UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA
DIPARTIMENTO DI INFORMATICA, SISTEMISTICA E COMUNICAZIONE
PH.D. IN COMPUTER SCIENCE

Symbolic Reasoning for Contrastive Explanations

Supervisor: Prof. Mario Mezzanzanica
Co-supervisor: Prof. Fabio Mercorio
Ph.D. Coordinator and Tutor: Prof. Leonardo Mariani

Ph.D. Candidate:
DR. ANDREA SEVESO, 781856
XXXV CYCLE

Academic Year 2021/2022

"We live on an island surrounded by a sea of ignorance. As our island of knowledge grows, so does the shore of our ignorance."

John Archibald Wheeler

Abstract

The need for explanations of Machine Learning (ML) systems is growing as new models outperform their predecessors while becoming more complex and less comprehensible for their end-users. An essential step in eXplainable Artificial Intelligence (XAI) research is to create interpretable models that aim at approximating the decision function of a black box algorithm.

Though several XAI methods have been proposed in recent years, not enough attention was paid to explaining how models change their behaviour in contrast with other versions (e.g., due to retraining or data shifts). In such cases, an XAI system should explain why the model changes its predictions concerning past outcomes. In several practical situations, human decision-makers deal with more than one machine learning model. Consequently, the importance of understanding how two machine learning models work beyond their prediction performances is growing, to understand their behaviour, their differences, and their likeness.

To date, interpretable models are synthesised for explaining black boxes and their predictions and can be beneficial for formally representing and measuring the differences in the retrained model's behaviour in dealing with new and different data. Capturing and understanding such differences is crucial, as the need for trust is key in any application to support human-Artificial Intelligence (AI) decision-making processes.

This is the idea of `ContrXT`, a novel approach that (i) traces the decision criteria of a black box classifier by encoding the changes in the decision logic through Binary Decision Diagrams. Then (ii) it provides global, model-agnostic, Model-Contrastive (M-contrast) explanations in natural language, estimating why -and to what extent- the model has modified its behaviour over time. We implemented and evaluated this approach over several supervised ML models trained on benchmark datasets and a real-life application, showing it is effective in catching majorly changed classes and in explaining their variation through a user study.

The approach has been implemented, and it is available to the community both as a python package and through REST API, providing contrastive explanations as a service.

Acknowledgements

Thank you, to my parents Maurizio and Cristina, who have always supported me.

Thank you, to my girlfriend Sara, for her love and patience to bear with me.

Thank you, to professors and supervisors Fabio Mercorio and Mario Mezzanzanica, for their wisdom and constant guidance.

Thank you, to all the researchers from the CRISP Research Centre. It is a pleasure to work with you every day.

Thank you, to professor Federico Cabitza, who led me to continue my studies after my Master's Degree, and introduced me to research activities.

Thank you, to the reviewers who evaluated this thesis, professors Benedetto Intrigila and Erik Cambria, for their valuable comments and suggestions.

Andrea

Contents

List of Figures

List of Tables

Acronyms

1 Introduction

1.1 Motivations and Scope

In the last decade, advances in artificial intelligence have led to increasingly powerful machine learning models. Their high performance, however, comes at cost of a clear interpretation of how these systems work internally. Because of their opaqueness, they are called black boxes. Considering their wide range of sensitive applications, from health care to insurance risk and credit scores, it has become apparent that being able to explain the logic behind those models is crucial, especially in such critical scenarios when a user believes there is a mismatch between the algorithm's outcome and its own expectation.

The field of eXplainable AI (XAI) is rapidly growing to answer this need [56]. Since those models are often used for guiding decision-makers to choose between different outcomes, they must be interpretable and understandable.

