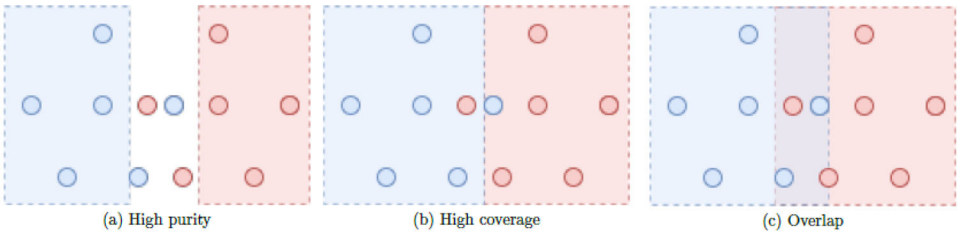
The cluster alone does not provide much information about the prediction outcome. That is why you need to combine this piece of global knowledge to the local behavior of the model around the target instance, to provide a clear and comprehensive explanation. Figure 1b shows the result of CASTLE applied to the above example. The identified clusters are represented in the feature values range under the form of rules, which means the patient is labeled as *"Abnormal"* because his *Grade of Spondylolisthesis* level lies in the range $[-11.1, 418.5]$ and his *Sacral Slope* value lies in $[13.4, 121.4]$. Furthermore, the green and red areas show the supporting and opposing directions to the prediction, respectively. This means, should the patient *Grade of Spondylolisthesis* level increase, the likelihood of him being considered *"Abnormal"* would increase. While an increase in his *Sacral Slope* value would weaken the current prediction.

The overall explanation extraction process in CASTLE consists of two main phases:

1. ***Preparation***: one-time fitting of a clustering model which represents the global knowledge of the predictive model.
2. ***Explanation***: extraction of both global and local behavior of the classification model. Global behavior is retrieved by finding the cluster which most likely applied to the

[2] http://archive.ics.uci.edu/ml/datasets/Vertebral+Column

Fig. 2 Clusters properties



target instance, while local behavior is obtained by analyzing model's outputs for instances in the neighborhood of the target.

Preparation Phase

The goal of the preparation process is to find a set of clusters grouping instances which share a common behavior and which are homogeneously classified by the predictive model. More in detail, the classification model $f(\cdot)$ is used to obtain labels for data on which the model has been trained, which are the basis of its knowledge. Then, a clustering procedure aims to mine the knowledge acquired by the model, and representative points for each cluster (named *pivots*) are eventually pulled out and will be exploited by the explanation phase. The clustering procedure, through the identification of the axis-aligned hyper-rectangle which includes the cluster instances, enables the possibility to provide logic rules as a part of the explanations, which can be easily understood by humans. For this reason, not only should clusters respect the usual clustering properties (like low intra-variance and high inter-variance), as shown in Fig. 2, but they should also have the following characteristics:

- ***High purity***: The user expects the logic rule to be trustworthy and, hence, that every instance lying in the hyper-rectangle has the label specified by the rule (the same label as the *target instance* being explained). Given a cluster C_j, its *purity* can be defined as the percentage of instances lying in C_j with the majority-class label.
- ***High coverage***: It is simple to obtain completely pure clusters, even in highly overlapped datasets, if we allow the clustering algorithm to not cover every instance (refer, for an example, to Fig. 2a). At the same time, it is simple to cover every instance if there are no purity restrictions (refer to Fig. 2a). In real-world scenarios, where there usually is no well-defined boundary between classes, it is necessary to handle a trade-off between purity and coverage: a high purity increases the user trust towards the explanation system, but the not-covered areas of the instance-space result in the inability to provide logic rules when required.
- ***Low overlap***: The area shared by clusters representing different classes must be as small as possible. As a matter of fact, the model's behavior is uncertain in these areas, and that could jeopardize users' trust towards the system. Figure 2c shows an example of overlap. Formally, overlap is defined as the mean of the *Intersections-over-Unions (IoUs)* between the clusters (axis-aligned hyper-rectangles) representing different classes. So, given a classification problem with s classes and, consequently, s cluster sets $\mathbb{C}^0, \mathbb{C}^1, ..., \mathbb{C}^{s-1}$, overlap can be computed as follows:

$$overlap = \frac{1}{\sum_{i=0}^{s-2} |\mathbb{C}_i| \left| \bigcup_{j=i+1}^{s-1} \mathbb{C}_j \right|} \sum_{i=0}^{s-2} \sum_{C_w \in \mathbb{C}_i} \sum_{C_z \in \bigcup_{j=i+1}^{s-1} \mathbb{C}_j} IoU(C_w, C_z) \tag{2}$$

Once the appropriate cluster set is found, the last step of the *preparation* phase consists of extracting one or more representative points, called ***pivots***, from each cluster. They will be exploited to retrieve the explanation in the following phase. A simple, yet efficient choice for pivots could be the clusters' centroids. Note that this phase is totally absent in *LIME*, as it lays the foundation for the "global" understanding of the model, described in the following section. So, if on the one hand it adds overhead to the methodology, on the other hand, it allows for a complete and exhaustive explanation.

Explanation Phase

The explanation phase represents the core of the proposed methodology. Its aim is to extract a comprehensive explanation for the target instance outcome, basing both on the ***global*** and ***local*** behavior of the predictive model.

The "global" understanding of the model is achieved by finding the cluster which is most likely applied to the target instance. This could be simply implemented by selecting the cluster whose centroid is closer to the instance. Once the target has been assigned to a cluster, the extraction of the rule consists of finding the axis-aligned hyper-rectangle that represents it (for example, you could select all the instances

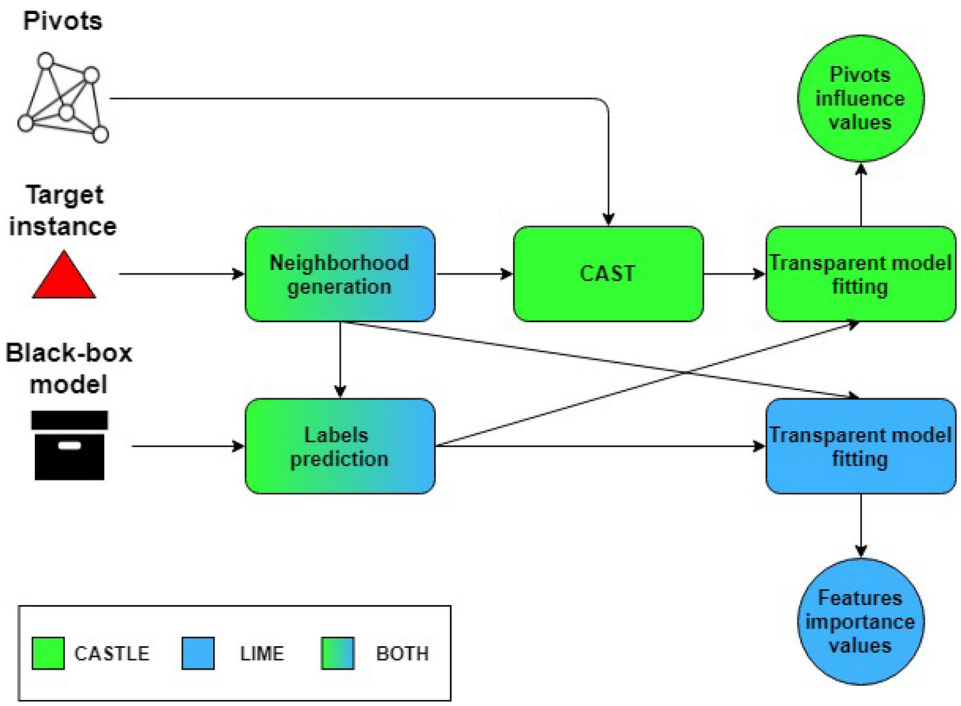


Fig. 3 Explanation phase: local behavior

belonging to the target cluster and define the boundaries by computing the minimum and maximum value for each feature).

