POLITECNICO DI TORINO

DEPARTMENT OF CONTROL AND COMPUTER ENGINEERING

Master of Science in Computer Engineering

Master Degree Thesis

Explainable AI for Clustering Algorithms

Supervisors
Prof. Elena Baralis
Doctor Eliana Pastor

Candidate
Marcello Cannone

Academic Year 2019-2020

To my family

Abstract

Technological progress has brought artificial intelligence closer to people, assuming an important role in many fields thanks to its support. Artificial Intelligence, AI, is a technology capable of being transversal in many fields, from medicine to finance, from legal to security, from autonomous driving to military and so on. As AI becomes involved in context of high sensitivity and risk, the user needs more to be able to understand what the AI decision-making process suggests. The understanding, and so also the comprehensibility, of the result is closely linked to the interpretability that the model is capable of providing through its result's explanations. Although AI systems are becoming more useful providing huge benefits, their involvement is limited by the model's inability in explaining to users a given decision and action. This leads many user to consider it as untrustworthy. Today's challenge is to make AI explainable, gaining users trust and helping them to understand and manage AI outcomes. Not all AI models have this lack of interpretability, some of the simpler ones are interpretable by nature, which, although less accurate, make them preferable to user. So trust and understanding are key to a growing adoption of AI models by users. To achieve them, the generation of the upcoming AI models is also making a greater effort on interpretability, not only to understand a result but also to validate a model, finding possible issues. Inspection of internal processes of a model is not always possible, it depends on the model, which could also be a black-box one. This is a problem for the majority of the most used machine learning algorithms, for which today's effort is in developing tools and libraries to understand what is behind the model outcome, in order to provide a reasonable explanation. The explanation method approaches are divided into two type, model-dependent and model-agnostic, with the first limited on a single class of model and the second independent from the applied model. As

the explanation is the basis and essence of the interpretability of a result, it influences the name given to the topic, being known as eXplainable Artificial Intelligence, XAI.
This thesis is focused on exploring the actual available tools and solutions to provide explanations for unsupervised clustering learning. State-of-the-art explanation techniques for supervised clustering are tailored and adapted for unsupervised clustering applications. The proposed approach is model agnostic, i.e. it is applicable to explain the clustering results of any unsupervised techniques. Clustering results are firstly learned and modeled exploiting supervised techniques. State-of-the-art explainers are then applied to provide explanations. The proposed explanation approach allows the understanding of clustering results at different scopes. It provides (i) a global understanding of clustering results (ii) individual cluster interpretability, highlighting which attributes values mostly contribute to a specific cluster under analysis, and (iii) a local explanation for a single cluster instance. Experimental results on artificial and real datasets compare multiple explainers and underline which is most suitable for the scope of interpretability of interest.

Contents

List of Figures

Chapter 1

Introduction

Technological progress has brought artificial intelligence closer to people, assuming an important role in many fields thanks to its support. Artificial Intelligence, AI, is a technology capable of being transversal in many fields, from medicine to finance, from legal to security, from autonomous driving to military and so on [18]. As AI becomes involved in context of high sensitivity and risk, the user needs more to be able to understand what the AI decision-making process suggests. The understanding, and so also the comprehensibility, of the result is closely linked to the interpretability that the model is capable of providing through its result's explanations. Although AI systems are becoming more useful providing huge benefits, their involvement is limited by the model's inability in explaining to users a given decision and action [19, 20]. This leads many user to consider it as untrustworthy [2].
Today's challenge is to make AI explainable, gaining users trust and helping them to understand and manage AI outcomes. Not all AI models have this lack of interpretability, some of the simpler ones are interpretable by nature, which, although less accurate, make them preferable to user. So trust and understanding are key to a growing adoption of AI models by users [19]. To achieve them, the generation of the upcoming AI models is also making a greater effort on interpretability, not only to understand a result but also to validate a model, finding possible issues [21]. Inspection of internal processes of a model is not always possible, it depends on the model, which could also be a black-box one. This is a problem for the majority of the most used machine learning algorithms, for which today's effort is in developing tools and libraries to understand what is behind the

model outcome, in order to provide a reasonable explanation [22]. As the explanation is the basis and essence of the interpretability of a result, it influences the name given to the topic, being known as eXplainable Artificial Intelligence, XAI [18].
Also from a legal point of view there is now a higher focus on the ethical aspect of AI models [20, 22]. In fact, with the approval of General Data Protection Regulation, GDPR [23], by the European Union Parliament in April 2016, became law in May 2018, are stated clauses on automated decision-making process, which introduce a sort of "right of explanation" for all users of getting a "meaningful explanation about the logic involved", in cases where an automated decision is made. This ethical aspect of AI has coined another denomination for the XAI and is that of "Responsible artificial intelligence" [24].
This thesis is focused on exploring the actual available tools and solutions to provide explanations for unsupervised clustering learning. State-of-the-art explanation techniques for supervised clustering are tailored and adapted for unsupervised clustering applications. The proposed approach is model agnostic, i.e. it is applicable to explain the clustering results of any unsupervised techniques. Clustering results are firstly learned and modeled exploiting supervised techniques. State-of-the-art explainers are then applied to provide explanations. The proposed explanation approach allows the understanding of clustering results at different scopes. It provides (i) a global understanding of clustering results (ii) individual cluster interpretability, highlighting which attributes values mostly contribute to a specific cluster under analysis, and (iii) a local explanation for a single cluster instance. Experimental results on artificial and real datasets compare multiple explainers and underline which is most suitable for the scope of interpretability of interest.

Thesis overview The thesis is structured as follows. Chapter 2 provides a background on the XAI's terminology and concepts, in particular reports examples pointing out the increasing needs in terms of fair and interpretable outcomes resulting from artificial intelligence's and machine learning's models. The Chapter 3 consists of two main sections. The first section reports an overview on the current XAI clustering state-of-the-art, with references to novel approaches and milestones in the development of

interpretable models. The second section, instead, describes the methods that are mostly used in clustering XAI approaches, made for Machine Learning classification. These methods are also involved in the proposed experimental approach, extensively described in Chapter 4, reason thy are explained from a technical and conceptual point of view, with a distinction between global methods, in the first part, and local methods, in the second part. The Chapter 4 also describes the theory and the technical implementation of both the clustering and classification algorithms. Chapter 5 focuses on the qualitative and quantitative analysis of the experimental results obtained on both artificial and real datasets with the proposed experimental approach. Finally, in Chapter 6 conclusions are collected and possible future enhancements are illustrated to line out more suitable solutions for explainable clustering's topic.

Chapter 2

Explainable AI

The chapter consists of two sections and its purpose is to provide a contextualization of explainable artificial intelligence. The first section focuses on defining key concepts and terms, describing their relationship. The second section illustrates why explanations are becoming even more necessary, providing example for the main motivations.

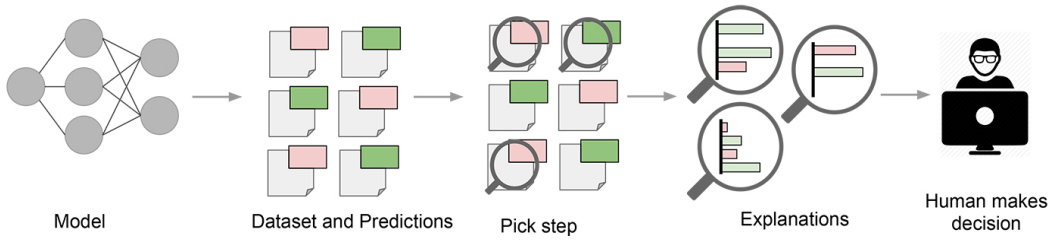
2.1 Key concepts

When we talk about explainable AI, the attention immediately falls on the term "explainable" and the first impression is the one of a AI system that gives explanations. What is meant by the term **Explanation**? [25] As defined in [26], "Explanation" is a noun that means "a statement, fact or situation that tells you why something happened; a reason given for something". So an explanation makes something clear or interpretable. Here a reflection arises on how an explanation of AI can be considered clear or interpretable. For whom does it become such? As anticipated in the introduction chapter, a limit to the adoption of AI models is above all the difficulty that humans have in understanding what, for an AI, has led to obtain a given result or to suggest a specific action [20]. Since the human is the reference for the explainable AI, we speak of **human interpretable** [27, 28]. What is interpretability? So the **interpretability** of a AI model is the ability to explain or to present the choices taken to get a result, using terms that are human understandable. An important aspect is to whom the generated explanation is interpretable, which means also

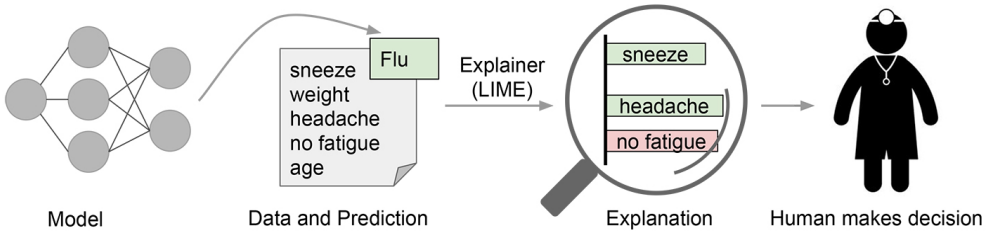
that there is not an explanation universally interpretable for all possible user [22, 27]. For example an end users could be interested to know if it is been treated fairly, if the decision is contestable or what it is necessary to improve and getting a positive decision. Another example is a technician that could be just interested to understand if the system is working properly, in order to debug it [22, 27].

2.1.1 Taxonomy of XAI

Some models by their nature have a good interpretability and are called **white** or **transparent box**, while others are called **black box** because their internal structure is almost unknown to the user or even if known it is not humanly understandable (e.g. Neural Network) [22, 25, 27]. An example of model interpretable by design is the decision tree, whose result is easy to explain by looking the decision path taken for a given instance of interest. An interesting characteristic of decision tree is that provides both global and local explanations, two key concepts, on which the models explanations usually differ. A model that is **global interpretable** has an internal logic which is clearly understandable, and allows to follow the inner reasoning that lead to all various possible results. A **local interpretable** model, instead, offers only the chance to understand a specific outcome and its reasons, being so limited to the single instance prediction [1, 3, 22].
The interpretability of a model can be also distinguishes in **intrinsic** or **post-hoc** [2, 3]. When the interpretability is given by the restriction of the internal model complexity, it is defined as intrinsic, or ante-hoc. The new upcoming generation of AI models, as anticipated in the introduction chapter, are developed with the aim of being more interpretable by design. The post-hoc interpretability, instead, is achieved applying interpretation methods to analyze a trained model, as for example the permutation feature importance method. It is also possible to apply post-hoc methods on model that are intrinsically interpretable, for example on decision trees can be applied the permutation feature importance method.
Another important distinction is on the type of interpretation methods, which can be **model-specific** or **model-agnostic** [3, 22, 25, 27]. As the name suggests, model-specific methods are limited to specific models. The model-agnostic methods, instead, are not depending from the model, can be applied on any model, after it has been trained. The independence



(a) Explaining a model to a human decision-maker



(b) Explaining individual predictions to a human decision-maker

Figure 2.1: Examples of model and individual explanation [1].

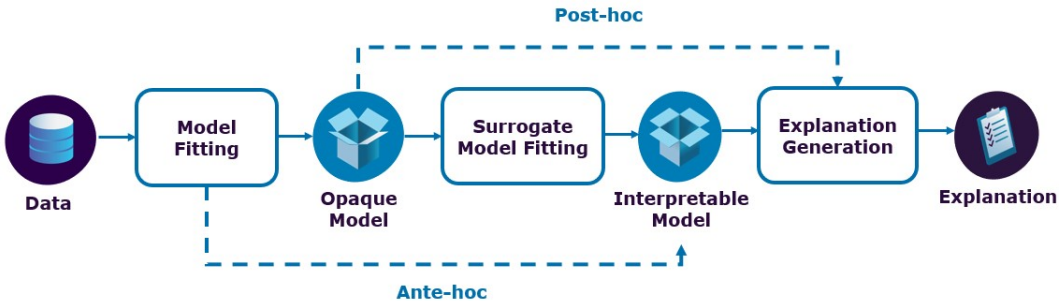


Figure 2.2: Ante-hoc and post-hoc techniques [2]

from the model does not allow so to access the internals of the model, as structural information or weights. Model-agnostic methods have desirable aspect, they are not only flexible on the model but have flexibility on the explanation and on the representation. There is not a limitation on the form of explanation, sometimes might be preferable to have a graph of feature importances while in others a linear formula. The model-agnostic methods can use different feature representation, for example with a text classifier, that is based on word embedding, can be preferable to use the word presences for the explanation [3, 25, 27]. An example of explainable AI process is shown in Figure 2.3, where starting from the real world, the

data passes through many layers, where they are filtered according to the layer, finally reaching the human with human understandable explanations [3].

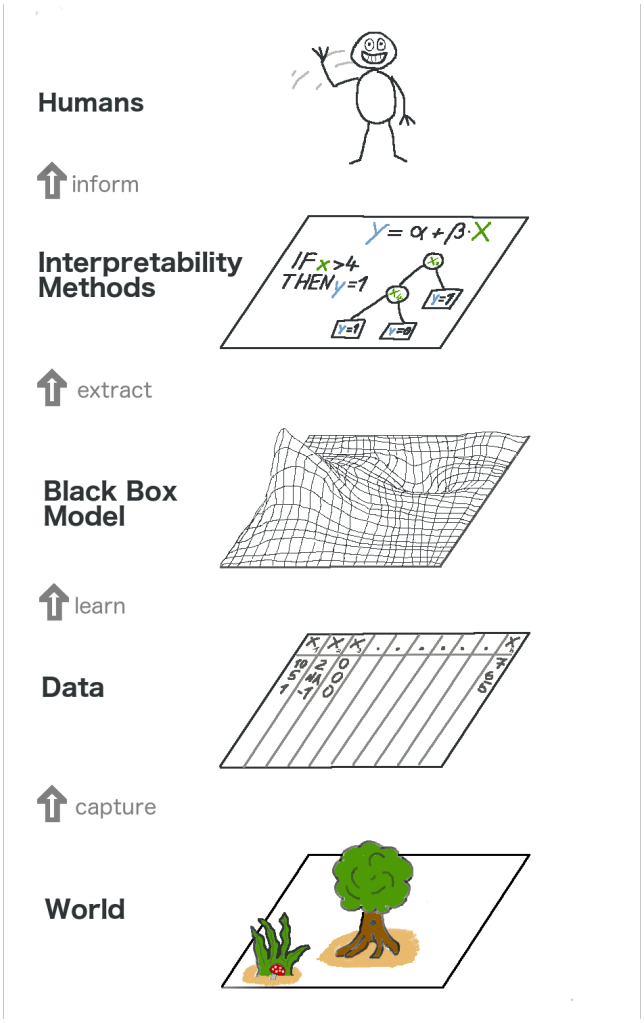


Figure 2.3: Example of explainable AI process, from the real world to the human, passing through artificial intelligence [3].

2.2 Need for explainability

Not always algorithms need to be explainable, this depends a lot on the context, if at low or high risk [2, 19, 20, 29, 30]. For example, if the Netflix algorithm suggests a wrong movie, the reason for suggesting it has no

particular importance and impact on a person's life. The algorithms that are asked to be explainable are therefore those that have an impact on people's lives, whether financial, health, social or political, so the more an AI's decision affects a person's life, the more important it is to get an explanation from the model [18, 20, 22]. The needs of explainable AI may stem from at least four reasons that highlight the different motivations for interpretability, despite they seems to be overlapping: **(i)** explain to discover, **(ii)** explain to improve, **(ii)** explain to justify, and **(iv)** explain to control [20, 21].

Explain to improve

The explanations also help to improve a model by detecting its weaknesses [20, 21]. A model able to explain its result in understandable terms is more easily improved. AI models can take up biases from the data, in some cases leading to a racists model that discriminates minorities. This was the case reported by ProPublica in their analysis [4] on the software COMPAS, Correctional Offender Management Profiling for Alternative Sanctions, a criminal risk assessment tool. The analysis founds that the COMPAS predictions were affected by a racial bias, which made them unreliable. Figure 2.4 represents one of the cases analyzed by ProPublica, that compares two drug possession arrests. An example of the racial bias in the prediction can be noted comparing the risk scores of the two candidates with also their respective prior and subsequent offences.

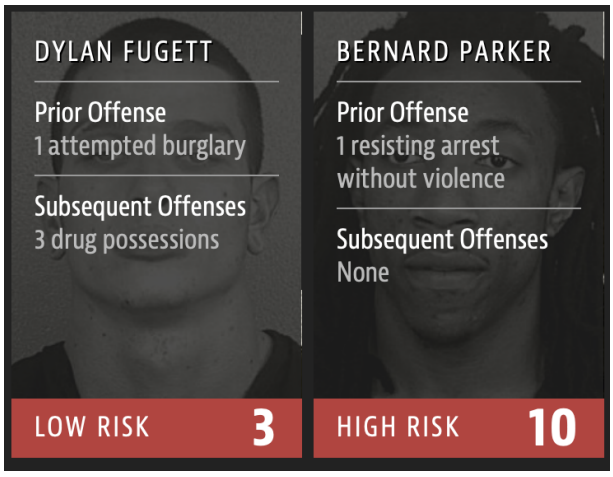


Figure 2.4: Comparison of the predicted risk scores for two drugs possession arrests [4].

Explain to justify

So in the past years there were multiple controversies about the adoption of AI system, in particular because of discriminatory or biased results [20]. This increases the needs of explaining decision, that means understanding the reason that justify the result. The explanations become even more important in case of decisions that are unexpected by the user. So AI systems can only be debugged and audited when it is possible to get an explanation to justify a behaviour, and also to get a verification of the system [20,21]. A motivating example for this aspect is the one presented in [5] where a classifier was trained to distinguish a wolf from an Husky. By looking only on accuracy score it was seems to be a good classifier with an high accuracy score, but when asked about a wrong outcome, the explanation was quite unexpected, as shown in Figure 2.5. In fact the classifier did not learn to distinguish wolf and husky but become a snow detector, because it learned to use snow as relevant feature on classifying instances as wolf one.

(a) Husky classified as wolf (b) Explanation

Figure 2.5: Original data and explanation of a bad model's prediction in the "Husky vs Wolf" task by LIME [5]

Explain to control

Explanations are also important from a legal point of view, to control the fairness of the predictions [20,21]. The European Parliament recently

adopted the General Data Protection Regulation (GDPR), which has become law in May 2018 [22, 23]. An interesting aspect is the introduction of a "right of explanation" for all individuals to obtain “meaningful explanations of the logic involved” when automated decision making takes place. Figure 2.6 shows an example of a loan requests, a scenario where an automated decision has an important impact on people’s life and needs a compliance control. A user whose request has been rejected, could ask for the reasons that lead to the denied loan. So the user by explanations could understand if the system is fair and trustworthy or also what needs to be improved to get the loan request accepted.

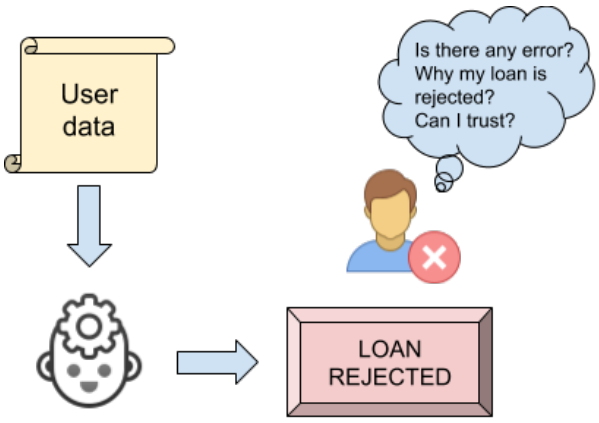


Figure 2.6: Example of need to control the explanation, scenario of loan request rejected.

Explain to discover

A system that provides explanations, especially in research, offers the chance to find new insights on the model and on why certain outcomes are suggested by the model, allowing to learn from the system. The explanations thus allow to increase the informativeness and to discover causality, thanks to the huge amount of data processed by the system [20, 21].

2.2.1 Summary of XAI desiderata

A model that is able to explain its outcomes allows to verify important aspect as [31]:

Privacy: assuring the protection of sensitive information

Reliability or Robustness: guarantee that small variations in the input do not make great differences in prediction.

Causality: verify that are considered only causal relationship

Trust: if humans are able to understand the decisions explained by the system, they can easily place more trust in it.

Fairness: ensuring that the model makes unbiased predictions, without doing any discrimination, as implicit or explicit.

Chapter 3

Related works

This chapter has the goal of providing an overview of the state-of-the-art in XAI, with a distinction on clustering and classification approaches. In the first section are described the two main philosophies of XAI approaches for clustering tasks and the current litereature's approaches.The second section is focused on the methods that are used for machine learning classification task, which are exploited in existing approaches for XAI clustering. The described method are also used in this thesis work, to understand the linkage between the XAI clustering and the existing approaches.

3.1 XAI in Clustering

In recent years, the research concerning the Explainable AI has been focused mainly in the classification's field, only a few years ago there has been an increase in clustering's studies. This delay is given by its concept, in fact the clustering is understood as intrinsically interpretable as it is suitable for grouping similar entities.
The two main lines of research are quite distinct, on one hand researchers try to generate explanations from a black-box model, while the other is focused on improving the transparency of decision-making processes thanks to more interpretable models. Before the advent of AI explainability, clustering researches were all heavily focused on improving performance metrics, such as execution time, scalability and accuracy, while leaving out any aspect of interpretability [32]. To overcome this, some studies applied soft clustering methods, while remaining far from guaranteeing solutions.

Other methods, instead, are aimed to specific application domains, on which fairness can be increased [33, 34].

3.1.1 Research philosophies

There are mainly two philosophies, which distinguish the research, the first one regards the creation of new clustering algorithms, made with explainability purposes and parameters interpretable oriented. The other philosophy has a two steps approach, that combines two models, one of which also involves a classifier (e.g from unsupervised to supervised, with a clustering method followed by a classification model, so from unsupervised to supervised, or through an explanatory framework on a trained classifier). Some studies of both philosophies have an interesting effort in generating rules and exploiting prior knowledge.

Creation of algorithms interpretable by design

Starting from research on the formulation of new algorithms, some give importance to prior-knowledge that can help to better identify some patterns. Two interesting examples are proposed by [35] and [36]. The first method, Discriminative Rectangle Mixture also called DReaM, offers a probabilistic discriminative model, which is able to learn rectangular decision rules for each cluster. The DReaM model is opposite to the model proposed by Pelleg and Moore [37], which is a probabilistic generative one and assumes tailed rectangular distributions. DReaM allows to use domain expert's knowledge, introducing informative prior distribution to the decision boundaries, that influence the generation of soft decision rectangles, the key element of this clustering methods [35]. DReaM extracts two type of rules, one related to clusters structure's preservation, which are represented as set of feature called "cluster-preserving features", instead the other set, called "rule-generating features", is used to extract the explanation by taking also in consideration the features of interest suggested by domain experts. The method proposed by S. Saisubramanian, S. Galhotra and S. Zilberstein [36] addresses the problem of extracting clusters that are interpretable, considering features that are meaningful for the end-user. The proposed algorithm has also a parameter that gives to the user the chance to ask for a certain level of explainability, that is ensured

as strong when it is equal to 1. This method measures the cluster's interpretability based on the homogeneity of the cluster's nodes, respect to the given feature of interest. To quantify the interpretability is computed an Interpretability score, that refers to a cluster respect to a feature's value and it is bounded between zero and one. The interpretability score of a cluster is given by the maximum of the feature's value scores, while the Interpretability score of clustering is the minimum cluster interpretability score respect to a feature of interest.

Two steps approaches

Some researches are focused on the generation of tree-based rules, using unsupervised decision trees and different metrics [38, 39]. The researchers of *ExKMC (Expanding Explainable k-Means Clustering)* method [40] propose a novel extension of their previous explainable k-means algorithm [41], which is based on the traditional k-means clustering, and returns a tree-based k-clustering of a dataset. The algorithm proposed in [42] is a two-step process, with a first stage for traditional clustering, to extract the class label of cluster assignments, and a second stage for tree-based supervised classification, that exploits the class labels. As the authors mentioned, the algorithm leverages the Mixed Integer Optimization techniques to generate interpretable tree-based clustering models. It provides user defined inputs to give flexibility on cluster quality measure and tree depth and complexity. The score given to the assignment considers the intra-cluster compactness of the sample and its separation from other cluster's samples. Two common measure are the Silhouette Metric [43] and the Dunn Index [44], the first compares the internal distances between samples of the same cluster with the distances from samples of the second closer cluster, while the latter considers the compactness as the biggest intra-cluster distance and the separation as minimum inter-cluster distance, so an high score means that the inter-cluster distance is proportionally high respect to the intra-cluster distance. Also interesting are similar approaches as the one proposed by E. Horel, K. Giesecke, V. Storchan and N. Chittar [45] that, after the classification stage, for each cluster runs a *Single Feature Introduction Test* method, called *SFIT*, that identifies the statistically significant features, characterizing a given cluster.

The IBM's research group in their work "Interpretable Clustering for Prototypical patient understanding" [46], defines a pipeline in which firstly is computed a similarity estimation of the patients, applying their *Locally Supervised Metric Learner (LSML)* method. The next stage employs a hierarchical clustering, generating sub-groups and learning the main feature that define the different outcomes. As described in [47] the clustering stage is at the beginning of a personalized prediction model, in fact also a single patient can be considered for patient similarity, that so identifies the K similar patient, which are used for the classification stage and which influence the subset of feature to consider.
Another interesting approach is the one proposed in [48], which is focused on learning concise rule on single instance. The SECPI algorithm, that stands for Search for Explanations for Clusters of Process Instances, takes in input the single instance to explain and returns as output a set of rule as explanation. For the classification stage, in this cited work is adopted a Support Vector Machine as a classifier. In Politecnico di Torino is developed an interesting framework for Unsupervised Network Traffic Analysis [49]. Also this approach is based on multiple steps, but it exploits the explanations given by a notorious framework called LIME [1], that provides local explanations, as can be see in the following Section 3.2.3. As [48] this work uses a Support Vector Machine for the classification step.

3.2 XAI in Machine Learning Classification

The development of XAI methods is divided into two macro areas, which differ in the granularity of the explanation, on one side there are methods focused on obtaining a global explanation, which involves the whole dataset, while on the other side there are methods which aim is to understand the explanation of a single sample. Both approaches are described in the following sections, with an overview of the available related methods. A particular mention is given to SHAP [50], a library which methods are able to extract both global and example explanations, described in the following dedicated chapter.

3.2.1 Global insight

The global insight is a first approach to broadly understand what characterized the model, bringing to light the most influential features for that model. This information is usually obtained by investigating the trained model [22].

Random Forest's feature importance

For Random Forest model, the scikit-learn's [51] implementation allows to extract the feature importance by an inner function that returns the features with their importance score respect the whole dataset, so it represents a score of global importance. As stated by scikit-learn developers, their method is impurity-based, and computes the importance score as the normalized total reduction of the criterion brought by that feature. It is also known as the Gini importance. Although this computation is fast, it can be misleading for continuous features or high-cardinality features, so with many unique values. A proper alternative is the Permutation Importance, which is suggested [51] and compared [52] by scikit-learn.

Permutation feature importance

Permutation feature importance is a technique that measures the increase in the prediction error of the model after a feature's value permutation. In that way, the relationship between the feature and the true outcome is broken. A feature is considered important when shuffling its values the model error increases, that means the model relies on that feature for the prediction. Otherwise when the model error is unchanged, the feature is not considered as important because it is ignored by the model for the prediction.
This kind of measurement was introduced by Breiman [53] for random forests, and starting form this idea was proposed a model-agnostic version called model reliance [54]. Permutation importance method has advantages, one is its highly compressed global insight of the model's behavior, another advantage is that provides measurements that are comparable across different problem thanks to the usage of error ratio. The feature permutation allows to save time because it does not require to retrain the model, considering so a fixed model. All the feature interaction are taken into account by the importance measure, in fact the feature's permutation

destroys the interaction effect with the other features, which also represents a disadvantages from a feature's dependencies point of view. The method is applicable to any model and is a reliable technique. An unclear aspect is whether compute the feature importance on training or test data. Being linked to the error of the model, in some cases is a disadvantage because it is not clear on the robustness of the model's output when features are tweaked and how much of the model's output variance is explained by each feature. The randomness given by the feature's shuffling allows repeated permutation which results might vary greatly, stabilizing the measure thanks to the averaging of the results but requiring more time for the computation. As anticipated, feature dependencies can be broken by permutation that may create unrealistic data instances that are considered for the importance measure computation (i.e. A 2 meter tall person weighting 30 kg). Another critic aspect is that adding a correlated feature can decrease the importance of the associated feature by splitting the importance between both features.

3.2.2 Local insight

Sometimes explanations are requested to better understand a specific instance o subgroup of instances, so an explanation focused on the instance's locality is necessary. If global explanations try to answer questions like "How does the trained model make predictions?", the local explanations, instead, try to answer questions like "Why did the model make a certain prediction for an instance or group of instances?" [3]. The main, and also the first, research for local explanation is done with LIME [5]. A novel interesting approach is LORE [55], which is a rule-based explanations approach.

3.2.3 LIME

One of the first approach created with a focus on the machine learning interpretability is LIME, acronym that stands for Local Interpretable Model-Agnostic Explanations [5].

Intuition behind LIME

LIME's authors Ribeiro, Singh and Guestrin proposed it as a model-agnostic method, so with the purpose of explaining predictions of any classifier [5]. The method emphasizes the local analysis for the single explanation, differently from previous methods which were looking for a global explanation. As the authors state, to be model-agnostic, they learn the behaviour of the underlying model perturbing the input and observing how the prediction changes [1]. LIME is interesting because it merges

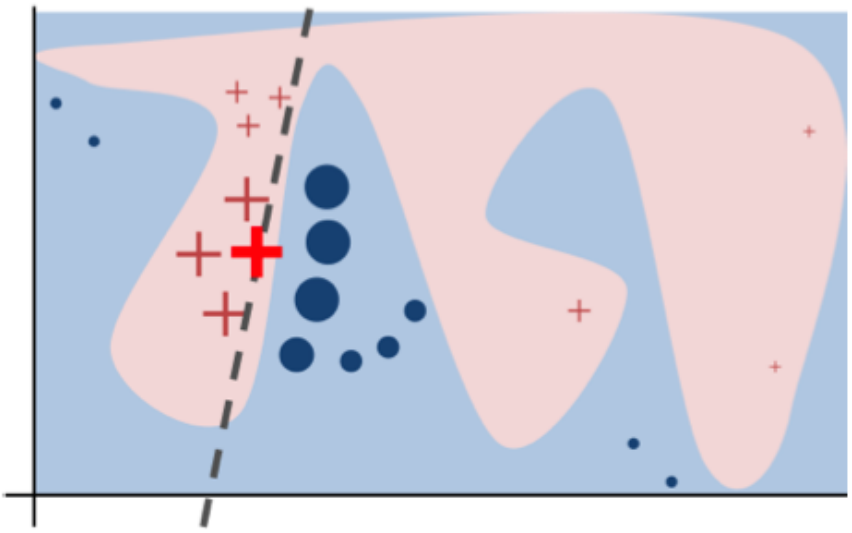


Figure 3.1: Visual presentation of LIME's intuition on global and local explanations [5].

two aspects, **(i)** a simple model is more interpretable and **(ii)** a complex model can be approximated locally with a simpler model. Figure 3.1 is a visual presentation of LIME's intuition. The blue and pink background represents the complex decision function given by the black-box model. The dashed line is the local explanation, which is the linear model approximation for the bold red cross, the instance to be explained [5]. Observing Figure 3.1, we can feel how it is difficult to explain a single prediction of interest with the global context. LIME focuses on the local of the prediction to be explained. A simpler linear model is used to approximate the local behaviour of the complex model, learning an interpretable model only on that region of interest. The local linear model is learned on the

perturbations of the original instance of interest, which are weighted by their similarity respect to the instance to explain. The impact of the locality is reflected only on local and not global behavior, therefore what is extracted for the region of interest may not be reflected in the context of the global model [5].

How is the explanation computed?

The computation is based on the following building blocks [5]:

g **:** an explanation is defined as a model $g \in G$, one of the potential interpretable models of the class G. That class is composed of models such as linear one, decision trees, or falling rule lists. The model domain is $\{0,1\}^{\mathrm{d}'}$, i.e. the model g acts over the absence or presence of the interpretable features.

$\Omega(g)$ **:** is a measure of complexity of the explanation $g \in G$, it is the opposite of interpretability and it is defined because not every model may be clearly interpretable. For example, as proposed by its authors, a decision trees the measure $\Omega(g)$ may be the depth of the tree, instead for linear models it may refer to the number of non-zero weights.

x **:** it is the original representation on d feature, $x \in R^{\mathrm{d}}$, of an instance being explained

f **:** it represents the model being explained, which is denoted $f : R^{\mathrm{d}} \rightarrow R$, while $f(x)$ is the probability that a given instance x belongs to a certain class.

$\pi_{\mathbf{x}}(z)$ **:** is a proximity measure of x with an instance z, defining so the locality of x.

$\mathcal{L}(f, g, \pi_{\mathbf{x}})$ **:** it is a measure of how wrongly g is approximating f in the π_{x} locality.

To achieve a good local fidelity-interpretability trade-off it must be minimized the $\mathcal{L}(f, g, \pi_{\mathrm{x}})$ while $\Omega(g)$ needs to be low with a tolerance given by the human's interpretability [5]. For these reasons the explanation given by LIME is obtained by the following formula, that can consider different explanation families G, fidelity functions $\mathcal{L}$, and complexity measures Ω [5]:

$$\xi(x) =_{g \in G} \ \mathcal{L}(f, g, \pi_{\mathrm{x}}) \ + \ \Omega(g) \tag{3.1}$$

LIME provides explanations depending on the type of data used, like Tabular data, text data or image. Figure 3.2 shows an example of LIME explanations on image, in particular is explaining an image classification prediction made by Google's Inception neural network. The classes with highest prediction score are "Electric Guitar" (p = 0.32), "Acoustic guitar" (p = 0.24) and "Labrador" (p = 0.21). Considering now the benefits

(a) Original Image

(b) Explaining *Electric guitar*

(c) Explaining *Acoustic guitar*

(d) Explaining *Labrador*

Figure 3.2: LIME's image explanations [5].

of LIME, it offers an intuitive explanation based on weighted sum computation of the prediction, it is also simple to compute, in fact requires short time of computation [3]. The disadvantages are mostly linked to the local linear approximation, that can only represent linear relationships. So with a lack of predictive performance because the model can only learn

restricted and oversimplified relationships.

3.2.4 SHAP

SHAP is the acronym of SHapley Additive exPlanations, as the name suggests, it is a method which takes inspiration by the optimal Shapley values from game theory [50], but with a kernel-based estimation approach, which is also reflected on the name of the proposed estimator *KernelSHAP* [3]. SHAP's aim is to explain a single prediction, computing the contribution of each feature contribution in the resulting prediction. The way of extracting the explanation via single contribution is taken from the coalitional game theory, so SHAP considers a feature value as a player in a coalition [50]. The Shapley values point out how distinguish the feature values impact on a prediction, among the involved features. The player involved depends on the nature of data, e.g. for tabular data it can be a single feature value, instead for an image, the player is a group of feature values, the pixels that are grouped to significant super pixels. SHAP method is used taking two optional assumptions that simplifies the computation of the expected value, they are feature independence and model linearity, so *KernelSHAP* merges two philosophy, the *Linear LIME* and the Shapley values [50]. SHAP is based also on additive feature attribution methods, that have an explanation model that is a linear function of binary variables [50]:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i \tag{3.2}$$

where M is the number of simplified input features, $\phi_i \in \mathcal{R}$ and $z' \in \{0,1\}^M$. The explanation model that follows the (3.2) assigns to each feature an effect ϕ_i and then sums the effects of all feature attributions approximates the original model output $f(x)$, where f is the original prediction model to be explained and g the explanation model, x is the single input [3]. The shap values guarantee three important properties like local accuracy, missingness and consistency [50]. For the first property, when the original model f is approximated for a given instance x, local accuracy imposes a match with the explanation model output of f for the simplified input x', that corresponds to the original instance x.