As an example, once a machine learning-based system has been deployed to perform a task, the problem of keeping it updated often requires retraining the model with new data (see, e.g. [76]). In many real-life AI applications the data is gathered incrementally from different sources. This often forces the machine learning engineers to retrain their models to work on a dataset that grows periodically, and whose distribution changes frequently. Consequently, the underlying learning function of the newly trained model might change the system's logic concerning the past, which leads to producing unexpected outcomes that might contradict past predictions without providing any reason to the user. This is rapidly becoming a critical issue in several contexts, especially in data science applications that deal

with natural language (see, [130]). To clarify the matter, let us consider a machine learning model ψ_1 trained using a dataset D_1 and deployed to perform a single-label classification task, that is, to assign a Boolean value to each pair $(d_j, c_i) \in D_1 \times C$ where D_1 is a set of documents while C is a set of predefined categories. Although the accuracy reached during the training phase may have been high, the introduction of new datasets $D_{t_2}, ..., D_{t_n}$ at time $t_2, ..., t_n$ compels to retrain the model accordingly. Let us then consider a second machine learning model ψ_2, trained over D_{t_2}. One might ask the following:

Q1: Can we estimate to what extent ψ_2 classifies D_{t_2} coherently to the past predictions made by ψ_1?

Q2: Why does the criteria used by ψ_1 result in class c, but ψ_2 does not classify on c anymore?

Q3: Can we use natural language to transform the differences between models' classification paths to make them comprehensible for the final users ?

To face these questions, machine learning-based systems have to synthesise explanations that are transparent and comprehensible to technicians and decision-makers, so that they can understand the rationale that the algorithm used, enabling them to make decisions more consciously (see, e.g. [57]). In this thesis, we propose a method that tries to answer these questions.

Q2 is posed as a *"why question"*, requiring a contrastive explanation or a response to a counterfactual case, which is termed *foil*. That is, people usually do not ask *"why event P happened"* but *"why P rather than Q"* [89]. Most of the current work in XAI, as argued by [89], are either centred mainly around causal attribution (see e.g. [108]), or despite raising contrastive questions, fail to provide contrastive explanations. Notably, contrastive explanations can be multifaceted: one might ask *"Why does object A have property P at time t_i, but property Q at time t_j?"*, i.e., a *T-contrast*, firstly introduced by [121] and formalised in [89]. Answering Q2 requires observing the system as a whole (i.e., global explanation) rather than at a single instance level (i.e., local explanation). A way for generating global explanations from a black box model requires using the training data to fit a surrogate model to the black box and deriving a surrogate model capable of mimicking the original model *globally* (see Figure 1.1).

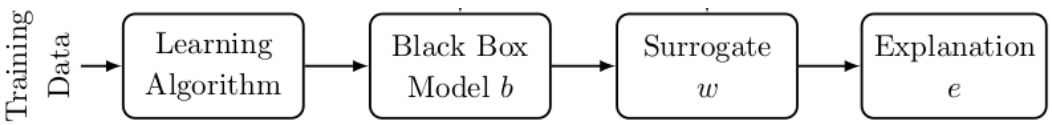


Figure 1.1 Fitting a global surrogate, taken from [18].

Novelty The novelty of our approach is the encoding of the *differences* in the logic of both ψ_1 and ψ_2 through compacted representation of boolean formulae, i.e., Binary Decision Diagram (BDD), to derive contrastive explanations for text and tabular classifiers. Though contrastive approaches are gaining importance in the literature (see Part I), there is no work that the authors are aware of that computes T-contrast or model-contrast explanation globally, as clarified by the most recent state-of-the-art surveys on XAI for supervised machine learning (see Burkart and Huber [18], Mueller et al. [92], Miller [88]). Clearly, we consider the black box is neither interpretable by nature (i.e. models which can be interpreted without using any interpretability method, like decision trees and linear models) nor by design, as recently clarified by [18].