The process for understanding the local behavior of the model is shown in detail in Fig. 3. As a matter of fact, CASTLE and LIME share the same phases in this process, applied in a slightly different way. So, you can both get information about the main features contributing to the prediction, and also estimate how much the prediction is influenced by the proximity of the target instance to the pivots retrieved from the clusters.

First of all, the neighbors of the target instance define its locality. They are commonly generated by random perturbation of the instance to be explained. Every neighboring instance generated is forecasted by the input black-box model, with the underlying concept being that we can reconstruct the model's specific behavior within the vicinity established by these generated instances.

The novel idea of our approach is an instance-space transformation, named CAST, such that each feature represents the proximity of the target instance to a pivot. Let us consider a binary two-dimensional problem where instances are preliminarily divided into two clusters, and their centroid is the pivots p_0, p_1. Given a target instance $t = (x_0, x_1)$, it can be encoded into a new variable $t' = (x'_0, x'_1)$, where $x'_i = proximity(t, p_i)$ is a measure of similarity between t and the pivot p_i. A simple example of a proximity function is as follows:

$$proximity(t, p_i) = \frac{1}{1 + d(t, p_i)} \tag{3}$$

where $d(t, p_i)$ is the distance between t and p_i. Figure 4 shows an example of transformation using Eq. 3 as a proximity function. As you can see, red points in the new space are on the top-left region of the plot because they have a small proximity to pivot 1 and a big proximity to pivot 2, and vice versa.

Finally, the *transparent model fitting* phase consists in learning a transparent model (e.g., linear regressor, decision tree) on the neighbors, exploiting the black-box model predictions. In other words, the proposed model encapsulates the internal mechanisms of the black-box model within the proximity of the target instance. More precisely, the interpretable model, trained on modified data, calculates a weight for each pivot, signifying the extent to which the proximity of the target instance impacts the prediction. For instance, the output $y(t)$ of a linear model can be expressed as follows:

$$y(t) = \sum_{p \in \mathbb{P}} proximity(t, p) \cdot w_p, \tag{4}$$

where t is the target instance, $\mathbb{P}$ the set of pivots, and w_p the learned weight for pivot p which represents its influence to the prediction.

It is worth noting that the space transformation has no impact on the behavior of the black-box model, as it was queried with the neighbors in the original space.

Furthermore, an additional interpretable model can be trained using the original neighbors, akin to the approach in the LIME framework. This helps in understanding the influence of data features on the local prediction. It can be

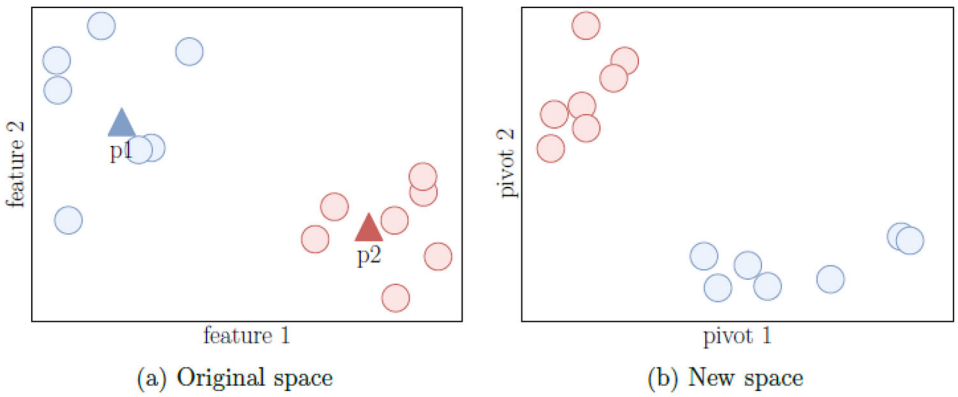


Fig. 4 Example of the pivot-based space transformation

insightful to view the output of the interpretable model fitted on CAST data as if it were a combination of gravitational forces acting on the target instance to guide it towards its predicted outcome, as depicted in Fig. 5. Each pivot, denoted as p, exerts an attractive force when its weight w_p is positive (indicating a positive contribution to the prediction), and conversely, a repulsive force when the weight is negative. The impact of each pivot is determined by both its weight and its proximity to the target instance: reducing the distance between the target instance and a positively weighted pivot enhances the likelihood of predicting the corresponding class. The resultant force represents a vector pointing towards the most supportive region for the prediction $y(t)$, with its magnitude signifying the predicted value.

How to Choose Pivots

When performing a transformation of the instance space, it is crucial to guarantee the preservation of information from the original space in the new one. From a mathematical standpoint, this entails ensuring the injective property of the space transformation: every element in the transformed space corresponds to at most one element in the original space. Achieving this requires the involvement of an appropriate number of pivots in the transformation process. To be more specific, it is imperative to select a minimum of $d + 1$ pivots (where d represents the dimensionality), as illustrated in Fig. 6 through a simple two-dimensional example.

The choice of pivot selection strategy within the system may vary depending on the user involved. On one hand, the level of comprehensibility required by the audience differs based on whether they are end-users or considered "expert" analysts, as discussed in [27]. End-users may primarily seek an understanding of why the model made a specific prediction, such as why their loan application was not approved. Conversely, expert analysts may aim to scrutinize the model's vulnerabilities and seek improvements, such as identifying potential biases stemming from the training data. On the other hand, the level of familiarity with the model also differs among these user groups: developers possess knowledge about the training process and can access model-related data, while end-users are limited to interacting with the model through its interface.

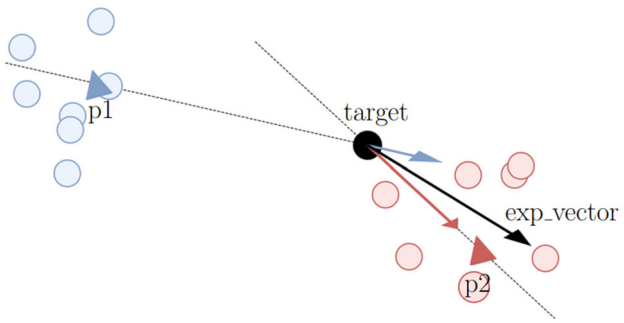


Fig. 5 Explanation vector as a sum of pivot contributions to the prediction

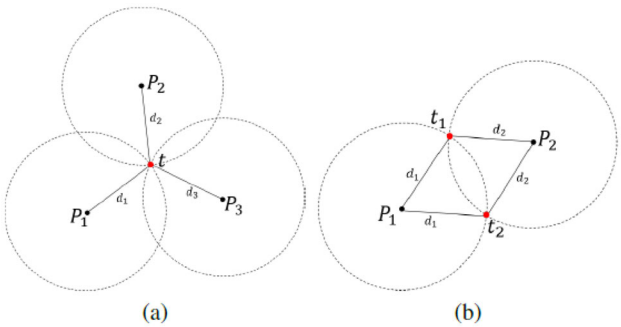


Fig. 6 **a** In a two-dimensional space featuring three distinct pivots, if the Euclidean distance is employed as the proximity metric, the instance denoted as t can be transformed into $\hat{t} = (d_1, d_2, d_3)$. The presence of three non-aligned pivots ensures that $\hat{t}$ corresponds exclusively to the original instance t, as it is the solitary point of intersection among three circles centered on the pivots and having their respective distances as radii. **b** When only two pivots are selected, the transformed instance $\hat{t} = (d_1, d_2)$ matches with two distinct points, namely, t_1 and t_2, which represent the intersections of the two circles formed by the selected pivots. This scenario may result in a potential loss of information