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i \tag{3.3}$$

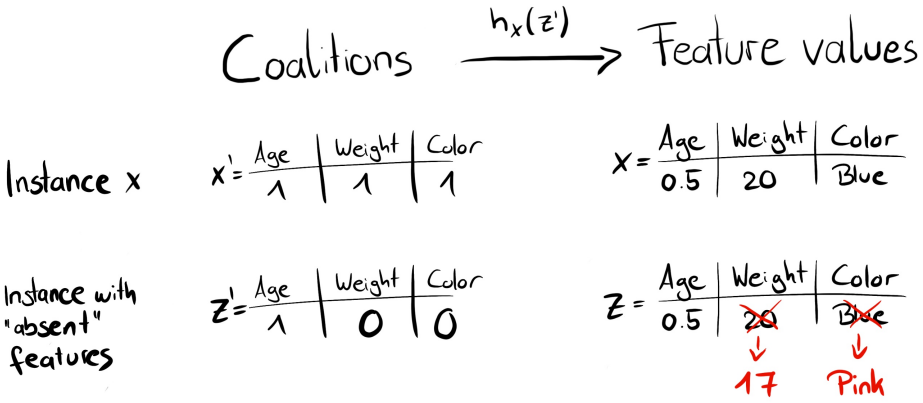
The explanation model matches the original model when $x = h_x(x')$ and where $\phi_0 = f(h_x(0))$, which means that all simplified inputs are toggled off. The missingness property can be considered as the dual of the simplified input, so instead of taking into account the impact for a feature presence, considers a missing feature as one that does not provide any impact [50].

$$x'_i = 0 \implies \phi_i = 0 \tag{3.4}$$

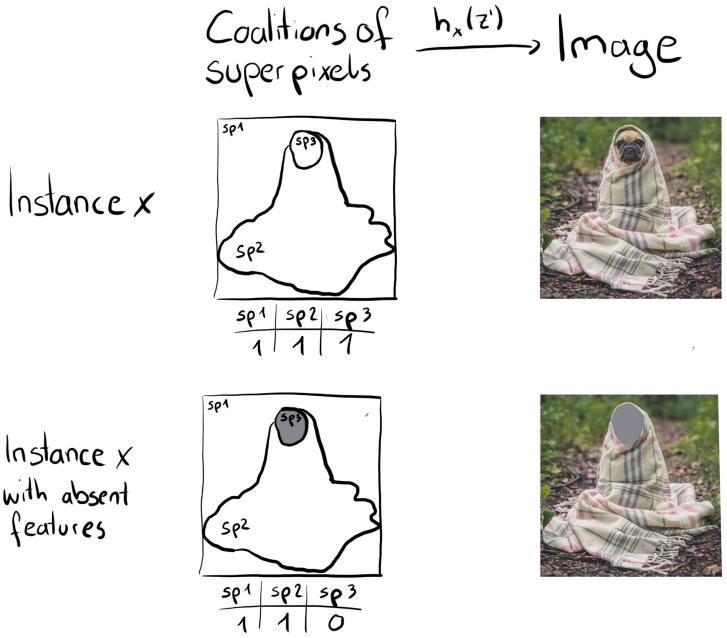
Consistency property states that the attribution of a given instance, independently of other instances, should not decrease when the model changes in a way that the impact of the simplified input increases or stays the same [50].

$$\phi_i(f, x) = \sum_{z' \subseteq x'}^{M} \frac{|z'|!(M - |z'| - 1)!}{M!}[f_x(z') - f_x(z' \backslash i)] \tag{3.5}$$

These properties are encountered in the definition of ϕ 3.5, where $|z'|$ is the number of non-zero features in z', with $z' \subseteq x'$ represents all z' vectors where the non-zero features are a subset from the non-zero features in x [50]. Figure 3.3 shows two example of simplified instances. In Figure 3.3a the simplified instance is computed on a tabular data, so the coalition represent the presence of the features [3]. In the example only the value of the feature Age is maintained, while for the features that are not in the coalition, the feature value is replaced by random feature value from data. Figure 3.3b is the equivalent in case of images, where the feature is intended as a super pixels and whose absence is represented with a grey area [3]. The SHAP authors propose explainers with different approaches, the main are *BruteForceExplainer*, *KernelExplainer*, *SamplingExplainer* and *TreeExplainer*, *DeepExplainer* [6,50,56]. The main functions for these explainers are *shap_values* and *explain*. As the name suggests *shap_values* is in charge of estimating the SHAP values for the passed dataset. For models with a single output is returned a matrix of SHAP values of SxF dimension, where S is the number of samples and F is the number of features. Each sample is represented with a row that sums to the difference between the expected value of the model output and the one observed by the model for that sample. In case of models with a vector of output, the function returns a list of that per-output matrices. The explainers *BruteForceExplainer*, *KernelExplainer*, *SamplingExplainer* share the same *shap_values* function, the one defined for *KernelExplainer*, but have a



(a) Tabular data



(b) Image

Figure 3.3: Examples of simplified input [3].

different philosophy on the computation of samples behind the explain function. The *DeepExplainer* is a high-speed approximation algorithm for SHAP values computed for deep learning models.

BruteForceExplainer

This explainer is a brute force implementation of SHAP values intended

for unit tests, because of its high computation costs in terms of time and resources [6]. The *explain* method defined for this approach, as for others, works on a single instance to explain. Given that instance, given the model used for the prediction, is built up a section for the estimation of the ϕ values for each possible feature combination. For each feature in analysis are computed the combination taking only in consideration the other features. The intuition behind this is that it is possible to learn most on a single feature if it is possible to understand its effect in isolation [3]. A coalition of a single feature allows to understand the feature isolated main effect on the prediction, instead when the coalition consists of all but one feature, it is possible to understand about the features total effect, compounding the main effect with the feature interactions. If are considered only half of the features, the coalition provides little understanding about individual feature contribution, because of the high number of coalition made with half of the features [6]. Then the mask is built up for the prediction without the feature of interest, starting from a mask of zeros and taking only the value for the feature in a given combination. The masker function is in charge of changing the values of the whole dataset for the present feature, in this manner for all the instances the present feature value is the same. The masked dataset and the original one are passed to the classification model, which returns back the predictions. The weighted difference between the predictions made on the original dataset and those of the masked dataset, define the ϕ value for the considered combination and i-th feature. So the ϕ value for the i-th feature is an incremental update from the weighted difference between $i_{present}$ and i_{absent} results, over all possible presence combination made considering all but one feature.

KernelExplainer

KernelExplainer uses a special weighted linear regression to compute the importance of each feature, which are shapley values from game theory and coefficients from the local linear regression [57]. From some points of view this explainer can be seen as the fusion of linear LIME with Shapley values. LIME's equation 3.1 allows to recover Shap values, but differently from linear LIME, the parameters can not be chosen heuristically. Equation 3.6 shows how to find the proper loss function L, weighting kernel $\pi_{x'}$, and regularization term Ω to recover the Shapley values, according to the

definition of additive feature attribution methods 3.2 [50].

$$\begin{aligned} \Omega(g) &= 0 \\ \pi_{x'} &= \frac{(M-1)}{(M\text{choose}|z'|)|z'|(M-|z'|)} \\ \mathcal{L}(f,g,\pi_{\text{x'}}) &= \sum_{z'\in Z}[f(h_x(z')) - g(z')]^2\ \pi'_x(z') \end{aligned} \tag{3.6}$$

where $|z'|$ is the number of non-zero elements in z'.

SamplingExplainer

SamplingExplainer computes SHAP values under the assumption of feature independence and extends the algorithm proposed in [58]. Even if it is based on a *KernelExplainer* class, it is an alternative especially for cases with a large dataset as background set, instead of a single instance. Differently from the *KernelExplainer* the passed data can be the whole training dataset, even if a high dimensional one, and this is possible thanks to the sampling on the dataset done by the *SamplingExplainer*. The explain method allows the user to define the number of samples to generate, a way to speed up the method.

TreeExplainer

As stated by SHAP's authors in their paper [56], despite the compelling theoretical advantages of shap values, their practical use is hindered by two problems: the challenge of estimating efficiently the shap value and the exponential complexity of the Equation 3.5. *TreeExplainer*, also called *TreeShap*, is focused on tree models and proposes fast SHAP value estimation methods for trees and ensembles of trees, this is reason why it has its own definition for *shap_values*. Starting from a definition of a slow but straightforward algorithm the *TreeShap* one is a more complex and faster version, that allows to a memory usage of $O(D^2 + M)$ and an estimation time of $O(TLD^2)$ instead of $O(TL2^M)$, where T is the number of trees, L is the maximum number of leaves in any tree, M is the number of features and D is the depth of the trees, that becomes $D = \log L$ for balanced trees [56]. Since the weights in the equation depend on |z'|, to get a proper weighting of subsets, the algorithm keeps track of each possible subset size during the recursion. The *TreeShap* method has three main functions *recurse*, *extend* and *unwind*. The last two are in charge

of creating or removing the extension of the path in exam. The *extend* method grows the set of subsets, considering a given mask of ones and zeros, so this method is used to descend the tree whereas the *unwind* is used to undo the previous when the current split is on the same feature twice, and to undo the extension of the path inside a leaf, to compute weights for all the features involved in the path [56].

SHAP explanation visualization

SHAP offers the methods *Summary_Plot*, *Dependence_Plot*, *Decision_Plot*, *Force_Plot*, which are described below [6]. The summary plot provides a global interpretation, not only for the model but also for the single classes, it distinguish each feature's impact for each class. The dependence plot is a scatter plot which shows the effect of a single feature on the model's predictions. The decision plots show how complex models arrive at their predictions. The force plot gives a visual representation of the impact of each feature for a group of explanations or for a single one, its equivalent for image data is *Image_Plot*. Figure 3.4 shows an example of the *Image_Plot* applied on a MNIST data. It explains, for each of the four different images, the output for each possible result, which are the digits from 0 to 9. The pixel in red are the one that drive to a given output by increasing its score, on the other hand the blue pixels decrease the outcome score. The images on the left are the input, while the images in a nearly transparent grayscale, provide a visual match with explainations. It is interesting to observe how for the zero is important the blank in the middle, while for the four is important the absence of a line on the top to do not consider it as nine instead of four.

3.3 XAI tool development rush

The growing interest on artificial intelligence fairness has taken also the interest of important IT companies, that now are more involved in new accountable, fair and transparent tool development. One of the most recent tool is *WIT (What-If Tool)* [59] by Google, which offers the chance of testing performance in hypothetical situations, analyzing the importance of different data features, visualizing the given scenario across multiple models and subsets of data, considering different fairness metrics [59]. Microsoft is releasing the alpha of its open-source package, *InterpretML* [60],

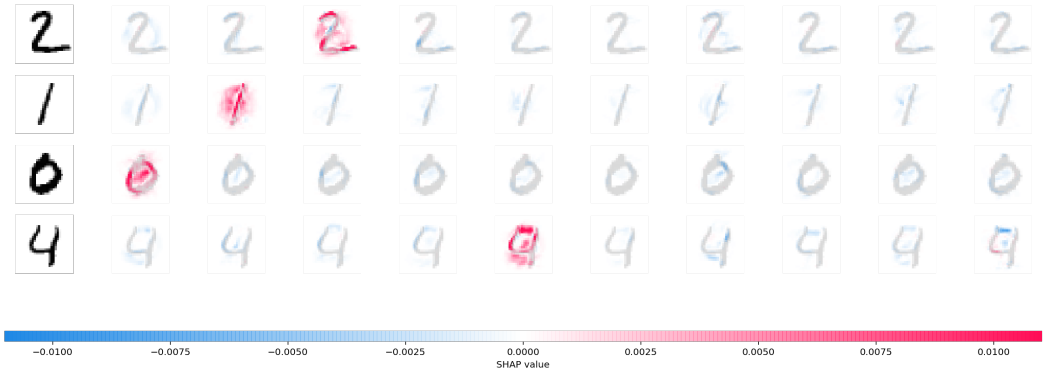


Figure 3.4: Example of feature attributions obtained with image_plot on MNIST data [6].

that incorporates state-of-the-art machine learning interpretability techniques under one roof. It also allows to train interpretable glass-box models and explain black-box systems [60]. Like Microsoft, H2O has also focused on interpretability in its closed-source platform called *H2O Driverless AI* [61]. Finally, IBM has released an open-source tool, *AI Explainability 360*, that implements the state-of-the-art algorithms for interpretability and explainability of machine learning models [62].

Chapter 4

Explanation extraction's approach

This chapter introduce the thesis work approach, with a focus on the theoretical background of the algorithms used for clustering and classification, explaining the possible configurations.

4.1 Proposed experimental approach

The approaches for explainable clustering, as already said in the Section 3, offer a limited set of possibilities. The clustering techniques are often used for exploration phases, thanks to their ability in identifying groups of subjects, a peculiarity on which the proposed approach is based. The proposed approach belongs to the two phases approaches philosophy described in Section 3. As the aim is to address the extraction of explanation for unsupervised learning problems, by clustering, the input dataset is assumed to be suitable for the clustering. This assumption is important to grant the applicability of the proposed approach. Moreover the dataset is pre-processed, removing empty or null values. Once the dataset is ready, the approach can begin, following the schema shown in Figure 4.1, which steps are described next. The first phase has the goal of labeling the dataset instances, by clustering algorithm. Depending on the considered algorithm, the number of clusters can be required as parameter. If it is not known from the given problem, it is tuned over a range of

possible numbers of cluster, by comparing the Silhouette scores achieved by the different cluster numbers. The chosen number of cluster is the one with the highest Silhouette score. The labels extracted by the clustering algorithm of the first phase are used in the second phase. The main actors of the second phase are the classifier and the explainer. Firstly a classifier is trained on the input dataset, considering the labels extracted in the first phase. After that is possible to use an explainer, by passing the dataset and the classifier model. This approach is suitable to both the used families of interpretability methods, LIME and SHAP. At this point by using SHAP's methods it is possible to visualize global explanations about the processed dataset, with a well defined contribute of each cluster on the model explanation.
When it is request to process a single instance, it is passed to the explainer together with the classifier model.

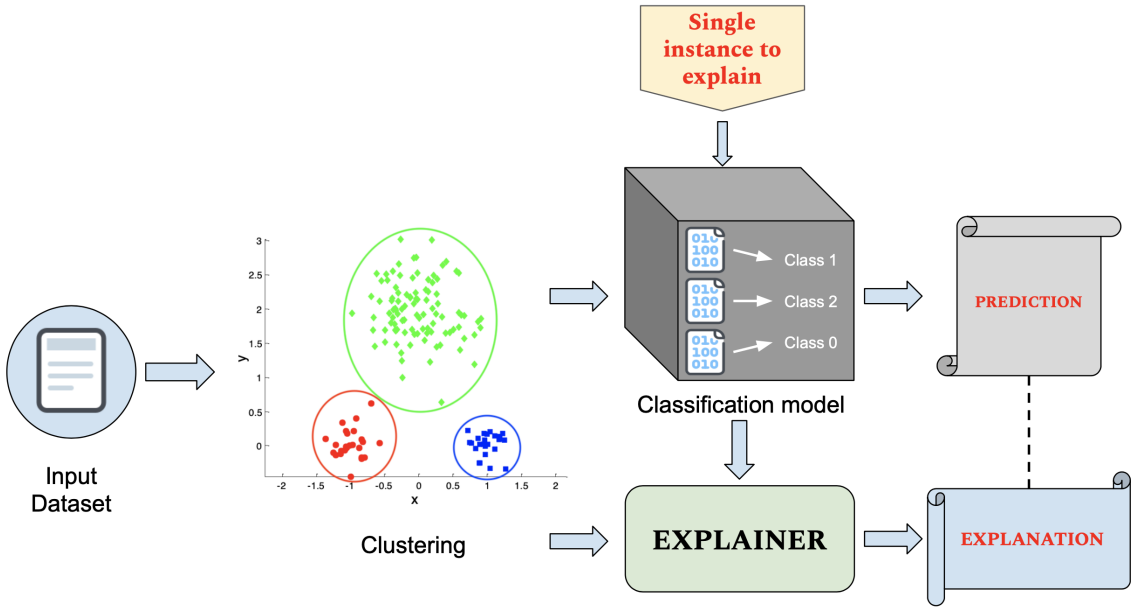


Figure 4.1: Conceptual scheme of the proposed approach

4.1.1 Algorithms lineup

To obtain a wide variety of possible cases, for each analyzed data set, the clusters are extracted and classified considering singularly all the possible algorithm's combinations, both for clustering and classification. Also for

the explanations are exploited all the explainable functions presented in Section 3.2.3 and 3.2.4. It is so necessary to get a background knowledge of the used algorithms, also for clustering and classification, on which are focused the next sections.

4.2 Clustering

This section explains the chosen clustering algorithm and the given configuration for the experiments with artificial data set and real data set. The clustering phase is done using the scikit-learn implementation of the following algorithms: KMeans, DBSCAN, OPTICS, Spectral and Agglomerative [7, 63].

4.2.1 KMeans

It is an iterative algorithm which identifies K subgroups, not overlapped, called clusters, where K is specified by the user. Each cluster is composed of data points that are as much as possible similar between them [64].

How does it work? [7, 63]

1. First of all is specified the K number of clusters.
2. The algorithm initialize K-centroids, it can be done passing an ndarray of shape (K, number of features) or can be randomly defined by the algorithm that choose K random samples from the data set.
3. After that the K-means starts to loop the following two steps:
   (a) For each sample computes the distance from all the centroids and assign the sample to the nearest one.
   (b) New centroids are created considering the new mean value given by the data points belonging to the appropriated centroid.

The loop stops when the difference between the old centroids and the new one is under a certain threshold. The algorithm choose centroids that minimize the within-cluster sum of squares, where x_i is the sample and the μ_j is the j-th centroid, so the mean of the j-th cluster.

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2)$$

KMeans has some limitations. It has problem when data contains outliers, and also when it is composed by clusters that are different in density, size or when their shape is non globular. As shown in Figure 4.2, respect to

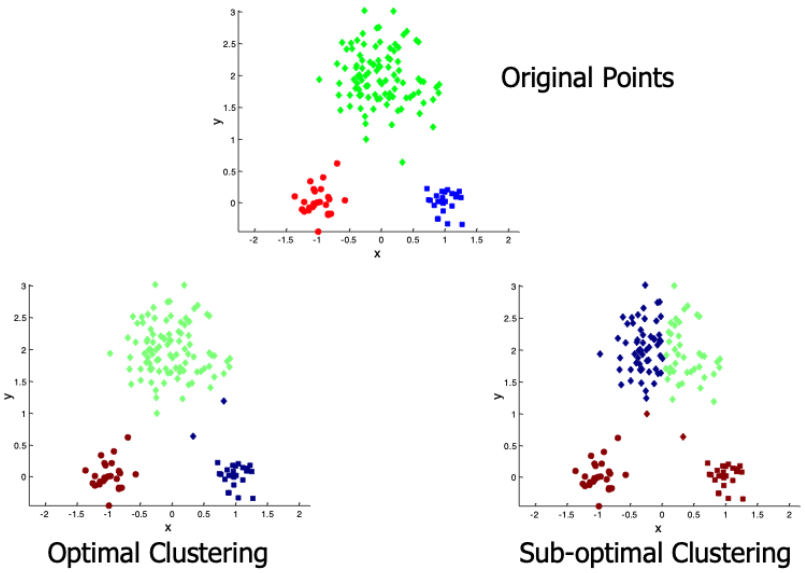


Figure 4.2: Importance of centroids initialization [7]

the original point, the KMeans algorithm could extract optimal or not optimal clusters. In both the cases are detected three clusters from the data, but they are different because of the initialization of centroids. A wrong choice of initial centroids cause a sub-optimal clustering, as shown in Figure 4.2, where the big cluster in the middle is divided in two clusters and the two small clusters on the sides are considered as a unique one.

4.2.2 DBSCAN

DBSCAN, which is the acronym for Density Based Spatial Clustering of Application with Noise, is one of the most famous unsupervised learning method [8]. In fact, differently from the previous algorithm, it does not requires a number of cluster to find, which instead are considered as high density areas separated from low density ones. For that reason DBSCAN can identify clusters of any shape, differently from the K-Means that assumes clusters have a convex shape [63]. The concept of density is given by the two parameters *min_samples* and *eps*, the first one defines the

number of neighborhood to consider a point as a core one, so it defines the tolerance of the algorithm towards noise, the second one defines the core distance measure, the maximum distance between samples to be considered neighbors. The default distance metric used by this algorithm is the euclidean one. The *eps* definition is particularly important because it states the local neighborhood of the data, and should be choose coherently with the data set values. A small *eps* will consider most of the data as not clustered samples, labeled as noise, on the other hand a large *eps* value will cause the collapse of close clusters in a single one [63]. A cluster is composed of a set of core samples and a set of non-core, or border, samples, the first are samples, from the dataset, that have *min_samples* other samples within an *eps* distance, the seconds are neighbors samples of a core one but are not themselves a core sample. The non-core samples can be formally considered as the edge of the cluster, out of it there are outliers, samples that have a distance of at least *eps* from any other sample. Figure 4.3 provides an example of DSCAN application, considering *min_samples* equal to 4. The core point is represented with red dot, while the blue dot are used to identify the noise point, that dos not belong to any cluster. The yellow dot represents the border point, that is still part of the cluster thanks to a distance within *eps* respect to a core point, but it can not be consider as a core point because it does not satisfies the *min_samples* criteria [8].

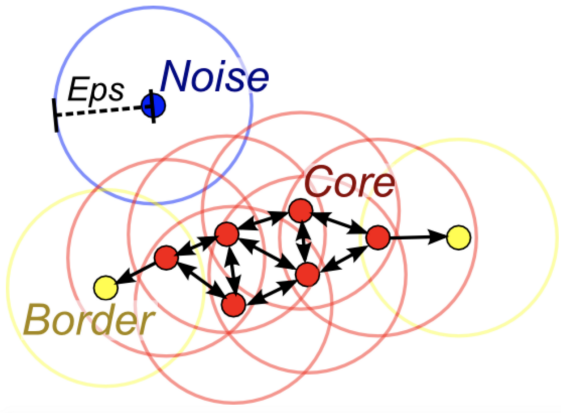


Figure 4.3: DBSCAN example [8]

4.2.3 OPTICS

This algorithm takes inspiration from the DBSCAN, of which can be considered a generalization [65]. The main parameters of the algorithm are *min_samples* and *max_eps*, the first is the same of the DBSCAN, the second one, instead, represents the maximum distance between two samples to be considered neighbors, differently from the DBSCAN it has infinite as default value, which means it identifies clusters across all scales, this also means that requires more memory and computational power. The most important difference is that the OPTICS extract a reachability graph, which gives to each sample a reachability distance, that enables this algorithm to identify clusters with variable density [63]. The reachability distance is computed for a sample respect to a core sample and returns the smallest distance from the core sample, so it cannot be smaller than the core distance of that core sample. A core distance is the minimum *eps* to consider a point as a core one, satisfying the *min_samples* criteria. The default distance metric used by this algorithm is the minkowski one [65]. Figure 4.4 shows an example that visualize the already described concepts of core-distance and reachability distance. The example underlined the different reachability distance for the point p respect to $q1$ and $q2$. The reachability distance respect to $q1$ is influenced from the smaller core distance of p, which is ϵ', instead the reachability distance respect to $q2$ is equal to the distance from p.

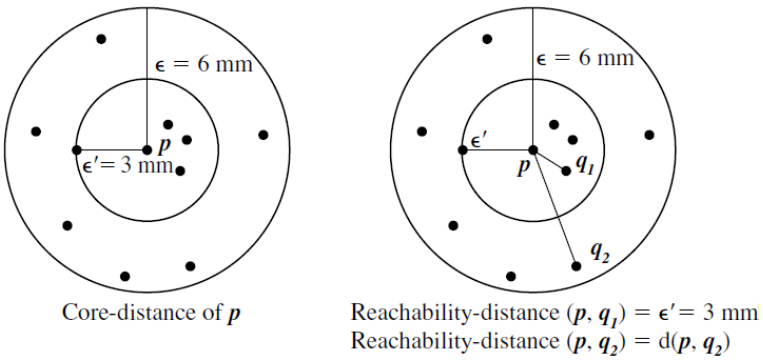


Figure 4.4: OPTICS example [9]

4.2.4 Spectral

This algorithm has become popular thanks to its easy configuration and interesting performance with graph-based clustering [66]. The algorithm was built to address the complex cluster shape problem, which is an issue for the KMeans and DBSCAN application, for the first one because of its ability to find spherical clusters and for the second one because of its parameter sensitivity. It is based on 3 phases [66]:

1. First of all it is necessary to build up a similarity graph between the samples. The similarity graph can be computed in many ways, the most used are the *k-nearest neighbors* and the *eps-neighborhood*, which respectively use a K numbers of samples and a sample's distance eps to get the local structure of the data.
2. Then are calculated the first K eigenvectors of its Laplacian matrix. The eigenvectors are used to compute the sample's feature vectors.
3. On that feature is executed a K-Means algorithm, to identify the K clusters.

A similarity graph represents the dataset as a weighted graph, where all vertices that can be reached from each other by a path form a connected component. When there is only one connected component in the graph, it is defined fully connected [10]. Figure 4.5 is an example of similarity graph build on 6 vertices, that represent the data points, composed of 8 weighted edges showing pairwise similarity between points. The two main

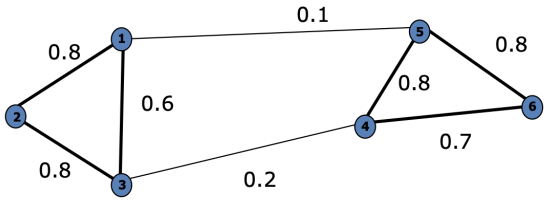


Figure 4.5: Example of similarity graph [10]

type of graph are *k-nearest neighbors* and *eps-neighborhood*. The *eps-neighborhood* computes pairwise distance between any couple of points, and connects each point to all other points having a distance smaller than a threshold *eps*. This approach can be unweighted and so there is an edge if a point belongs to the eps–neighborhood of another point, instead

when the graph is weighted, the distance is transformed to similarity, that is considered as edge weight [10]. The *k-nearest neighbors* approach defines edges between a point and its k-nearest neighbors, each point is connected to at least k points. The kNN graph considers a single edge between to points (e.g. considering x_i and x_j, there is an edge that can be from x_i to x_j or from x_j to x_i), while the MutualkNN considers the edge set as a subset of that in the kNN graph, then edges are taken for both the directions [10]. An example of the described approaches is shown in Figure 4.6, where for a given dataset are computed similarity graph considering different approaches. On the top right corner it is used an *eps-neighborhood* approach, with an epsilon equal to 0,3, instead the two approaches on the bottom of image are based on the two approaches of *k-nearest neighbors* [10].

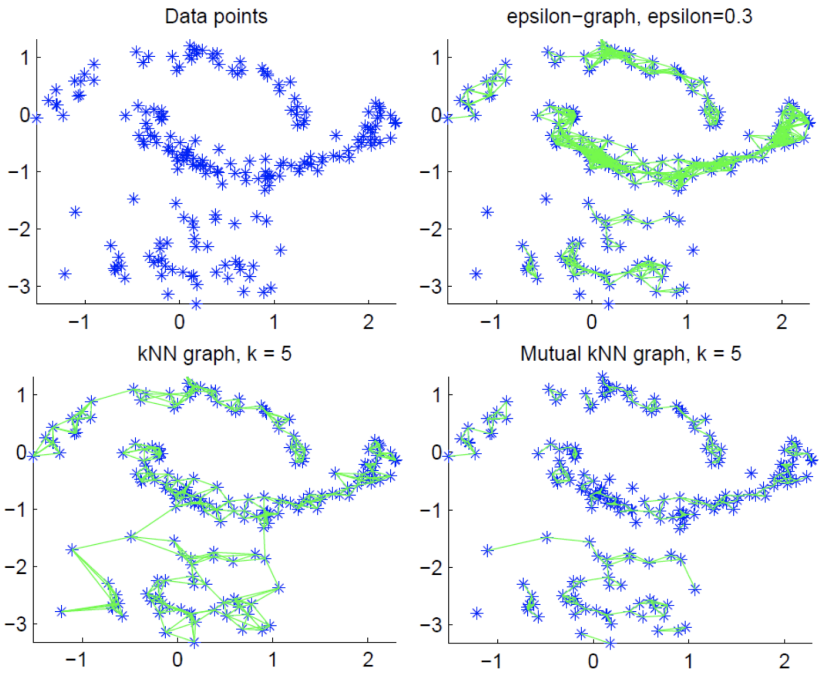


Figure 4.6: Different similarity graphs [10]

4.2.5 Agglomerative

This algorithm is one of the most used of the hierarchical clustering methods [63]. The clusters obtained with this family of clustering algorithms are build by merging or splitting them afterwards. The results can be displayed with a dendrogram or a tree, which root represents the unique clusters, that collects the whole dataset, and the leaves represent the standalone sample's cluster. For these algorithms is not necessary to assume a given number of cluster, because it can be obtained by cutting the dendrogram at a suitable threshold. The Agglomerative clustering algorithm has a bottom-up approach [63]:

- At the beginning, each sample is considered as a single cluster
- At each step, the two closest clusters are merged and the distance matrix is updated. It stops when remains a number of clusters equal to 1 or K, if given.

The merging strategy is determined by the linkage criteria [7, 63]:

MAX or Complete : minimizes the maximum distance between observations of two clusters.

MIN or Single : minimizes the distance between the closest samples of two clusters.

Average : minimizes the average of the distances between all samples of two clusters.

Ward : minimizes the sum of squared differences within all clusters.

Centroid distance : The distance between two clusters is computed considering their centroid, which are the center of the cluster.

An example of how the linkage strategy influences the cluster detection is provided in Figure 4.7, where for a given set of points are shown the result by using as linkage the min, the max, the average and the Ward.

4.3 Classification

The obtained cluster labels and the related data sets are classified with the following classification methods: Random Forest, Decision Tree, SVM, K-Nearest Neighbors, Neural Network Multi Layer Perceptron.

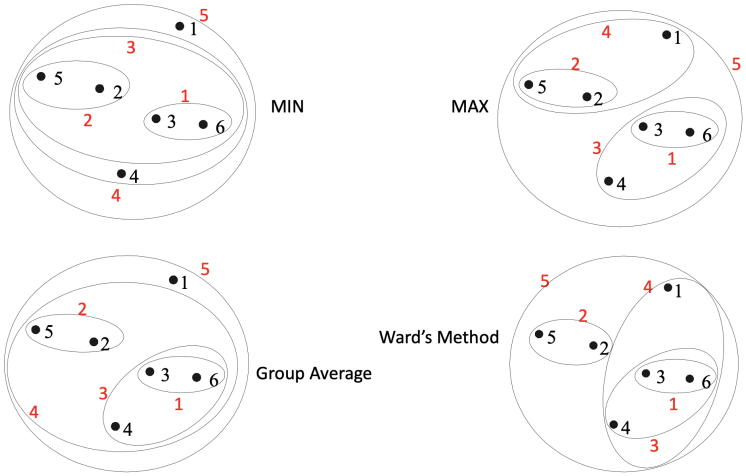


Figure 4.7: Different linkage strategies [7].

4.3.1 Decision Tree

Decision Tree is a supervised learning method with the aim of defining a model by learning decision rules from the dataset [11]. The decision path is made step by step, for each one the algorithm defines the best feature to split on, rewarding the feature with the most homogeneous class distribution, that also means a lowest impurity. The measure of impurity can be computed with many different methods, depending from the algorithm. The most used are Gini Index [67], Entropy [68] and Misclassification error [11]. The decision path ends when all the records belong to the same class or have similar attribute values. Advantages of decision trees [11]:

- Is easy to understand to such an extent that it is interpretable for nature, thanks to its easy visualization.
- Is able to handle both numerical and categorical data, requiring little data preparation.
- Has a cost that is logarithmic in the number of data points used to train the tree and offers the possibility to validate a model by statistical tests, providing a certain reliability to the model.

Disadvantages of decision trees [11]:

- Could end up to over-complex trees that do not generalize the data well, taking to overfitting. A partial solution is the prune the decision tree

limiting the minimum number of samples required at a leaf node or setting a maximum depth of the tree.

- Can be unstable because small variations in the data may completely change the generated tree.
- Cab be created biased trees if some classes dominate, reason why it is recommended to balance the dataset prior to fitting with the decision tree.
- Is sensible to missing values, data fragmentation and Tree Replication.

The example in Figure 4.8 shows a decision tree extracted for the IRIS [69] dataset, using the Gini Index as measure of impurity, a minimum number of sample for the split equal to 2, which becomes one for the leaf nodes. All the leafs that classify samples as belonging to a class are represented with the same color, so to each class is associated a color. The color of the internal node is referred to the class most represented in the set of samples considered in that path.

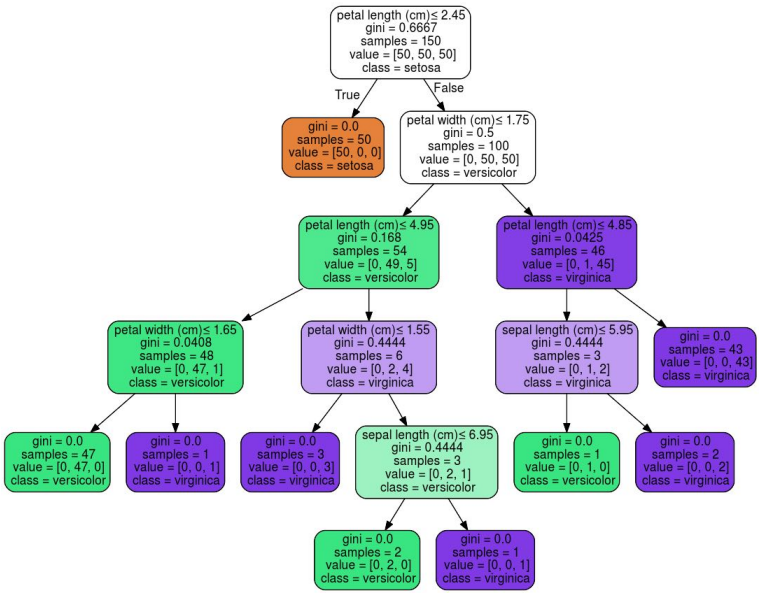


Figure 4.8: Visualization of a Decision Tree on Iris dataset [11]

4.3.2 Random Forest

Random Forest is an ensemble learning technique based on Decision Trees, which are combined to avoid overfitting and improve accuracy [12, 70].

Each tree of the ensemble is created from a sample drawn with replacement (i.e. a bootstrap sample) from the training dataset. The best split of a node can be found in two ways, considering all input features or a random subset, also known as feature bagging [12]. These sources of randomness helps to decrease the variance of the forest estimator and reducing overfitting, and also decorrelates the trees [70]. The main advantages of this technique is that it has an higher accuracy than decision trees, it is robust to noise and outliers, it has a fast training phase and provides global feature importance, i.e. an estimate of which features are important in the classification. The drawbacks are a slight increase in bias and the chance of making results that are difficult to interpret [12]. Figure 4.9 represents an example of the random forest approach, that defines the class by majority voting.

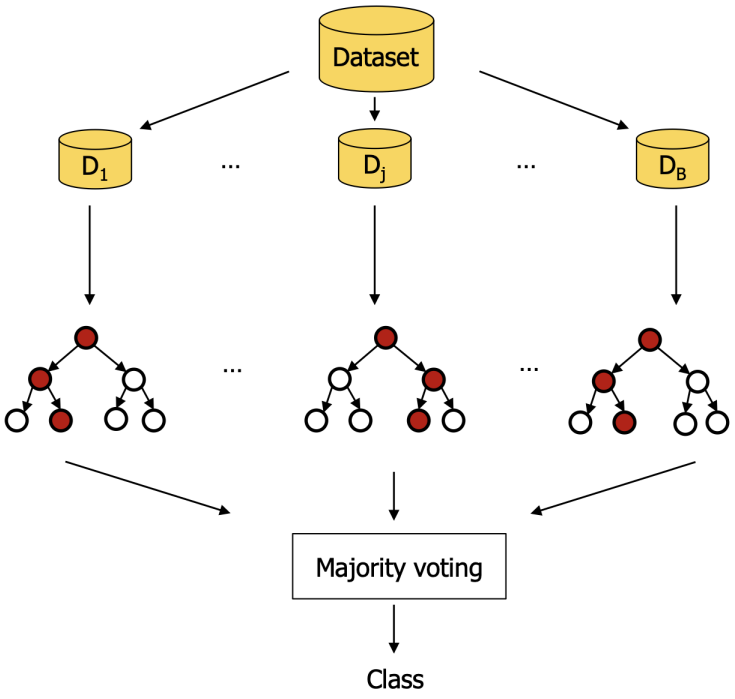


Figure 4.9: Random Forest example [12].

4.3.3 Support vector machine

The support vector machine, as known as SVM, is a supervised learning method which aim is to find an hyperplane that well separates and classifies data points [71]. There are many possible hyperplanes that could be chosen to separate classes of sample, for that reason is necessary to find the hyperplane that has the maximum margin from the nearest points. Hyperplanes are decision boundaries that allow the classification of samples and its dimension is related to the sample's number of features. When there are two features the hyperplane is simply a line, instead when the features are three the hyperplane is a bi-dimensional plane [71]. The samples that are closer to the hyperplane compose the support vector, which defines the position and the orientation of the hyperplane, with the goal of maximizing the margin. As shown in Figure 4.10 the margin can be of two types, hard margin or soft margin. SVMs implementing the hard margin are more sensible to outliers, because the margin is intended to strictly separating the classes. When the SVM implements a soft margin, it has a tolerance on samples that are closer to the hyperplane but on the side of the other class. This technique has advantages like being effective in high

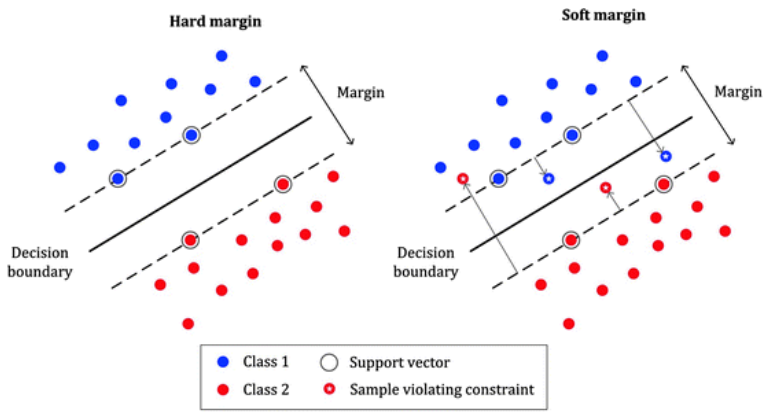


Figure 4.10: Hard margin vs Soft margin [13]

dimensional spaces, even when the number of dimensions is greater than samples one [72]. It also memory efficient thanks to the support vector that help the decision function considering a subset of training samples. It is really versatile thanks to its different possible kernel's function, that can be specified and customized. The example in Figure 4.11 shows how

the SVM, with an appropriate kernel, can separate samples through non-linear boundaries projecting the input into a higher dimensional space, to make it linearly separable [72]. Some important disadvantages are that

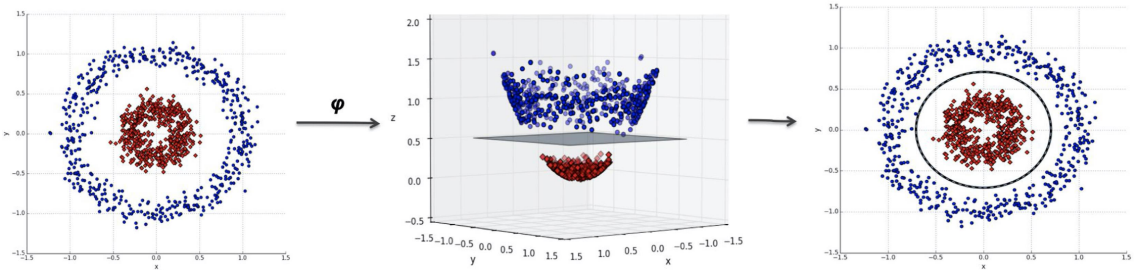
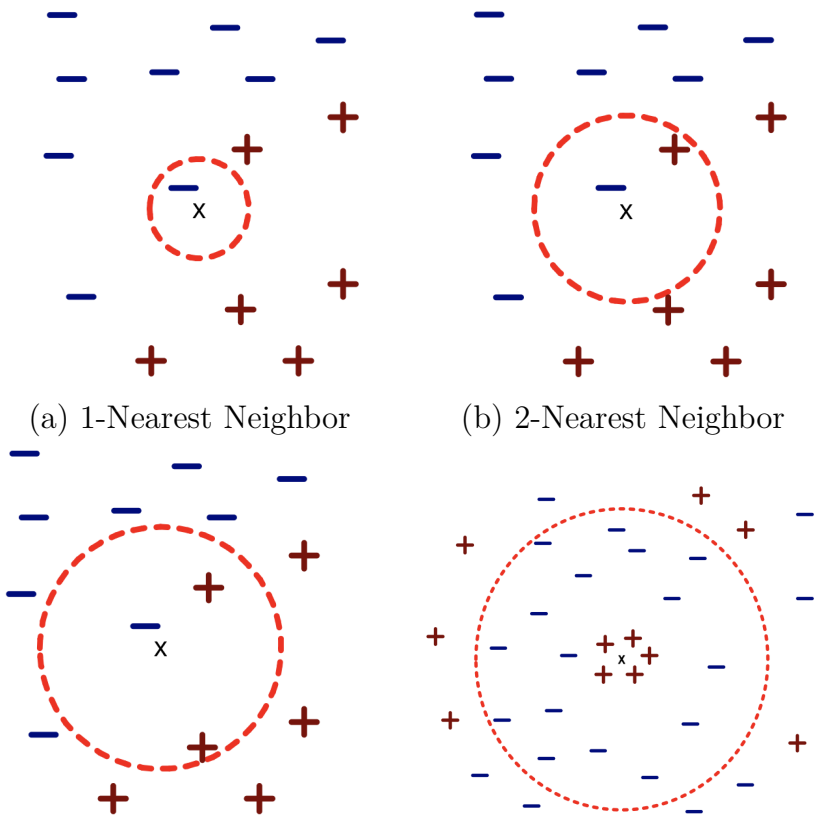


Figure 4.11: Feature space by kernel trick [14].

if the number of features is much higher than the number of samples, is necessary to fine tune well the choice of kernel functions and the regularization term, to avoid overfitting. This technique does not provide any direct probability estimates, which are computed by cross-validation [72].