1.2 Motivating Example

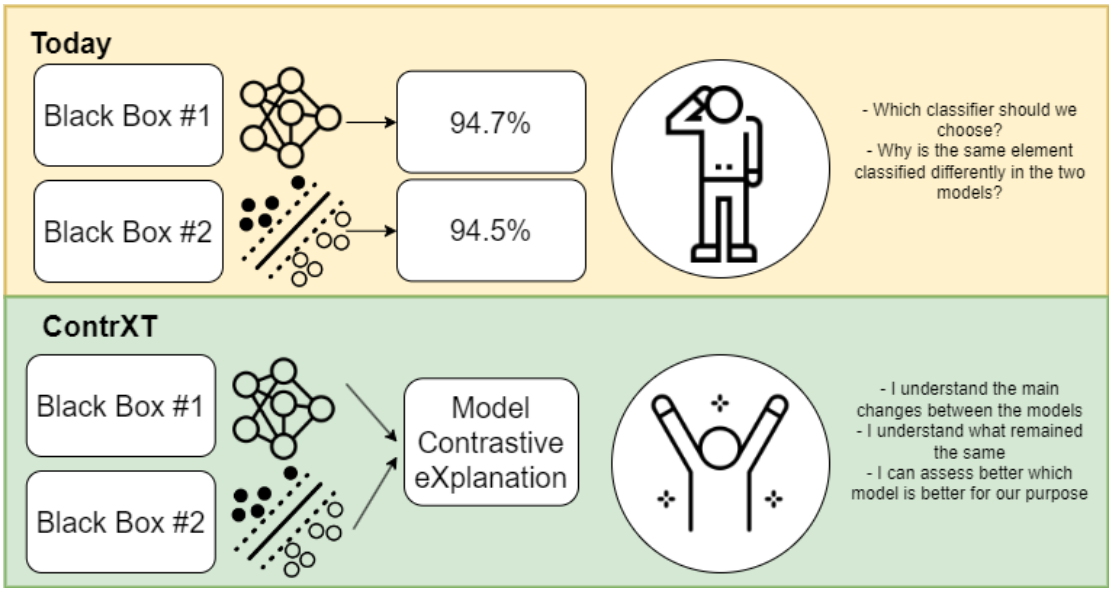


Figure 1.2 Enhanced decision-making process with model contrastive explanations.

When choosing between two different black-box machine learning models, a user might be uncertain about which one is better for his decision-making purpose. The two models might be similar in predictive accuracy given the same task while completely different in their inner working - e.g., being trained on different substrata of data or using a different learning algorithm. The same data instance might be classified differently between the two models, with the user not understanding the reasons behind the dissimilarity.

The approach proposed in this Thesis can enhance the decision-making process by providing model contrastive explanations, which describe how the models differ between

themselves as in Figure 1.2. Model contrastive natural language explanations provide a clearer understanding of why a specific instance was classified differently by providing the main classification paths that have changed or remained the same. Overall, these explanations can help choose which model is more beneficial for the user's purpose.

The example below is inspired by a real-life problem in the field of text classification of multilingual online job ads within an EU project [25, 38]. To clarify the matter, let us consider an organisation that needs to classify millions of online job ads to analyse labour market dynamics over time across borders. In such a scenario, training a machine learning model would be helpful to support questions such as: *Which occupations will grow in the future and where? What skills will be demanded the most in the next years?* However, once such a model has been trained and deployed (see, e.g., [30, 11]) it needs to be periodically re-trained as the labour market demand constantly changes through time, mainly due to rise of new emerging occupations and skills [52]. This, in turn, leads policy makers to ask if - and to what extent - the re-trained model is coherent in classifying new job ads with respect to the past criteria.

As an example, let us consider the *systems analysts*, an occupation that changed a lot in the last years driven by technological progresses [77]. A policy maker might ask: *"how systems analysts are now classified by the updated model, and how the updated model differs with respect to the previous one?"*

```
The model now uses the following classification rules for this class:
This class has 6 added classification rules, but only 3 are used to classify the 80%
of the items.

 - Having Consultant but not Analyst, and SystemsEngineer.
 - Having BusinessAnalyst but not Analyst, SystemsEngineer, and Consultant.
 - Having DataScientist but not Analyst, SystemsEngineer, Consultant, and BusinessAnalyst.

The model is not using the following classification rules anymore:
This class has 6 deleted classification rules, but only 3 are used to classify the
80% of the items.

 - Having Systems but not Analyst.
 - Having System but not Analyst, and Systems.
 - Having TestAnalyst but not Analyst, Systems, System, SystemsAdministrator, and ItSystems.

The following classification rules are unchanged throughout time:
This class has 1 unchanged classification rule.

 - Having Analyst.
```

Figure 1.3 `ContrXT` output for the class "Systems Analysts", showing the change in classification paths between the updated and the older model.