CASTLE provides users with access to training data and knowledge about the training process, such as developers and expert analysts, with the means to thoroughly investigate the dynamics of the model. This investigation involves the analysis of both the supporting directions and the relevance values of the chosen pivots. Expert analysts with data access can exercise control in extracting pivots by leveraging clustering techniques and their domain expertise. This approach enables them to select pivots that ensure a significant separation between classes within the transformed space. For instance, there are several strategies based on pivot selection metrics that can be effectively employed to enhance pivot selection and subsequently improve nearest neighbor or range query searches, as elaborated in [37].

When an end-user has data access but lacks the expertise or time to manually choose pivots, an automated explanation process can be facilitated using a random selection strategy. It has been empirically demonstrated that this approach does not negatively impact performance. In the broader context, for end-users who have neither data access nor training information, a fully automated procedure can be employed, where pivots are selected through random perturbation of the target instance.

CASTLE on Big Data Architecture

The process has been carried out with a *bottom-up* approach, by identifying the main steps and entities that characterize an explanation process and connecting them together to create the final framework that relies on big data technologies. This is mainly because of their modularity: in fact, it is easily replace the underlying machine learning model or just as easily replace the interpretation method. These are the reasons why it is possible to create a generalizable architecture, which can just as easily be extended to embrace more and more algorithms as the research goes on. The milestone steps of the post hoc local explanation process are as follows:

1. Get the class predictions for your data through a **black-box** ML algorithm.
2. Generate a **neighborhood** around the target instance you want to provide an explanation for.
3. Make the black-box model (used in the first step) **predict** the class label for the new instances.
4. Fit a simple, **transparent model** on the newly generated neighbors to get an explanation of the model behavior in the locality of the target instance.

By following this schema, it was possible to implement the framework shown in Fig. 7. For the sake of readability and clarity, only the main methods (directly involved in the explanation process) are displayed. In the following sections, each entity represented in the architecture will be discussed.

The first package, named *data*, contains the entities to handle different data (*TextData*, *ImageData*, or *TabularData*) from an external source. Successively, model connectors are defined in order to implement the independence of the explainer from the specific black-box classifier. In particular, ***Model*** is $< interface >$, which provides the method *predict()* to get the label predictions of the black-box model for the instances generated in the target instance neighborhood while ***ScikitConnector*** and ***SparkMLConnector*** are respectively the connector for Python *scikit-learn* machine learning library and *MLLib* from *Apache Spark* for integrating whichever ML model. The explainer package, then, represents the core of the whole explanation process, whose main entities are as follows:

Explainer This is the general interface that allows to produce an explanation for the behavior of a black-box model. As you can see from the architecture Fig. 7, it makes use of the model it is going to explain, and of course the data to retrieve all the information it needs. Once the explainer has computed all the steps, it instantiates an *Explanation* object, which will be analyzed later. In particular, it is possible to distinguish between global and local explainers, basing on your scope. *GlobalExplainer* can be used for those techniques, whose goal is to provide a global explanation of the whole ML model, so as to understand the entire logic behind it. On the other hand, *LocalExplainer* can be exploited to produce an explanation for a target instance, without having to go into the details of the whole model understanding.

LIMEExplainer This is the main class for the implementation of *LIME* [18]. Being a local explanation strategy, it expands *LocalExplainer* and provides an explanation for a target instance. According to the type of data you are going to handle, you can either use *LIMETextExplainer, LIMEImageExplainer* or *LIMETabularExplainer*. This is the class responsible for the neighborhood generation process and the definition of the locality of the target instance. Once the needed computations have been completed, it calls the method *explain_instance_with_data()*, provided by *LIMEBase*, to generate the explanation.

LIMEBase This is the final step of the explanation process, where the simple, transparent model is fitted on the neighborhood data, previously labeled by the classifier, to generate an interpretable explanation for the target instance outcome.

CASTLEExplainer It implements another local explanation methodology. Its inner working is similar to *LIME*, but it additionally exploits global knowledge about the data, through a clustering technique, to provide more information about the outcome.