4.3.4 K-Nearest Neighbors

The K-Nearest Neighbors is a neighbors-based algorithm, a kind of method that implements instance-based learning or non-generalizing learning, in fact it does not build a general model, but stores the instances of the training data [73]. The classification is defined by majority voting of the nearest neighbors of each sample, the winning class is the one with the most representatives within the nearest neighbors of the sample [15]. Parameter K defines the number of closest points to be considered for the voting phase of the sample in exam. Choosing the value of K, as shown in Figure 4.12, is not simple because if K is too small it makes the classification algorithm sensitive to noise points, otherwise if K is too large, the neighborhood could include points from other classes [15]. One of the issues of this algorithm is the feature domain, that usually needs to be normalized, to avoid distance measures from being dominated by one of the features. The distance measures can also provoke another issue, the curse of dimensionality, which is caused by high dimensional data.



(a) 1-Nearest Neighbor (b) 2-Nearest Neighbor

(c) 3-Nearest Neighbor (d) Large K-Nearest Neighbor

Figure 4.12: Definition of K-Nearest Neighbor [15]

4.3.5 Neural Network Multi Layer Perceptron

Neural Network Multi-layer Perceptron (NN-MLP) is a nonlinear supervised learning algorithm that is used in this work as a representative of Neural Network classifiers [74]. This model is inspired to the structure of the human brain, in fact the perceptron, the fundamental building block of a neural network, is also called neuron because it mimics the human brain's functions done by the neuron [17]. An example of a Neural Network Multi-Layer Perceptron is shown in Figure 4.13. The input is taken from the layers on the left, a set of neurons representing the input features. The middle layers, also called hidden layers, transform the values from the previous layer with a weighted linear summation then followed by a non-linear activation function. The output layer gets the values from the last hidden layers, transforming them into output values [16]. Looking

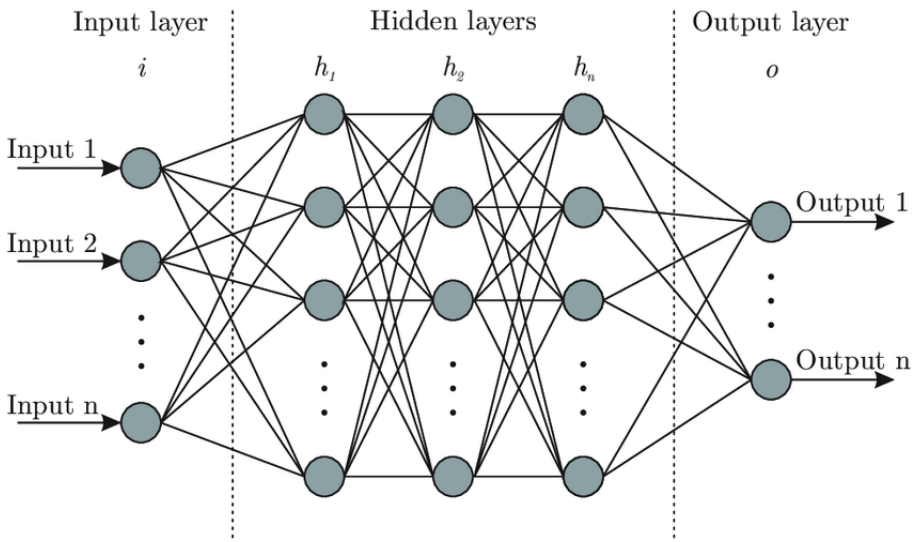


Figure 4.13: Structure of a neural network [16]

more strictly on a single node, the neuron's structure is the one of Figure 4.14. At the beginning are assigned random values to weights and offsets [17]. Training samples are processed one at a time. For each neuron the result is given applying weights, offset and activation function for the instance. A forward propagation is done until the output is computed. The computed output is then compared with the expected output and is evaluated the error, which is back propagated, updating weights and offset for each neuron. The training process ends when the percentage of accuracy is above a given threshold or the percentage of parameter variation (error) is below a given threshold or when the maximum number of epochs is reached [17]. The most relevant advantages are the high accuracy, the robustness to noise and outliers but also the support to both discrete and continuous output [17, 74]. The drawbacks are that this classifiers take a long training time, is weakly scalable in training data size, it has a complex configuration, the application domain knowledge cannot be exploited in the model. The heaviest disadvantage, from the thesis work point of view, is that the model is not interpretable [17, 74].

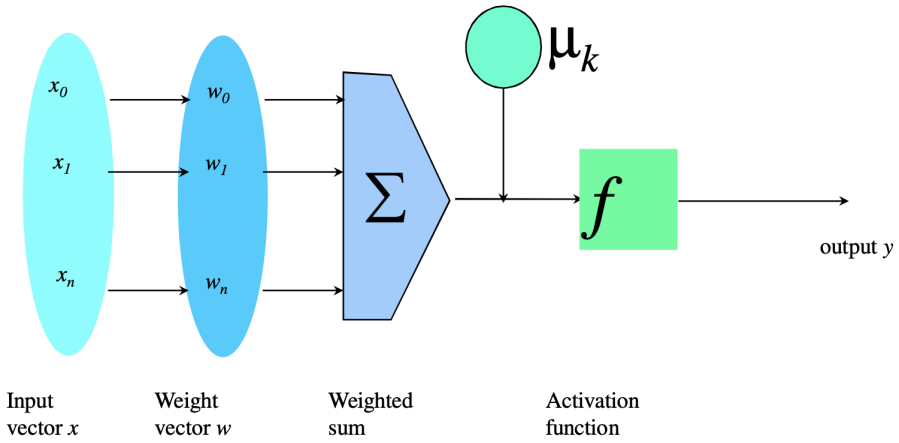


Figure 4.14: Structure of a neuron [17]

Chapter 5

Experiments

The chapter consists of three sections, the first two are provide the results and analysis for the datasets, of which the first section is on artificial datasets while the second one is on real datasets. Finally, the third section is dedicated to observations on time's performances, considering for the first part all the experiments on artificial datasets and on the second one all the experiments on real datasets.
In the first part are used artificial data sets, in order to take in account how much this methods gives us back what we expect as background truth. The reason behind that choice is that, to be enough sure of the answer, in this case is truly important knowing, in advance, the true answer. In that way the explanation given by the explainer is really comparable with the expected features relationship. Another interesting aspect is that this data sets also allow to compare the explanations given by the different explainers, which are depending by the model and its configuration. This provides us some insight on the model behind the explanation.
The second part is focused on the analysis of real data sets. These data sets are chosen taking in consideration aspects as their dimensionality, intended in terms of both number of samples and features cardinality. Some of the data sets are chosen as literature knowledge example, so it helps to stress and recognise some relationship. Since in real cases data scientists do not have the complete knowledge on the dataset problem, the main feature for the relationship are provided by domain experts, that suggest the feature to take into account, also called "feature of interest" (FOI).

5.1 Artificial data sets experiments

I have created two different couples of data sets, which are made with the intent of stressing some importance's relationship. All the data set are build on a set of five features, named X, Y, Z, K, W. The first couple is named "**Blobs**" and its data sets are called **2D-Blobs** and **3D-Blobs**, while the second couple is composed by a data set called **Moons** and another called **Circles**.

5.1.1 Dataset description

2D-Blobs

This dataset provides two main features, X and Y, which are the only useful to define the designed clusters, while the other features can not provide any contribution, as for example the feature Z in the 3-dimensional plot shown in Figure 5.1. So the aim of the dataset is getting different importance for a single feature, depending on the associated cluster. As shown

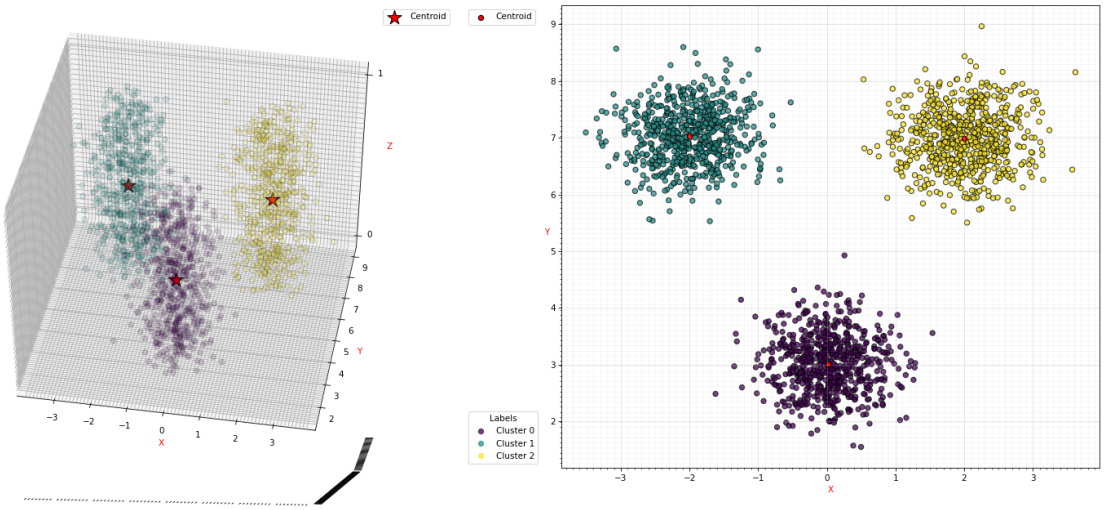


Figure 5.1: 2D-Blobs of 2000 samples, clusters extracted by KMeans algorithm

in Figure 5.1, the data set is composed of three globular clusters, its structure stresses an higher importance of feature Y for a specific cluster (the violet one), that is placed in a way that the data in its range probably

belongs to it, so that enables to increase its weight as probable solution. Feature X, on the other hand, emphasizes the difference between the clusters on the top, which are not distinguishable by the feature Y, but that are clearly distinct on the X's space. It is also given a global consideration, in fact along feature Y, is possible to clearly distinguish only one cluster, while feature X allows to a partial distinction of two or three cluster.

3D-Blobs

The 3D-Blobs is a modified version of the 2D-Blobs, which difference is on the domain space of the feature Z. In this data set, also the feature Z as an impact on defining the cluster, and its values are chosen to have the same behaviour as the ones of feature Y. In Figure 5.1 is possible to

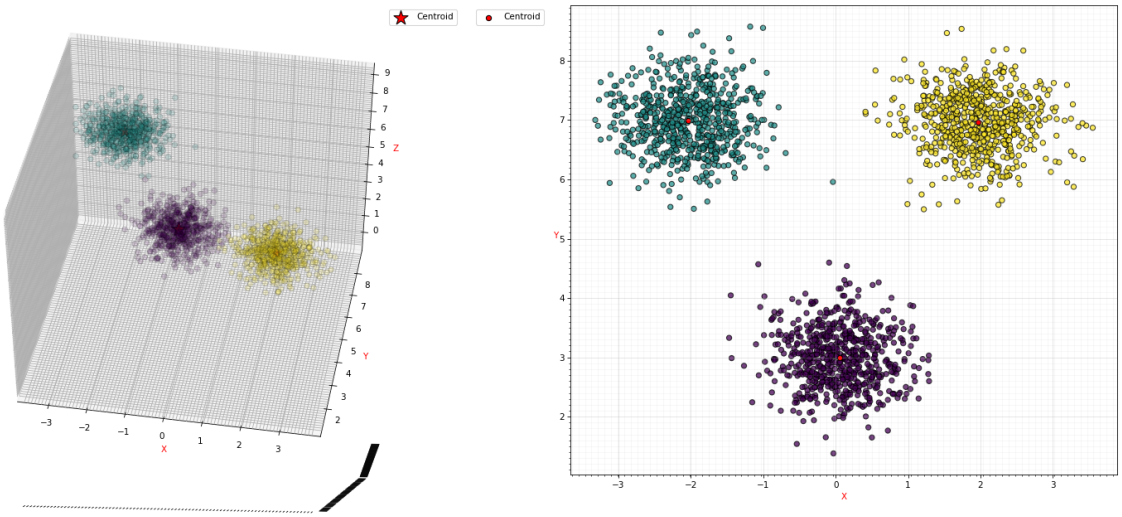


Figure 5.2: 3D-Blobs of 2000 samples, clusters extracted by OPTICS algorithm

observe that feature Y, on the XY cartesian plane, distinguish the violet cluster from the green and yellow ones. Moreover Z feature's values are created to clearly separate the yellow cluster from the green and violet one, considering only that feature. In that way we can evaluate if the feature Z has more importance, influencing the explanation.

Moons

Respect to the previous data sets, data set Moons provides a more challenging importance definition for the features X and Y, for the position and the shape given to the two clusters. The first cluster is placed cen-

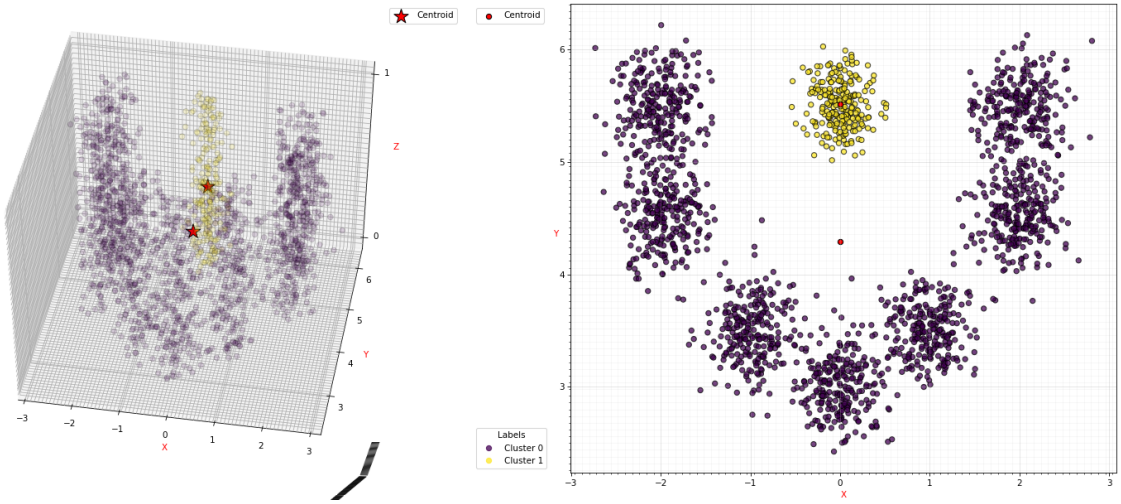


Figure 5.3: Moons of 2000 samples, clusters extracted by OPTICS algorithm

tered on the top of the XY cartesian plane, and has a full moon's shape, while the second cluster has an halfmoon shape, a lower semi circumference which has the center that point the center of the first cluster. It is clearly up to the explainer to give an importance's gap between the two feature, that are both important.

Circles

The Circles is a data set used to better investigate the influence of a feature, given by the shape of clusters, which, respect to the Moons ones, is the same for the internal cluster, instead the external one has a closed circular shape, a full moon instead of the half moon.

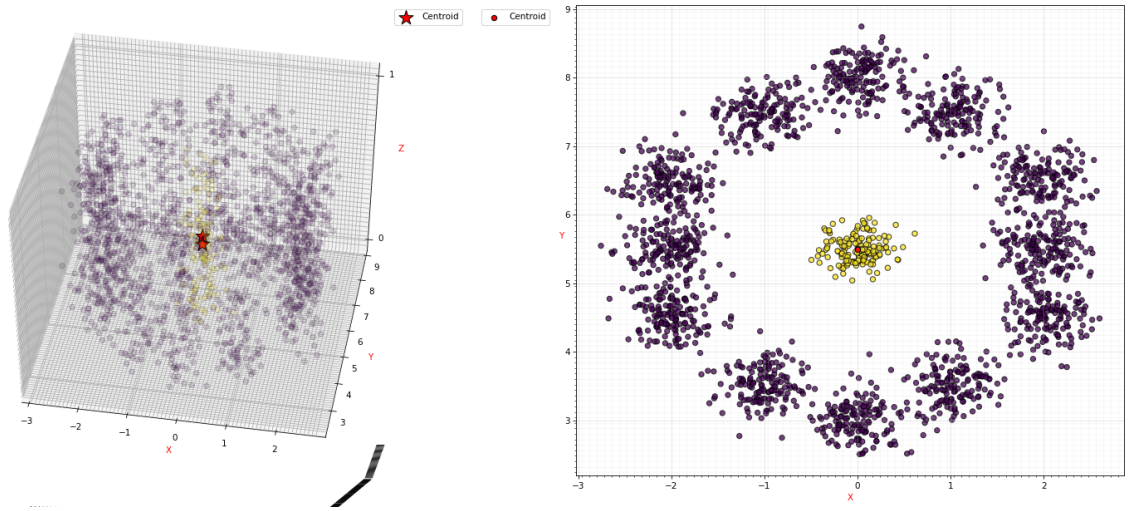


Figure 5.4: Circles of 2000 samples, clusters extracted by Spectral algorithm

5.1.2 Experimental settings

All the experiments done with this data sets are made with different data set size: 300, 500, 1000, 2000 and 3000 samples.

Clustering algorithms configurations

For these artificial data sets I used algorithms with the following configurations, common for each experiment. When is required a random number, it is used a common seed number, that ensure reproducibility.

K-Means For the parameters of this algorithm I used the default values.

DBSCAN DBSCAN required a tuning of parameters, to better identify the appropriate parameters for the artificial dataset. The resulting setting is with *eps* value of 0.9 and a minimum number of samples equal to 10.

OPTICS The parameters for OPTICS algorithm are the minimum number of samples, which has a value of 20 and the minimum cluster size of 0,1, while for the parameters *max_eps*, distance metric and *xi* are used the default values. The maximum default eps is infinite, to get cluster

across all scales, as distance metric the Minkowski is the default one. The parameter *xi* represent the minimum steepness, that defines the cluster boundary, and its default value is 0,05.

Spectral Clustering For the Spectral clustering it is passed the correct number of clusters in the tested data set, while the affinity matrix is constructed by computing a graph of nearest neighbors. The eigenvectors are computed with one of the most used strategy for large scale eigenvalues problem, the ARPACK. The K number of cluster, as already said, is the real one, but thanks to the projection of the samples into a non-linear embedding, It is also possible to obtain the number of cluster by analyzing the eigenvalues of the Laplacian matrix. This because when a similarity graph is not fully connected, the number of cluster estimation is given by the multiplicity of eigenvalue with 0 value.

Agglomerative Clustering The only parameter needed to the Agglomerative clustering algorithm is the number of clusters of the data set in exam, for which a number of clusters equal to those expected from the dataset is passed.

Classification algorithms configurations

For all the artificial experiment were used the same algorithm's setting. When required by the algorithm setting, the random state is common for all methods, for reproducibility.

Decision Tree The only parameter defined for the Decision Tree is the minimum number of samples for a leaf, which is defined with at least 5 samples. The quality measure of a split is given by the "gini" [67] criteria, that measure the Gini impurity and the entropy for the information gain. As splitting strategy, the chosen one is the "best" that considers the best split over all the possible features. The maximum depth is defined as "None", in that way the nodes are expanded until all leaves are pure or with less than the minimum needed samples for a split, which is 2 as default.

Random Forest Some of the Decision Tree's paremeters are unchanged for the Random Forest, as the quality measure of a split, the splitting strategy and the minimum samples for a split. The maximum depth, for this classifier is 6, even if the nature of the artificial datasets mostly let the algorithm reach less than 6 features, it is limited to 6 to avoid too specific path in the generated trees. As number of estimators, 10 trees resulted enough, because of the low feature dimensionality and variability.

Support Vector Machine In scikit-learn the SVM implementation for classification task is called SVC and offers different kernel type that can be used in the algorithm as *linear*, *polynomial*, *rbf* and *sigmoid*. In this work I have selected the *rbf* kernel, which stands for Radial Basis Function. This kernel is one of the most used in literature, also thanks to its capability to extract non-linear boundaries. Regularization parameter C used is the default one, which value is 1. The probability estimates are enabled even if it slows the classifier's execution, because internally it uses a cross-validation.

K-Nearest Neighbors

The parameters tuned for this algorithm are the *n_ neighbors* and the weight function. The first one is influenced by the nature of the artificial dataset, so it is defined a small number of neighbors equal to 2. The chosen weight function is the uniform one, all points in each neighborhood are weighted equally. The algorithm used to compute the nearest neighbors is *auto*, it will attempt to settle the most appropriate algorithm based on the values passed to fit method. The distance metric to used is minkowski, which is the default one for scikit-learn.

Neural Network Multi Layer Perceptron

The neural network adopted is the same used by SHAP's creator, to get a reasonable simple model. It is build on 5 hidden layers, with a *lbfgs* solver for the weight optimization, an optimizer in the family of quasi-Newton methods.

Why scikit-learn's algorithms implementations?

There are many other Machine Learning packages providing an implementation of these clustering algorithms but I choose scikit-learn because

of its more extended community of developers, which means its methods are well tested and optimized in resources usage. For the permutation importance I used the Eli5 [75] implementation, instead of the scikit-learn one, because it provides a colored representation, which visually makes the order of importance of the features clearer.

5.1.3 Experimental results

This section reports the importance weights extracted from each combination of the clustering and classification algorithms. The results of each dataset are divided in three parts. In the first part are shown the results extracted by combining clustering algorithms with the Random Forest classifier. For this model, in Figures 5.5, 5.8, 5.11 and 5.14 are displayed the importance weights of SHAP's explainers BruteForce, Kernel, Sampling, Tree and in addition are shown the results from the feature importance function by Random Forest model. The second part shows the results extracted for the Decision Tree classifier, for which 5.6, 5.9, 5.12 and 5.15 displays the importance weights of SHAP's explainers BruteForce, Kernel, Sampling and Tree. Finally, the third part provides the results for the single combinations of clustering algorithms with the remaining classifiers, Support Vector Machine, K-Nearest Neighbors and Neural Network Multi Layer Perceptron. The importance weights in 5.7, 5.10, 5.13 and 5.16 refer only to the ones by SHAP's explainers BruteForce, Kernel, Sampling and Tree.
SHAP's results are intended as the absolute values of the mean of the SHAP values [50], while the feature importance's results are computed as Gini importances [51, 67].

Importance weights extracted on 2D Blobs

Random Forest		SHAP explainers Brute - Kernel - Sampling			TreeExplainer		Feature Importance		
Clustering Algo	# sample	X	Y	3rd feature	X	Y	X	Y	3rd feature
Agglomerative	300	0,75	0,6	0,01	0,75	0,6	0,63	0,31	0,03
	500	0,71	0,66	0,03	0,71	0,66	0,57	0,4	0,01
	1000	0,75	0,65	0,01	0,75	0,65	0,62	0,36	0,01
	2000	0,6	0,8	0,02	0,7	0,66	0,55	0,44	0,01
	3000	0,68	0,86	0,02	0,7	0,67	0,5	0,49	0,01
DBSCAN	300	0,75	0,65	0,04	0,75	0,65	0,53	0,43	0,02
	500	0,72	0,7	0,02	0,7	0,6	0,62	0,36	0,02
	1000	0,74	0,72	0,02	0,7	0,6	0,61	0,37	0,01
	2000	0,67	0,82	0,01	0,76	0,6	0,65	0,34	0,01
	3000	0,67	0,8	0,01	0,76	0,6	0,63	0,37	0,01
KMeans	300	0,8	0,6	0,03	0,8	0,6	0,64	0,33	0,01
	500	0,8	0,63	0,03	0,8	0,63	0,55	0,44	0,01
	1000	0,75	0,67	0,01	0,75	0,67	0,69	0,29	0,01
	2000	0,69	0,8	0,02	0,8	0,62	0,6	0,4	0,01
	3000	0,65	0,9	0	0,8	0,61	0,54	0,46	0,01
OPTICS	300	0,8	0,6	0,03	0,8	0,6	0,68	0,27	0,03
	500	0,65	0,75	0,03	0,65	0,75	0,5	0,46	0,01
	1000	0,69	0,73	0,03	0,7	0,72	0,51	0,47	0,01
	2000	0,69	0,78	0,02	0,71	0,78	0,55	0,44	0,01
	3000	0,63	0,8	0,02	0,64	0,82	0,51	0,49	0,01
Spectral	300	0,72	0,62	0,05	0,72	0,62	0,62	0,33	0,02
	500	0,7	0,705	0,05	0,73	0,67	0,65	0,33	0,01
	1000	0,69	0,73	0,03	0,69	0,73	0,6	0,38	0,01
	2000	0,65	0,8	0,01	0,73	0,67	0,61	0,38	0,01
	3000	0,63	0,85	0,01	0,74	0,66	0,54	0,45	0,01

Figure 5.5: Importance weights by Random Forest

Decision Tree		SHAP explainers Brute - Kernel - Sampling			TreeExplainer	
Clustering Algo	# sample	X	Y	3rd feature	X	Y
Agglomerative	300	0,8	0,7	0	0,7	0,85
	500	0,7	0,9	0	0,7	0,85
	1000	0,66	0,9	0	0,68	0,83
	2000	0,67	0,92	0	0,7	0,85
	3000	0,67	0,9	0	0,7	0,85
DBSCAN	300	0,64	0,84	0,02	0,7	0,85
	500	0,69	0,78	0	0,7	0,85
	1000	0,68	0,9	0	0,7	0,85
	2000	0,68	0,9	0	0,7	0,85
	3000	0,68	0,9	0	0,7	0,85
KMeans	300	0,7	0,9	0	0,7	0,85
	500	0,8	0,7	0	0,7	0,85
	1000	0,7	0,9	0	0,7	0,85
	2000	0,67	0,89	0	0,7	0,85
	3000	0,7	0,9	0	0,7	0,85
OPTICS	300	0,7	0,9	0	0,7	0,85
	500	0,7	0,9	0	0,7	0,85
	1000	0,7	0,9	0	0,7	0,85
	2000	0,67	0,89	0	0,7	0,85
	3000	0,68	0,9	0	0,7	0,85
Spectral	300	0,7	0,85	0	0,84	68
	500	0,7	0,85	0	0,7	0,85
	1000	0,65	0,9	0	0,7	0,85
	2000	0,7	0,9	0	0,7	0,85
	3000	0,68	0,9	0	0,7	0,85

Figure 5.6: Importance weights by Decision Tree

		SVM			K-NN			NN-MLP		
Clustering Algo	**# sample**	**X**	**Y**	**3rd feature**	**X**	**Y**	**3rd feature**	**X**	**Y**	**3rd feature**
Agglomerative	300	0,66	0,76	0	0,65	0,85	0	0,0011	0,0012	0,004
	500	0,65	0,85	0	0,7	0,9	0	6,5 x 10(-5)	8 x 10(-5)	-
	1000	0,65	0,85	0	0,65	0,9	0	3,8 x 10(-4)	3,6 x 10(-4)	1,5 x 10(-4)
	2000	0,66	0,85	0	0,64	0,9	0	1,3 x 10(-4)	1,25 x 10(-4)	4 x 10(-5)
	3000	0,63	0,88	0	0,66	0,9	0	0,79	0,7	0,02
DBSCAN	300	0,75	0,71	0,01	0,68	0,9	0	0	0	0
	500	0,59	0,9	0	0,6	0,95	0	0,69	0,71	0,04
	1000	0,66	0,82	0	0,65	0,9	0	4,7 x 10(-5)	6,7 x 10(-5)	2,5 x 10(-5)
	2000	0,65	0,85	0	0,65	0,9	0	0,77	0,8	0
	3000	0,68	0,84	0	0,66	0,9	0	0,79	0,69	0
KMeans	300	0,65	0,75	0	0,7	0,9	0	0,00139	0,0014	0,0005
	500	0,65	0,8	0	0,65	0,9	0	1,5 x 10(-4)	2x 10(-4)	-
	1000	0,65	0,85	0	0,65	0,9	0	3,2 x10(-4)	3,5x10(-4)	1 x 10(-4)
	2000	0,65	0,85	0	0,67	0,89	0	0,74	0,72	0,01
	3000	0,63	0,86	0	0,66	0,9	0	1,18 x 10(-4)	1,2x 10(-4)	4,5 x 10(-5)
OPTICS	300	0,68	0,72	0	0,7	0,9	0	0,6	0,9	0
	500	0,65	0,8	0	0,7	0,9	0	2,5 x 10(-5)	3,5 x 10(-5)	-
	1000	0,65	0,85	0	0,65	0,9	0	0,045	0,02 (4th)	0,05 (Z)
	2000	0,67	0,89	0	0,67	0,9	0	0,74	0,8	0,01
	3000	0,66	0,86	0	0,66	0,9	0	2,2 x 10(-7)	3,2 x 10(-7)	1,3 x 10(-7)
Spectral	300	0,6	0,8	0	0,6	0,9	0	0,6	0,9	0
	500	0,65	0,8	0	0,65	0,9	0	0,65	0,9	0
	1000	0,65	0,85	0	0,65	0,9	0	1,65 x 10(-4)	1,6 x 10(-4)	6 x 10(-5)
	2000	0,67	0,89	0	0,65	0,9	0	1,1 x 10(-4)	1,05 x 10(-4)	4 x 10(-5)
	3000	0,67	0,84	0	0,66	0,9	0	0,7	0,8	0,02

Figure 5.7: 2D Blobs classified by SVM, K-NN and NN-MLP

Importance weights extracted on 3D Blobs

Random Forest		SHAP explainers Brute - Kernel - Sampling			Tree Explainer			Feature Importance		
Clustering Algo.	**# sample**	**X**	**Y**	**Z**	**X**	**Y**	**Z**	**X**	**Y**	**Z**
Agglomerative	300	0,55	0,45	0,43	0,63	0,55	0,25	0,43	0,29	0,26
	500	0,59	0,58	0,35	0,5	0,58	0,42	0,48	0,36	0,15
	1000	0,44	0,63	0,48	0,45	0,51	0,49	0,41	0,38	0,2
	2000	0,43	0,69	0,425	0,43	0,56	0,55	0,36	0,4	0,23
	3000	0,47	0,75	0,34	0,4	0,65	0,53	0,38	0,42	0,2
DBSCAN	300	0,61	0,57	0,37	0,6	0,45	0,03	0,43	0,37	0,18
	500	0,75	0,67	0,27	0,66	0,4	0,5	0,47	0,35	0,17
	1000	0,36	0,66	0,5	0,43	0,57	0,55	0,37	0,32	0,3
	2000	0,455	0,68	0,45	0,33	0,75	0,56	0,44	0,31	0,25
	3000	0,38	0,7	0,54	0,43	0,67	0,48	0,346	0,3	0,347
KMeans	300	0,5	0,68	0,28	0,48	0,7	0,36	0,38	0,44	0,16
	500	0,46	0,495	0,52	0,47	0,51	0,36	0,37	0,33	0,29
	1000	0,41	0,6	0,515	0,45	0,59	0,44	0,38	0,3	0,31
	2000	0,41	0,66	0,48	0,45	0,68	0,42	0,39	0,31	0,3
	3000	0,37	0,67	0,5	0,38	0,62	0,47	0,347	0,355	0,3
OPTICS	300	0,62	0,55	0,29	0,58	0,56	0,34	0,46	0,37	0,16
	500	0,59	0,5	0,43	0,47	0,59	0,43	0,43	0,35	0,21
	1000	0,54	0,52	0,48	0,53	0,61	0,38	0,44	0,26	0,39
	2000	0,38	0,67	0,5	0,44	0,7	0,4	0,367	0,371	0,26
	3000	0,74	0,46	0,38	0,44	0,73	0,37	0,34	0,46	0,2
Spectral	300	0,57	0,47	0,45	0,55	0,6	0,32	0,45	0,3	0,25
	500	0,64	0,65	0,19	0,47	0,53	0,46	0,44	0,45	0,1
	1000	0,4	0,68	0,44	0,49	0,64	0,41	0,388	0,387	0,22
	2000	0,45	0,68	0,4	0,5	0,64	0,35	0,39	0,36	0,25
	3000	0,45	0,74	0,39	0,41	0,6	0,46	0,35	0,45	0,2

Figure 5.8: Importance weights by Random Forest

Decision Tree		SHAP explainers Brute - Kernel - Sampling			Tree Explainer		
Clustering Algo.	# sample	X	Y	Z	X	Y	Z
Agglomerative	300	0	0,69	0,8	0,83	0	0,7
	500	0	0,86	0,72	0,69	0,84	0
	1000	0	0,68	0,87	0	0,87	0,68
	2000	0	0,87	0,68	0	0,68	0,87
	3000	0,68	0,87	0	0	0,9	0,68
DBSCAN	300	0,68	0,1	0,88	0,02	0,7	0,9
	500	0,1	0,69	0,88	0,05	0,9	0,7
	1000	0	0,67	0,9	0	0,68	0,9
	2000	0,9	0,67	0	0,06	0,68	0,9
	3000	0,02	0,67	0,9	0,02	0,9	0,7
KMeans	300	0	0,7	0,88	1,37	0	0
	500	0,68	0,88	0	0	0,7	0,86
	1000	0	0,68	0,9	0	0,68	0,86
	2000	0	0,68	0,9	0	0,68	0,86
	3000	0	0,68	0,9	0	0,68	0,86
OPTICS	300	0,81	0,75	0	0	0,86	0,7
	500	0	0,86	0,7	0	0,86	0,7
	1000	0,66	0,9	0	0	0,7	0,84
	2000	0	0,7	0,9	0	0,68	0,86
	3000	0	0,9	0,68	0	0,86	0,68
Spectral	300	0,81	0,73	0	0	0,7	0,86
	500	0	0,87	0,69	0	0,7	0,86
	1000	0	0,68	0,86	0,7	0,86	0,7
	2000	0	0,68	0,86	0	0,7	0,86
	3000	0	0,68	0,86	0,7	0,86	0,7

Figure 5.9: Importance weights by Decision Tree

		SVM			K-NN			NN-MLP		
Clustering Algo.	# sample	X	Y	3rd feature	X	Y	3rd feature	X	Y	3rd feature
Agglomerative	300	0,3	0,6	0,47	0,05	0,68	0,84	0,28	0,59	0,47
	500	0,29	0,58	0,56	0,09	0,6	0,95	0,13	0,61	0,82
	1000	0,24	0,63	0,57	0,1	0,64	0,89	0,22	0,58	0,5
	2000	0,19	0,67	0,58	0,05	0,63	0,89	0,25	0,68	0,73
	3000	0,23	0,65	0,62	0,1	0,63	0,9	0,07	0,62	0,88
DBSCAN	300	0,48	0,53	0,42	0,05	0,67	0,89	0,45	0,68	0,5
	500	0,47	0,63	0,47	0,18	0,69	0,82	0	0	0
	1000	0,32	0,7	0,52	0,05	0,66	0,9	0	0	0
	2000	0,25	0,62	0,58	0,1	0,62	0,9	0,01	0,67	0,675
	3000	0,46	0,71	0,48	0,1	0,63	0,9	0	0	0
KMeans	300	0,29	0,61	0,48	0,05	0,69	0,85	0,15	0,61	0,41
	500	0,28	0,61	0,52	0,08	0,66	0,9	0,12	0,63	0,35
	1000	0,24	0,63	0,58	0,1	0,64	0,9	0,4	0,34	0,8
	2000	0,23	0,65	0,58	0,1	0,64	0,9	0,48	0,55	0,68
	3000	0,2	0,69	0,6	0,1	0,64	0,9	0,05	0,64	0,3
OPTICS	300	0,3	0,59	0,5	0,05	0,65	0,91	0,18	0,6	0,8
	500	0,31	0,59	0,52	0,1	0,6	0,92	0,1	0,58	0,48
	1000	0,24	0,62	0,59	0,09	0,64	0,91	0,67	0,16	0,75
	2000	0,21	0,65	0,58	0,09	0,68	0,89	0,07	0,66	0,89
	3000	0,2	0,69	0,6	0,1	0,63	0,9	0,27	0,65	0,58
Spectral	300	0,3	0,59	0,49	0,03	0,67	0,9	0,17	0,54	0,53
	500	0,3	0,6	0,5	0,1	0,62	0,9	0,39	0,49	0,74
	1000	0,24	0,64	0,56	0,08	0,65	0,88	0,28	0,34	0,68
	2000	0,22	0,65	0,57	0,08	0,65	0,88	0,31	0,55	0,8
	3000	0,22	0,65	0,61	0,08	0,64	0,88	0,33	0,62	0,56