Figure 1.3 shows the difference in the criteria between the two classifiers for the class *"Systems Analysts"*. The Figure shows that the updated model considers *business analysts*

as *Systems Analysts*. Furthermore, the user can easily discover that a novel occupation, i.e., *"data scientist"*, is considered as a system analyst by the updated model. On the other side, Figure 1.3 clarifies to the user that the updated model changed its criterion in regard to the term "test analyst", that now does not characterise the class anymore. Being able to catch those differences -class by class- is helpful to end users as it allows understanding to what extent the updated model is coherent with past predictions, as well as its ability to catch the novelty in the domain and terms that might lead the model to misclassification.

1.3 Contributions

The contributions of this work are as follows:

1. We propose a novel approach to compute model-agnostic global T-contrast explanations from black box text classifiers. Our approach (i) encodes the differences in the classification criteria over multiple training phases through BDDs, and (ii) estimates to what extent the models are congruent or different;
2. We implement this approach as an off-the-shelf Python tool, namely `ContrXT` (short for **Contr**astive e**X**plainer for **T**ext/**T**abular classifier), publicly available to the community on Github[1], evaluating it on (i) *20newsgroups* [65] a well-known multi class benchmark which is partitioned evenly across 20 different topics (classes), and (ii) a real-application in the field of Online Job Ads classification (see [11, 51, 52]), as a research activity of an ongoing EU Project [38].
3. We then generalize the approach above in [78], namely symbolic reasoning moving from time contrastive to *model contrastive* dealing with both textual and tabular data;
4. The effectiveness on tabular data is evaluated over multiple benchmarks, including multiple methods of generating the surrogate models (e.g.,*Rulefit*), and a feature that enhances the natural language explanations computed by the tool providing real data examples to the users. We also tackle the issue of trade-off between fidelity with the original black box model and interpretability.

1.4 Thesis Outline

The thesis is divided in four main parts, organized as follows:

Part I presents the theoretical basis and related work in the field of XAI and contrastive explanations.

[1] https://github.com/Crisp-Unimib/ContrXT

Part II formalizes the approach, shows the implementation of the tool and presents the experimental results on benchmark datasets.

Part III introduces different works on real-world data, dealing with contrastive explanations and machine learning in the domain of Labour Market Intelligence (LMI) and Online Job Advertisement (OJA).

Part IV, lastly, draws the conclusions and proposes several ideas for future research activities.

Part I

Background and Theoretical Basis

In the first part, we introduce the formal notation and concepts that will be used in the Thesis. In chapter 2 we give an introduction to Machine Learning (ML), focusing on black box and white box classification and model evaluation techniques. Then, we look at eXplainable AI (XAI) methodologies in chapter 3, paying attention to the different kinds of explanations and diving deeper in the ones that will be used in the Thesis.

2

Black Boxes and White Boxes

In this section, we will give a brief introduction to supervised machine learning, with keys definitions and concepts that will be used through the Thesis.

There are two main types of supervised learning: classification and regression. Classification deals with predicting discrete class labels, while regression predicts a continuous variabile. The methods developed in the Thesis focus on the former.

2.1 Supervised Black and White Box Classification

Black boxes are models whose behaviour and internal logic is hidden to end-users. As our methodology deals with *classification*, our formalisation relies on the definition of [111].

Definition 2.1.1 (Classifier)**.** Let $\mathcal{D} = \{d_1, \ldots, d_n\}$ be a set of documents, the classification of $\mathcal{D}$ under the class set C consists of $|\mathrm{C}|$ independent problems of classifying each document $d \in \mathcal{D}$ under a given class c_i for $i = 1, \ldots, |C|$. Then, a *classifier* for c_i is a function $\psi : \mathcal{D} \times C \to \{0, 1\}$ that approximates an unknown target function $\breve{\psi} : \mathcal{D} \times O \to \{0, 1\}$. When dealing with a single-label classifier, $\forall d \in \mathcal{D}$ the following constraint must hold: $\sum_{c \in C} \psi(d, c) = 1$.