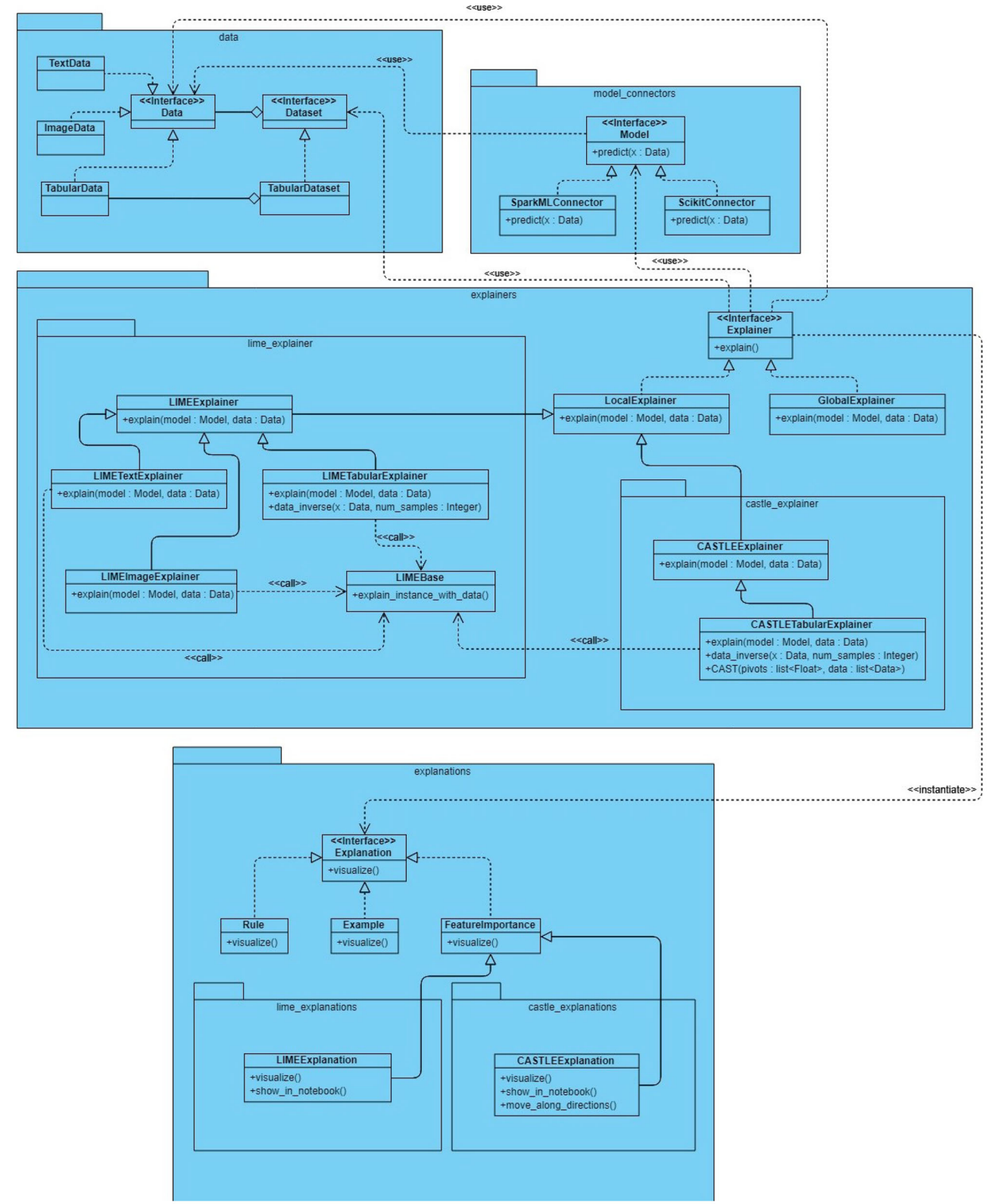


Fig. 7 General framework for our method

Table 2 Datasets characterization

Dataset	Original size	Scaled size	#features	#labels
Avila[a]	20.867	1.053.430	10	12
Diabetes[b]	768	2.001.408	8	2
Magic Gamma Telescope (MGT)[c]	19.020	3.005.160	10	2
HTRU[d]	17.898	4.009.152	8	2
SUSY[e]	5.000.000	/	18	2

[a] https://archive.ics.uci.edu/ml/datasets/Avila
[b] https://archive.ics.uci.edu/ml/datasets/diabetes
[c] https://archive.ics.uci.edu/ml/datasets/magic+gamma+telescope
[d] https://archive.ics.uci.edu/ml/datasets/HTRU2
[e] https://archive.ics.uci.edu/ml/datasets/SUSY

Finally, the explanation package contains the possible explanation forms (i.e., rule, example, or FeatureImportance) you can get from an explainer.

Evaluation

In this section, we will show the performance results obtained from the implementation of the proposed technique (*CASTLE*) in a big data environment. As a matter of fact, the main difference between LIME and CASTLE is that the former is a "simple local explainer," which means, it does not require any preparation phase and it only focuses on the locality of the target instance. On the other hand, CASTLE provides an explanation by combining both local and global behavior of the prediction model. Hence, when handling large-scale datasets, not only does the adoption of big data paradigms allow for better performance, but it even becomes necessary if the amount of data to be processed exceeds the computational capabilities of traditional architectures. Thus, in a nutshell, the evaluation process can be split into two phases. The former is the ***Clustering evaluation*** to assess the effectiveness of the clustering model, by finding the appropriate number of clusters for the related dataset and by evaluating the quality metrics proposed in the "Preparation Phase" section. The latter is the ***Performance evaluation*** to evaluate the efficiency of the technique by comparing it to the original "non-Big Data" algorithm after identifying the best clustering configuration for each dataset.

The framework has been tested on different open-source datasets, whose main features are summarized in Table 2. In order to show the differences in performance as the amount of data increases, a simple process of "*data augmentation*" has been applied to most of the datasets by replicating the original instances.

The black-box classification model used on the large-scale datasets is a *RandomForestClassifier*, provided by spark.ml library.[3] In configuring our RandomForest model, we selected parameters to optimize both accuracy and computational efficiency. The model is set with a maximum depth of 4, allowing sufficient complexity while avoiding overfitting. We used 32 maximum bins for discretizing continuous features, ensuring efficient computation. A minimum of one instance per node and a minimum information gain of 0.0 were chosen to capture detailed patterns in the data. The "gini" impurity measure was employed for its effectiveness in node purity assessment. The forest consists of 100 trees, balancing performance and computational demand. A subsampling rate of 1.0 allows each tree to learn from the entire dataset, enhancing model robustness. No minimum weight fraction per node was set, offering flexibility in tree growth, and bootstrap sampling was used to promote diversity in the forest's trees.. This model has been trained on each dataset, guaranteeing good prediction performances ($> 80\%$). The validation procedure employed is a held-out method, wherein 70% of the data instances are allocated for training, 20% for validation, and 10% for the test set. Similarly, in order to compare the proposed framework to the stand-alone version, we have used a *RandomForestClassifier* from Python *scikit-learn* library,[4] tuned in the same way.

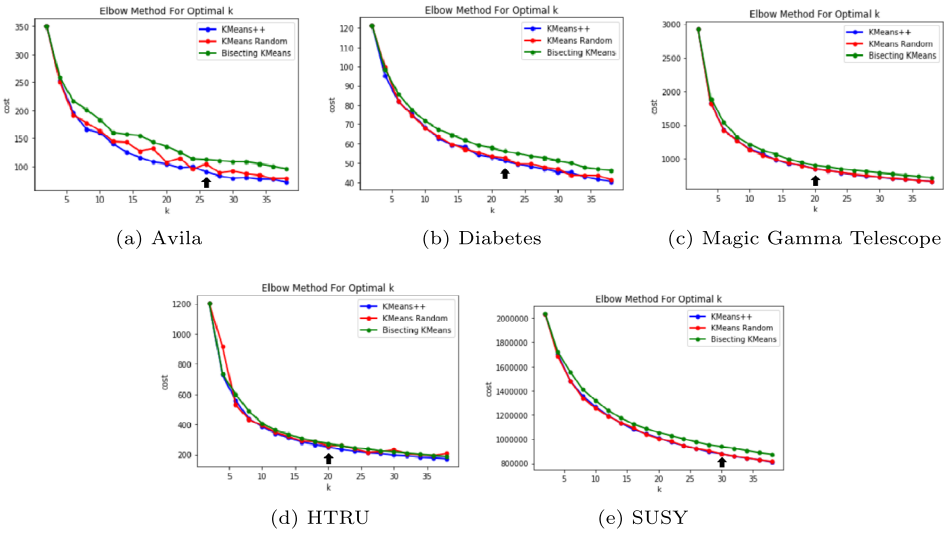
Clustering Evaluation

The first step of the framework evaluation concerns the clustering phase. In this section, we will first compare two clustering algorithms, provided by *spark.ml*, on the different datasets. Then, once established the best configuration, we will evaluate the quality metrics proposed in the "Preparation Phase" section. The clustering methods used are as follows:

- ***KMeans***: Given a set of n observations, KMeans clustering aims to partition them into $k(\geq n)$ sets so as to minimize the within-cluster sum of squares. *spark.ml* allows to initialize this algorithm in two modes: (i) *KMeans Random*, representing the standard KMeans method where k centers are chosen randomly and then,

[3] https://spark.apache.org/mllib/

[4] https://scikit-learn.org/stable/



(a) Avila (b) Diabetes (c) Magic Gamma Telescope

(d) HTRU (e) SUSY

Fig. 8 Inertia method for clustering algorithms

in every iteration, the algorithm assigns each instance to the closest center and recomputes the centers, until convergence, and (ii) *KMeans //*, a parallelized variant of the *KMeans++* method where the first k centers are computed basing on the data distribution, in order to find the optimal clusters in fewer iterations [38].

- ***Bisecting KMeans***: This approach combines elements of both divisive hierarchical clustering (top-down clustering) and KMeans clustering. Instead of dividing the dataset into k clusters in each iteration, the bisecting KMeans algorithm progressively divides a single cluster into two sub-clusters at each bisecting step, accomplished through KMeans, until a total of k clusters are achieved.