Figure 5.10: 3D Blobs classified by SVM, K-NN and NN-MLP

Importance weights extracted on Moons

Random Forest		SHAP explainers Brute - Kernel - Sampling			TreeExplainer		Feature Importance		
Clustering Algo.	# sample	X	Y	3rd feature	X	Y	X	Y	3rd feature
Agglomerative	300	0,67	0,14	0,01	0,57	0,13	0,88	0,07	0,03
	500	0,68	0,14	0	0,68	0,12	0,92	0,07	0,01
	1000	0,7	0,09	0	0,64	0,1	0,94	0,05	0,01
	2000	0,56	0,18	0	0,63	0,53	0,91	0,08	0,01
	3000	0,62	0,15	0	0,65	0,13	0,92	0,08	0,01
DBSCAN	300	0,22	0,24	0,04	0,29	0,31	0,54	0,32	0,05
	500	0,21	0,25	0,025	0,24	0,27	0,6	0,28	0,05
	1000	0,25	0,27	0,02	0,235	0,24	0,54	0,34	0,06
	2000	0,2	0,31	0,02	0,23	0,3	0,64	0,3	0,04
	3000	0,25	0,27	0,01	0,29	0,27	0,54	0,35	0,05
KMeans	300	0,95	0,01	0,011	0,9	0,01	0,95	0,02	0,02
	500	0,91	0	0	0,88	0,01	0,96	0,02	0,01
	1000	0,97	0,02	0	0,97	0,05	0,94	0,03	0,01
	2000	0,9	0	0	0,97	0,02	0,98	0,01	0,004
	3000	0,88	0	0	0,97	0,01	0,98	0,01	0,003
OPTICS	300	0,4	0,3	0,05	0,71	0,37	0,65	0,26	0,05
	500	0,64	0,45	0,05	0,86	0,38	0,7	0,23	0,03
	1000	0,25	0,28	0,02	0,745	0,75	0,75	0,2	0,02
	2000	0,245	0,255	0,01	0,24	0,29	0,62	0,33	0,02
	3000	0,24	0,28	0,01	0,25	0,27	0,73	0,25	0,01
Spectral	300	0,245	0,23	0,02	0,19	0,26	0,41	0,44	0,07
	500	0,24	0,27	0,04	0,25	0,24	0,61	0,28	0,07
	1000	0,26	0,27	0,02	0,22	0,3	0,55	0,35	0,04
	2000	0,265	0,25	0,02	0,26	0,31	0,47	0,44	0,05
	3000	0,18	0,26	0,01	0,25	0,27	0,54	0,37	0,03

Figure 5.11: Importance weights by Random Forest

Decision Tree		SHAP explainers Brute - Kernel - Sampling			TreeExplainer	
Clustering Algo.	# sample	X	Y	3rd feature	X	Y
Agglomerative	300	0,68	0,05	0	0,66	0,05
	500	0,71	0,05	0	0,77	0
	1000	0,7	0,04	0,01	0,69	0,09
	2000	0,6	0,18	0	0,68	0,1
	3000	0,67	0,11	0	0,69	0,09
DBSCAN	300	0,31	0,27	0	0,37	0,35
	500	0,315	0,31	0	0,33	0,325
	1000	0,32	0,33	0,01	0,35	0,34
	2000	0,35	0,35	0	0,35	0,34
	3000	0,32	0,3	0,01	0,34	0,32
KMeans	300	0,95	0,1	0	0,97	0
	500	0,95	0	0	0,9	0
	1000	0,91	0,05	0	0,97	0,05
	2000	0,98	0	0	0,97	0
	3000	0,9	0,01	0	0,97	0
OPTICS	300	1,2	0,28	0	0,84	0,35
	500	0,9	0,34	0,01	1	0,47
	1000	0,34	0,36	0	0,86	0,9
	2000	0,33	0,35	0	0,33	0,32
	3000	0,34	0,31	0	0,32	0,31
Spectral	300	0,34	0,31	0	0,32	0,325
	500	0,27	0,3	0	0,295	0,29
	1000	0,32	0,34	0	0,35	0,33
	2000	0,33	0,35	0	0,35	0,34
	3000	0,32	0,31	0	0,32	0,31

Figure 5.12: Importance weights by Decision Tree

		SVM			K-NN			NN-MLP		
Clustering Algo.	**# sample**	**X**	**Y**	**3rd feature**	**X**	**Y**	**3rd feature**	**X**	**Y**	**3rd feature**
Agglomerative	300	0,55	0,2	0,001	0,58	0,22	0,001	0,6	0,1	0,05
	500	0,59	0,19	0	0,6	0,22	0	0,62	0,18	0,01
	1000	0,63	0,2	0	0,63	0,2	0	0,7	0,13	0,01
	2000	0,61	0,18	0	0,62	0,21	0	0,68	0,16	0
	3000	0,6	0,21	0	0,62	0,22	0	0,6	0,21	0,01
DBSCAN	300	0,37	0,29	0,001	0,41	0,29	0,001	0,36	0,23	0,02
	500	0,38	0,32	0,001	0,39	0,315	0,001	0,19	0,11	0,07
	1000	0,4	0,33	0,001	0,39	0,325	0,001	0,39	0,27	0,03
	2000	0,44	0,35	0,001	0,43	0,34	0,001	0,42	0,31	0,02
	3000	0,44	0,33	0,001	0,42	0,31	0,001	0,41	0,26	0,01
KMeans	300	0,9	0,1	0,05	0,9	0,1	0,02	0,95	0	0,01
	500	0,9	0,02	0	0,89	0,05	0	0,89	0	0
	1000	0,91	0,05	0,01	0,9	0,05	0,02	0,98	0,01	0
	2000	0,9	0,02	0	0,91	0,02	0	0,91	0	0
	3000	0,9	0,02	0	0,88	0,02	0,01	0,9	0	0
OPTICS	300	0,95	0,5	0,01	1,1	0,55	0,01	0	0	0
	500	0,78	0,46	0	0,81	0,45	0,01	0,7	0,6	0,03
	1000	0,41	0,35	0	0,41	0,35	0	4,5 x10(-7)	8x10(-7)	4x10(-7)
	2000	0,42	0,35	0	0,42	0,37	0	0	0	0
	3000	0,42	0,34	0	0,42	0,32	0	0,39	0,28	0,02
Spectral	300	0,41	0,3	0	0,43	0,3	0	0,42	0,25	0,03
	500	0,36	0,32	0	0,37	0,31	0,01	0,45	0,3	0,02
	1000	0,39	0,34	0	0,39	0,34	0	0,46	0,34	0,01
	2000	0,42	0,37	0	0,42	0,36	0	0,37	0,29	0,02
	3000	0,42	0,35	0	0,42	0,34	0	0,4	0,28	0,01

Figure 5.13: Moons classified by SVM, K-NN and NN-MLP

Importance weights extracted on Circles

Random Forest		SHAP explainers Brute - Kernel - Sampling			TreeExplainer		Feature Importance		
Clustering Algo.	**# sample**	**X**	**Y**	**3rd feature**	**X**	**Y**	**X**	**Y**	**3rd feature**
Agglomerative	300	0,55	0,29	0,03	0,38	0,68	0,61	0,28	0,05
	500	0,61	0,26	0,01	0,57	0,26	0,6	0,33	0,04
	1000	0,55	0,51	0,01	0,66	0,31	0,49	0,48	0,01
	2000	0,52	0,48	0,01	0,64	0,33	0,44	0,53	0,01
	3000	0,66	0,28	0,01	0,54	0,59	0,67	0,31	0,007
DBSCAN	300	0,15	0,124	0,035	0,11	0,1	0,39	0,25	0,14
	500	0,167	0,105	0,026	0,18	0,12	0,4	0,34	0,11
	1000	0,148	0,17	0,018	0,19	0,11	0,34	0,5	0,07
	2000	0,2	0,1	0,01	0,19	0,13	0,51	0,37	0,05
	3000	0,21	0,15	0,01	0,16	0,15	0,46	0,47	0,02
KMeans	300	0,55	0,4	0,02	0,48	0,41	0,48	0,43	0,04
	500	0,39	0,5	0,05	0,42	0,44	0,43	0,51	0,03
	1000	0,44	0,455	0,02	0,49	0,45	0,5	0,46	0,02
	2000	0,47	0,5	0,01	0,47	0,42	0,46	0,53	0,006
	3000	0,42	0,46	0,02	0,47	0,5	0,47	0,51	0,01
OPTICS	300	0,9	0,2	0,04	0,59	0,09	0,77	0,13	0,04
	500	0,54	0,07	0,01	0,1	0,6	0,89	0,07	0,02
	1000	0,09	0,55	0,01	0,16	0,13	0,07	0,91	0,01
	2000	0,125	0,16	0,01	0,2	0,13	0,432	0,437	0,06
	3000	0,19	0,135	0,01	0,15	0,14	0,51	0,41	0,03
Spectral	300	0,145	0,064	0,04	0,13	0,08	0,33	0,25	0,2
	500	0,13	0,123	0,02	0,15	0,12	0,4	0,35	0,12
	1000	0,133	0,147	0,01	0,145	0,15	0,45	0,4	0,06
	2000	0,17	0,145	0,02	0,18	0,13	0,41	0,47	0,05
	3000	0,175	0,16	0,01	0,18	0,14	0,41	0,5	0,03

Figure 5.14: Importance weights by Random Forest

Decision Tree		SHAP explainers Brute - Kernel - Sampling			TreeExplainer	
Clustering Algo.	**# sample**	**X**	**Y**	**3rd feature**	**X**	**Y**
Agglomerative	300	0,68	0,37	0	0,75	0,45
	500	0,72	0,33	0	0,65	0,4
	1000	0,55	0,58	0	0,68	0,37
	2000	0,57	0,5	0	0,65	0,4
	3000	0,7	0,32	0	0,56	0,6
DBSCAN	300	0,28	0,18	0	0,21	0,25
	500	0,19	0,23	0	0,26	0,21
	1000	0,21	0,28	0	0,17	0,24
	2000	0,26	0,2	0,01	0,24	0,24
	3000	0,27	0,22	0	0,25	0,25
KMeans	300	0,6	0,45	0,02	0,59	0,55
	500	0,52	0,57	0	0,45	0,6
	1000	0,48	0,55	0	0,55	0,54
	2000	0,53	0,5	0	0,52	0,54
	3000	0,5	0,53	0	0,53	0,59
OPTICS	300	1	0,1	0	0,6	0,1
	500	0,6	0,06	0	0,06	0,6
	1000	0,05	0,62	0	0,19	0,25
	2000	0,26	0,21	0	0,25	0,2
	3000	0,2	0,26	0	0,2	0,25
Spectral	300	0,31	0,18	0	0,24	0,28
	500	0,26	0,18	0	0,2	0,24
	1000	0,2	0,25	0	0,19	0,25
	2000	0,21	0,25	0	0,19	0,23
	3000	0,27	0,21	0	0,2	0,25

Figure 5.15: Importance weights by Decision Tree

		SVM			K-NN			NN-MLP		
Clustering Algo.	**# sample**	**X**	**Y**	**3rd feature**	**X**	**Y**	**3rd feature**	**X**	**Y**	**3rd feature**
Agglomerative	300	0,64	0,34	0,01	0,65	0,36	0,01	0,7	0,33	0,07
	500	0,7	0,31	0,01	0,7	0,32	0	0,7	0,33	0,01
	1000	0,51	0,55	0,01	0,51	0,54	0,01	0,545	0,55	0,01
	2000	0,52	0,525	0,01	0,54	0,52	0,01	0,55	0,54	0,01
	3000	0,7	0,32	0	0,68	0,32	0	0,7	0,31	0,01
DBSCAN	300	0,31	0,32	0	0,34	0,32	0	0,3	0,04	0,05
	500	0,31	0,29	0	0,33	0,29	0	0,27	0,09	0,04
	1000	0,32	0,31	0	0,32	0,31	0	0,18	0,04	0,03
	2000	0,3	0,29	0	0,3	0,27	0	0,3	0,23	0,01
	3000	0,32	0,31	0	0,32	0,3	0	0,11	0,04	0,03
KMeans	300	0,54	0,48	0,02	0,53	0,49	0,02	0,58	0,46	0,05
	500	0,49	0,5	0,01	0,47	0,5	0,02	0,45	0,58	0,01
	1000	0,47	0,52	0	0,46	0,52	0,02	0,45	0,57	0,01
	2000	0,5	0,52	0	0,5	0,52	0,02	0,5	0,55	0
	3000	0,49	0,54	0	0,49	0,52	0,02	0,48	0,58	0
OPTICS	300	0,7	0,5	0,01	0,77	0,48	0,02	0,0004	0,00038	0,0002
	500	0,48	0,2	0	0,5	0,2	0,01	0,64	0,01	0,02
	1000	0,18	0,54	0	0,15	0,56	0	0,2	0,56	0,01
	2000	0,29	0,295	0	0,29	0,28	0	0,24	0,13	0,01
	3000	0,32	0,31	0	0,31	0,3	0	0,21	0,1	0,01
Spectral	300	0,32	0,3	0	0,34	0,3	0	0,05	0,009	0,01
	500	0,27	0,26	0	0,28	0,26	0	0,075	0,025	0,005
	1000	0,29	0,29	0	0,29	0,29	0	0,32	0,25	0,01
	2000	0,31	0,3	0	0,31	0,29	0	0,29	0,2	0,02
	3000	0,32	0,31	0	0,31	0,3	0	0,27	0,21	0,02

Figure 5.16: Circles classified by SVM, K-NN and NN-MLP

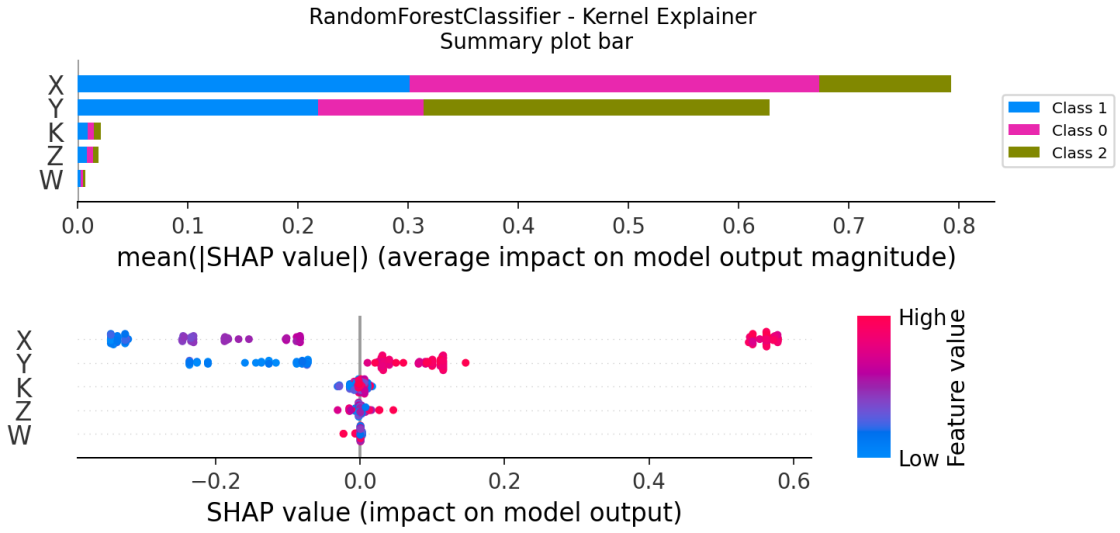
5.1.4 Explanations

This section is focused on the explanation analysis of Section 5.1.3 results, taking care of individuating the driving features for a cluster attribution. The SHAP explainers Kernel, Sampling and BruteForce have similar, almost identical behavior, however even if the Tree Explainer follows the trend of the other explainers it stresses less certain differences (basically all the features tend to be more relevant, almost reaching to equalize the X with the Y).

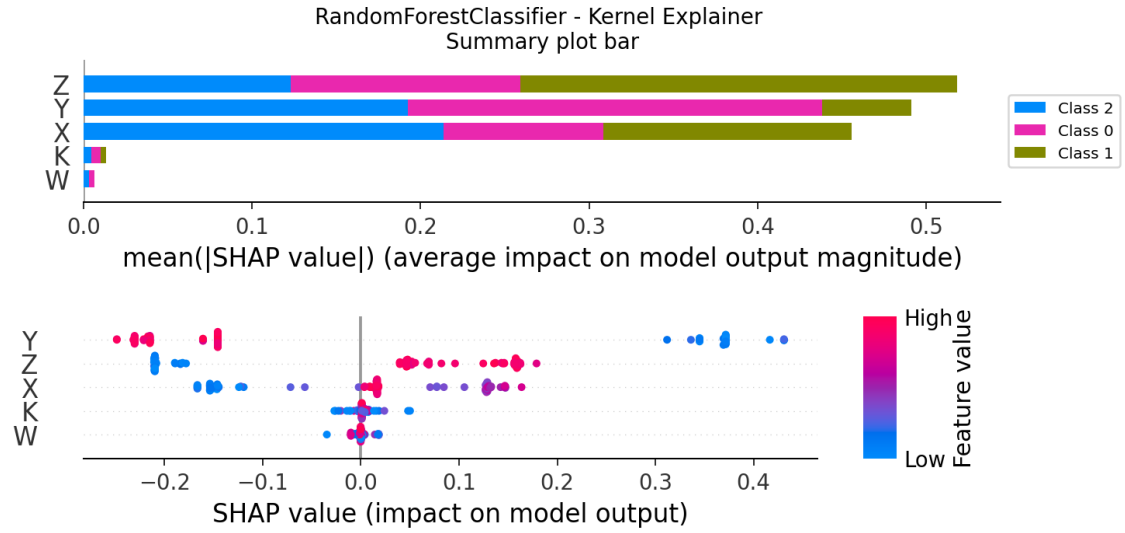
2D Blobs and 3D Blobs

The aim of this comparison is to understand the influence given by a different domain of a feature. The main features expected for 2D Blobs are the X and the Y, while for 3D Blobs it is also important the feature Z.
The SHAP's explanations find Y as most relevant feature and are almost completely in agreement on all the tested classifiers. For Random Forest and NN-MLP the difference on the X is so little that in some cases it becomes more relevant than the Y, in particular on small datasets. The explanations resulting from the 3D Blobs are more oscillating on identifying the most relevant feature, this is due to the nature of the features Y and Z, that are designed with the same criteria, as already described in the datasets description's section. Only the tree-based classifiers give a little weight to the X, instead of zero. A focused observation needs to be done on the Random Forest results, that, differently from the other classifiers, alternates the X to the Y as main feature, instead of the Z. In fact for this classifier the X alternates on the first and third place in the importance's rank, while the Z alternates the second and the third place. The described behaviour depends on the role given to the Y, because when it is considered as main feature, the X is always the third one, while when the X is the main feature the Z is always the third in importance. The results on the 2DBlobs dataset are more coherent for Tree explainer, which almost always indicates the X as main feature for Random Forest, while for Decision Tree the more relevant one is Y. For the 3D Blobs data set, as the other explainers, the Decision Tree returns Z and Y as the two main features, instead the Random Forest explanations stress an higher weight for the Y, alternating the X and the Z as second one. Note that the weights difference is smaller for the *TreeExplainer*.

Figure 5.17 shows two graphs for each of the two datasets. The first



(a) 2D Blobs



(b) 3D Blobs

Figure 5.17: Global weight comparison

graph, on the top, provides the singles feature's contributions for each cluster, highlighting the global importance that each feature has on the dataset and also on single cluster. The second graph of each pair provides the feature importance on a single cluster, in these case it refers to Cluster 0. The importance weights are expressed as the mean of the

SHAP values, differently from the first graph, where the impact is computed as mean of the absolute SHAP values. Another relevant difference is that provides also the trend on the feature's value, for example in Figure 5.17a an high value of both X and Y gives a positive importance score to identify the Cluster 0, the yellow cluster in Figure 5.1. An interesting observation can be done on the 3D Blobs's graphs in Figure 5.17b. In this case the feature's imporance order for Cluster 0 differs from the global model one. In fact, it considers Y more important than Z to identify the Cluster 0, while considering the whole dataset, the feature Z is slightly more important than Y to distinguish all the clusters. From the feature importance function, provided by the Random Forest, can be observed that on 2D Blobs dataset the function fully agrees on all the experiments, that recognise the feature X as first in importance and the Y as second. It states the same feature importance given by the *TreeExplainer*, but partially disagrees with the other SHAP's explainers that invert the Y and X's importance roles on increasing dimensions. It disagrees on all the Spectral's evaluations. For the 3D blobs dataset the explainers strongly disagree with the features importance of Random Forest, in fact, the latter, almost always considers feature X as the main, alternating it in a few cases with the Y and indicating the Z as 3rd in importance. The SHAP's explainers usually considers the Y as the most important, alternating with the X which, however, turns out to be the 3rd, on almost half of the tests, alternating with the Z. So the SHAP's explainers are able to find a certain confirmation between the features Y and Z, which, as the data set is designed, they can clearly identify one of the three clusters. The same results are extracted from the permutation importance's methods, as is shown in Figure 5.18. From Figure 5.19 we can understand the impact

Weight	Feature
0.4840 ± 0.1009	X
0.3060 ± 0.0515	Y
0 ± 0.0000	K
0 ± 0.0000	W
0 ± 0.0000	Z

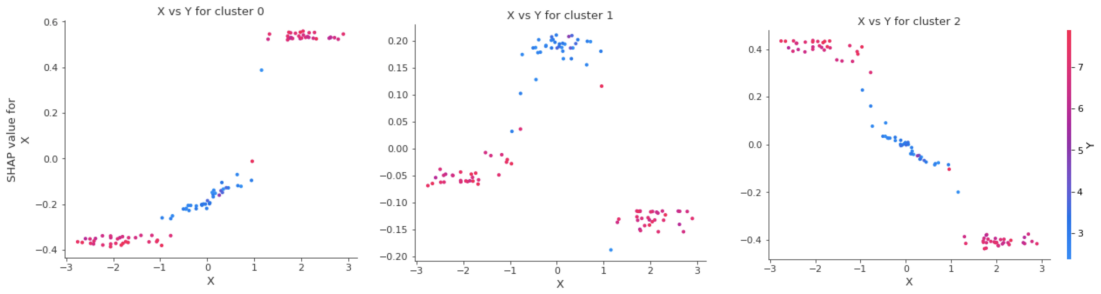
(a) Weights on 2D Blobs.

Weight	Feature
0.3420 ± 0.0753	Z
0.3180 ± 0.1031	Y
0.2080 ± 0.0774	X
0 ± 0.0000	K
0 ± 0.0000	W

(b) Weights on 3D Blobs.

Figure 5.18: Eli5 Permutation Importance on KMeans clusters with Random Forest

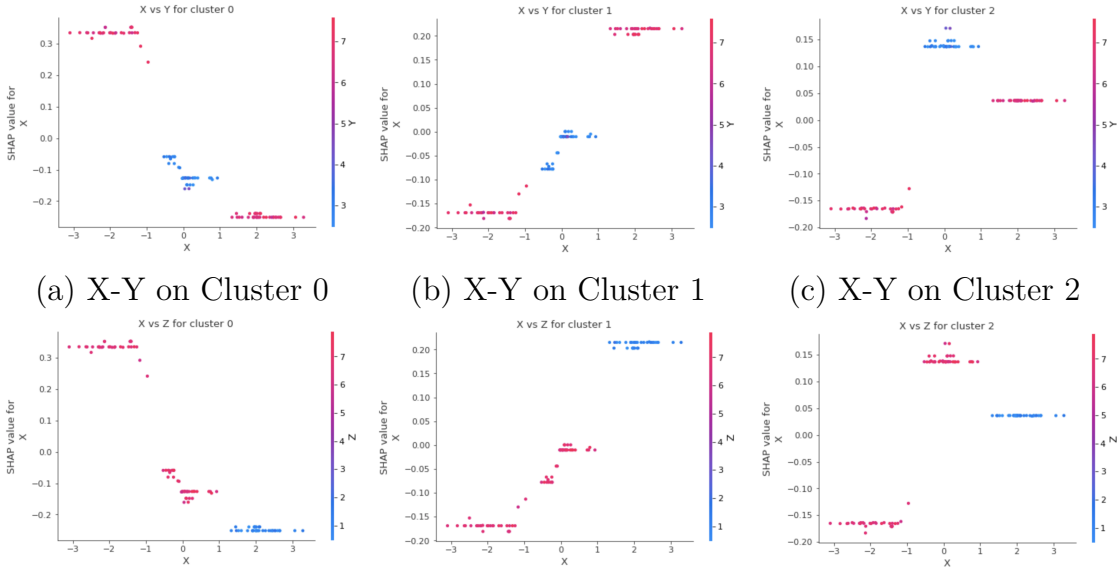
of the feature X, when combined with the Y, on the prediction of single cluster, noting its correlation on the X's range of the three clusters. In



(a) Cluster 0 (b) Cluster 1 (c) Cluster 2

Figure 5.19: 2D Blobs's dependence plot on X and Y

Figure 5.20, instead, we can observe how the equal philosophy for the Y and Z samples generation is reflected on their dependencies, which are the same.



(a) X-Y on Cluster 0 (b) X-Y on Cluster 1 (c) X-Y on Cluster 2

(d) X-Z on Cluster 0 (e) X-Z on Cluster 1 (f) X-Z on Cluster 2

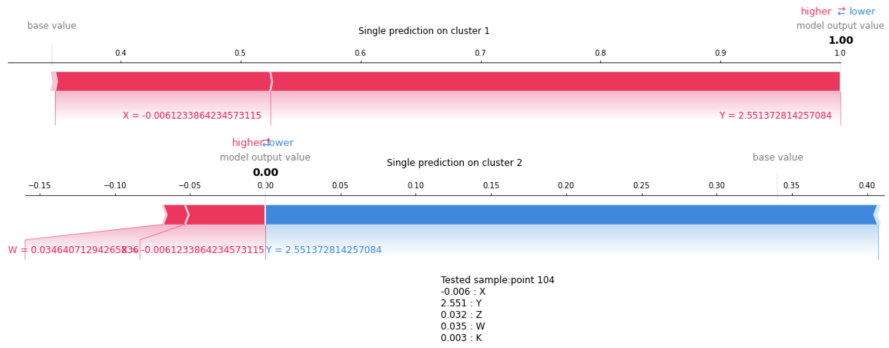
Figure 5.20: 3D Blobs's dependence plot comparison on X-Y and X-Z

Single sample's explanations

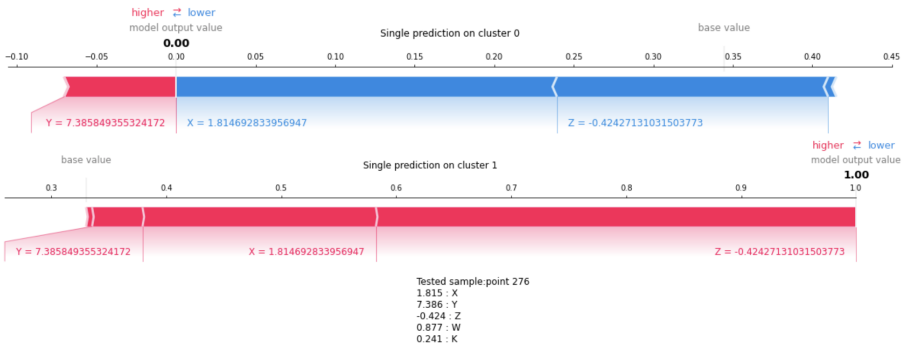
The knowledge that can be extracted from global importance characterizes the clusters, but does not provide any certainty on singles samples characteristics and their variability. For these more strict observations, both SHAP and LIME, as already said in SHAP's Section 3.2.4, provides function for single sample prediction's explanation.

SHAP

Each cluster has its own dependencies, as already observed in the global weights plot made by singles cluster contributions, but these dependencies can be better understood with singles explanation as in Figure 5.21. In fact Figure 5.21a stresses the importance of Y, which is predominant, especially for clusters 0 and 1 as shown in Figure 5.17a. In Figure 5.21b is interesting observing a higher importance given to the feature X for Cluster 1 and in particular for Cluster 0, respect to its 3rd position in the rank of global importance.



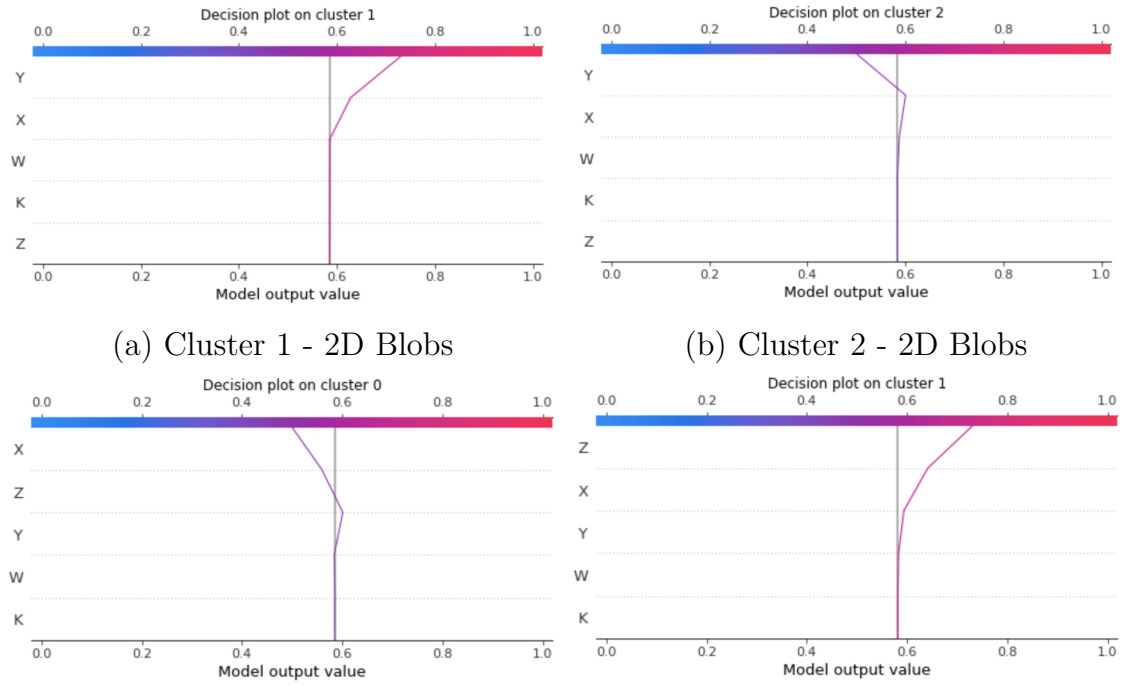
(a) Explained sample from 2D Blobs



(b) Explained sample from 3D Blobs

Figure 5.21: SHAP's single explanation

To get an idea of the model inner workings, the decision plot shows how models make decisions and how a single feature can drive them [76]. The contribution given by X is evident in Figure 5.22, where it is always present as one of the first two features that drive the prediction.



(a) Cluster 1 - 2D Blobs

(b) Cluster 2 - 2D Blobs

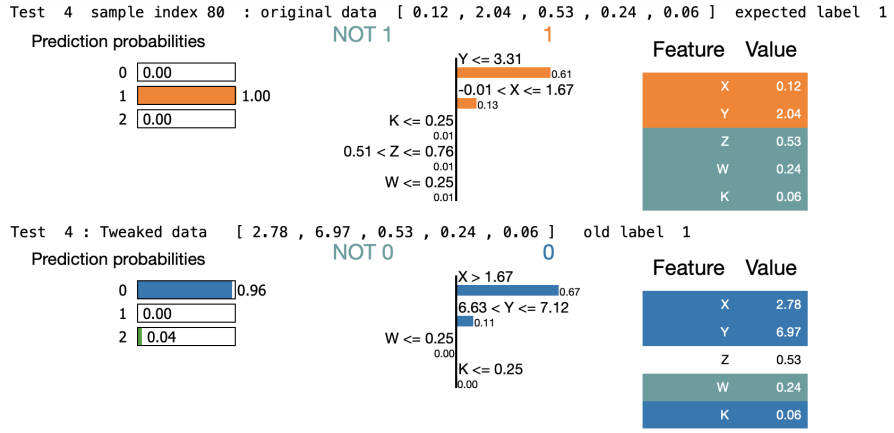
(c) Cluster 0 - 3D Blobs

(d) Cluster 1 - 3D Blobs

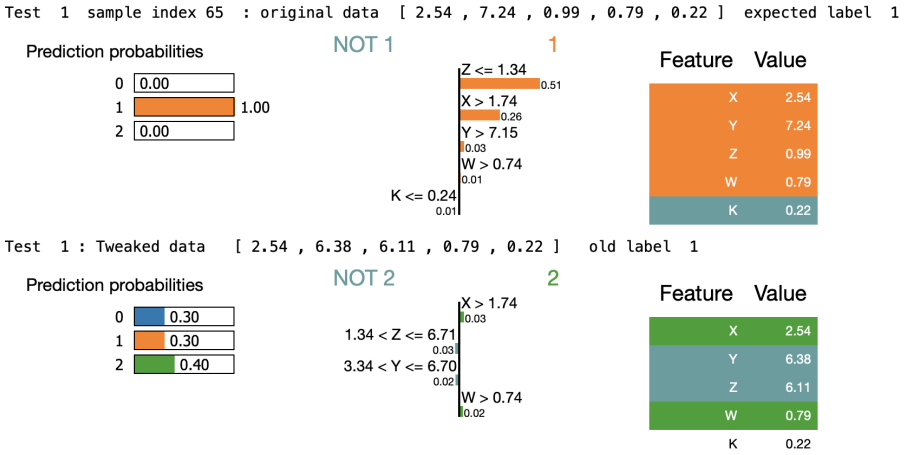
Figure 5.22: Decision plots for single prediction

LIME

Considering local explanations provided by LIME, the modification on most characterizing features increases the chance of different prediction result, respect to the original sample one, even if in some cases lead to a uncertain prediction, as shown in Figure 5.23.