On the other hand, white box models, or interpretable models, are comprehensible by humans on their own, without the need of additional explanations. This method of training is also called *ante-hoc* explainability. A definition of a white box is provided by [18]:

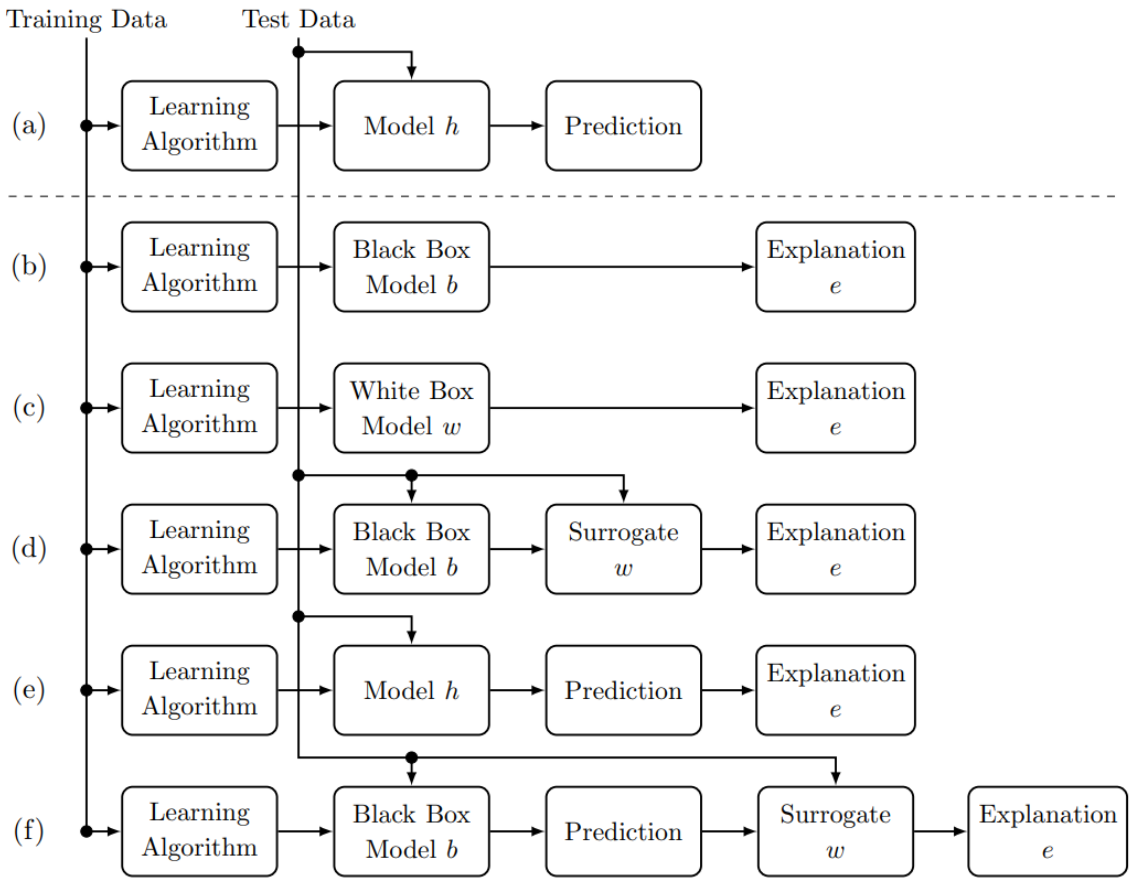


Figure 2.1 [18] shows different ways to gain explainability, starting from (a) standard supervised, black box machine learning without explanation. (b)-(d) Model/global explanations: (b) post-hoc black box model explanation, (c) interpretable by nature, i.e., white box model explanation, and (d) explaining a black box model by means of a global surrogate model (see Sec. 3.2). (e)-(f) Instance/local explanations: (e) directly or (f) with a local surrogate.

Definition 2.1.2 (White Box Classifier)**.** Given training data $\mathcal{D}$, a white box classifier aims for solving the optimization problem

$$p_g* = \arg\min_{p_g \in I} \frac{1}{n} \sum_{i=1}^{n} S(p_g(x_i), y_i) \tag{2.1}$$

where the averaged error S over all the n training instances is minimized and p_g* is the resulting model from the hypothesis space of interpretable models I.

Figure 2.1 shows the differences between various types of black and white box approaches, starting from a completely black box model to different kinds of white box explanations. We also like additional insights on different types of interpretable models for supervised learning, focusing on algorithms that are most relevant to the Thesis.