Both the clustering algorithms require you to set the number of clusters k. So, in order to optimize the results, we tried to find the best trade-off between k and the $cost$. The latter refers to the inertia value, which is defined as the average squared distance between each instance and its nearest centroid. In accordance with this definition, a lower inertia value indicates a better-performing model. The method consists in running the algorithm several times while increasing the number of clusters ($k \in [2, 40]$), and plotting the graph of inertia as the number of clusters increases. Note that the evaluation has been executed on the original sized datasets and the inertia value is strictly dependent on the size of the dataset. As a matter of fact, *SUSY*, which already had 5.000.000 instances, has much higher inertia values than the other datasets, but the process is of course unchanged.

Hyperparameters Tuning

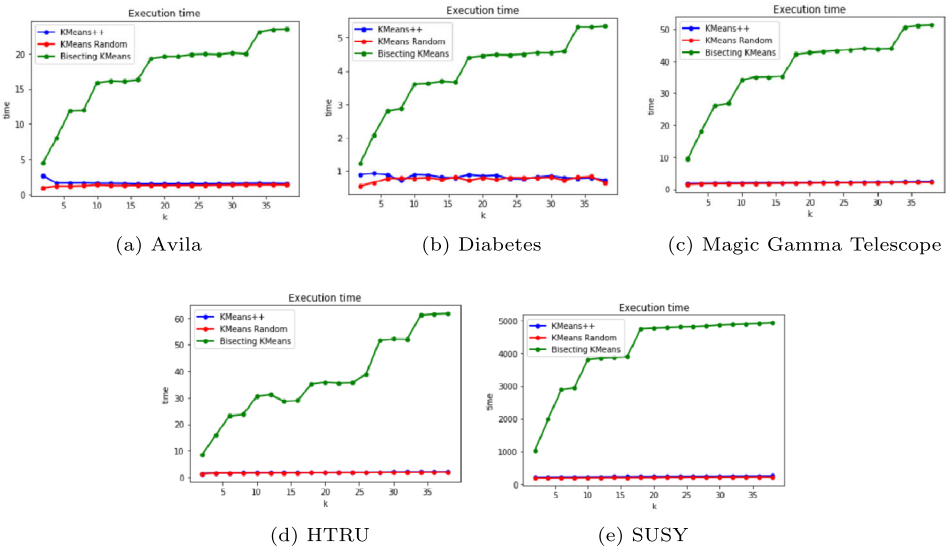
The initial phase of our evaluation involves identifying the optimal value of k for the two clustering algorithms we have previously delineated. Prior to executing these algorithms, we employed a MinMaxScaler from spark.ml to preprocess the data, enhancing the efficacy of the clustering process. The outcomes of this procedure on various datasets are illustrated in Fig. 8.

It is noteworthy that the algorithms exhibit minimal divergence in terms of inertia, attributable to their shared foundation in the KMeans methodology. Both KMeans Random and KMeans// yield remarkably similar results, which aligns with expectations given their primary distinction lies in the initialization phase. Conversely, Bisecting KMeans consistently underperforms in comparison, as evidenced across all datasets. An additional crucial aspect is the execution duration; the time required for cluster computation significantly contributes to the overall explanation process, necessitating adherence to reasonable time constraints. Figure 9 displays the execution times for the aforementioned algorithms. While the inertia values exhibited marginal differences, a stark contrast is apparent in execution times between KMeans and Bisecting KMeans. The sequential nature of Bisecting KMeans results in substantially longer execution times compared to KMeans, rendering it less feasible in practical scenarios.

In conclusion, Table 3 presents the selected hyperparameter k for each dataset.

Clustering Quality

As detailed in the "Preparation Phase" section, the desired characteristics of a clustering result to be useful within the explanation process are as follows: (i) *High purity*, so as to guarantee the trustworthiness of the final explanations, which rely on the clustering results, (ii) *High coverage*, since you want the clusters to cover as many instances as possible, and (iii) *Low overlap* between clusters representing different classes.



(a) Avila (b) Diabetes (c) Magic Gamma Telescope

(d) HTRU (e) SUSY

Fig. 9 Execution time for clustering algorithms

Aiming to take count of all these properties, we merged them together in a single quality metric, as shown below:

$$quality = w_p \cdot purity + w_c \cdot coverage + w_o \cdot (1 - overlap) \quad (5)$$

where w_p, w_c, w_o represent the weights associated with $purity$, $coverage$, and $overlap$, respectively. These weights are useful if, for some reason, the application prefers one property over the others. For example, if the dataset contains a lot of noise, it is advisable to set a low weight w_c for the $coverage$ term, in order to facilitate the rejection of noise. For my experiments, we assumed equal contributions and set $w_p = w_c = w_o = \frac{1}{3}$. The metric was evaluated using the optimal k retrieved in the previous section, but this time, instead of preprocessing the data with a scaler, we referred to the original instances, since the clusters' description is computed in the original space. Table 4a, b, and c show the final results for the different algorithms.

The analysis of the quality evaluation, as delineated in the table, reveals consistently high performance across most metrics for the clustering algorithms. A critical observation is that the coverage (C) metric uniformly registers at 1.0 across all datasets for each algorithm. This is an inherent characteristic of the KMeans methodology, which inherently ensures the inclusion of all instances in the clustering process, thus guaranteeing complete coverage.

Focusing on the overall quality (Q), the Bisecting KMeans algorithm generally lags behind its counterparts, with the notable exception being the Magic Gamma Telescope (MGT) dataset. In this specific instance, Bisecting KMeans outperforms due to a marginally superior purity (P) value. Purity, in this context, reflects the homogeneity of the clusters, suggesting that Bisecting KMeans, despite its overall lower efficiency, can achieve better homogeneity under certain conditions.

However, it is crucial to balance these quality metrics against the practical considerations of execution time. The Bisecting KMeans algorithm, while occasionally offering slightly higher purity, is significantly outpaced in terms of computational efficiency by the other KMeans variants. Given the substantial discrepancy in execution times, with Bisecting KMeans being markedly slower, the decision to favor the more time-efficient KMeans algorithms for all datasets appears justified. This decision is based on the premise of achieving optimal overall efficiency, which encompasses not just the quality of clustering but also the practical feasibility of the algorithms in terms of computational resources and time constraints.

Performance Analysis

The instance-space transformation, central to this paper, necessitates the calculation of a *proximity function*. This

Table 3 Selection of hyperparameter k

Hyperparameter	Avila	Diabetes	Magic Gamma Telescope	HTRU	SUSY
k	26	22	20	20	30

Table 4 Quality evaluation of different methods, in where C, P, O, and Q are respectively coverage, purity, overlap, and quality

(a) KMeans// quality evaluation				
Dataset	***C***	***P***	***O***	***Q***
Avila	1.0	0.8810	0.9975	0.9595
Diabetes	1.0	0.8188	0.9986	0.9391
MGT	1.0	0.7873	0.9962	0.9278
HTRU	1.0	0.9708	0.9999	0.9902
SUSY	1.0	0.7470	0.9903	0.9124
(b) KMeans Random quality evaluation				
Dataset	***C***	***P***	***O***	***Q***
Avila	1.0	0.8480	0.9980	0.9486
Diabetes	1.0	0.8228	0.9985	0.9404
MGT	1.0	0.7666	0.9942	0.9202
HTRU	1.0	0.9664	0.9999	0.9887
SUSY	1.0	0.7472	0.9877	0.9116
(c) Bisecting KMeans quality evaluation				
Dataset	***C***	***P***	***O***	***Q***
Avila	1.0	0.8461	0.9976	0.9479
Diabetes	1.0	0.8006	0.9973	0.9326
MGT	1.0	0.8017	0.9953	0.9323
HTRU	1.0	0.9665	0.9988	0.9884
SUSY	1.0	0.7341	0.9875	0.9072

function produces a numerical value that indicates the closeness or similarity between two data points, and it relies on the computation of the *distance* between these two instances.