(a) LIME on 2D Blobs

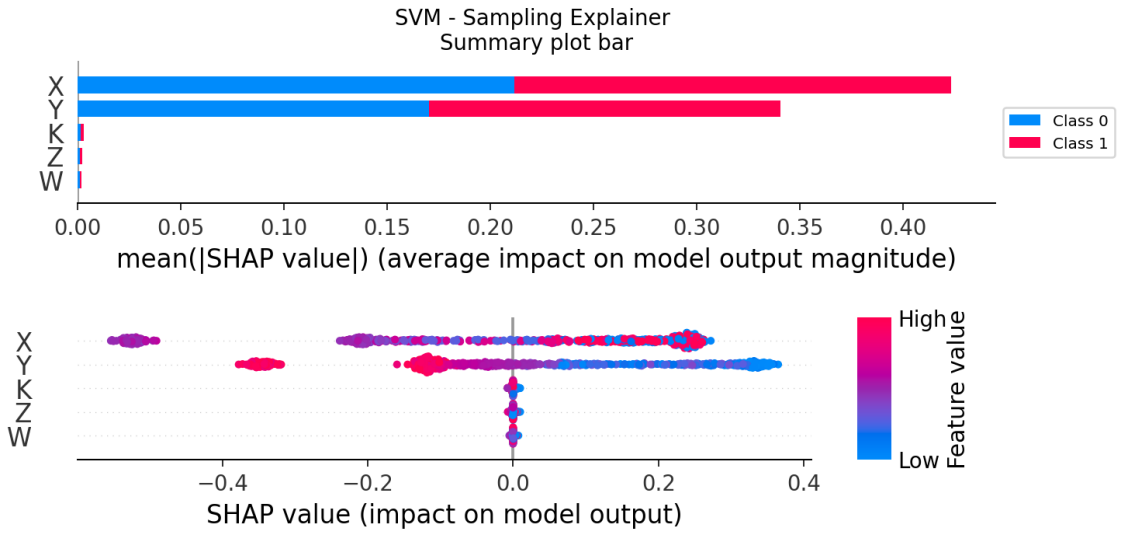


(b) LIME on 3D Blobs

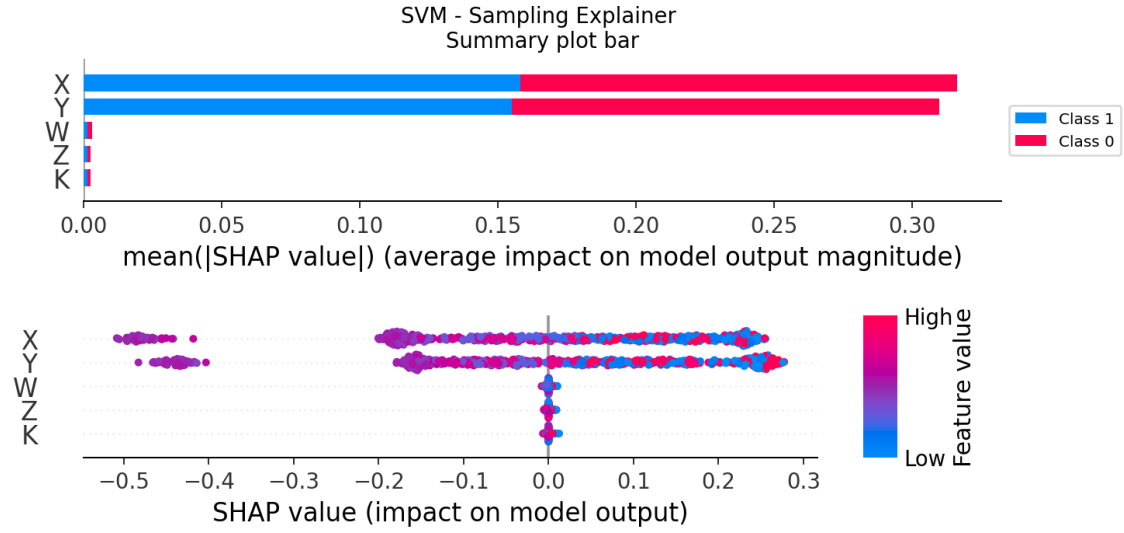
Figure 5.23: Local explanations

Moons and Circles

From this couple of dataset the comparison is focused only on the two main features, X and Y, taking also in observation the weight's distance on the third main feature. It is important to point out that for wrong generated clusters, the weights are conditioned from the clusters boundaries. In fact, considering a classic XY cartesian plane, when there is a vertical boundary, it increases the weight on the feature X, instead when it is horizontal, a bigger weight is given to the feature Y. In case of diagonal boundaries, it depends on slope, but tends to get a more similar weights on X and Y when it is a 45 degree's slope. This behaviour is observed on the datasets clustered with the algorithms Agglomerative and K-Means, that have vertical boundaries for the Moons datasets and diagonals for Circle ones. The OPTICS algorithm gives correct results for datasets with at least 2000 samples, instead the DBSCAN and the Spectral always get correct clusters. The SVM, KNN and NN-MLP classifiers agrees on giving an higher importance value to the feature X, respect to the Y, with a slightly difference for SVM and KNN, in particular on the Spectral 1000 samples results, where their weights are equal. The explanation of the tree based models are less certain on giving more importance to the feature X, in particular for Moons datasets on DBSCAN, where the difference is little but for Random Forest is always in advantage of X, while for the Decision Tree has an alternated attribution on the two features. An interesting difference on this two methods is that the Decision Tree almost always gives a zero value also from the third feature weight, stressing more the difference between the first two features. The explanation, on Moons dataset, with *TreeExplainer* agrees for the Random Forest experiments, stressing more the importance of the Y, being in disagree only few times, when the two main weights are almost equal. For the Decision Tree result, instead, the X is always the main feature, also when other explainers consider Y as first feature, even if with a little advantage on the second one. For the Circles data set, is interesting to note how the Random Forest explanations hardly point the feature X as main one according with other explainers, even if the Decision Tree completely disagree, taking the feature Y as most relevant. In Figure 5.24 is possible to observe the described behaviour, extracted by Shape, on clusters given by Spectral algorithm and learned the classification with a SVM. Also with the function feature importance for the Moons dataset, where the clusters



(a) Moons



(b) Circles

Figure 5.24: Global weight comparison on Spectral clusters

are identified as designed, the X is always the main feature, even if with a little difference on the Y. This keep the feature importance function in a divergent position respect to the SHAP one. Also for the Circle dataset, like for the Moons one, the X is often considered as the main feature, but respect to it the feature Y acquires more importance. So in this case the shape of external cluster shows an influence on the score of the features.

The permutation importance's results, shown in Figure 5.25, confirm that the weights of X and Y are more similar in Moons dataset.

Weight	Feature
0.2240 ± 0.0105	X
0.1897 ± 0.0311	Y
0 ± 0.0000	K
0 ± 0.0000	W
0 ± 0.0000	Z

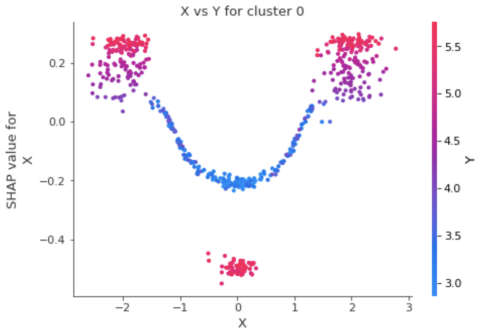
(a) Weights on Moons.

Weight	Feature
0.1977 ± 0.0248	X
0.1830 ± 0.0122	Y
0 ± 0.0000	K
0 ± 0.0000	W
0 ± 0.0000	Z

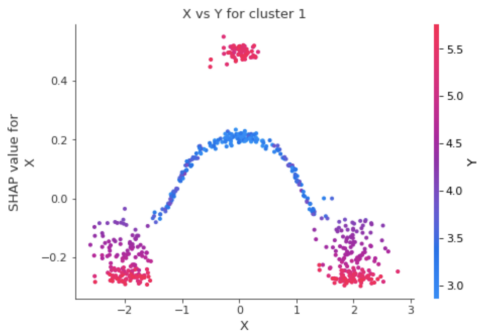
(b) Weights on Circles.

Figure 5.25: Eli5 Permutation Importance on Spectral clusters with SVM

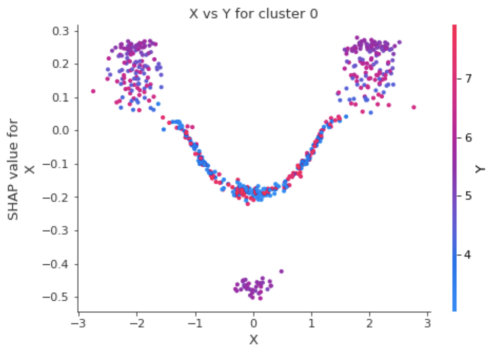
The dependence plots in Figure 5.26 are particularly interesting because, especially for Moons, is stressed the correlation on the value of Y, in fact the central part of the X range is composed of two distinct group of Y's values, the blue one that represents low values of Y and the red one that point the Y's values related also to the internal cluster.



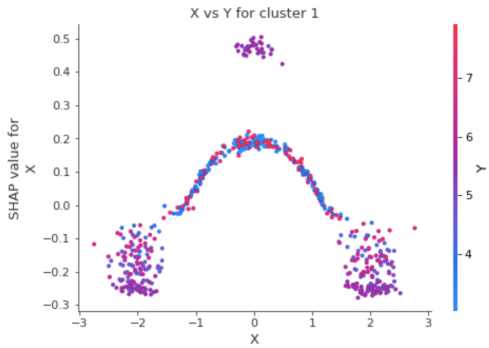
(a) Cluster 0 - Moons



(b) Cluster 1 - Moons



(c) Cluster 0 - Circles



(d) Cluster 1 - Circles

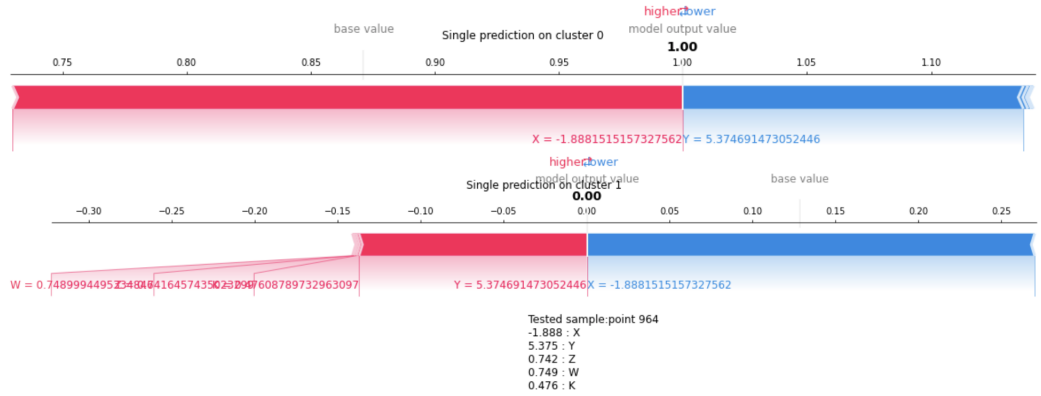
Figure 5.26: Dependence plot comparison on X-Y

Single sample's explanations

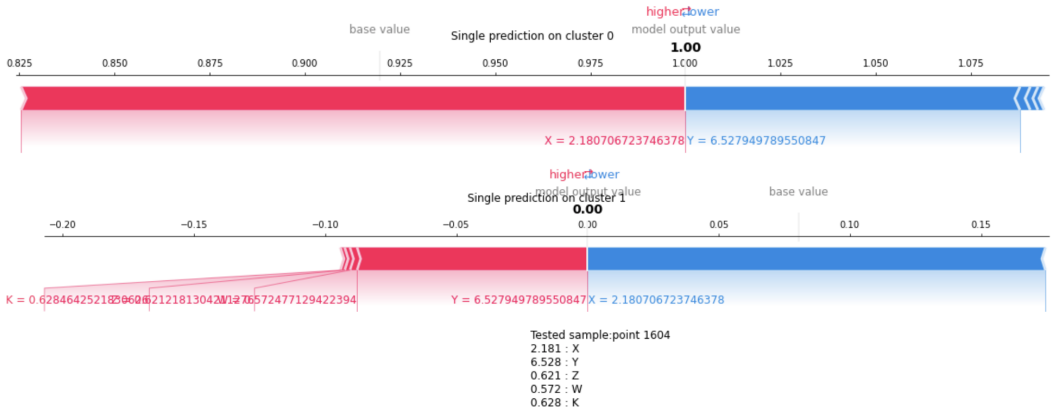
For this couple of dataset is considered the experimental setting with Spectral Clustering and SVM.

SHAP

From the single explanation in Figure 5.27 and the correspondent decision plot of Figure 5.28, extracted from Sampling Explainer, is not possible to note such difference between the features X and Y of the two datasets, observed for the global weights.

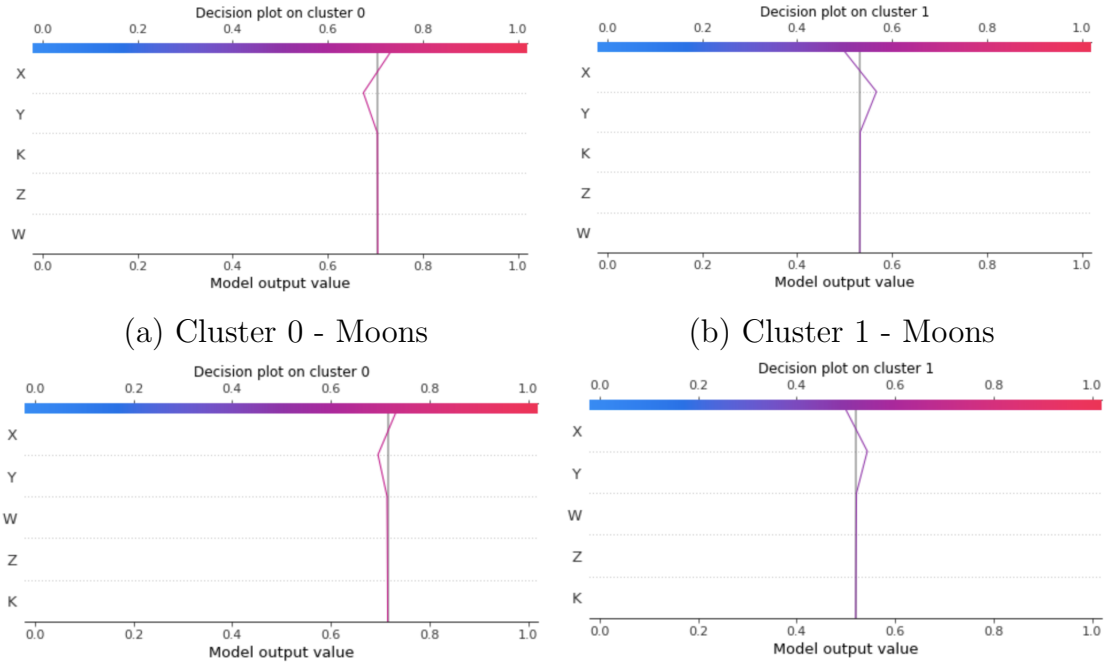


(a) Explained sample from Moons



(b) Explained sample from Circles

Figure 5.27: SHAP's single explanation



(a) Cluster 0 - Moons

(b) Cluster 1 - Moons

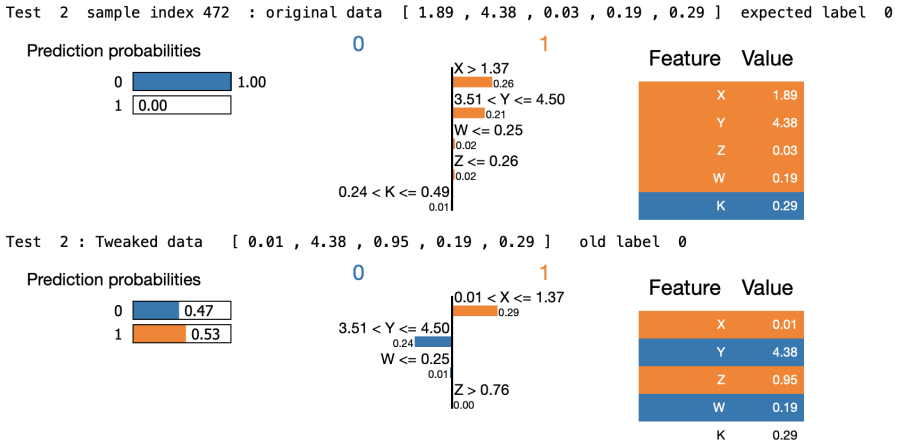
(c) Cluster 0 - Circles

(d) Cluster 1 - Circles

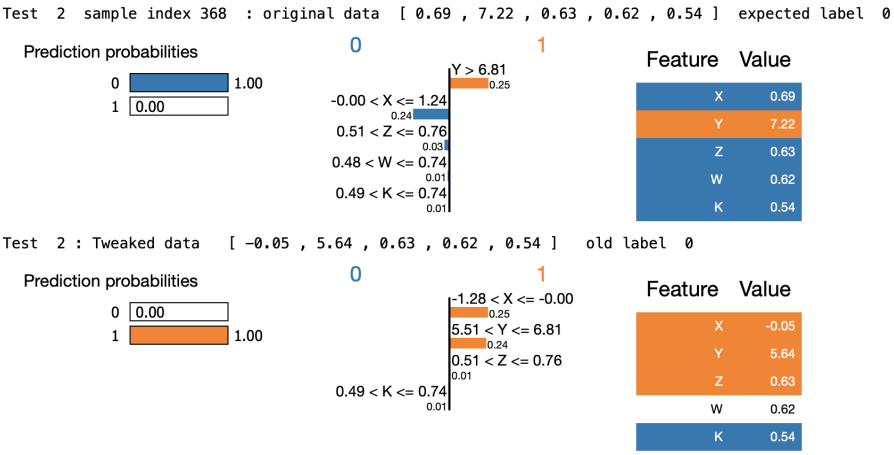
Figure 5.28: Decision plots for single prediction

LIME

The local explanations provided in Figure 5.29 shows that tweak on an important feature could also create indecision in the prediction of the model, perhaps identifying its boundaries with other clusters, as for Figure 5.29a, while for Figure 5.29b the tweak on both the two main features gives a more stable prediction.



(a) LIME on Moons



(b) LIME on Circles

Figure 5.29: Local explanations

5.2 Real datasets experiments

The purpose of this section is to verify how good are the methods used for artificial datasets and the validity of the explainer's results.
The first dataset of this section is Iris [69], well known in the literature and one of the most used to test new methodologies on a dataset with few samples and few features. The second dataset is Mall customers [77], and it relies on the customer segmentation, a way to better profile the customers. As the Iris, also this dataset has a low number of features, just five. Given the size of the first two dataset, a third one is chosen with the purpose of testing a dataset with a higher feature dimension. To do so, the dataset Adult is taken in analysis.

5.2.1 Dataset description

Iris

The Iris flower dataset was introduced by Ronald Fisher in his paper [69] in 1936, and it is one of the most famous and used dataset in literature, especially for classification tasks. The dataset contains morphological variation on Iris flowers of three species, which are Setosa, Virginica and Versicolor. Each species is represented with 50 samples, containing the measures of the four features "Petal length", "Petal width", "Sepal length" and "Sepal width". Because of its values, is not common for cluster analysis, in fact the dataset contains only two separated clusters, one representing the Iris Setosa while the other contains both Iris Virginica and Iris Versicolor. Thanks to those characteristics, the Iris dataset is a good example for differentiating the supervised techniques from the unsupervised ones.
Figure 5.30 shows the real classified samples, considering a pair of measure at each block. The red dots are the samples from Setosa class, while the green ones are from Versicolor and the blue dots are the samples of Virginica specie.

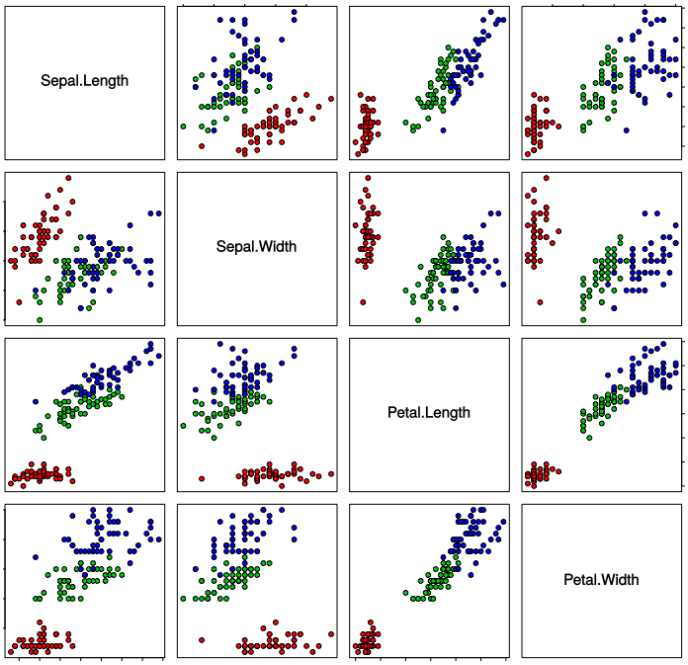


Figure 5.30: Scatter plot of original Iris dataset.

Mall customers

The mall customers is a dataset used in literature for the customer segmentation [77]. The customer segmentation is a practice similar to the clustering one, in fact the intent is to define groups of customer, understanding their common characteristics. This results reflect on the interaction to have with a specific cluster of customer, with the aim of maximizing the business.
The dataset is composed by 200 samples of 5 features each and does not provide any label. For that lack of labels, it s necessary to tune the number of clusters to identify, which is an important parameters for some clustering algorithms. (For an exhaustive investigation, the optimal number of clusters was also searched for the Iris dataset, which was two, as expected due to the strong similarity of the two species Virginica and Versicolor. But knowing that the number of real clusters is three it was decided to use it as a parameter value.) The metric chosen to identify the number of clusters is the Silhouette, which is often used in literature to evaluate dataset when a ground truth labels is not known [42, 43]. The Silhouette coefficient is calculated using the mean intra-cluster distance and the mean

nearest-cluster distance for each sample. The best Silhouette score is 1, which means the cluster's sample are better defined, while the worst is -1. A zero's value, instead, gives a warning on the cluster overlapping [43].

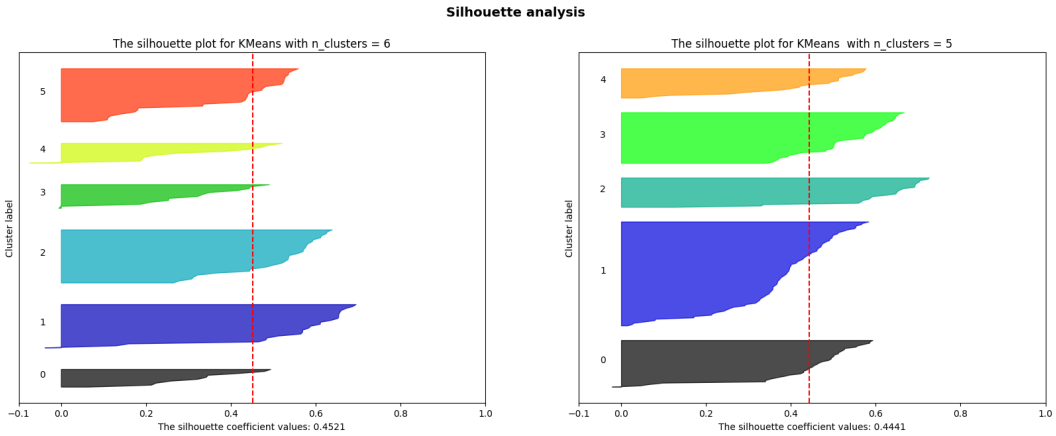
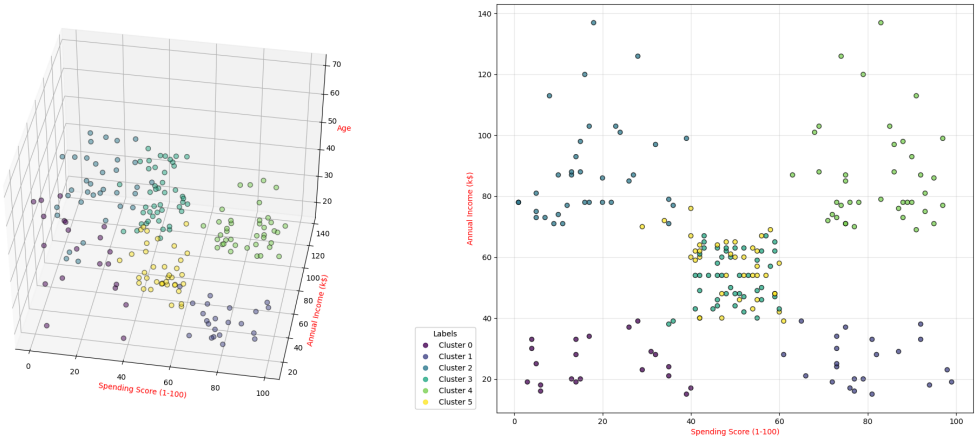


Figure 5.31: Example of tuning for the number of clusters

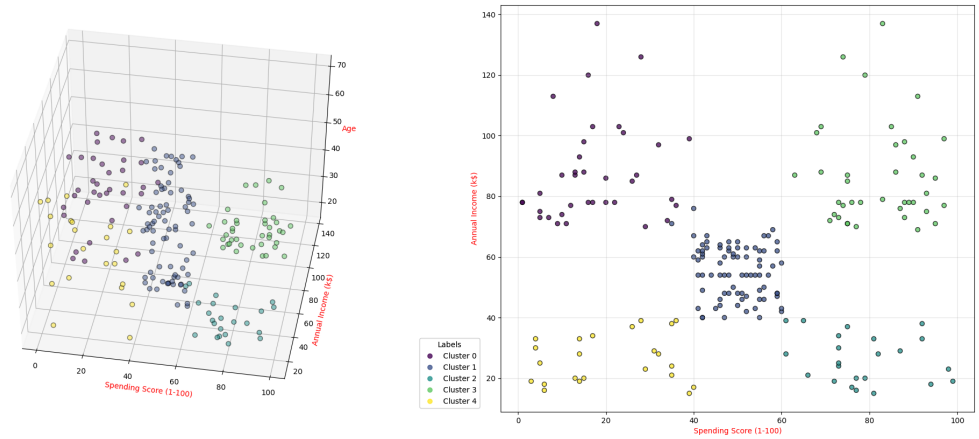
As shown in Figure 5.31 the difference between the two silhouette score is so little that also 5 can be a correct value for the parameter "number of cluster". To better understand which one can be more appropriated, I've analyzed combinations of features in pairs, and, thanks to this pair plot, come out a pair of feature where the cluster can be more distinct. From Figure 5.32 is possible to compare the clusters extracted for the two setting, having also a better idea on their quality. Even if the silhouette is greater with a number of cluster equal to 6, for a parameter value of 5, the extracted clusters do not overlap, which also means a better inter-cluster distinction. This observation leads me to consider 5 the real number of dataset's clusters.

Framingham Heart Study

This dataset is from the Framingham Heart Study, which is a large-scale clinical study. This study is one of the most famous for research into heart disease risk, to the point of giving its name to a risk score. Most of the framingham risk score calculators (as [78] or [79]) are based on some of the features present in the framingham study dataset, of which, for this work, it is used a subset of 3658 samples of 16 features each [80]. Framingham dataset is chosen to have an experiment over a higher number



(a) Pair plot for 6 clusters



(b) Pair plot for 5 clusters

Figure 5.32: Comparison between 6 and 5 cluster's result

of features, respect to the previous ones, analysing so how time scales over more complex dataset. The goal of the Framingham risk factor is to identify patients at high risk of developing a heart attack or coronary artery disease in the next 10 years.

5.2.2 Experimental settings

Clustering algorithms configurations

For Iris and Mall customer datasets, the experimental settings of Kmeans, Spectral and Agglomerative are the same used for the artificial dataset

experiments, while for density based algorithms the tuning of parameters has been omitted due to the complexity in finding parameters capable of correctly identifying the clusters known from the literature.

Classification algorithms configurations

The settings of classification algorithms used for the artificial dataset experiments is valid also for experiments on the considered real datasets, so it was not necessary to modify them.

5.2.3 Experimental results

This section provides the results for real datasets, for each one there are three parts. The first part is for Random Forest results (5.33, 5.36), the second one is for Decision Tree results (5.34, 5.37), while the third one is for the results of Support Vector Machine, K-Nearest Neighbors and Neural Network Multi Layer Perceptron classifiers (5.35, 5.38). For some configurations the explainer was not able to extract the importance weights, in these cases no value was reported, replaced by a **-** symbol.

Importance weights extracted on IRIS

	Feature	Agglomerative	DBSCAN	KMeans	OPTICS	Spectral
SHAP explainers Brute - Kernel - Sampling	Petal Width	0,45	0,5	0,44	0,5	0,47
	Petal Length	0,66	0,39	0,68	0,35	0,72
	Sepal Width	0,06	0,01	0,04	0,01	0,02
	Sepal Length	0,17	0,02	0,16	0,03	0,12
Tree Explainer	Petal Width	0,46	0,36	0,46	0,3	0,47
	Petal Length	0,62	0,33	0,62	0,31	0,62
	Sepal Width	0,05	0,12	0,04	0,2	0,04
	Sepal Length	0,25	0,19	0,2	0,39	0,24
Feature Importance	Petal Width	0,3	0,45	0,39	0,45	0,39
	Petal Length	0,49	0,35	0,4	0,33	0,42
	Sepal Width	0,03	0,01	0,03	0,01	0,04
	Sepal Length	0,17	0,19	0,17	0,22	0,15

Figure 5.33: Importance weights by Random Forest

	Feature	Agglomerative	DBSCAN	KMeans	OPTICS	Spectral
SHAP explainers Brute - Kernel - Sampling	Petal Width	0,1	0,9	0,1	0,28	0,1
	Petal Length	1,2	0,03	1,2	0,31	1,2
	Sepal Width	0	0	0	0,21	0
	Sepal Length	0,05	0	0	0,39	0
Tree Explainer	Petal Width	0,85	-	0	-	0,87
	Petal Length	0,56	-	1,2	-	0,6
	Sepal Width	0	-	0	-	0
	Sepal Length	0	-	0	-	0

Figure 5.34: Importance weights by Decision Tree

	Feature	Agglomerative	DBSCAN	KMeans	OPTICS	Spectral
SVM	Petal Width	0,22	0,2	0,2	0,26	0,2
	Petal Length	0,81	0,57	0,8	0,4	0,86
	Sepal Width	0,05	0,05	0,05	0,14	0,05
	Sepal Length	0,18	0,1	0,18	0,265	0,17
K-NN	Petal Width	0,17	0,01	0,1	0,42	0,1
	Petal Length	1,1	0,9	1,05	0,68	1,1
	Sepal Width	0,05	0,01	0,02	0,28	0,02
	Sepal Length	0,2	0,01	0,2	0,54	0,19
NN-MLP	Petal Width	0	0,33	0	0	0
	Petal Length	0	0,5	0	0,44	0
	Sepal Width	0	0,12	0	0	0
	Sepal Length	0	0,03	0	0	0

Figure 5.35: IRIS classified by SVM, K-NN and NN-MLP

Importance weights extracted on Mall Customers

	Feature	Agglomerative	DBSCAN	KMeans	OPTICS	Spectral
SHAP explainers Brute - Kernel - Sampling	Spending Score	0,82	-	0,83	-	1
	Annual Income	0,73	-	0,85	-	0,63
	Age	0,18	-	0,15	-	0,1
	Gender	0	-	0	-	0
Tree Explainer	Spending Score	0,82	-	0,8	-	1
	Annual Income	0,74	-	0,9	-	0,65
	Age	0,2	-	0,18	-	0,1
	Gender	0	-	0,01	-	0
Feature Importance	Spending Score	0,46	-	0,44	-	0,53
	Annual Income	0,45	-	0,47	-	0,38
	Age	0,08	-	0,08	-	0,09
	Gender	0	-	0	-	0

Figure 5.36: Importance weights by Random Forest

	Feature	Agglomerative	DBSCAN	KMeans	OPTICS	Spectral
SHAP explainers Brute - Kernel - Sampling	Spending Score	0,59	-	0,64	-	1,13
	Annual Income	1,1	-	1,25	-	0,7
	Age	0,02	-	0	-	0
	Gender	0	-	0	-	0
Tree Explainer	Spending Score	0,59	-	0,64	-	1,2
	Annual Income	1,18	-	1,3	-	0,65
	Age	0,01	-	0	-	0
	Gender	0	-	0	-	0

Figure 5.37: Importance weights by Decision Tree

	Feature	Agglomerative	DBSCAN	KMeans	OPTICS	Spectral
SVM	Spending Score	0,8	-	0,79	-	0,86
	Annual Income	0,7	-	0,72	-	0,66
	Age	0,08	-	0,06	-	0,07
	Gender	0	-	0	-	0
K-NN	Spending Score	0,92	-	0,93	-	1
	Annual Income	0,76	-	0,8	-	0,71
	Age	0,11	-	0,1	-	0,1
	Gender	0	-	0	-	0
NN-MLP	Spending Score	-	-	-	-	-
	Annual Income	-	-	-	-	-
	Age	-	-	-	-	-
	Gender	-	-	-	-	-

Figure 5.38: Mall Customers classified by SVM, K-NN and NN-MLP

Importance weights extracted on Framingham

KMeans	Kernel Explainer				Sampling Explainer			
	1st feature	2nd feature	3rd feature	4th feature	1st feature	2nd feature	3rd feature	4th feature
RandomForest	totChol (0,58)	sysBP (0,09)	prev.Hyp (0,085)	heartRate (0,08)	totChol (0,85)	heartRate (0,07)	Age (0,06)	sysB(0,06)
DecisionTree	totChol (1,0)	diaBP(0,01)	-	-	totChol (1,0)	diaBP (0,02)	-	-
SVM	totChol (0,9)	CigsPerDay(0,02)	Glucose (0,01)	heartRate (0,01)	totChol (0,9)	CigsPerDay(0,02)	Glucose (0,01)	heartRate (0,01)
KNN	totChol (0,77)	CigsPerDay(0,14)	Glucose (0,06)	heartRate (0,03)	totChol (0,78)	CigsPerDay(0,14)	Glucose (0,07)	sysBP (0,03)
NNMLP	-	-	-	-	-	-	-	-

Spectral	Kernel Explainer				Sampling Explainer			
	1st feature	2nd feature	3rd feature	4th feature	1st feature	2nd feature	3rd feature	4th feature
RandomForest	totChol (0,54)	Age (0,08)	sysBP (0,065)	Glucose(0,055)	totChol (0,53)	Age (0,09)	Glucose (0,08)	Male(0,05)
DecisionTree	totChol (0,83)	diaBP (0,06)	Glucose (0,04)	heartRate (0,01)	totChol (0,85)	diaBP(0,04)	Glucose (0,02)	heartRate(0,01)
SVM	totChol (0,82)	sysBP (0,04)	Glucose (0,001)	Age (0,001)	totChol (0,8)	sysBP (0,04)	Glucose (0,03)	Age (0,02)
KNN	totChol (0,78)	sysBP (0,04)	Glucose (0,03)	CigsPerDay(0,01)	totChol (0,77)	sysBP (0,02)	Age (0,015)	Glucose (0,01)
NNMLP	-	-	-	-	-	-	-	-

Agglomerative	Kernel Explainer				Sampling Explainer			
	1st feature	2nd feature	3rd feature	4th feature	1st feature	2nd feature	3rd feature	4th feature
RandomForest	totChol (0,4)	sysBP (0,1)	BMI (0,09)	diaBP (0,075)	totChol (0,4)	sysBP(0,1)	BMI (0,09)	diaBP(0,07)
DecisionTree	totChol (0,83)	sysBP (0,15)	diaBP(0,11)	Age(0,10)	totChol (0,85)	sysBP(0,18)	diaBP(0,16)	Age(0,1)
SVM	sysBP (0,33)	totChol (0,24)	diaBP(0,08)	Age(0,05)	sysBP (0,33)	totChol (0,24)	diaBP(0,08)	Age(0,05)
KNN	sysBP (0,27)	totChol (0,19)	Age (0,17)	CigsPerDay(0,16)	sysBP (0,24)	totChol (0,22)	Age (0,17)	CigsPerDay(0,17)
NNMLP	-	-	-	-	-	-	-	-

	Tree Explainer				
	1st feature	2nd feature	3rd feature	4th feature	
RandomForest	totChol (0,95)	sysBP (0,09)	Age (0,08)	BMI (0,07)	KMeans
DecisionTree	totChol (0,8)	heartRate(0,02)	sysBP (0,01)	-	
RandomForest	totChol (0,49)	Age (0,08)	Glucose (0,07)	diaBP(0,065)	Spectral
DecisionTree	totChol(0,083)	diaBP(0,04)	Glucose(0,02)	heartRate(0,01)	
RandomForest	totChol (0,39)	sysBP (0,17)	Age (0,11)	BMI (0,1)	Agglomerative
DecisionTree	totChol (0,84)	sysBP (0,17)	diaBP (0,12)	Age(0,09)	

(a) Results on 300 samples.

KMeans	Kernel Explainer				Sampling Explainer			
	1st feature	2nd feature	3rd feature	4th feature	1st feature	2nd feature	3rd feature	4th feature
RandomForest	totChol (0,6)	Age (0,09)	sysBP (0,01)	BMI(0,01)	totChol (0,6)	Age (0,1)	BMI (0,01)	sysBP (0,01)
DecisionTree	totChol (0,81)	sysBP (0,01)	-	-	totChol (0,79)	sysBP (0,03)	BMI (0,005)	-
SVM	totChol (0,81)	sysBP (0,01)	-	-	totChol (0,81)	sysBP (0,01)	Glucose (0,01)	heartRate (0,01)
KNN	totChol (0,8)	sysBP (0,01)	-	-	totChol (0,8)	Age (0,02)	heartRate (0,01)	Glucose (0,01)
NNMLP	-	-	-	-	-	-	-	-

Spectral	Kernel Explainer				Sampling Explainer			
	1st feature	2nd feature	3rd feature	4th feature	1st feature	2nd feature	3rd feature	4th feature
RandomForest	totChol (0,8)	Age (0,05)	diaBP (0,02)	sysBP(0,01)	totChol (0,8)	Age (0,05)	sysBP (0,02)	BMI (0,015)
DecisionTree	totChol (1,1)	sysBP (0,01)	-	-	totChol (1,1)	heartRate (0,01)	sysBP (0,01)	-
SVM	totChol (1,1)	sysBP (0,02)	heartRate (0,01)	Glucose (0,001)	totChol (1,0)	sysBP (0,04)	heartRate(0,03)	BMI (0,001)
KNN	totChol (1,0)	sysBP (0,05)	heartRate (0,03)	Age (0,01)	totChol (1,0)	CigsPerDay(0,04)	sysBP (0,04)	heartRate(0,01)
NNMLP	-	-	-	-	-	-	-	-

Agglomerative	Kernel Explainer				Sampling Explainer			
	1st feature	2nd feature	3rd feature	4th feature	1st feature	2nd feature	3rd feature	4th feature
RandomForest	sysBP (0,48)	totChol (0,3)	Age (0,11)	diaBP (0,05)	sysBP (0,49)	totChol (0,3)	Age (0,13)	diaBP (0,05)
DecisionTree	sysBP (0,62)	totChol (0,26)	Age (0,1)	CigsPerDay(0,05)	sysBP (0,63)	totChol (0,26)	Age (0,1)	diaBP (0,04)
SVM	sysBP (0,55)	totChol (0,32)	Age (0,1)	diaBP (0,03)	sysBP (0,52)	totChol (0,31)	Age (0,1)	diaBP (0,035)
KNN	sysBP (0,67)	totChol (0,26)	Age (0,08)	CigsPerDay(0,04)	sysBP (0,7)	totChol (0,26)	Age (0,06)	CigsPerDay(0,04)
NNMLP	-	-	-	-	-	-	-	-

	Tree Explainer				
	1st feature	2nd feature	3rd feature	4th feature	
RandomForest	totChol (0,62)	Age (0,11)	sysBP (0,04)	heartRate (0,04)	KMeans
DecisionTree	totChol (0,8)	sysBP (0,02)	Glucose (0,001)	BMI (0,001)	
RandomForest	totChol (0,7)	Age (0,1)	sysBP (0,04)	heartRate (0,03)	Spectral
DecisionTree	totChol (1,1)	sysBP (0,03)	heartRate (0,01)	diaBP (0,01)	
RandomForest	sysBP (0,52)	totChol (0,28)	Age (0,12)	Male (0,04)	Agglomerative
DecisionTree	sysBP (0,72)	totChol (0,24)	Age (0,06)	CigsPerDay(0,04)	

(b) Results on 3000 samples.