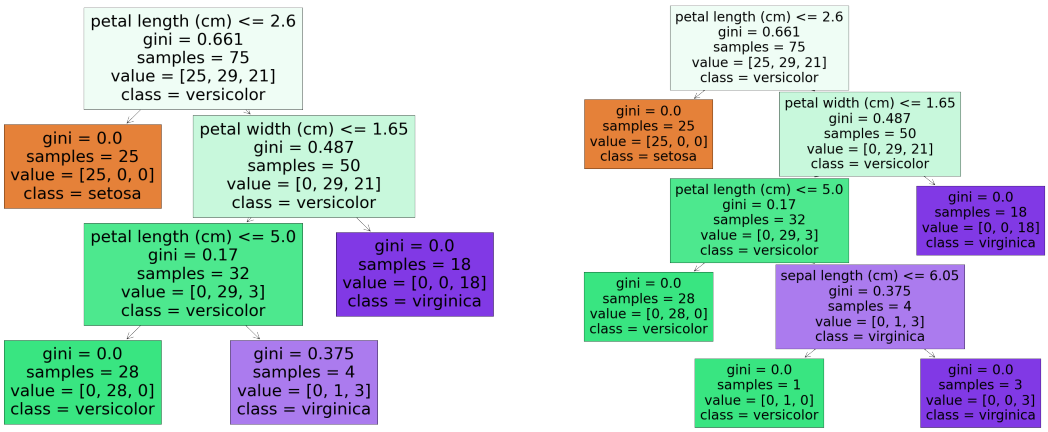


Figure 2.2 Examples of decision trees trained on the Iris benchmark dataset. The maximum depth of the tree on the left is set to 3, while on the right, it is set to 4.

2.1.1 Decision Trees

Decision trees are composed by nodes and leafs, and are trained as in Def. 2.1.2. Figure 2.2 shows a decision tree in action. A splitting feature and a splitting value are given to tree nodes. In the case of classification, leaf nodes are given a class label; in the case of regression, they are given the average value. The classification procedure for a test instance d travels downward from the root node at the top of the tree, and follows a path to a leaf node where the class value is assigned.

A specific feature value is compared to the splitting value at each intermediate node. The direction of travel is either left or right depending on the results of this comparison. Typically, decision trees are greedily built top-down, so that once a feature and its value are chosen as the splitting criterion, these are fixed and cannot be changed [74]. To enhance a tree's capacity to generalise to new data, trees are typically first grown to their maximum size, and then pruned afterwards by removing unnecessary splits. Decision trees can be converted in rule-based models, while the opposite is not always possible.

There are various different algorithms that can be used to build decision trees. The ID2of3 [31] algorithm uses a hill climbing search process to learn M-of-N splits for each node of the tree. C4.5 [104] is a divide and conquer algorithm, based on information theory concepts. It uses the gain ratio criterion, an entropy-based node splitting strategy. In this type of algorithm, input features can be categorical as well as numerical, since the latter are automatically split in a discrete set of intervals. The trees are converted in sets of if-then rules, which are then evaluated to find the best ordering that minimises the gain ratio. It also applies pruning by selectively removing a certain rule precondition, if the accuracy of the

tree increases by doing so. C4.5 is also extended by C5.0T [120], which typically builds smaller sets of rules while also using less memory and being more accurate. Finally, the Classification And Regression Trees (CART) [15] technique is also based on the divide and conquer principle. It splits the input features and their values using the Gini index as the criterion, using the feature and threshold that return the highest information gain during each split. It also supports regression supervised machine learning tasks, and in contrast to C4.5, it does not compute rule sets. CART is the algorithm used by the popular machine learning toolkit package Scikit-Learn [101], in an highly optimised version.