We conducted an assessment of our approach concerning variations in both distance and proximity functions. Specifically, we examined the performance of the *Euclidean*, *Minkowski*, *Chebyshev*, and *Cosine* distance functions, along with the following proximity functions:

$$P_1(x_1, x_2) = \frac{1}{1 + \delta(x_1, x_2)} \tag{6}$$

$$P_2(x_1, x_2) = e^{-\delta(x_1, x_2)} \tag{7}$$

$$P_3(x_1, x_2) = -\delta(x_1, x_2) \tag{8}$$

$$P_4(x_1, x_2) = 1 - \frac{\delta(x_1, x_2) - min(\delta(x_1, x_2))}{max(\delta(x_1, x_2)) - min(\delta(x_1, x_2))} \tag{9}$$

$$P_5(x_1, x_2) = max(\delta(x_1, x_2)) - \delta(x_1, x_2), \tag{10}$$

x_1 and x_2 being the two data points, $\delta(x_1, x_2)$ representing a generic distance function and $min(\delta(x_1, x_2))$ and $max(\delta(x_1, x_2))$ representing the minimum and maximum distance observed in the whole dataset, respectively.

The findings presented in Table 5 demonstrate that the optimal proximity functions are those described by Eqs. 8 and 10, where the proximity exhibits a linear relationship with the distance. On average, cosine distance outperforms other methods while the most effective configurations typically involve euclidean or Minkowski distances.

The evaluation of the CASTLE methodology will be computed according to two main factors:

1. The **efficacy** of the technique will be evaluated considering whether CAST transformation affects the explanation process. Specifically, we have collected the *adjusted coefficient of determination* R^2 of the linear models learned on the original and transformed spaces. *Adjusted* R^2 is an indicative measure of the level of explained variability in the data set, used in statistical analysis to assess how well a model explains and predicts future outcomes; differently than R^2, it is independent of the number of variables which the linear model used during its fitting. Note that the datasets have been split into two sets in order to explain instances which neither the explainer nor the cluster algorithm had never seen, and confidence intervals have been computed.
2. The **efficiency** of the methodology will be evaluated by comparing the execution time on the original Python algorithm and on Spark, so as to observe the difference between a traditional environment and a big data engine when dealing with huge amounts of data.

Efficacy

As mentioned above, the efficacy of the technique will be evaluated through the $adjusted R^2$ score:

$$Adjusted - R^2 = 1 - \frac{n-1}{n-k-1} \cdot \frac{RSS}{TSS} \tag{11}$$

where n is the number of samples in the neighborhood (set to 1000), k is the number of features on which the model has been fitted, and RSS and TSS are the *residual sum of squares* and the *total sum of squares*, respectively, which provide an indicative measure of the level of explained variability in the data set.

Table 6 presents a comprehensive comparison of the performance metrics in the original space (LIME) versus the transformed space (CASTLE). This comparison is quantified using the average adjusted R^2 values, accompanied by their respective 95% confidence intervals, across various datasets.

A detailed analysis of the results reveals that an astute selection of pivot points in CASTLE does not adversely affect its performance relative to LIME. This is evident from the relatively close adjusted R^2 values for both methods across different datasets. For instance, in the Avila dataset, CASTLE

Table 5 Effects of distance and proximity functions

Dataset	Proximity	Distance Euclidean	Minkowski	Chebyshev	Cosine	***Mean***
	P_1	0.227	0.279	0.25	0.382	0.284
	P_2	0.216	0.215	0.158	0.571	0.29
	P_3	**0.689**	0.683	0.412	0.684	**0.617**
	P_4	0.33	0.333	0.26	0.648	0.393
	P_5	0.686	0.686	0.392	0.685	0.612
Bank	***Mean***	0.43	0.439	0.294	**0.594**	
	P_1	0.278	0.278	0.162	0.452	0.292
	P_2	0.152	0.157	0.141	0.428	0.219
	P_3	0.512	**0.517**	0.176	0.511	**0.429**
	P_4	0.395	0.395	0.139	0.436	0.341
	P_5	0.5	0.5	0.18	0.501	0.42
Titanic	***Mean***	0.367	0.369	0.161	**0.466**	
	P_1	0.312	0.315	0.209	0.637	0.368
	P_2	0.146	0.148	0.199	0.639	0.283
	P_3	0.701	0.702	0.216	0.718	0.584
	P_4	0.584	0.584	0.268	0.707	0.536
	P_5	0.692	0.694	0.333	**0.738**	**0.614**
Diabetes	***Mean***	0.487	0.489	0.245	**0.634**	
	P_1	0.136	0.138	0.225	0.436	0.234
	P_2	0.084	0.087	0.155	0.444	0.192
	P_3	0.581	**0.586**	0.15	0.563	0.47
	P_4	0.469	0.469	0.31	0.555	0.451
	P_5	0.576	0.574	0.147	0.554	**0.463**
Magic	***Mean***	0.369	0.371	0.197	**0.51**	
	P_1	−0.045	−0.045	0.007	0.254	0.043
	P_2	−0.061	−0.061	−0.042	0.368	0.051
	P_3	**0.582**	0.581	0.197	0.507	0.467
	P_4	0.44	0.437	0.123	0.534	0.383
	P_5	0.576	0.58	0.221	0.511	**0.472**
Spambase	***Mean***	0.298	0.298	0.1	**0.435**	
	P_1	0.078	0.078	0.027	0.278	0.115
	P_2	−0.001	−0.002	0.039	0.411	0.112
	P_3	0.563	0.56	0.097	0.509	**0.432**
	P_4	0.451	0.453	0.07	0.497	0.368
	P_5	0.559	**0.565**	0.086	0.513	0.431
Digits	***Mean***	0.33	0.331	0.064	**0.442**	

Values in bold indicate the highest results for each dataset

shows a slightly higher adjusted R^2 value (0.610 ± 0.016) compared to LIME (0.591 ± 0.015), suggesting that the transformation in CASTLE retains the explanatory power.

However, it is crucial to delve deeper into these results to understand the implications fully. The confidence intervals indicate the range within which we can expect the true adjusted R^2 value to lie with 95% certainty. A narrower interval denotes a higher precision of the estimate. For example, in the Diabetes dataset, the interval width for LIME is 0.058, while for CASTLE, it is slightly narrower at 0.047, indicating a more precise estimation in the transformed space.

Table 6 $Adjusted - R^2$ of LIME and CASTLE

Dataset	LIME	CASTLE
Avila	0.591 ± 0.015	0.610 ± 0.016
Diabetes	0.691 ± 0.058	0.715 ± 0.047
Magic Gamma Telescope	0.569 ± 0.083	0.605 ± 0.082
HTRU	0.587 ± 0.062	0.637 ± 0.036
SUSY	0.622 ± 0.071	0.619 ± 0.063

Efficiency

Figure 10 offers a detailed analysis of the computational efficiency by contrasting the execution time of the original CASTLE algorithm with its adaptation for the Spark engine, hereafter referred to as CastSpark. The comparison with LIME is not included in this analysis due to the additional computational steps inherent to CASTLE, namely the preliminary clustering phase and subsequent data transformation, which introduce additional complexity and processing time to the workflow.