Figure 5.39: Framingham importance weights

5.2.4 Explanations

Iris

Almost all of the experiments detect Petal Length as most important feature, followed by the Petal Width. For both Random Forest and Decision Tree the explanations given on the DBSCAN and OPTICS, disagree this rank of relevance, in particular because these two density based algorithms are the only to detect 2 clusters instead of 3. *TreeExplainer*'s explanations for the Random Forest agree on the main importance of Petal Length, while for the Decision Tree the explanations over Spectral and Agglomerative algorithm's results, take Petal Width as most relevant. Also the Random Forest's feature importance function agrees on the importance of these two feature, even if with a slightly difference between them. The Neural Network's experiments failed on this dataset, returning a valid result only for the DBSCAN's run, that confirms the Petal Length as main feature. The results described next refer to the experiment with Spectral

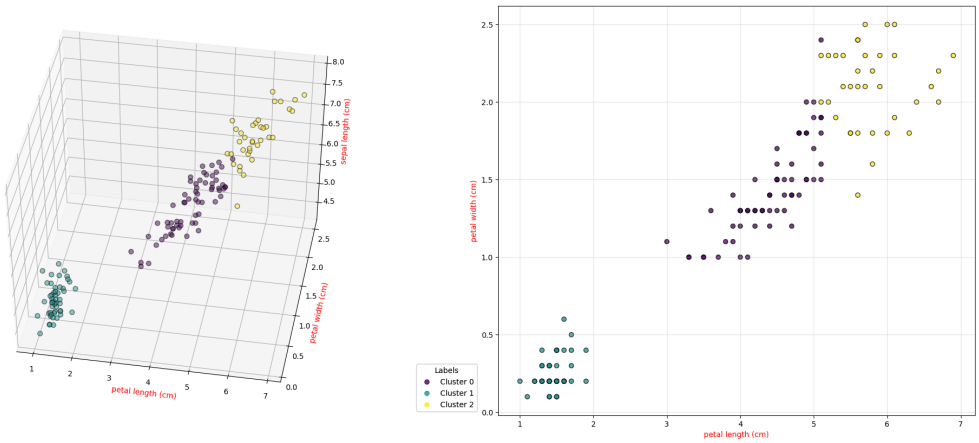


Figure 5.40: Partial feature representation of IRIS dataset.

Clustering and SVM. The obtained clusters, shown in Figure 5.40, when learned by a SVM, provides interesting global results, as the ones given by Kernel Explainer displayed in Figure 5.41. As can be easily note, the length of the petal, as already said, is clearly the most important feature, but is interesting to point out the difference between the second and third feature, that is little also for the permutation importance results of Figure 5.42. To better understand the difference with the Petal length is useful to observe the dependence plots shown in Figure 5.43 and Figure 5.44, that

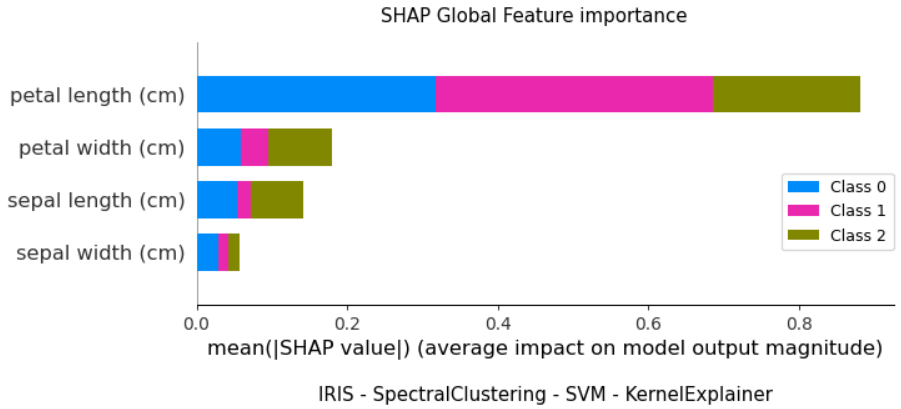


Figure 5.41: Global weights extracted by Kernel Explainer.

```
sepal length (cm) scores:       Mean: 0.1197    Max: 0.2333     Min: 0.0333
sepal width (cm) scores:        Mean: 0.0097    Max: 0.0667     Min: -0.0333
petal length (cm) scores:       Mean: 0.4927    Max: 0.7333     Min: 0.3000
petal width (cm) scores:        Mean: 0.1310    Max: 0.2333     Min: -0.0333
```

(a) Scikit-learn weights.

Weight	Feature
0.5200 ± 0.1373	petal length (cm)
0.1533 ± 0.0680	petal width (cm)
0.1000 ± 0.0730	sepal length (cm)
0.0067 ± 0.0267	sepal width (cm)

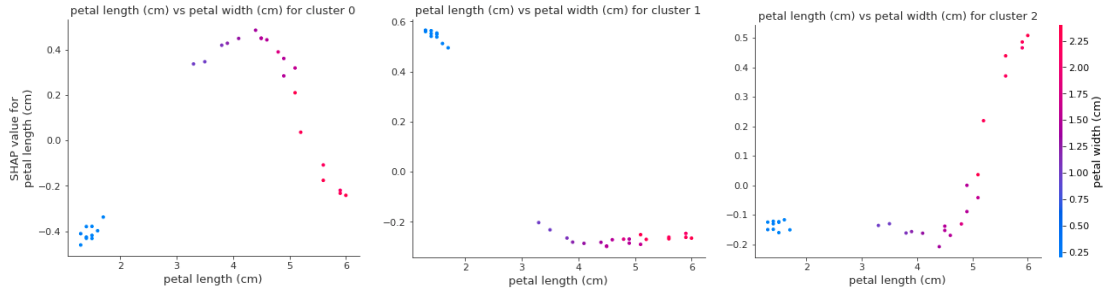
(b) Eli5 weights.

Figure 5.42: Permutation Importance on Spectral clusters with SVM

highlight a similar trend when considered the petal length's value. This observation not only gives an idea of the petal length's relevance but also suggests how the sepal width and the petal width can be interchangeable, resulting similarly weighted.

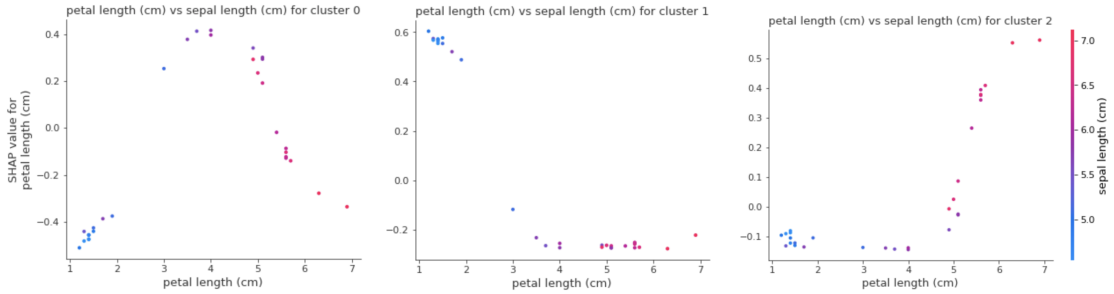
Single sample's explanations

As for the previous datasets, also for IRIS dataset is run a test on singles samples, randomly chosen from the test set, to extract explanations for single sample's predictions.



(a) Cluster 0 (b) Cluster 1 (c) Cluster 2

Figure 5.43: Dependence plot on petal length and petal width



(a) Cluster 0 (b) Cluster 1 (c) Cluster 2

Figure 5.44: Dependence plot on petal length and sepal width

SHAP

The single explanation reflects the base of knowledge learned on the whole dataset, as can be observed in Figure 5.45. It is clear the relevance of petal length over all the clusters, but is interesting also to note the maintained weight rapport between petal width and sepal length, especially for cluster 2. Even from Figure 5.46 we can observe these behaviours, with also a more clear visualization for each cluster's feature rank.

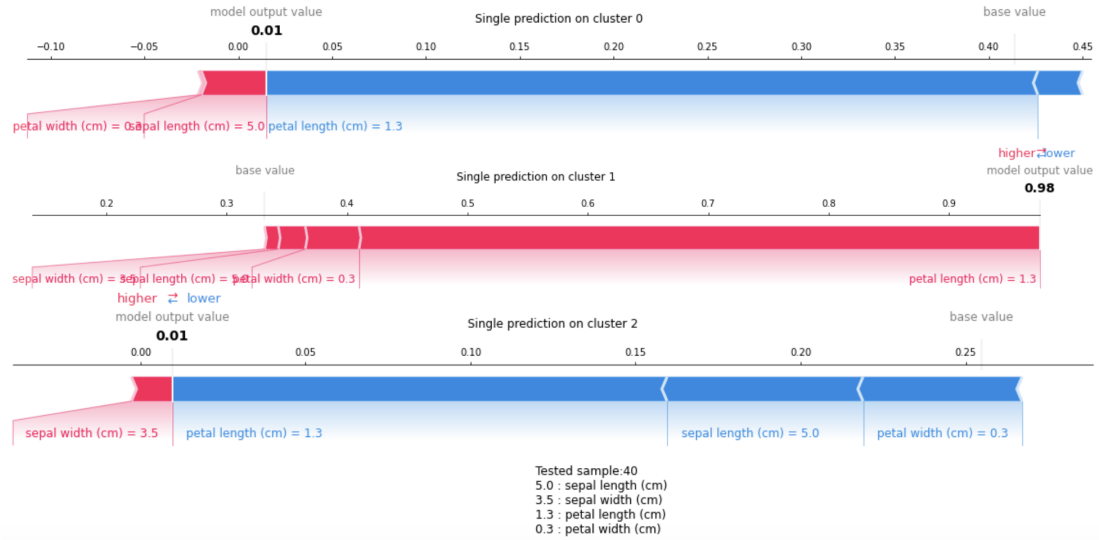
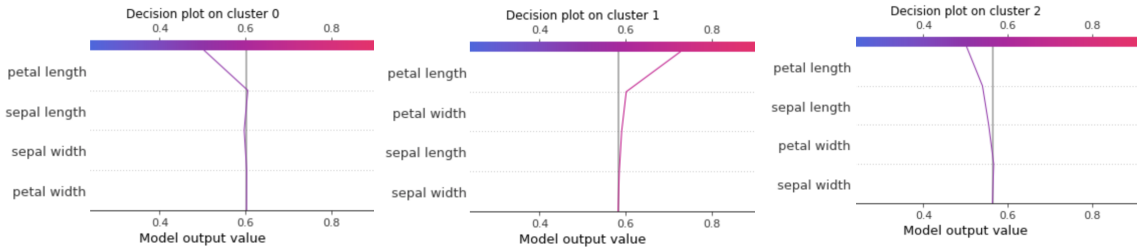


Figure 5.45: Single explanation for Iris sample extracted by SHAP



(a) Cluster 0 (b) Cluster 1 (c) Cluster 2

Figure 5.46: Decision plots for single prediction

LIME

From the local explanations of Figure 5.47, with LIME is also possible to understand the influence of a value on the feature's impact over a cluster's prediction. The performed tweak is inspired by the SHAP's one, in which a random sample is considered as source of the modified data, which replace the original ones, in order to extract the new prediction explanation. The two couples of prediction show how the feature's impact differs when the two most important feature are modified. In fact, for the first couple, the feature have a collaborative influence to drive the prediction of both the original sample and the modified one. In the second couple, the features has a contrasting influence, with the petal width in favour of predicting the sample as belonging to cluster 1 and the petal length in disagree.

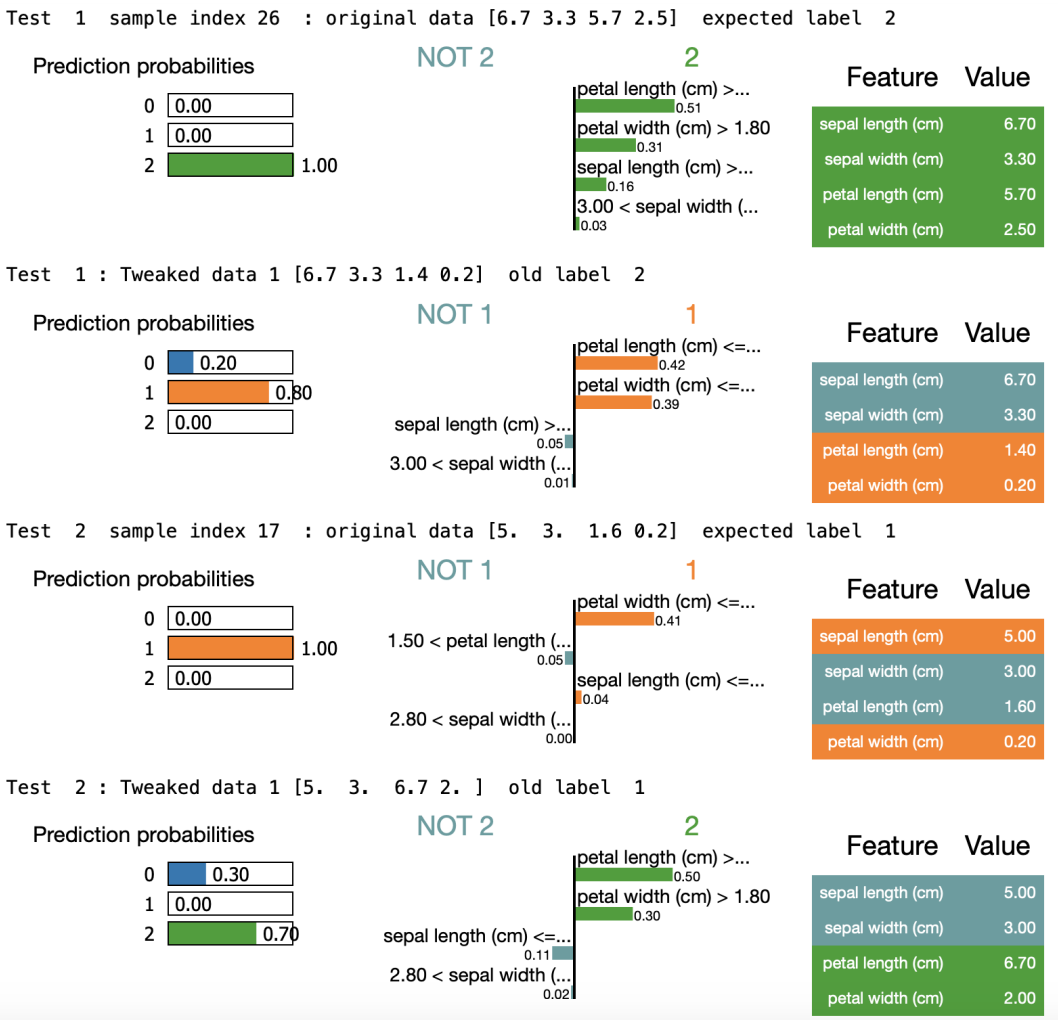


Figure 5.47: LIME explanations on single feature's tweak

Mall customers

This dataset underlined the importance of tuning properly the clustering algorithms, also as a way to preliminary understand which algorithm could be inadequate. In fact, for this dataset, it is possible to obtain meaningful results only for some algorithms, the ones that are not density based. The results obtained for the remaining clustering algorithms, agrees on considering "Spending Score" and "Annual Income" as most relevant features. This is a results that is coherent with the considerations done on figure. An interesting observation is on the difference of weight that are given to this features, that for the Random Forest's function, the so called

feature importance, is smaller than the SHAP's. In most of the combination of clustering and classification algorithms, SHAP gives a slightly higher importance to the "Spending Score" attribute. Only the Decision Tree gives more importance to the "Annual Income" feature, in particular for the clusters obtained with the Agglomerative algorithm, where it doubles the weight respect to the "Spending Score" one. From Figure 5.48 it

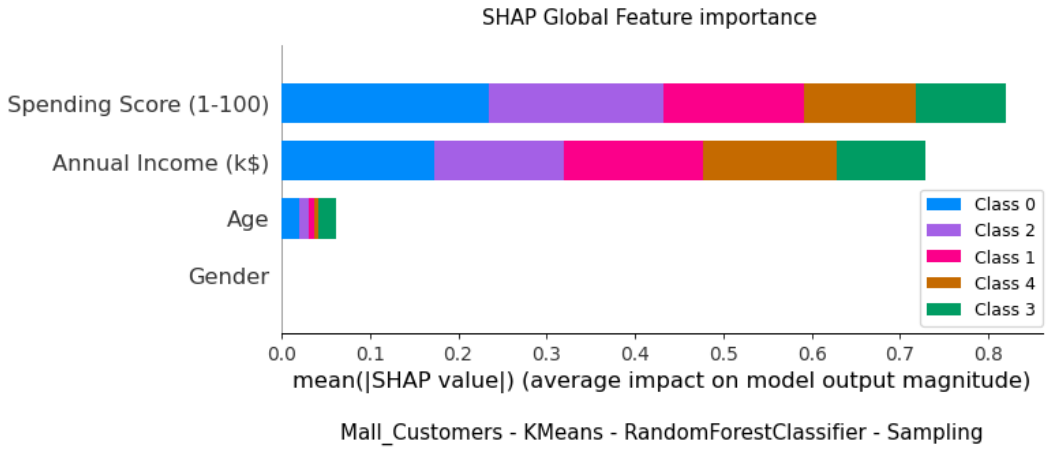


Figure 5.48: Shap summary plot of a Sampling Explainer on KMeans clusters and Random Forest

is possible to get an insight on how feature's weight changes considering each cluster. For Class 0 the feature Spending Score is more important than Annual Income, instead for Class 4 the Annual Income has an higher specific weight respect to the Spending Score one. Not only SHAP's result finds this global insight, also the permutation importance results agree on giving an higher importance to the Spending Score, even if it is almost the same of Annual Income. An interesting observation, from Figure 5.49, is on the Age weight, that gains importance respect to the SHAP one. Some interesting insights can be observed putting the focus on what characterizes a specific feature weight. As represented by SHAP's dependence plot in Figure 5.50, which shows the SHAP value of Spending Score, considering also the value of Annual Income of the associated sample, for cluster 1 is important that the Spending Score belongs to the range of 40 to 60, that usually means that the Annual Income is around 40/70 k$.

```
Scores for Gender
        Mean: 0.0
        Max: 0.0
        Min: 0.0
Scores for Age
        Mean: 0.27925
        Max: 0.42499
        Min: 0.17499
Scores for Annual Income (k$)
        Mean: 0.43474
        Max: 0.625
        Min: 0.275
Scores for Spending Score (1-100)
        Mean: 0.47725
        Max: 0.625
        Min: 0.32499
```

(a) Scikit-learn weights

Weight	Feature
0.4650 ± 0.0510	Spending Score (1-100)
0.4500 ± 0.1304	Annual Income (k$)
0.2550 ± 0.0970	Age
0 ± 0.0000	Gender

(b) Eli5 weights

Figure 5.49: Permutation Importance on KMeans clusters with Random Forest

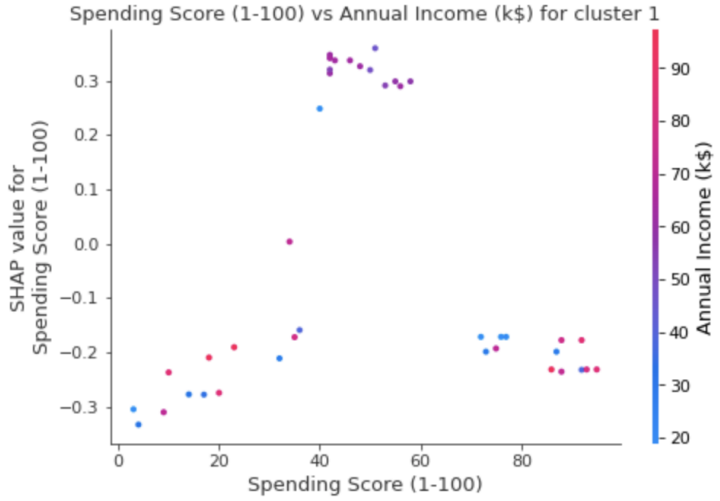


Figure 5.50: SHAP dependence plot made by Sampling Explainer on KMeans clusters and Random Forest

Single sample's explanations

Also for this dataset is run a test for a single sample, randomly taken from the test set, with the aim of explaining single sample's predictions.

SHAP

As reported in Figure 5.51 the chosen sample has a prediction that is strongly driven by the two most relevant feature.The explanation provided for the prediction of cluster 2, stresses the importance of a Spending Score value out of cluster's range, that has so a negative influence for the prediction as a sample of cluster 2, despite of the positive score for the Annual Income's value, that is in favour of that association. The same hint can

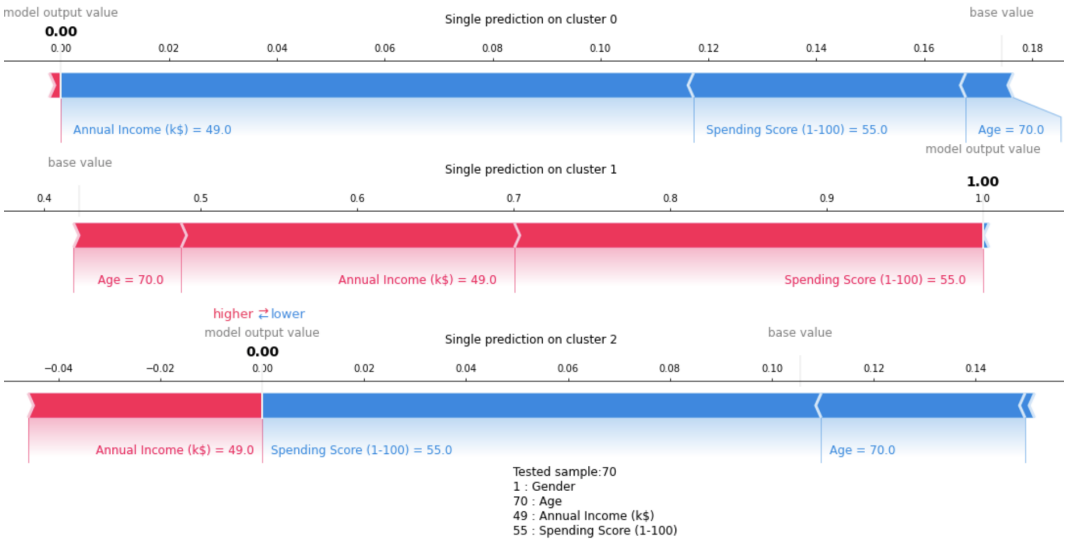


Figure 5.51: Single explanation from BruteForce Explainer

be observed with the decision plot of Figure 5.52, with the cluster 0 conditioned more by Annual Income and the cluster 2 strongly influenced by Spending Score.

(a) Cluster 0 (b) Cluster 1

Figure 5.52: Decision plot of BruteForce Explainer

LIME

Thanks to its local explanation, LIME provides other interesting insight, that more strictly reflects the relevance of a value, extracting its weight, related to a feature's value interval. Also for this sample, are reported an extract of an exhaustive research of sample feature's tweaking.

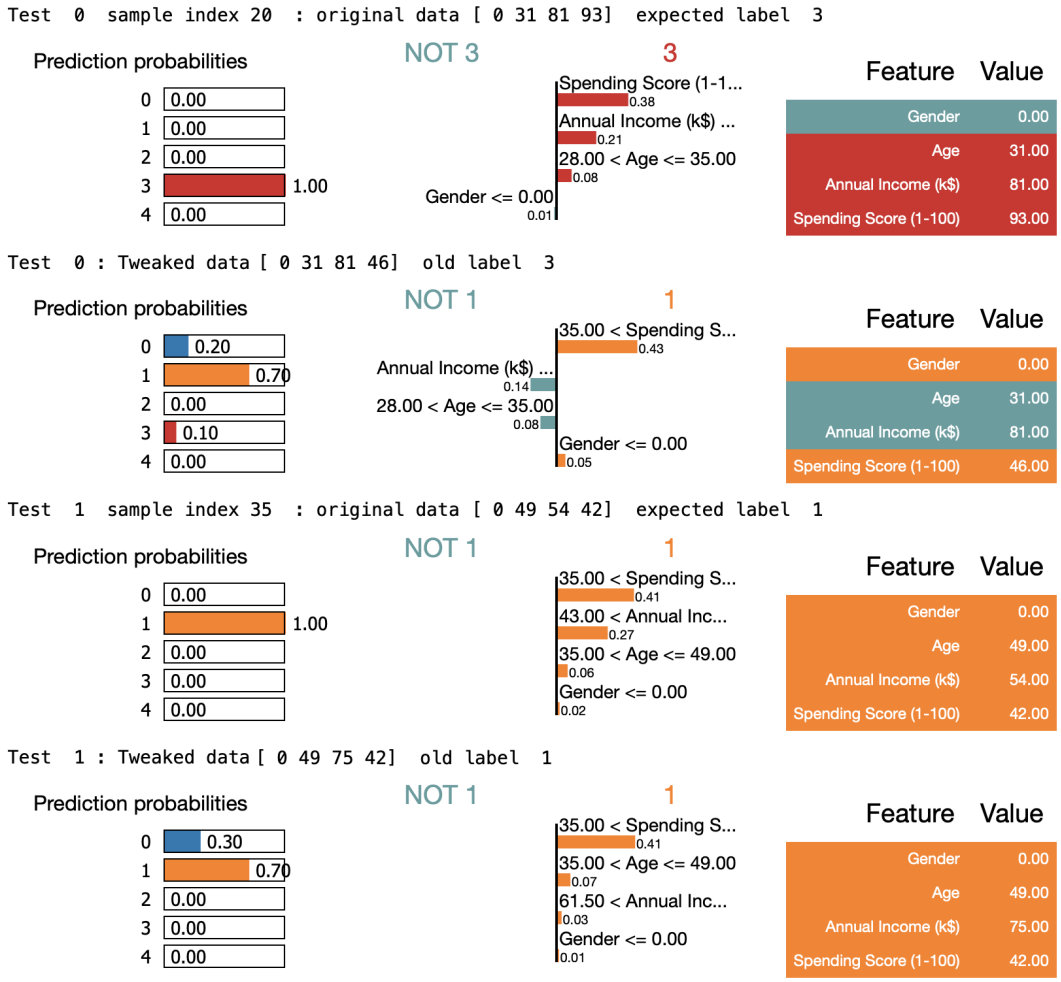


Figure 5.53: LIME explanations on single feature's tweak

In the first test sample of Figure 5.53, an augmented value for Spending Score, radically change the features weights and the associated class. In fact while the original sample is associated to cluster 3, thanks to the positive weights combination of Spending Score and Annual Income, the modified sample gets a less sure association, which is strongly influenced by Spending score, despite the Annual Income and Age would drive to

different associations, with a 20% of probability for a cluster 0 prediction and a 10% probability of association as cluster 1. Also for the second test is tweaked only a feature's value, in that case the Annual Income. The prediction remains the same, though the Annual Income's weight is less supporting the cluster 1 prediction, in favour of cluster 0, which gains prediction's probability from 0% up to 30%. The tests reported in Figure 5.54 are based on a combined tweak of two features, taken from the same random sample source. The features modified for Test 2 are Annual

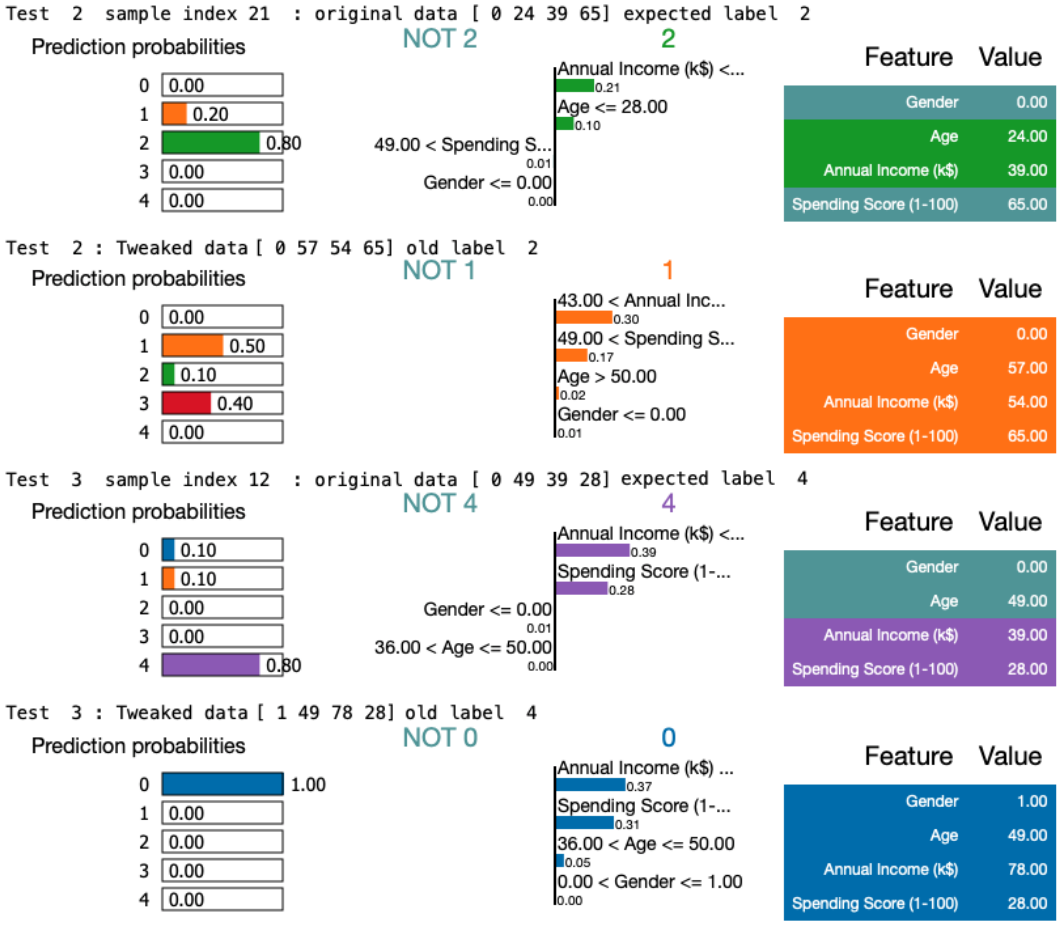


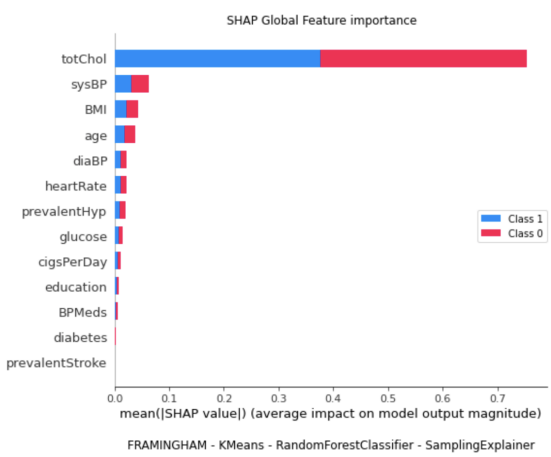
Figure 5.54: LIME explanations on multiple feature's tweak

Income and Age, the two that influence the original sample prediction to cluster 2, but for that modification downgrade the prediction probability of cluster 2 from 80% to 10% , while the one for cluster 1 increases up to 50% and the probability for cluster 2 became 40%. The small difference between this two last probability, highlights how the sample could be

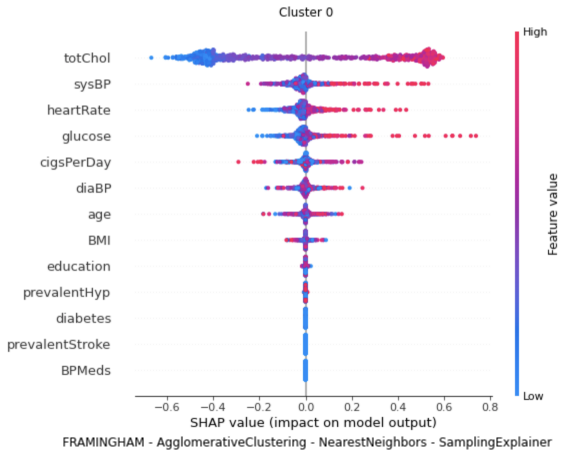
at the edge of two clusters. The last test sees the Gender change, a feature considered not influential and the modification of which does not contribute to the new prediction. In fact, even the prediction has changed, the credit for this is due to the tweak of Annual Income's value.

Framingham Heart Study

Due to the nature of the Framingham study dataset and its clustering results, some experiments were not considered, particularly those involving density-based clustering algorithms. For this dataset are considered two versions, one of 300 samples and one of 3000 samples, which aim to understand the impact of the number of features on the computational time. The classes for this dataset are two: low risk of a coronary artery disease and high risk. The classifiers return mutually agreed results, identifying the first two features for importance in "totChol" and "sysBP". For clusters extracted with the KMeans and Spectral algorithms the most important feature is "totChol", while for clusters extracted with Agglomerative the "sysBP" is the main one, with the exception of the results on 300 samples, obtained with the Random Forest and Decision Tree classifiers, which indicate "totChol" as the main feature. Compared to the 300 samples version, in the 3000 samples version the third feature is more often associated to the feature "Age". The explainers finds that the most relevant features to predict the risk of a coronary heart disease are "totChol", which identifies the total cholesterol of a patient, and "sysBP", that stands for systolic blood pressure and indicates the pressure in the arteries when the heart beats and pumps out blood [81]. Other important features are "Age", "BMI" (Body Mass Index), "Glucose", "CigsPerDay" and "heartRate". The BMI is a metric that considers weight and height of a patient into account to measure body size and defines six obesity class range, from underweight to severe obesity [82].
The following reported analyzes refer to the dataset of 3000 samples. The example in Figure 5.55a shows the global importances extracted by a Sampling Explainer on Agglomerative clusters and Nearest Neighbors. The first feature has a much greater importance than the second, as can also be seen from the results shown in Figure 5.39. As shown in Figure 5.56, the permutation importance's results agree on the first two features but not on the next ones. The insight from the dependencies of the two main features in Figure 5.57, points out how low values for both "totChol" and



(a) Global feature importance



(b) Cluster 0 feature importance

Figure 5.55: Shap summary plot of a Sampling Explainer on Agglomerative clusters and Nearest Neighbors

```
age scores:     Mean: 0.0151    Max: 0.0301     Min: 0.0027
education scores:       Mean: 0.0000    Max: 0.0000     Min: 0.0000
cigsPerDay scores:      Mean: 0.0397    Max: 0.0574     Min: 0.0178
BPMeds scores:  Mean: 0.0000    Max: 0.0000     Min: 0.0000
prevalentStroke scores:         Mean: 0.0000    Max: 0.0000     Min: 0.0000
prevalentHyp scores:    Mean: 0.0000    Max: 0.0000     Min: 0.0000
diabetes scores:        Mean: 0.0000    Max: 0.0000     Min: 0.0000
totChol scores:         Mean: 0.4023    Max: 0.4399     Min: 0.3443
sysBP scores:   Mean: 0.0728    Max: 0.0915     Min: 0.0505
diaBP scores:   Mean: 0.0246    Max: 0.0369     Min: 0.0137
BMI scores:     Mean: 0.0063    Max: 0.0191     Min: -0.0014
heartRate scores:       Mean: 0.0470    Max: 0.0710     Min: 0.0314
glucose scores:         Mean: 0.0546    Max: 0.0724     Min: 0.0369
```

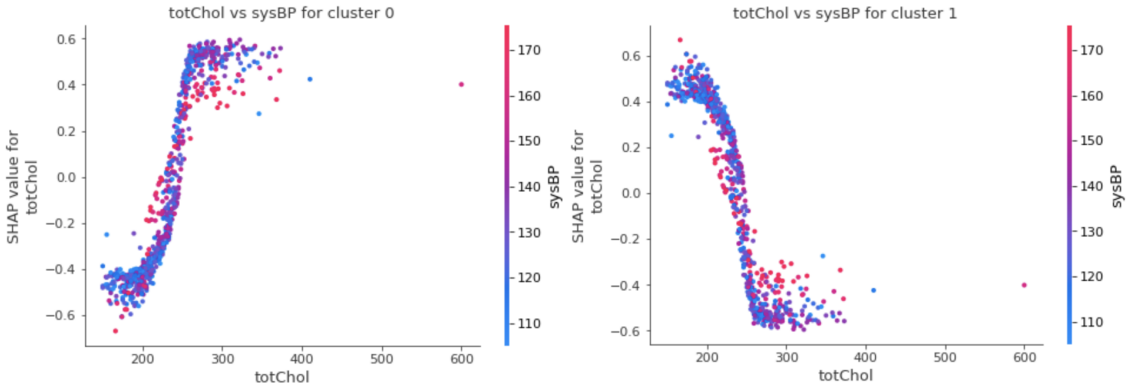
(a) Scikit-learn weights

Weight	Feature
0.3910 ± 0.0196	totChol
0.0730 ± 0.0132	sysBP
0.0568 ± 0.0162	glucose
0.0525 ± 0.0070	heartRate
0.0429 ± 0.0191	cigsPerDay
0.0284 ± 0.0160	diaBP
0.0131 ± 0.0087	age
0.0052 ± 0.0047	BMI
0 ± 0.0000	diabetes
0 ± 0.0000	prevalentHyp
0 ± 0.0000	prevalentStroke
0 ± 0.0000	BPMeds
0 ± 0.0000	education

(b) Eli5 weights

Figure 5.56: Permutation Importance on Agglomerative clusters with Nearest Neighbors

"sysBP" positively influence the association to the cluster 1, that refers to a low risk of coronary artery disease. This observation can also be confirmed by the importance of the features of cluster 0, provided in Figure 5.55b.



(a) Cluster 0 (b) Cluster 1

Figure 5.57: SHAP dependence plot made by Sampling Explainer on Agglomerative clusters and Nearest Neighbors

Single sample's explanations

Also for this dataset is run a test for a single sample, randomly taken from the test set, with the aim of explaining single sample's predictions.