The data presented in Fig. 10 indicates that CastSpark demonstrates a sub-linear growth pattern in execution time across varying dataset sizes. This suggests an improved scalability in the Spark-based architecture as the size of the data increases. Conversely, the stand-alone implementation of CASTLE displays a trend that could be characterized as exponential, with execution time escalating significantly with larger datasets.

The sub-linear trend observed with CastSpark can be attributed to Spark's distributed computing features, which enhance the efficiency of processing large datasets by leveraging parallel execution across multiple nodes. This is particularly evident in the comparison presented in subfigure (a), which displays the raw execution times, and is further elucidated by the trend line in subfigure (b), which more clearly delineates the comparative growth rates of execution time for both versions of the algorithm.

These findings underscore the importance of optimizing algorithmic implementations for scalable data processing frameworks like Spark, especially when dealing with large-scale data that can benefit from distributed computing paradigms. The results suggest that the CastSpark implementation is more suitable for larger datasets, where the computational overhead can be more effectively managed and mitigated.

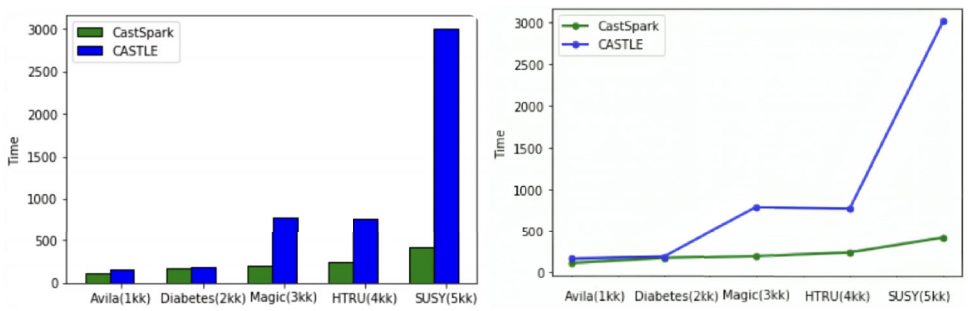
Conclusion and Future Work

The aim of the paper was to design a novel XAI methodology over a big data architecture. More in detail, *CASTLE* is a novel state-of-the-art XAI methodology, which provides a clear, exhaustive explanation for the predictions of a black-box classifier. Its strength lies in the exploitation of both the global (through clustering) and the local (through neighborhood generation) behavior of the model, through which not only does it generate a rule explanation, but it also provides supporting and opposing directions for the prediction. Two of the main drawbacks of the original algorithm are the overhead added with the clustering phase (which is highly related to the size of the dataset) and the *curse of dimensionality*, which strongly affects the computation of CAST transformation.

Instead, in a distributed environment like Spark, the first issue is addressed using specifically optimized clustering techniques. As for CAST transformation, it is a completely parallelizable process, so it can be easily and efficiently computed in a distributed architecture.

The main findings from our results can be summarized as follows:

- Our adaptation of CASTLE with PySpark shows sub-linear growth in execution time, indicating better handling of large datasets.
- We have achieved high purity and coverage with our clustering procedure.
- CASTLE shows similar adjusted R^2 to LIME and narrower confidence intervals, indicating more precise estimation in transformed space
- The cosine distance generally outperforms other distance methods.
- Bisecting KMeans shows marginally superior purity.



(a) Execution time (second) (b) Execution Time Trend (second)

Fig. 10 Execution time comparison between CASTLE and CastSpark

Future works will be devoted to extend the methodology to work with multimodal data (e.g., text and images), by of course redefining the concepts of neighborhood and clusters accordingly. In addition, we want to validate the applicability of our work for different kinds of problems (e.g., regression, text, or image generation). Finally, instead of focusing on general-purpose datasets, we would like to target specific application domains (e.g., finance, social network analysis) to investigate the practical utility of the framework.

Funding Open access funding provided by Università degli Studi di Napoli Federico II within the CRUI-CARE Agreement.

Data Availability The authors do not have permission to share data.

Declarations

Research Involving Human and Animal Participants This article does not contain any studies with human participants performed by any of the authors.

Conflict of Interest The authors declare no competing interests.

References

1. Firouzi F, Farahani B, Marinšek A. The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT). Inf Syst. 2022;107:101840. https://doi.org/10.1016/j.is.2021.101840.
2. Cao L. AI in finance: challenges, techniques, and opportunities. Comput Surv. 2022. https://doi.org/10.1145/3502289.
3. Huang C, Zhang Z, Mao B, Yao X. An overview of artificial intelligence ethics. IEEE Trans Artif Intell. 2023;4(4):799–819. https://doi.org/10.1109/TAI.2022.3194503.
4. Strouse D, McKee K, Botvinick M, Hughes E, Everett R. Collaborating with humans without human data. Adv Neural Inf Process Syst. 2021;34:14502–15.
5. Li Z, Li S, Luo X. An overview of calibration technology of industrial robots. IEEE CAA J Autom Sin. 2021;8(1):23–36. https://doi.org/10.1109/JAS.2020.1003381.
6. Zhou X, Chai C, Li G, Sun J. Database meets artificial intelligence: a survey. IEEE Trans Knowl Data Eng. 2022;34(3):1096–116. https://doi.org/10.1109/TKDE.2020.2994641.
7. Jiao L, Zhang R, Liu F, Yang S, Hou B, Li L, Tang X. New generation deep learning for video object detection: a survey. IEEE Trans Neural Netw Learn Syst. 2022;33(8):3195–215. https://doi.org/10.1109/TNNLS.2021.3053249.
8. Li J, Sun A, Han J, Li C. A survey on deep learning for named entity recognition. IEEE Trans Knowl Data Eng. 2022;34(1):50–70. https://doi.org/10.1109/TKDE.2020.2981314.
9. Li Z, Li S, Francis A, Luo X. A novel calibration system for robot arm via an open dataset and a learning perspective. IEEE Trans Circuits Syst II Express Briefs. 2022;69(12):5169–73. https://doi.org/10.1109/TCSII.2022.3199158.
10. Li Z, Li S, Bamasag OO, Alhothali A, Luo X. Diversified regularization enhanced training for effective manipulator calibration. IEEE Trans Neural Netw Learn Syst. 2023;34(11):8778–90. https://doi.org/10.1109/TNNLS.2022.3153039.
11. Castelnovo A, Cosentini A, Malandri L, Mercorio F, Mezzanzanica M. FFTREE: a flexible tree to handle multiple fairness criteria. Inf Process Manage. 2022;59(6):103099. https://doi.org/10.1016/j.ipm.2022.103099.
12. Bharati S, Mondal MRH, Podder P. A review on eXplainable Artificial Intelligence for healthcare: why, how, and when? IEEE Trans Artif Intell. 2023. https://doi.org/10.1109/TAI.2023.3266418.
13. Minh D, Wang HX, Li YF, Nguyen TN. eXplainable Artificial Intelligence: a comprehensive review. Artif Intell Rev. 2022. https://doi.org/10.1007/s10462-021-10088-y.
14. Ali S, Abuhmed T, El-Sappagh S, Muhammad K, Alonso-Moral JM, Confalonieri R, Guidotti R, Del Ser J, Díaz-Rodríguez N, Herrera F. eXplainable Artificial Intelligence (XAI): what we know and what is left to attain trustworthy artificial intelligence. Inf Fusion. 2023;99:101805. https://doi.org/10.1016/j.inffus.2023.101805.
15. Cambria E, Malandri L, Mercorio F, Mezzanzanica M, Nobani N. A survey on XAI and natural language explanations. Inf Process Manage. 2023;60(1):103111. https://doi.org/10.1016/j.ipm.2022.103111.
16. Viswan V, Shaffi N, Mahmud M, Subramanian K, Hajamohideen F. eXplainable Artificial Intelligence in Alzheimer's disease classification: a systematic review. Cogn Comput. 2023. https://doi.org/10.1007/s12559-023-10192-x.
17. Schwalbe G, Finzel B. A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts. Data Min Knowl Disc. 2023. https://doi.org/10.1007/s10618-022-00867-8.
18. Ribeiro MT, Singh S, Guestrin C. "Why should I trust you?" Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016. p. 1135–44. https://doi.org/10.1145/2939672.2939778
19. Dwivedi R, Dave D, Naik H, Singhal S, Omer R, Patel P, Qian B, Wen Z, Shah T, Morgan G, Ranjan R. Explainable AI (XAI): core ideas, techniques, and solutions. ACM Comput Surv. 2023. https://doi.org/10.1145/3561048.
20. Bodria F, Giannotti F, Guidotti R, Naretto F, Pedreschi D, Rinzivillo S. Benchmarking and survey of explanation methods for black box models. Data Min Knowl Disc. 2023. https://doi.org/10.1007/s10618-023-00933-9.
21. Gunning D, Aha D. DARPA's Explainable Artificial Intelligence (XAI) Program. AI Mag. 2019;40(2):44–58. https://doi.org/10.1609/aimag.v40i2.2850.
22. Adadi A, Berrada M. Peeking inside the black-box: a survey on eXplainable Artificial Intelligence (XAI). IEEE Access. 2018;6:52138–60. https://doi.org/10.1109/ACCESS.2018.2870052.
23. Guidotti R, Monreale A, Ruggieri S, Pedreschi D, Turini F, Giannotti F. Local rule-based explanations of black box decision systems. arXiv:1805.10820 [Preprint]. 2018: [cs.AI]. Available from: http://arxiv.org/abs/1805.10820.
24. Ribeiro MT, Singh S, Guestrin C. Anchors: High-precision model-agnostic explanations. In: Thirty-Second AAAI Conference on