SHAP

Figure 5.58 reports the single prediction for a sample that belongs to the cluster 1. For both the clusters explanations the main feature is "totChol", followed by "sysBP" and "heartRate", showing an evident opposite influence on the cluster's associations. Figure 5.59 highlights the features

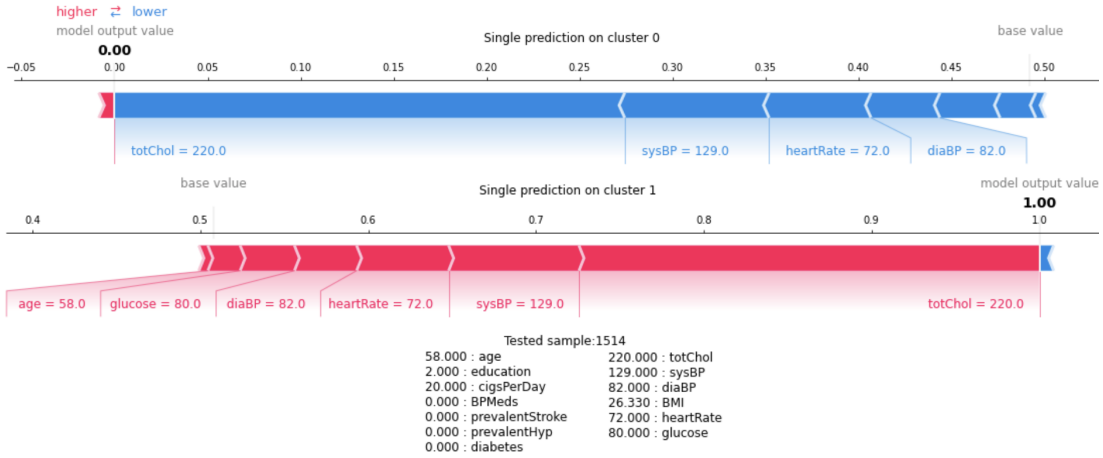
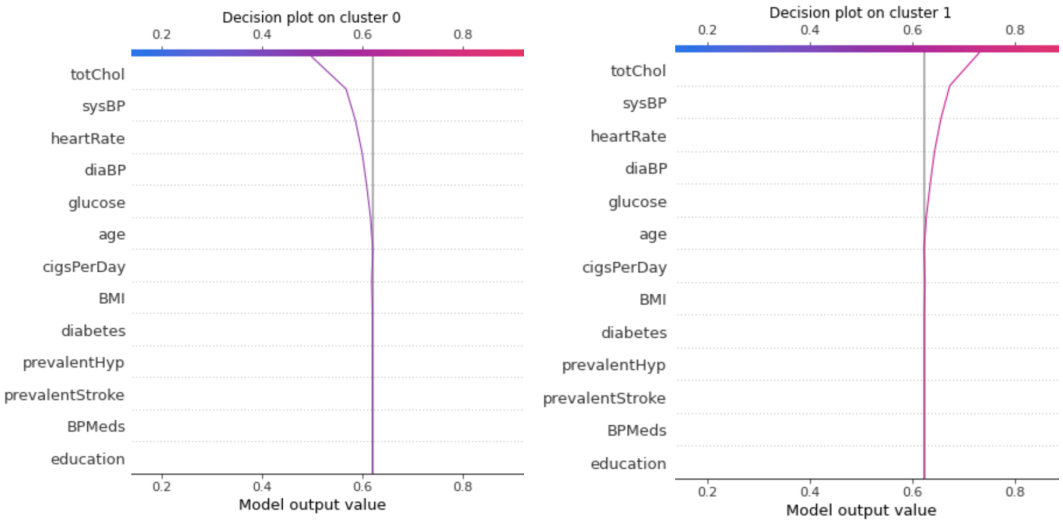


Figure 5.58: Single explanation from Sampling Explainer

that influenced the decision-making process, with a clear impact of the "totChol" value importance, which drives in favour of the association to cluster 1.

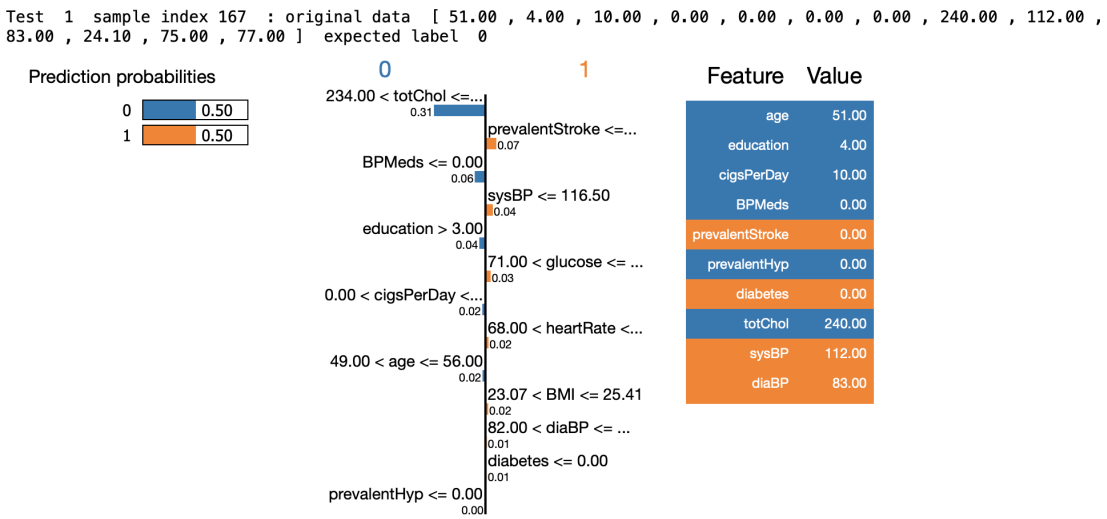


(a) Cluster 0 (b) Cluster 1

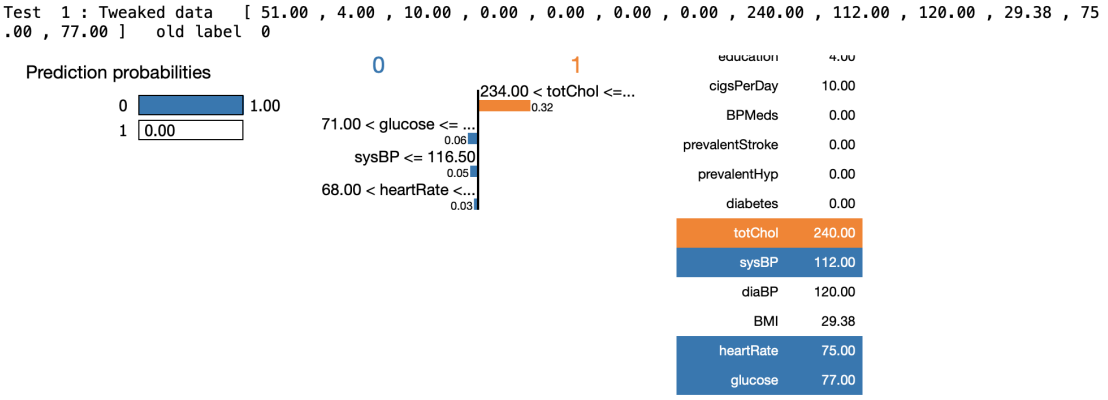
Figure 5.59: Decision plots of Sampling Explainer

LIME

Figure 5.60 provides an example run on the LIME explainer, which reports an interesting case where the uncertainty of the original sample, is strongly influenced by tweaking a feature with a low importance. Although the tweak on the BMI consists of a slight increase in the value, belonging to the same BMI class as the original sample, it has a huge impact on cluster association, according to local LIME explanations.



(a) Original sample



(b) Sample with modified BMI value

Figure 5.60: LIME explanations on single feature's tweak

5.3 Time performance analysis

This section shows the time performances of each used explainer over all experiments of the treated datasets. All values refer to a time performance in seconds.

5.3.1 Artificial datasets

2D Blobs execution times

		Random Forest					Decision Tree				
# sample	Explainer	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS
300	BruteForce	15	15	15	15	15	2	2	2	2	2
	Kernel	3	3	3	3	3	3	3	3	3	3
	Sampling	3	3	3	3	3	2	2	2	2	2
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001
500	BruteForce	27	27	27	27	28	4	4	4	4	4
	Kernel	9	9	9	9	9	8	8	8	9	9
	Sampling	6	6	6	6	5	4	4	4	4	4
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001
1000	BruteForce	64	63	64	63	65	10	9	9	10	9
	Kernel	41	35	36	36	37	34	32	34	33	34
	Sampling	12	12	12	12	10	9	9	9	9	8
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001
2000	BruteForce	178	163	170	175	170	25	24	24	24	24
	Kernel	148	137	157	155	151	134	125	129	130	129
	Sampling	25	21	25	28	25	19	16	18	18	18
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001
3000	BruteForce	287	285	289	290	296	44	43	45	45	45
	Kernel	324	302	325	314	325	276	280	286	284	286
	Sampling	38	37	38	37	33	29	29	30	29	30
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001

Figure 5.61: Times for Random Forest and Decision Tree experiments

		SVM					K-Nearest Neighbors				
# sample	Explainer	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS
300	BruteForce	6	8	5	6	6	11	10	11	11	11
	Kernel	4	4	3	4	4	4	4	4	4	4
	Sampling	3	2	3	3	3	3	2	3	3	3
500	BruteForce	15	15	16	15	21	26	26	27	27	27
	Kernel	11	10	11	11	12	11	11	12	12	12
	Sampling	5	5	5	5	4	5	5	5	5	4
1000	BruteForce	61	54	60	60	78	107	105	106	105	106
	Kernel	43	41	43	43	47	50	0:47	50	49	50
	Sampling	10	10	10	10	9	12	11	12	12	10
2000	BruteForce	231	309	241	218	227	477	495	485	485	477
	Kernel	186	179	170	165	167	210	209	212	212	210
	Sampling	20	17	20	20	20	24	20	24	24	24
3000	BruteForce	539	536	562	554	537	1242	1229	1242	1228	1236
	Kernel	381	371	379	378	365	499	494	503	497	488
	Sampling	32	30	31	31	31	38	37	38	38	37

Figure 5.62: Times for SVM and K-Nearest Neighbors experiments

NN - MLP		Clustering algorithm				
# sample	Explainer	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS
300	BruteForce	4	4	4	4	4
	Kernel	3	3	3	3	3
	Sampling	2	2	2	2	2
500	BruteForce	7	7	8	7	8
	Kernel	12	12	12	12	12
	Sampling	4	5	4	5	4
1000	BruteForce	19	19	19	19	20
	Kernel	41	39	41	41	41
	Sampling	9	9	9	9	9
2000	BruteForce	60	58	55	55	60
	Kernel	148	143	147	147	147
	Sampling	19	16	19	19	16
3000	BruteForce	105	109	115	115	111
	Kernel	304	306	305	307	319
	Sampling	29	29	29	29	24

Figure 5.63: Times for NN-MLP experiments

3D Blobs execution times

		Random Forest					Decision Tree				
# sample	Explainer	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS
300	BruteForce	15	15	15	15	15	2	2	2	2	2
	Kernel	3	3	3	3	3	3	3	3	3	3
	Sampling	3	3	3	3	3	2	2	2	2	2
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001
500	BruteForce	28	27	27	27	27	4	4	4	4	4
	Kernel	9	9	9	9	9	8	8	8	8	8
	Sampling	6	5	6	6	6	4	3	4	4	4
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001
1000	BruteForce	63	63	63	64	63	9	9	9	9	9
	Kernel	34	36	35	35	34	31	33	33	33	32
	Sampling	13	10	12	13	12	9	7	9	9	10
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001
2000	BruteForce	168	158	157	159	159	25	24	24	25	25
	Kernel	134	139	138	146	135	125	127	129	129	125
	Sampling	26	21	25	25	26	19	0:15	19	19	19
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001
3000	BruteForce	302	297	274	282	290	45	44	43	45	46
	Kernel	292	317	305	321	297	270	291	282	276	274
	Sampling	37	33	36	38	37	28	23	27	28	28
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001

Figure 5.64: Times for Random Forest and Decision Tree experiments

		SVM					K-Nearest Neighbors				
# sample	Explainer	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS
300	BruteForce	6	8	6	6	6	11	11	11	11	12
	Kernel	3	4	4	3	3	4	4	4	4	4
	Sampling	3	2	3	3	3	3	2	3	3	3
500	BruteForce	15	21	15	15	16	28	28	29	27	29
	Kernel	10	12	10	10	10	11	12	12	11	11
	Sampling	5	4	5	5	5	5	4	5	5	6
1000	BruteForce	61	81	69	61	62	121	116	123	119	119
	Kernel	41	47	44	46	47	50	50	51	51	50
	Sampling	10	8	10	10	10	12	9	12	12	12
2000	BruteForce	263	335	289	246	264	576	595	592	579	599
	Kernel	168	186	178	168	168	223	233	245	226	230
	Sampling	21	17	21	21	21	25	21	25	26	26
3000	BruteForce	584	852	574	579	556	1501	1500	1340	1394	1445
	Kernel	373	441	380	374	395	542	591	520	535	543
	Sampling	32	27	31	31	32	39	32	37	39	39

Figure 5.65: Times for SVM and K-Nearest Neighbors experiments

		Clustering algorithm				
# sample	Explainer	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS
300	BruteForce	4	4	4	4	4
	Kernel	3	3	3	3	3
	Sampling	3	2	2	3	3
500	BruteForce	8	7	8	7	8
	Kernel	12	12	12	12	12
	Sampling	5	4	5	5	5
1000	BruteForce	20	19	20	20	20
	Kernel	42	40	41	41	41
	Sampling	11	8	10	10	10
2000	BruteForce	57	61	55	57	58
	Kernel	140	153	142	146	145
	Sampling	20	16	19	20	20
3000	BruteForce	110	119	115	118	116
	Kernel	304	325	308	317	317
	Sampling	30	25	29	31	30

Figure 5.66: Times for NN-MLP experiments

Moons execution times

		Random Forest					Decision Tree				
# sample	Explainer	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS
300	BruteForce	15	15	16	15	16	2	2	2	2	2
	Kernel	3	3	3	3	3	3	3	3	3	3
	Sampling	4	4	4	5	3	3	3	3	3	2
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001
500	BruteForce	27	27	28	27	28	4	4	4	4	4
	Kernel	9	9	9	9	9	8	8	8	8	8
	Sampling	8	7	8	8	6	6	6	6	6	4
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001
1000	BruteForce	64	61	63	63	65	9	9	9	9	9
	Kernel	35	34	35	35	35	31	31	31	31	32
	Sampling	16	15	16	16	16	13	12	13	13	13
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001
2000	BruteForce	153	153	154	150	159	23	23	23	22	23
	Kernel	136	131	133	135	134	124	121	121	121	121
	Sampling	32	31	31	32	31	25	25	25	25	25
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001
3000	BruteForce	275	272	286	280	277	42	41	43	44	41
	Kernel	307	299	305	307	306	275	278	272	270	273
	Sampling	46	47	48	48	49	38	38	37	37	37
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001

Figure 5.67: Times for Random Forest and Decision Tree experiments

		SVM					K-Nearest Neighbors				
# sample	Explainer	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS
300	BruteForce	6	4	4	5	11	10	10	11	11	11
	Kernel	3	3	3	3	5	4	4	4	4	4
	Sampling	4	3	4	4	2	4	4	4	4	2
500	BruteForce	17	11	11	11	27	25	25	26	25	26
	Kernel	11	9	10	9	13	11	11	11	11	11
	Sampling	7	6	7	6	4	7	7	7	7	4
1000	BruteForce	118	45	47	51	46	100	99	101	100	100
	Kernel	52	38	38	39	39	46	45	46	46	46
	Sampling	15	13	13	13	13	15	15	15	16	15
2000	BruteForce	494	172	171	211	186	477	473	472	460	420
	Kernel	213	148	148	156	151	205	200	201	199	205
	Sampling	31	26	26	27	26	32	31	31	31	32
3000	BruteForce	1448	423	391	539	405	1246	1258	1222	1187	1208
	Kernel	540	346	332	362	341	495	497	485	417	489
	Sampling	49	41	40	41	39	50	49	48	48	48

Figure 5.68: Times for SVM and K-Nearest Neighbors experiments

		Clustering algorithm				
# sample	Explainer	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS
300	BruteForce	3	3	3	3	4
	Kernel	5	5	5	5	3
	Sampling	3	3	3	3	3
500	BruteForce	6	6	6	6	8
	Kernel	12	12	12	12	12
	Sampling	6	6	6	6	5
1000	BruteForce	15	14	15	15	20
	Kernel	39	39	40	40	41
	Sampling	13	12	13	13	10
2000	BruteForce	38	36	38	39	65
	Kernel	142	137	143	143	150
	Sampling	26	25	26	26	15
3000	BruteForce	65	66	66	66	66
	Kernel	298	301	296	296	294
	Sampling	38	39	39	38	39

Figure 5.69: Times for NN-MLP experiments

Circles execution times

		SVM					K-Nearest Neighbors				
# sample	Explainer	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS
300	BruteForce	6	4	5	6	10	11	11	11	11	11
	Kernel	6	3	3	4	4	10	4	4	4	4
	Sampling	6	4	4	4	3	11	4	4	4	3
500	BruteForce	18	12	12	17	14	25	26	25	26	26
	Kernel	18	10	10	11	10	25	11	11	11	11
	Sampling	18	7	6	6	6	25	7	7	7	7
1000	BruteForce	105	46	48	72	55	100	102	99	99	99
	Kernel	106	39	40	44	41	100	47	47	46	47
	Sampling	106	13	13	14	14	100	15	13	15	15
2000	BruteForce	627	187	169	374	184	444	441	443	445	439
	Kernel	627	152	148	188	152	446	196	196	196	195
	Sampling	606	26	26	29	27	463	31	31	31	31
3000	BruteForce	1728	430	421	944	440	1105	1150	1149	1113	1110
	Kernel	1664	432	351	434	351	1106	1170	478	469	473
	Sampling	1645	431	41	43	40	1152	1151	49	48	48

Figure 5.70: Times for Random Forest and Decision Tree experiments

		Random Forest					Decision Tree				
# sample	Explainer	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS
300	BruteForce	15	15	15	15	16	2	2	2	2	2
	Kernel	15	3	3	3	3	2	3	3	3	3
	Sampling	16	4	4	4	3	2	3	3	3	2
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001
500	BruteForce	28	27	28	28	27	4	4	4	4	4
	Kernel	28	9	9	9	9	4	8	8	8	8
	Sampling	27	8	8	8	8	4	6	6	6	6
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001
1000	BruteForce	64	64	64	63	62	9	9	10	9	9
	Kernel	64	35	36	35	35	9	32	32	32	32
	Sampling	64	16	16	16	16	9	12	12	13	13
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001
2000	BruteForce	166	152	154	156	152	25	23	23	23	23
	Kernel	164	131	132	133	132	25	122	122	123	121
	Sampling	165	31	31	31	32	25	25	25	25	25
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001
3000	BruteForce	305	287	288	291	276	46	44	46	44	42
	Kernel	300	283	307	303	301	43	44	281	278	280
	Sampling	301	284	48	48	46	44	43	39	39	37
	Tree	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001

Figure 5.71: Times for SVM and K-Nearest Neighbors experiments

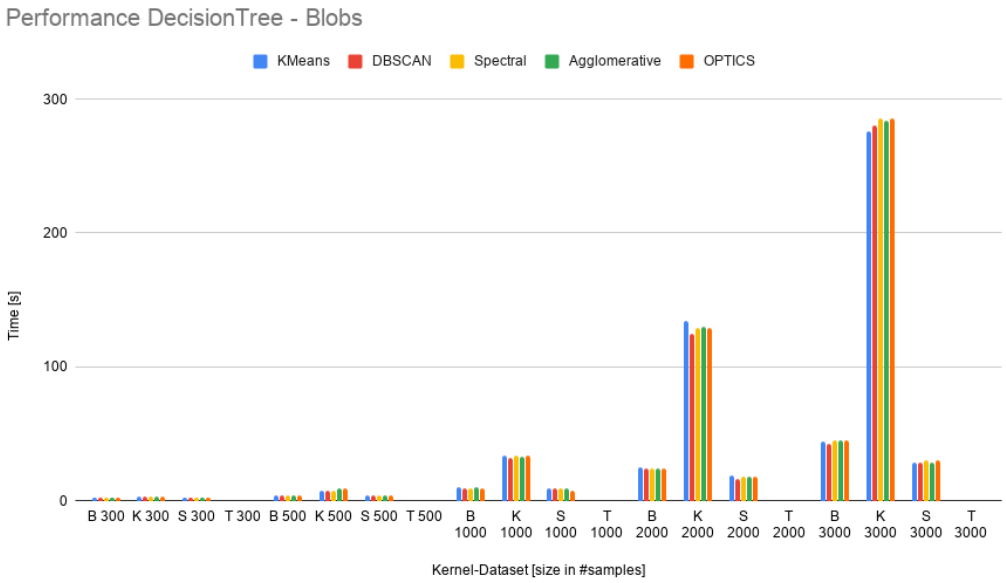
		Clustering algorithm				
# sample	Explainer	KMeans	DBSCAN	Spectral	Agglomerative	OPTICS
300	BruteForce	3	3	3	3	4
	Kernel	3	4	4	4	3
	Sampling	3	3	3	4	3
500	BruteForce	6	6	6	7	6
	Kernel	6	12	12	12	12
	Sampling	6	6	6	6	6
1000	BruteForce	15	16	15	15	15
	Kernel	15	40	40	40	40
	Sampling	15	13	13	13	13
2000	BruteForce	38	36	37	36	37
	Kernel	41	136	137	137	140
	Sampling	38	26	26	26	26
3000	BruteForce	74	67	70	66	65
	Kernel	74	67	301	298	300
	Sampling	79	67	40	39	38

Figure 5.72: Times for NN-MLP experiments

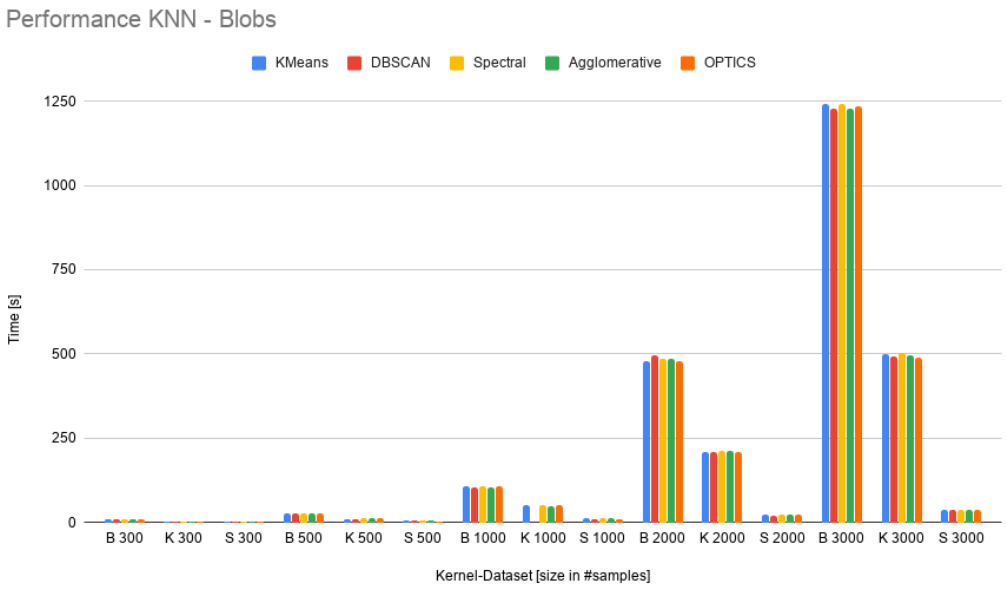
Time comparisons

The first observation, taken from the comparison between the various tests, is that the processing times of the explainer do not depend on the type of dataset. In fact, regardless of the value's distributions of the considered dataset, for the same size, the computational times were very similar. The classifier that takes less time is the Decision Tree, for which the heaviest dataset took less than 300 seconds, times that are higher for

KernelExplainer than for *BruteForceExplainer*. With Random Forest the times are slightly higher than those of the Decision Tree, but they are comparable, in particular the BruteForce and the Kernel times are very similar. The tests with the SVM took a little more time, on average 500 seconds, but reporting a slight difference between the BruteForce and the Kernel, with the latter slightly faster. The most difficult classifier in terms of temporal performance was the KNN, which takes 19 minutes for almost all datasets, with the exception of the 3D Blobs for which it required 25 minutes for the 3000 samples version. For this classifier the time required by the BruteForce is particularly longer than the Kernel. The NN-MLP, on the other hand, has times comparable with the other methods but, as seen for the treatment of weights, it extracts very low or zero weights. With all the classifiers, the times for BruteForce and Kernel explainers are increased, with an almost exponential trend, while the *SamplingExplainer* has a processing time with a much lower growth than the other explainer. This growth is moderate because, for the *SamplingExplainer*, it is possible to take up to a maximum number of background samples, therefore after the background data set (the one used by the classifier as training) has exceeded the maximum size, there is no increase of the times. For the Circles dataset, the differences between Kernel and BruteForce were much more stressed on Decision Tree and NeuralNetwork, while for the other classifiers there were significant peaks for the tests with K-Means and DBSCAN, also with *SamplingExplainer*. The *TreeExplainer*'s time performance are referring only to the Random Forest and Decision Tree's tests. It results to be the fastest over all the SHAP's explainers, taking less than 16 ms in tests with a dataset of 3000 samples. This explainer is quick also due to the low number of features, so as said in the treatment of the explainer's implementation, it limits the evaluation path. The graphs in Figure 5.73, provide an insight on how the nature of the classifier is relevant with respect to the time needed to the explainer.



(a) Decision Tree



(b) K-Nearest Neighbors

Figure 5.73: Time performances comparison

5.3.2 Real datasets

IRIS

		Clustering algorithm				
Classifier	**Explainer**	**KMeans**	**DBSCAN**	**Spectral**	**Agglomerative**	**OPTICS**
Random Forest	**BruteForce**	3	3	3	3	3
	Kernel	0,5	0,5	0,5	0,5	0,5
	Sampling	1,5	2	1,5	1,5	2
	Tree	0,004	0,004	0,004	0,004	0,004
Decision Tree	**BruteForce**	0,5	0,5	0,5	0,5	0,5
	Kernel	0,5	0,5	0,5	0,5	0,5
	Sampling	1	1,5	1	1	1,5
	Tree	0,004	0,004	0,004	0,004	0,004
SVM	**BruteForce**	1	0,6	1	1	0,6
	Kernel	0,6	0,6	0,6	0,6	0,5
	Sampling	1,4	1,4	1,2	1,2	1,5
K-Nearest Neighbors	**BruteForce**	1,5	1,5	1,5	1,5	1,5
	Kernel	0,6	0,6	0,6	0,6	0,6
	Sampling	1,4	1,8	1,4	1,4	1,7
NN-MLP	**BruteForce**	1	0,75	1	0,8	0,75
	Kernel	0,5	0,5	0,5	0,5	0,5
	Sampling	1	1,5	1,1	1,1	1,5

Figure 5.74: Time performances over all classifiers experiments on IRIS

Mall Customers execution times

		Clustering algorithm				
Classifier	**Explainer**	**KMeans**	**DBSCAN**	**Spectral**	**Agglomerative**	**OPTICS**
Random Forest	**BruteForce**	4	-	5	5	1
	Kernel	1	-	1	1	2
	Sampling	2	-	1,5	1,5	1
	Tree	0,001	-	0,001	0,001	0,001
Decision Tree	**BruteForce**	1	-	1	1	1
	Kernel	1	-	1	1	1
	Sampling	1	-	1	1	1,4
	Tree	0,001	-	0,001	0,001	0,001
SVM	**BruteForce**	3	-	3	3	2
	Kernel	1,5	-	1,5	1,4	1
	Sampling	1,5	-	1,5	1,4	1,8
K-Nearest Neighbors	**BruteForce**	2	-	2,3	2,4	2,4
	Kernel	1	-	1	1	1
	Sampling	2	-	1,5	1,4	1,8
NN-MLP	**BruteForce**	1,5	-	1,4	1,4	1
	Kernel	1	-	1	1	1
	Sampling	1,3	-	1,5	1,4	1,4

Figure 5.75: Time performances over all classifiers experiments on Mall Customers

Framingham execution times

Framingham 300 samples		Single sample			20% of dataset		
Classifier	**Explainer**	**KMeans**	**Spectral**	**Agglomerative**	**KMeans**	**Spectral**	**Agglomerative**
Random Forest	**Kernel**	3,53	3,53	3,55	215,33	215,33	216,55
	Sampling	3,68	3,96	3,72	224,48	241,56	226,92
Decision Tree	**Kernel**	3,23	3,23	3,24	197,03	197,03	197,64
	Sampling	4,32	4,7	4,46	263,52	286,7	272,06
SVM	**Kernel**	5,87	5,68	6,78	358,07	346,48	413,58
	Sampling	3,75	3,69	3,47	228,75	225,09	211,67
K-Nearest Neighbors	**Kernel**	6,21	5,54	6,2	378,81	337,94	378,2
	Sampling	3,47	3,59	3,47	211,67	218,99	211,67
NN-MLP	**Kernel**	3,28	3,29	3,28	200,08	200,69	200,08
	Sampling	4,4	4,41	4,38	268,4	269,01	267,18

(a) Time performances on 300 samples

Framingham 3000 samples		Single sample			20% of dataset		
Classifier	**Explainer**	**KMeans**	**Spectral**	**Agglomerative**	**KMeans**	**Spectral**	**Agglomerative**
Random Forest	**Kernel**	34,73	32,78	34,72	20872,73	19700,78	20866,72
	Sampling	3,58	3,66	3,56	2151,58	2199,66	2139,56
Decision Tree	**Kernel**	31,61	31,34	31,26	18997,61	18835,34	18787,26
	Sampling	4,4	4,52	4,31	2644,4	2716,52	2590,31
SVM	**Kernel**	161,5	164,43	210,62	97061,5	98822,43	126582,62
	Sampling	2,21	2,12	1,83	1328,21	1274,12	1099,83
K-Nearest Neighbors	**Kernel**	121,28	133,9	165,9	72889,28	80473,9	99705,9
	Sampling	2,47	2,31	2,11	1484,47	1388,31	1268,11
NN-MLP	**Kernel**	31,6	31,37	31,44	18991,6	18853,37	18895,44
	Sampling	4,31	4,37	4,39	2590,31	2626,37	2638,39

(b) Time performances on 3000 samples

Figure 5.76: Time performances over all classifiers experiments on Framingham

Time comparisons

The datasets Iris and Mall customer, as mentioned above, are small datasets with few features, which is why they take a very little amount of time to compute explanations. Figure 5.77 shows that the needed time is almost comparable among all the classifiers instances. An interesting observation is on the *SamplingExplainer*'s computational times, which are higher than BruteForce and Kernel, with an exception on the Random Forest where the slowest explainer is the BruteForce.

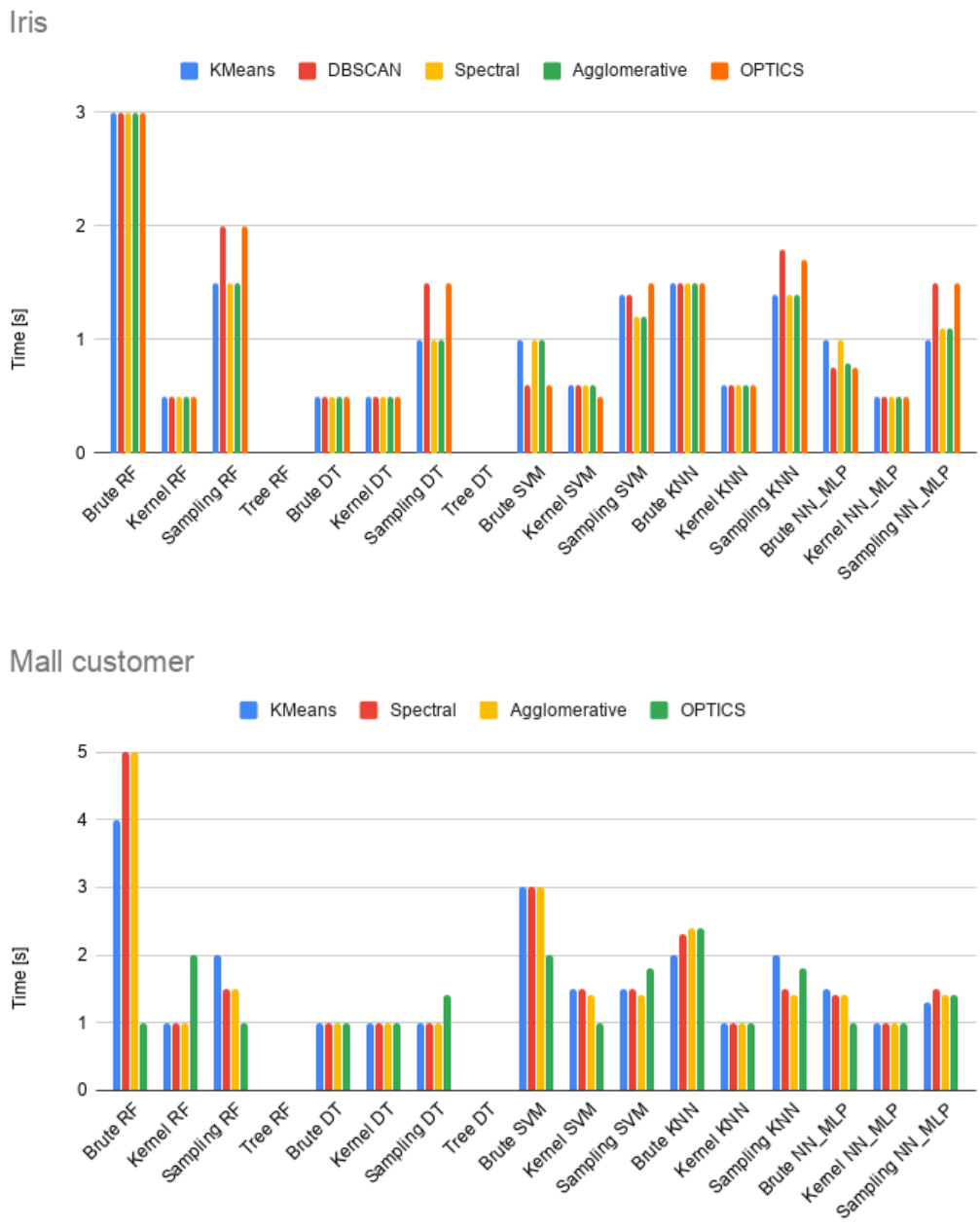


Figure 5.77: Comparison of real datasets time performances

Figure 5.76 reports the times needed to *KernelExplainer* and *SamplingExplainer* to compute the explanation depending on the algorithms used for clustering and classification. Figure 5.76 provides the time performances on a single prediction and on a test dataset equal to 20% of the

considered dataset. Despite the higher number of feature respect to the other datasets, the performances of the *TreeExplainer* were not affected, all being on a time of 1 ms. Explanations could not be extracted with *BruteForceExplainer* because the time needed for a single explanation was so long that the explainer was unable to provide a predicted time for the explanation. Time performances are similar for small dataset, while, as shown in Figure 5.78, for 3000 samples dataset the time performances become really different, highlighting the time effort needed to get a explanations from *KernelExplainer*.

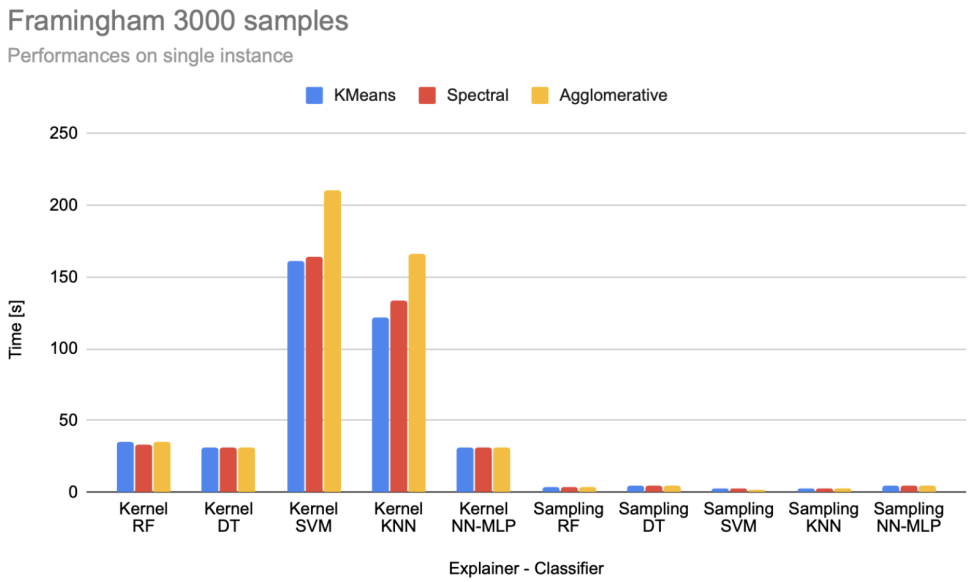


Figure 5.78: Time performances on single instance for 3000 samples Framingham dataset

5.3.3 Feature based time performances

For the Framingham dataset, times were measured for two versions of the dataset, the first consisting of 300 samples and the second of 3000 samples. This choice allows a comparison with datasets with few features, having a low number of samples as for the first version, while the second provides a comparison with datasets with more samples but less features than Framingham. By comparing the graphs of Figure 5.79 and Figure 5.73, given the same number of samples, the impact given by the number of features on the time needed to extract a strip is evident.