Artificial Intelligence (Vol. 32, No. 1). 2018. p. 1527–35. https://doi.org/10.1609/aaai.v32i1.11491
25. La Gatta V, Moscato V, Postiglione M, Sperlì G. CASTLE: Cluster-Aided Space Transformation for Local Explanations. Expert Syst Appl. 2021;179:115045. https://doi.org/10.1016/j.eswa.2021.115045.
26. Rjoub G, Bentahar J, Abdel Wahab O, Mizouni R, Song A, Cohen R, Otrok H, Mourad A. A survey on eXplainable Artificial Intelligence for cybersecurity. IEEE Trans Netw Serv Manage. 2023;20(4):5115–40. https://doi.org/10.1109/TNSM.2023.3282740.
27. Górriz JM, Álvarez-IllÁn I, Álvarez-Marquina A, Arco JE, Atzmueller M, Ballarini F, Barakova E, Bologna G, Bonomini P, Castellanos-Dominguez G, Castillo-Barnes D, Cho SB, Contreras R, Cuadra JM, Domínguez E, Domínguez-Mateos F, Duro RJ, Elizondo D, FernÁndez-Caballero A, Fernandez-Jover E, Formoso MA, Gallego-Molina NJ, Gamazo J, GonzÁlez JG, Garcia-Rodriguez J, Garre C, Garrigós J, Gómez-Rodellar A, Gómez-Vilda P, Graña M, Guerrero-Rodriguez B, Hendrikse SCF, Jimenez-Mesa C, Jodra-Chuan M, Julian V, Kotz G, Kutt K, Leming M, de Lope J, Macas B, Marrero-Aguiar V, Martinez JJ, Martinez-Murcia FJ, Martínez-TomÁs R, Mekyska J, Nalepa GJ, Novais P, Orellana D, Ortiz A, Palacios-Alonso D, Palma J, Pereira A, Pinacho-Davidson P, Pinninghoff MA, Ponticorvo M, Psarrou A, Ramírez J, Rincón M, Rodellar-Biarge V, Rodríguez-Rodríguez I, Roelofsma PHMP, Santos J, Salas-Gonzalez D, Salcedo-Lagos P, Segovia F, Shoeibi A, Silva M, Simic D, Suckling J, Treur J, Tsanas A, Varela R, Wang SH, Wang W, Zhang YD, Zhu H, Zhu Z, FerrÁndez-Vicente JM. Computational approaches to eXplainable Artificial Intelligence: advances in theory, applications and trends. Inf Fusion. 2023;100:101945. https://doi.org/10.1016/j.inffus.2023.101945.
28. Di Martino F, Delmastro F. Explainable AI for clinical and remote health applications: a survey on tabular and time series data. Artif Intell Rev. 2023;56(6):5261–315. https://doi.org/10.1007/s10462-022-10304-3.
29. Lamy J-B, Sekar B, Guezennec G, Bouaud J, Séroussi B. eXplainable Artificial Intelligence for breast cancer: a visual case-based reasoning approach. Artif Intell Med. 2019;94:42–53. https://doi.org/10.1016/j.artmed.2019.01.001.
30. Moscato V, Picariello A, Sperlí G. A benchmark of machine learning approaches for credit score prediction. Expert Syst Appl. 2021;165:113986. https://doi.org/10.1016/j.eswa.2020.113986.
31. Rong Y, Leemann T, Nguyen T-T, Fiedler L, Qian P, Unhelkar V, Seidel T, Kasneci G, Kasneci E. Towards human-centered explainable AI: a survey of user studies for model explanations. IEEE Trans Pattern Anal Mach Intell. 2023. https://doi.org/10.1109/TPAMI.2023.3331846.
32. Lundberg SM, Lee S-I. A unified approach to interpreting model predictions. In: Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Red Hook: Curran Associates Inc.; 2017. pp. 4768–77.
33. Jia Y, Bailey J, Ramamohanarao K, Leckie C, Ma X. Exploiting patterns to explain individual predictions. Knowl Inf Syst. 2019. https://doi.org/10.1007/s10115-019-01368-9.
34. Lakkaraju H, Bach SH, Leskovec J. Interpretable decision sets: a joint framework for description and prediction. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD'16. 2016. p. 1675–84. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2939672.2939874.
35. Guidotti R, Monreale A, Giannotti F, Pedreschi D, Ruggieri S, Turini F. Factual and counterfactual explanations for black box decision making. IEEE Intell Syst. 2019. https://doi.org/10.1109/MIS.2019.2957223.
36. Grover S, Pulice C, Simari GI, Subrahmanian VS. BEEF: Balanced English Explanations of Forecasts. IEEE Transactions on Computational Social Systems. 2019;6(2):350–64. https://doi.org/10.1109/TCSS.2019.2902490.
37. Chen L, Gao Y, Zheng B, Jensen CS, Yang H, Yang K. Pivot-based metric indexing. Proc VLDB Endow. 2017;10(10):1058–69. https://doi.org/10.14778/3115404.3115411.
38. Bahmani B, Moseley B, Vattani A, Kumar R, Vassilvitskii S. Scalable k-means++. Proc VLDB Endow. 2012;5(7):622–33. https://doi.org/10.14778/2180912.2180915.