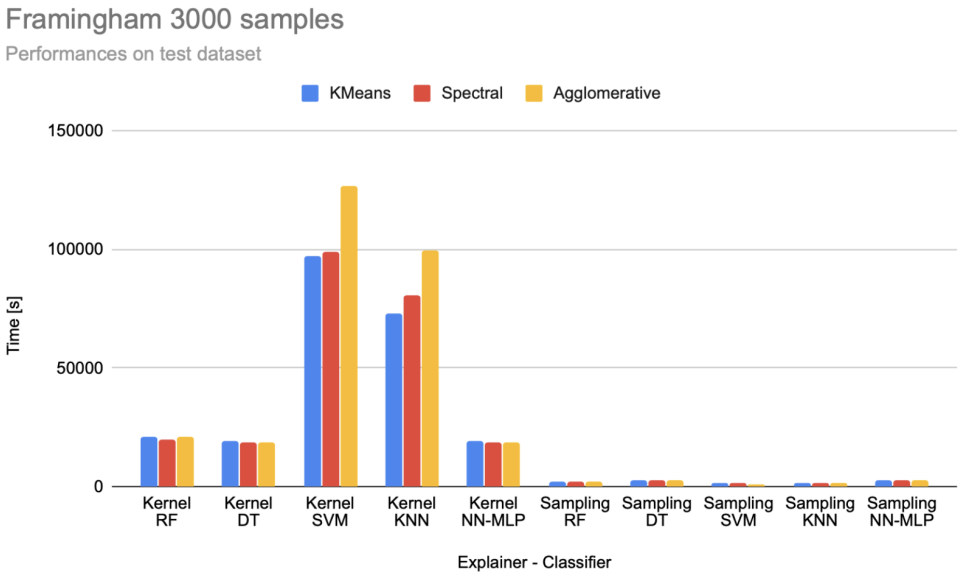


Figure 5.79: Time performances on test dataset from 3000 samples Framingham dataset

Chapter 6

Novel explanation approach

The first part of this chapter is dedicated to the presentation of a new approach. In the second part are presented the results obtained by using the proposed approach on the datasets described in Section 5.

6.1 Explanation approach definition

The approach is model agnostic, so it explains clustering results regardless of the algorithm that generated them. The idea behind the proposed solution is to overcome the intermediate stage of supervised techniques, extracting knowledge by exploiting cluster's algorithms and details. The goal is to gain global and individual cluster explanations, two methods have been implemented to achieve them. The proposed methods are inspired by related works like [35], already described in Section 3, and [75]. Figure 6.1 shows a schematic representation of the methods described below.

6.1.1 Permutation importance for clustering

This method is focused on the extraction of global explanations, considering the clusters provided on the entire dataset. To define the global importance of a feature, the method is based on the perturbation of feature's values.

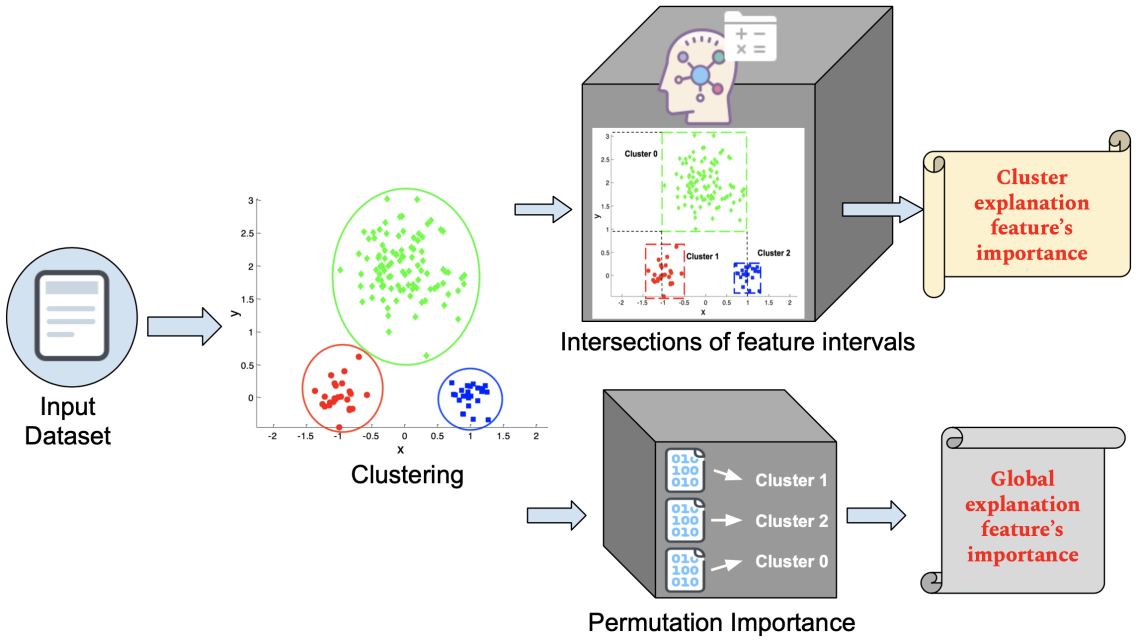


Figure 6.1: Novel explanation approach schema

The key idea is that the global importance of a feature can be measured considering the scores obtained by the cluster assignments on instances with perturbed feature's values. Since the object of the similarity distance is a categorical value, to measure the difference of cluster assignment between original instance and perturbed one are considered the distance metrics Cosine [83, 84] and Hamming [85]. The Hamming distance is the proportion of disagreeing components of two vectors.

The feature's value perturbation has the aim of emulating the feature's absence, for which a common approach is to replace the original feature's value with a feature's value taken from a random instance. Because of that perturbation approach, the method is a feature permutation one. The importance's scores are computed by considering single feature's permutation. Once the instance feature's value is perturbed, it is computed the cluster assignment. For each cluster the method extracts K representative instances, with the input parameter K defined by the user. These instances are the centroids computed over the single cluster instances. By reducing the number of instances, the process of cluster assignment is simplified, because the distances needed to assign a cluster to an instance are less respect to the whole dataset.

The extracted global explanations are computed considering the mean of the distance results. Each feature's permutation consist of *n_round* permutation's rounds, due to the random perturbation that characterize the sampling approach of this method.

6.1.2 Feature intervals intersections

This method focuses on finding cluster's explanations. The intuition behind this method is that when a cluster is well separated from others on a single feature, that feature might be important to associate a sample to that cluster. That knowledge is gained by comparing the feature's presence intervals of each cluster, which represent the range of feature's values defined by the cluster's instances. The presence interval of each feature is extracted over each individual cluster, allowing so a grid representation of clusters in the space of each couple of features. For a cluster i the importance score of a feature is computed by considering the overlap on others cluster. So a feature's interval with a low global overlap may characterize more a cluster from others. The feature importance score is extracted for each cluster, by considering all the cluster pairs.
As expressed in Equation 6.1, from the comparison of two feature k intervals only their intersection is taken into account for the score computation.

$$feat\_p\_score_{k,i} = \frac{1}{N-1} \sum_{j=0, j\neq i}^{N} 1 - \frac{feat\_int\_C_{k,i} \cap feat\_int\_C_{k,j}}{feat\_int\_C_{k,i}} \tag{6.1}$$

In Equation 6.1, the partial score of a feature k, for cluster i, depends on the mean of ratios of the intersection intervals to the cluster i feature's interval. For the final score are considered others feature's partial scores, scaling the sum of partial scores to 1 and than scaling each score to the corresponding proportion (Equation 6.2).

$$feat\_scaled\_score_{k,i} = \frac{feat\_p\_score_{k,i}}{\sum_{j=0, j\neq k}^{N} feat\_p\_score_{j,i}} \tag{6.2}$$

To mitigate zeros scores, the Sigmoid function in Equation 6.3 is applied to all scores.

$$feature\_score_{k,i} = \frac{1}{1 + e^{feat\_scaled\_score_{k,i}}} \tag{6.3}$$

6.2 Experiments results

This section shows the results obtained by applying the novel proposed approach to the artificial and real datasets used in the experiments of Section 5.

6.2.1 Artificial datasets

As described in Section 5, each artificial dataset type is created on five different sizes, from 300 samples up to 3000 samples. For simplicity, the results reported for global explanations on artificial datasets are the average of the results on different dataset size's experiments. Since cluster's explanations are agreeing on feature's importance score over most of the experiments, in the following sections, for each dataset, are reported only the result on two experiments.

2D Blobs	Hamming			Cosine		
	score X	score Y	score Z	score X	score Y	score Z
KMeans	0,328	0,44	0	0,15	0,29	0
Spectral	0,327	0,44	0	0,157	0,3	0
Agglomerative	0,328	0,44	0	0,21	0,25	0
DBSCAN	0,33	0,44	0	0,22	0,24	0
OPTICS	0,33	0,44	0	0,1	0,44	0

3D Blobs	Hamming			Cosine		
	score X	score Y	score Z	score X	score Y	score Z
KMeans	0,02	0,33	0,44	0,013	0,31	0,214
Spectral	0,017	0,33	0,44	0,005	0,1	0,316
Agglomerative	0,02	0,33	0,44	0,008	0,15	0,26
DBSCAN	0,02	0,33	0,44	0,01	0,17	0,27
OPTICS	0,02	0,33	0,44	0,01	0,1	0,33

Figure 6.2: Permutation importance feature's scores on 2D Blobs and 3D Blobs datasets

Figure 6.2 shows how the proposed permutation importance method identifies the feature Y as main one, with a result that is mostly the same over all the used clustering algorithms for the scores based on Hamming distance. Despite some differences, the feature Y is considered more important than feature X even for the scores based on Cosine distance. The scores computed on the other features are zero for both the distance metrics. The scores based on Hamming distance of the two main feature on

3D Blobs datasets are the same of the respective 2D Blobs ones. For 3D Blobs the proposed permutation importance method finds that feature Z is the main one, followed by feature Y. This result reflects the philosophy used to the generation of values for features Y and Z.
The results of feature intervals intersections method, shown in Figure 6.3, highlight the features that mostly characterize a specific cluster. The scores extracted on 2D Blobs experiments provide an intuition on the cluster identification, in fact over all experiments two clusters on three have almost the same score on all features. These clusters are the yellow and green ones in Figure 5.1, while the third one is the violet one. Those clusters are referred next as Cl. Y, Cl. G and Cl. V.
It is interesting to observe how the feature intervals intersections method recognize an higher importance on feature Y, with a score's gap that increases with the dataset size. 3D Blobs's scores reflect the feature generation process, the feature intervals intersections method identifies that for yellow cluster the feature Z is the most important, while for the cluster in violet the main feature is the Y one. The importance of these characterizing features, increases as the dataset size increases.

2D Blobs KMeans	300 samples			500 samples			1000 samples			2000 samples			3000 samples		
	Cl. V	Cl. Y	Cl. G	Cl. V	Cl. Y	Cl. G	Cl. V	Cl. Y	Cl. G	Cl. V	Cl. Y	Cl. G	Cl. V	Cl. Y	Cl. G
score X	0,4	0,56	0,6	0,34	0,59	0,56	0,36	0,57	0,57	0,34	0,57	0,58	0,32	0,55	0,57
score Y	0,44	0,28	0,24	0,5	0,24	0,27	0,48	0,26	0,26	0,51	0,27	0,26	0,52	0,29	0,27
score Z	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05

2D Blobs DBSCAN	300 samples			500 samples			1000 samples			2000 samples			3000 samples		
	Cl. V	Cl. Y	Cl. G	Cl. V	Cl. Y	Cl. G	Cl. V	Cl. Y	Cl. G	Cl. V	Cl. Y	Cl. G	Cl. V	Cl. Y	Cl. G
score X	0,4	0,6	0,59	0,34	0,6	0,56	0,36	0,58	0,57	0,34	0,58	0,57	0,32	0,57	0,55
score Y	0,445	0,24	0,25	0,5	0,24	0,27	0,48	0,26	0,26	0,51	0,26	0,27	0,52	0,27	0,29
score Z	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05

(a) 2D Blobs cluster's explanation extracted on KMeans and DBSCAN results

3D Blobs DBSCAN	300 samples			500 samples			1000 samples			2000 samples			3000 samples		
	Cl. V	Cl. Y	Cl. G	Cl. V	Cl. Y	Cl. G	Cl. V	Cl. Y	Cl. G	Cl. V	Cl. Y	Cl. G	Cl. V	Cl. Y	Cl. G
score X	0,358	0,36	0,48	0,32	0,47	0,35	0,32	0,48	0,35	0,23	0,42	0,33	0,25	0,45	0,33
score Y	0,367	0,16	0,2	0,38	0,21	0,16	0,38	0,2	0,16	0,45	0,22	0,16	0,44	0,21	0,16
score Z	0,16	0,364	0,19	0,18	0,2	0,37	0,18	0,2	0,37	0,19	0,22	0,38	0,18	0,21	0,39

3D Blobs Spectral	300 samples			500 samples			1000 samples			2000 samples			3000 samples		
	Cl. V	Cl. Y	Cl. G	Cl. V	Cl. Y	Cl. G	Cl. V	Cl. Y	Cl. G	Cl. V	Cl. Y	Cl. G	Cl. V	Cl. Y	Cl. G
score X	0,358	0,36	0,48	0,32	0,47	0,347	0,32	0,48	0,35	0,23	0,42	0,33	0,24	0,45	0,28
score Y	0,367	0,16	0,2	0,38	0,21	0,16	0,38	0,2	0,16	0,45	0,22	0,16	0,46	0,21	0,17
score Z	0,16	0,364	0,19	0,18	0,2	0,375	0,18	0,2	0,37	0,19	0,22	0,38	0,18	0,21	0,42

(b) 3D Blobs cluster's explanation extracted on DBSCAN and Spectral results

Figure 6.3: Feature intervals intersections method's scores on 2D Blobs and 3D Blobs clusters

Figure 6.4 provides the results extracted on Moons and Circles dataset, which agree on the recognition of feature X as more relevant than feature Y.

Moons	Hamming		Cosine	
	score X	score Y	score X	score Y
KMeans	0,47	0,03	0,4	0,02
Spectral	0,24	0,19	0,58	0,46
Agglomerative	0,33	0,1	0,66	0,2
DBSCAN	0,24	0,19	0,69	0,55
OPTICS	0,19	0,16	0,56	0,44

Circles	Hamming		Cosine	
	score X	score Y	score X	score Y
KMeans	0,3	0,33	0,31	0,34
Spectral	0,19	0,18	0,46	0,44
Agglomerative	0,36	0,23	0,46	0,29
DBSCAN	0,154	0,148	0,56	0,54
OPTICS	0,14	0,13	0,51	0,48

Figure 6.4: Permutation importance feature's scores on Moons and Circles datasets

In Figure 6.5 are provided cluster's explanations for Moons and Circles, showing both a correct and incorrect cluster results, respect to the structured ones. Such datasets does not have an expected and clear result, as already discussed in Section 5. From both the dataset, the Spectral clustering recognize the cluster as expected, providing always an equal score over all features, with a little difference in some cases. Those cases differs from the others because of a more extended interval, due to some border sample. The scores related to wrong clustering results, highlight the feature that dominates the cluster result, that is often the feature X because of clusters boundaries on that feature.

Moons Spectral	300 samples		500 samples		1000 samples		2000 samples		3000 samples	
	Cl 0	Cl 1	Cl 0	Cl 1	Cl 0	Cl 1	Cl 0	Cl 1	Cl 0	Cl 1
score X	0,2	0,2	0,06	0,17	0,2	0,2	0,2	0,2	0,2	0,2
score Y	0,2	0,2	0,78	0,32	0,2	0,2	0,2	0,2	0,2	0,2
score Z	0,21	0,21	0,06	0,17	0,2	0,2	0,2	0,2	0,2	0,2

Moons Agglomerative	300 samples		500 samples		1000 samples		2000 samples		3000 samples	
	Cl 0	Cl 1	Cl 0	Cl 1	Cl 0	Cl 1	Cl 0	Cl 1	Cl 0	Cl 1
score X	0,65	0,78	0,78	0,78	0,68	0,78	0,78	0,78	0,7	0,77
score Y	0,19	0,06	0,05	0,05	0,16	0,06	0,05	0,05	0,15	0,06
score Z	0,05	0,05	0,05	0,05	0,05	0,06	0,05	0,05	0,05	0,06

(a) Moons cluster's explanation extracted on Spectral and Agglomerative results

Circles Spectral	300 samples		500 samples		1000 samples		2000 samples		3000 samples	
	Cl 0	Cl 1	Cl 0	Cl 1	Cl 0	Cl 1	Cl 0	Cl 1	Cl 0	Cl 1
score X	0,19	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2
score Y	0,19	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2
score Z	0,22	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2

Circles Agglomerative	300 samples		500 samples		1000 samples		2000 samples		3000 samples	
	Cl 0	Cl 1	Cl 0	Cl 1	Cl 0	Cl 1	Cl 0	Cl 1	Cl 0	Cl 1
score X	0,63	0,46	0,62	0,59	0,65	0,63	0,39	0,35	0,3	0,26
score Y	0,19	0,21	0,18	0,13	0,16	0,13	0,33	0,37	0,42	0,49
score Z	0,06	0,11	0,07	0,09	0,06	0,08	0,09	0,1	0,09	0,08

(b) Circles cluster's explanation extracted on Spectral and Agglomerative results

Figure 6.5: Feature intervals intersections method's scores on Moons and Circles clusters

6.2.2 Real datasets

The first analyzed dataset is IRIS, which permutation importance's results are shown in Figure 6.6a. It is interesting to note that the results extracted on Iris reveal that petal length is the main feature, a result confirmed by the literature on the IRIS dataset. The feature intervals intersections method's scores on IRIS clusters, shown in Figure 6.6b, found that for the Setosa class the two main features are "petal length" and "petal width", as also known in literature. The other two classes, as already mentioned in the Section 5, are close to each other, but it is possible to distinguish

IRIS	Hamming			
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
KMeans	0,12	0,01	0,58	0,03
Spectral	0,09	0,01	0,56	0,06
Agglomerative	0,12	0,02	0,54	0,08
DBSCAN	0	0	0,46	0
OPTICS	0	0	0,43	0

IRIS	Cosine			
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
KMeans	0,2	0,02	0,38	0,05
Spectral	0,03	0,003	0,34	0,02
Agglomerative	0,2	0,03	0,38	0,12
DBSCAN	0	0	0,36	0
OPTICS	0	0	0,32	0

(a) Permutation importance feature's scores on IRIS datasets

	IRIS class	score sepal length	score sepal width	score petal length	score petal width
KMeans	Versicolor	0,23	0,1	0,44	0,22
	Setosa	0,21	0,09	0,34	0,34
	Virginica	0,29	0,08	0,43	0,19
DBSCAN	Versicolor	0,23	0,18	0,29	0,29
	Setosa	0,16	0,24	0,28	0,31
	Virginica	0,2	0,15	0,28	0,36
Spectral	Versicolor	0,23	0,1	0,44	0,22
	Setosa	0,21	0,1	0,34	0,34
	Virginica	0,29	0,08	0,44	0,19
Agglomerative	Versicolor	0,24	0,1	0,39	0,26
	Setosa	0,21	0,1	0,34	0,34
	Virginica	0,32	0,08	0,39	0,2
OPTICS	Setosa	0,12	0,09	0,39	0,39
	Versicolor and Virginica	0,25	0,07	0,34	0,34

(b) Feature intervals intersections method's scores on IRIS datasets clusters

Figure 6.6: Global and cluster explanations on IRIS dataset

them considering the feature "sepal length".

The second analyzed dataset is Mall customers, which permutation importance feature's scores are shown in Figure 6.7. For this dataset are

considered the five clusters shown in Figure 5.32, referred next as Cluster Yellow, Cluster Green, Cluster Cyan, Cluster Blue and Cluster Violet. The two main features of this dataset are Spending Score and Annual Income, which are also the main features for literature and supervised technique based proposed approach. For the permutation importance feature's scores based on Hamming distance, the Spending Score is always the most important feature, while for the Cosine distance based scores, the Annual Income is the main feature for the experiments where the clusters are computed via KMeans or Agglomerative.

The importance scores shown in Figure 6.7a refers to Mall customer dataset, also for the feature intervals intersections method not all the clustering algorithm provides a correct clustering result. The global explanations reveal that "Annual Income" and "Spending Score" represent the two main features, with a little gap, higher on clusters by Spectral. Figure 6.7b provides the cluster's explanation that confirm the global explanation, recognizing an almost equal score for the two main features, also for Spectral's clusters.

Mall Customers	Hamming			
	Gender	Age	Annual Income (k$)	Spending Score (1-100)
KMeans	0,01	0,04	0,44	0,5
Spectral	0,02	0,04	0,4	0,5
Agglomerative	0	0,03	0,41	0,5
DBSCAN	0	0	0	0
OPTICS	0	0	0,13	0,29

Mall Customers	Cosine			
	Gender	Age	Annual Income (k$)	Spending Score (1-100)
KMeans	0,001	0,02	0,28	0,26
Spectral	0,01	0,04	0,24	0,37
Agglomerative	0	0,03	0,29	0,27
DBSCAN	0	0	0	0
OPTICS	0	0	0,1	0,23

(a) Permutation importance feature's scores on Mall customers datasets

	Mall Customers	score Gender	score Age	score Annual Income	score Spending Score
KMeans	Cl. Violet	0,06	0,09	0,421	0,423
	Cl. Blue	0,06	0,09	0,426	0,426
	Cl. Cyan	0,06	0,09	0,426	0,426
	Cl. Green	0,06	0,07	0,43	0,438
	Cl. Yellow	0,06	0,11	0,42	0,41
Spectral	Cl. Violet	0,06	0,09	0,42	0,424
	Cl. Blue	0,06	0,09	0,42	0,431
	Cl. Cyan	0,06	0,09	0,425	0,428
	Cl. Green	0,06	0,07	0,427	0,44
	Cl. Yellow	0,06	0,11	0,419	0,408
Agglomerative	Cl. Violet	0,06	0,1	0,434	0,41
	Cl. Blue	0,06	0,09	0,43	0,42
	Cl. Cyan	0,06	0,09	0,378	0,47
	Cl. Green	0,06	0,07	0,43	0,44
	Cl. Yellow	0,06	0,12	0,385	0,434

(b) Feature intervals intersections method's scores on Mall customers datasets clusters

Figure 6.7: Global and cluster explanations on Mall customers dataset

Chapter 7

Conclusion

In recent years, AI has become increasingly present in society, being used in the most disparate fields, from finance to medicine, from justice to security and transport. With the growth of employment contexts, the need to understand the results of AI models has also increased. In fact, there are more and more high-risk tasks in which AI models are applied and for which it is necessary to be able to explain the result, making it responsible, as it can affect people's lives. The growing importance of the explainability of a result now leads to a crossroads with respect to accuracy. While some tasks require accuracy, although at the expense of explainability, for other tasks the compromise of less accuracy in favor of greater explainability is accepted.

In this thesis are analyzed the actual solutions aimed to increase the interpretability of existing models, addressing the imbalance regarding the offer of solutions based on clustering respect to the classification ones. Therefore, two approaches are proposed for the explainability of clustering results.

The first approach is based on state-of-the-art explanation techniques for supervised clustering, which are tailored and adapted for unsupervised clustering applications.The approach is model-agnostic, so it is independent from the type of used supervised algorithm and it is inspired by what is proposed in the literature mentioned in Chapter 3. The explanations are extracted on the trained supervised model using post-hoc interpretation methods. This methods allow not only to get a global explanation about the model build on the clusters results, highlighting which are the main features for the cluster association, but they also provide single sample's

explanation. This approach has the advantage of being model-agnostic, which grant to use any type of supervised model, exploiting well known methods in XAI literature. A limitation of this approach is the usage of an intermediate model between clusters results and explanation methods, because of possible influences due to the supervised-based techniques and to the need of retraining of the model for the state-of-the-art explainer.

The second approach is based on the cluster results analysis, extracting features importance globally and for each single cluster. The idea is that the global explanation can be achieved by single features contributions, observing how the prediction change when a feature changed. Even if a cluster is part of the model, it could not reflecting totally the global explanation's insight. Each cluster is characterized differently from features. This intuition is the basis of the cluster's explanations obtained by the feature intervals intersections method. The approach does not require retraining, providing a fast update of clusters details for new samples. The time needed to extract explanation is less respect to the supervised based approach. Being extracted directly on clustering results, the explanations are more sensitive on them.

7.1 Future work

To date, the research on explainable clustering has been marginal respect to the explainable classification, in some cases leaning on the usage of classification-based methods. This leaded to a growing interest in developing methods capable of increasing the interpretability of cluster results and developing new algorithms able to extract clusters which are interpretable by design. In future work the proposed methods could be supported by comparative methods such as counterfactual explanations, analyzing the differences identified between two similar objects belonging to different clusters. In such a way can be provided a single sample's explanation, focused on identifying the driving features for a given cluster association. Exploiting explanation on different granularity, from global to local, the future work can support domain expert to confirm their prior knowledge or to reveal new insights.

Bibliography

[1] S. S. Marco Tùlio Ribeiro and C. Guestrin, "Introduction to local interpretable model-agnostic explanations (lime)." 2016. [Online]. Available: https://www.oreilly.com/content/introduction-to-local-interpretable-model-agnostic-explanations-lime/

[2] R. MARSELIS, "Make your artificial intelligence more trustworthy with explainable ai." [Online]. Available: https://labs.sogeti.com/make-ai-trustworthy-with-explainable-ai/

[3] C. Molnar, "Interpretable machine learning.a guide for making black box models explainable." [Online]. Available: https://christophm.github.io/interpretable-ml-book/shap.html#definition

[4] S. M. Julia Angwin, Jeff Larson and P. Lauren Kirchner, "Machine bias," *ProPublica*, 2016. [Online]. Available: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing#disqus_thread

[5] a. S. S. Marco Tùlio Ribeiro and C. Guestrin, ""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, KDD '16*, pp. 1135–1144, 2016. [Online]. Available: https://arxiv.org/abs/1602.04938v3

[6] S. M. Lundberg and S.-I. Lee, "Shap's main page on github." [Online]. Available: https://github.com/slundberg/shap

[7] E. Baralis and T. Cerquitelli, "Clustering fundamentals," p. 18. [Online]. Available: https://dbdmg.polito.it/wordpress/wp-content/uploads/2019/10/DSL-4-DMClustering.pdf

[8] E. Lutins, "Dbscan: What is it? when to use it? how to use it." [Online]. Available: https://medium.com/@elutins/dbscan-what-is-it-when-to-use-it-how-to-use-it-8bd506293818

[9] C. Sinclair, "Clustering using optics." [Online]. Available: https:

//towardsdatascience.com/clustering-using-optics-cac1d10ed7a7

[10] J. Gao, "Clustering spectral methods," p. 8. [Online]. Available: https://cse.buffalo.edu/~jing/cse601/fa12/materials/clustering_spectral.pdf

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python." [Online]. Available: https://scikit-learn.org/stable/modules/tree.html#classification

[12] E. Baralis and T. Cerquitelli, "Random forest," p. 3. [Online]. Available: https://dbdmg.polito.it/wordpress/wp-content/uploads/2018/10/8-DMClassificationRandomForest.pdf

[13] MLMath.io, "Math behind svm(support vector machine)." [Online]. Available: https://medium.com/@ankitnitjsr13/math-behind-svm-support-vector-machine-864e58977fdb

[14] W. Gierke and C. Perscheid, "Trends in bioinformatics:bi-clustering with biological context information." [Online]. Available: https://hpi.de/fileadmin/user_upload/fachgebiete/plattner/teaching/TrendsBioinformatics/TiB2017/Final_Presentations/Willi_Gierke_neueFootnotes.pdf

[15] E. Baralis and T. Cerquitelli, "K-nearest neighbor," p. 103. [Online]. Available: https://dbdmg.polito.it/wordpress/wp-content/uploads/2010/12/4-DMClassification-x2.pdf

[16] L. Shukla, "Designing your neural networks." [Online]. Available: https://towardsdatascience.com/designing-your-neural-networks-a5e4617027ed

[17] E. Baralis and T. Cerquitelli, "Neural networks," p. 79. [Online]. Available: https://dbdmg.polito.it/wordpress/wp-content/uploads/2010/12/4-DMClassification-x2.pdf

[18] D. Gunning and D. A. R. P. A. (DARPA), "Explainable artificial intelligence (xai)." 2017. [Online]. Available: https://www.darpa.mil/attachments/XAIProgramUpdate.pdf

[19] V. Jhaveri, "Explainable ai- why is there a need to explain?" [Online]. Available: https://www.capgemini.com/gb-en/2020/04/explainable-ai-why-is-there-a-need-to-explain/

[20] A. Adadi and M. Berrada, "Peeking inside the black-box:

A survey on explainable artificial intelligence (xai)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8466590

[21] S. Srihari, "Explainable artificial intelligence." [Online]. Available: https://cedar.buffalo.edu/~srihari/CSE574/Chap17/17.2.1%20Explainable%20AI.pdf

[22] R. Guidotti, A. Monreale, F. Turini, D. Pedreschi, and F. Giannotti, "A survey of methods for explaining black box models," *ACM Computing Surveys (CSUR)*, vol. 51, pp. 1 – 42, 2018.

[23] B. Goodman and S. Flaxman, "European union regulations on algorithmic decision-making and a "right to explanation"," *AI Magazine*, vol. 38, no. 3, pp. 50–57, Oct. 2017. [Online]. Available: https://www.aaai.org/ojs/index.php/aimagazine/article/view/2741

[24] Microsoft, "Responsible ai principles from microsoft." [Online]. Available: https://www.microsoft.com/en-us/ai/responsible-ai?activetab=pivot1%3aprimaryr6

[25] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, 2018, pp. 80–89. [Online]. Available: https://arxiv.org/abs/1806.00069v3

[26] O. U. Press. [Online]. Available: https://www.oxfordlearnersdictionaries.com/definition/english/explanation?q=Explanation

[27] A. B. Arrieta, N. D'iaz-Rodr'iguez, J. Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garc'ia, S. Gil-L'opez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Inf. Fusion*, vol. 58, pp. 82–115, 2020.

[28] D. Sarkar, "The importance of human interpretable machine learning." [Online]. Available: https://towardsdatascience.com/human-interpretable-machine-learning-part-1-the-need-and-importance-of-model-interpretation-2ed758f5f476

[29] L. Costabello, F. Giannotti, R. Guidotti, P. Hitzler, F. Lecue, P. Minervini, and M. K. Sarker, ""explainable ai:from theory to motivation, applications and challenges," *AAAI-19:Thirty-Third AAAI Conference on Artificial intelligence*, January-February 2019.

[Online]. Available: https://xaitutorial2019.github.io/#

[30] E. in Chief | The Research Nest, "Explainable ai - what is it and why do we need it?" [Online]. Available: https://medium.com/the-research-nest/explainable-ai-what-is-it-and-why-do-we-need-it-261509e48cc

[31] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv: Machine Learning*, 2017. [Online]. Available: https://arxiv.org/pdf/1702.08608v2.pdf

[32] Rui Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, 2005.

[33] S. K. Bera, D. Chakrabarty, and M. Negahbani, "Fair algorithms for clustering," in *NeurIPS*, 2019.

[34] F. Chierichetti, R. Kumar, S. Lattanzi, and S. Vassilvitskii, "Fair clustering through fairlets," *ArXiv*, vol. abs/1802.05733, 2017.

[35] J. Chen, Y. Chang, B. Hobbs, P. Castaldi, M. Cho, E. Silverman, and J. Dy, "Interpretable clustering via discriminative rectangle mixture model," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016, pp. 823–828.

[36] S. Saisubramanian, S. Galhotra, and S. Zilberstein, "Balancing the tradeoff between clustering value and interpretability," *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020.

[37] D. Pelleg and A. W. Moore, "Mixtures of rectangles: Interpretable soft clustering," in *ICML*, 2001.

[38] R. Fraiman, B. Ghattas, and M. Svarc, "Interpretable clustering using unsupervised binary trees," *Advances in Data Analysis and Classification*, vol. 7, pp. 125–145, 2013.

[39] B. Liu, Y. Xia, and P. S. Yu, "Clustering through decision tree construction," in *CIKM '00*, 2000.

[40] N. Frost, M. Moshkovitz, and C. Rashtchian, "Exkmc: Expanding explainable k-means clustering," *ArXiv*, vol. abs/2006.02399, June 2020.

[41] S. Dasgupta, N. Frost, M. Moshkovitz, and C. Rashtchian, "Explainable k-means and k-medians clustering," *ArXiv*, vol. abs/2002.12538, 2020.

[42] D. Bertsimas, A. Orfanoudaki, and H. Wiberg, "Interpretable clustering via optimal trees," *ArXiv*, vol. abs/1812.00539, 2018.

[43] P. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and

validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[44] J. C. Dunn, "Well-separated clusters and optimal fuzzy partitions," 1974.

[45] E. Horel, K. Giesecke, V. Storchan, and N. Chittar, "Explainable clustering and application to wealth management compliance," *arXiv: Applications*, 2019.

[46] P.-Y. S. Hsueh and S. Das, "Interpretable clustering for prototypical patient understanding: A case study of hypertension and depression subgroup behavioral profiling in national health and nutrition examination survey data," in *AMIA*, 2017.

[47] K. Ng, J. Sun, J. Hu, and F. Wang, "Personalized predictive modeling and risk factor identification using patient similarity," *AMIA Summits on Translational Science Proceedings*, vol. 2015, pp. 132 – 136, 2015.

[48] P. D. Koninck, J. D. Weerdt, and S. K. L. M. vanden Broucke, "Explaining clusterings of process instances," *Data Mining and Knowledge Discovery*, vol. 31, pp. 774–808, 2016.

[49] A. Morichetta, P. Casas, and M. Mellia, "Explain-it: Towards explainable ai for unsupervised network traffic analysis," *Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*, 2019.

[50] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774. [Online]. Available: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "3.2.4.3.1. sklearn.ensemble.randomforestclassifier." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClas

[52] ——, "Permutation importance vs random forest feature importance (mdi)." [Online]. Available:

https://scikit-learn.org/stable/auto_examples/inspection/plot_permutation_importance.html#sphx-glr-auto-examples-inspection-plot-permutation-importance-py

[53] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, p. 5–32, Oct. 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[54] A. J. Fisher, C. Rudin, and F. Dominici, "Model class reliance: Variable importance measures for any machine learning model class, from the "rashomon" perspective," 2018.

[55] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti, "Local rule-based explanations of black box decision systems," *ArXiv*, vol. abs/1805.10820, 2018.

[56] S. M. Lundberg, G. Erion, and S.-I. Lee, "Consistent individualized feature attribution for tree ensembles," *ArXiv*, vol. abs/1802.03888, 2018.

[57] S. M. Lundberg and S.-I. Lee, "Github page for kernel explainer implementation code." [Online]. Available: https://github.com/slundberg/shap/blob/575e791cce2f755b5bcebfb33d0fc9a57ff6e77c/shap/explainers/kernel.py

[58] E. Strumbelj and I. Kononenko, "An efficient explanation of individual classifications using game theory," *J. Mach. Learn. Res.*, vol. 11, p. 1–18, Mar. 2010.

[59] Google, "What-if tool." [Online]. Available: https://pair-code.github.io/what-if-tool/

[60] InterpretML, "Interpretml." [Online]. Available: https://interpret.ml

[61] H20.ai, "H2o driverless ai - open source leader in ai and ml." [Online]. Available: https://www.h2o.ai/products/h2o-driverless-ai/

[62] IBM and A. Mojsilovic, "Introducing ai explainability 360." [Online]. Available: https://www.ibm.com/blogs/research/2019/08/ai-explainability-360/

[63] "2.3. clustering, author=Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E., url=https://scikit-learn.org/stable/modules/clustering.html."

[64] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel,

B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html?highlight=kmeans#sklearn.cluster.KMeans

[65] ——, "sklearn.cluster.optics." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.OPTICS.html#sklearn.cluster.OPTICS

[66] A. Aoullay, "Spectral clustering for beginners." [Online]. Available: https://towardsdatascience.com/spectral-clustering-for-beginners-d08b7d25b4d8

[67] C. Gini, "Measurement of inequality of incomes," *The Economic Journal*, vol. 31, no. 121, pp. 124–126, 1921. [Online]. Available: http://www.jstor.org/stable/2223319

[68] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948. [Online]. Available: https://ieeexplore.ieee.org/document/6773024

[69] R. A. FISHER, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x

[70] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, D. Passos, A. andCournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "1.11. ensemble methods," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: https://scikit-learn.org/stable/modules/ensemble.html#forest

[71] E. Baralis and T. Cerquitelli, "Classification fundamentals." [Online]. Available: https://dbdmg.polito.it/wordpress/wp-content/uploads/2019/11/DSL-5-Classification.pdf

[72] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, D. Passos, A. andCournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "1.11. ensemble methods," *Journal of Machine Learning Research*. [Online]. Available: https://scikit-learn.org/stable/modules/svm.html#svm

[73] ——, "1.11. ensemble methods," *Journal of Machine Learning Research*. [Online]. Available: https://scikit-learn.org/stable/modules/neighbors.html#classification

[74] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Neural network models (supervised)." [Online]. Available: https://scikit-learn.org/stable/modules/neural_networks_supervised.html

[75] K. L. Mikhail Korobov, "Permutation importance." [Online]. Available: https://github.com/TeamHG-Memex/eli5/blob/master/docs/source/blackbox/permutation_importance.rst

[76] F. Gilbert, "Introducing shap decision plots." [Online]. Available: https://towardsdatascience.com/introducing-shap-decision-plots-52ed3b4a1cba

[77] "Mall customer segmentation data." [Online]. Available: https://www.kaggle.com/vjchoudhary7/customer-segmentation-tutorial-in-python

[78] [Online]. Available: https://www.mdcalc.com/framingham-risk-score-hard-coronary-heart-disease

[79] [Online]. Available: https://reference.medscape.com/calculator/252/framingham-risk-score-2008

[80] F. H. Study, "Framingham heart study dataset." [Online]. Available: https://www.kaggle.com/amanajmera1/framingham-heart-study-dataset

[81] V. Higuera and S. Sampson, "What is coronary artery disease?" [Online]. Available: https://www.healthline.com/health/coronary-artery-disease#risks

[82] D. Moores and A. Biggers, "Obesity." [Online]. Available: https://www.healthline.com/health/obesity

[83] "Cosine similarity," *ScienceDirect*. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/cosine-similarity

[84] S. Prabhakaran, "Cosine similarity – understanding the math and how it works (with python codes)." [Online]. Available: https://www.machinelearningplus.com/nlp/cosine-similarity/

[85] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg,

J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Metrics and scoring: quantifying the quality of predictions." [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#hamming-loss