2.1.2 Rule Based Models

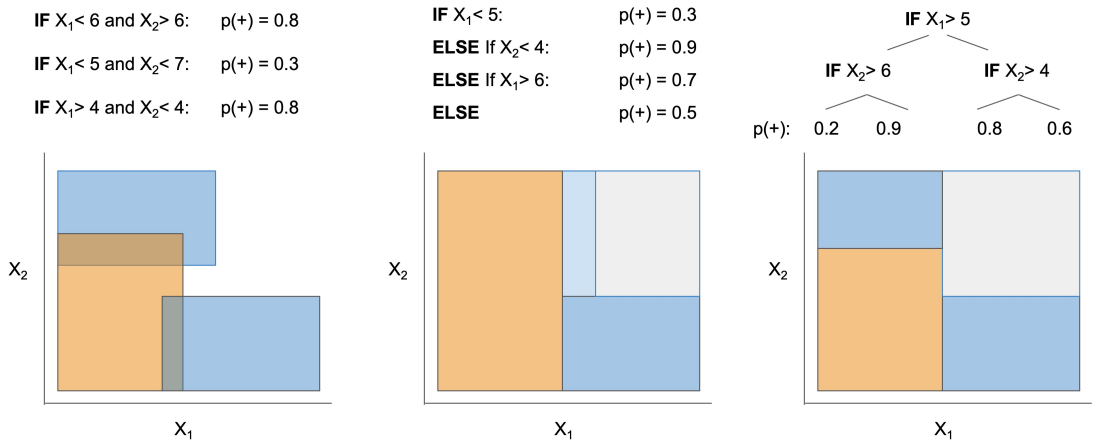
Decision rules are structured as *IF condition THEN label ELSE other label*. Simple decision rules, decision sets, decision tables, and m-of-n rules are a few examples of specific rule-based techniques. To expand on a set of rules, one can either expand a list by adding a rule with *"ELSE IF"* to an existing one, or create a set by adding another rule without any additional syntax. A rule's condition can be either a single feature, operator, or triple of values, which is commonly referred to as a literal in the literature [126]. Others define it as a predicate [72, 73]. The rule can also be a conjunction or disjunction of multiple predicates, the former being the most common case for building rule based models [63]. Figure 2.1 illustrates the structure of an example rule.

```
1  IF feature_1 > 128.5 THEN probability of diabetes: 68.6%
   (57.3%-78.8%)
2  ELSE IF feature_5 > 28.94 THEN probability of diabetes: 28.6%
   (19.1%-39.1%)
3  ELSE probability of diabetes: 3.9% (0.5%-10.6%)
```

Code 2.1 Example of a rule based model on diabetes benchmark dataset.

Figure 2.3 shows the difference between rule sets and rule lists: in rule sets, the rules are disjoined and can overlap with each other, while in rule lists, all the rules are conjoined by *ELSE IF* operations, ensuring no overlap occurs. The representation of decision trees is also added for comparison, the main difference being, rules generated by decision trees always begin with the same feature, the root of the tree, and follow its binary structure.

Rule based models are popular in the literature, with various different algorithms available. 1R [62] creates classification rules for objects based on a single attribute, using a set of training samples as input. Some algorithms use ant-based induction, such as AntMiner+ [81] algorithm, which creates one single rule at a time. There is a multitude of ant-based algorithms, such as cAntMinerPB [95], which creates a list of rules that also consider the



(a) Rule Set *(b) Rule List* *(c) Decision Tree*

Figure 2.3 Graphical visualization of the difference between rule sets, rule lists and rule trees - taken from [116].

interactions between each other. Among others are RIPPER [29], CN2 [80], C5.0R [120] and Re-RX [113].

Falling Rule Lists [125] consist of (as suggested by the name) a list of rules in descending order of class probabilities - meaning, the first rules have the highest probability of matching an instance of the selected class, while the last ones have a lower chance. Falling Rules Lists use Bayesian probabilities as their foundation, and as such, can integrate in the learning process a prior for the size of a decision list. Another upside of this method is the possibility of assigning probabilities also to unseen new instances, as long as they match a rule from the list.

Interpretable Decision Sets [72] are trained to learn a set of rules, optimised for being as short and accurate as possible. The rules are each independent from each other, and can be used singularily. Its algorithm works on a pre-mined rule space, where association rule mining is applied, generating the most frequent item-sets. Both accuracy and interpretability are considered in the optimisation process.

2.2 Evaluation Metrics in Machine Learning

In supervised machine learning methods, a thorough evaluation is key to understanding the performance of either a black or white box classifier.

It is not advised to estimate the goodness of a model on the same data it was trained on. There are different ways of splitting the available data for evaluation, the first one being the hold-out method. Training and performance assessment are performed on two different subsets which form a partition of the dataset. These are called training and test sets. A

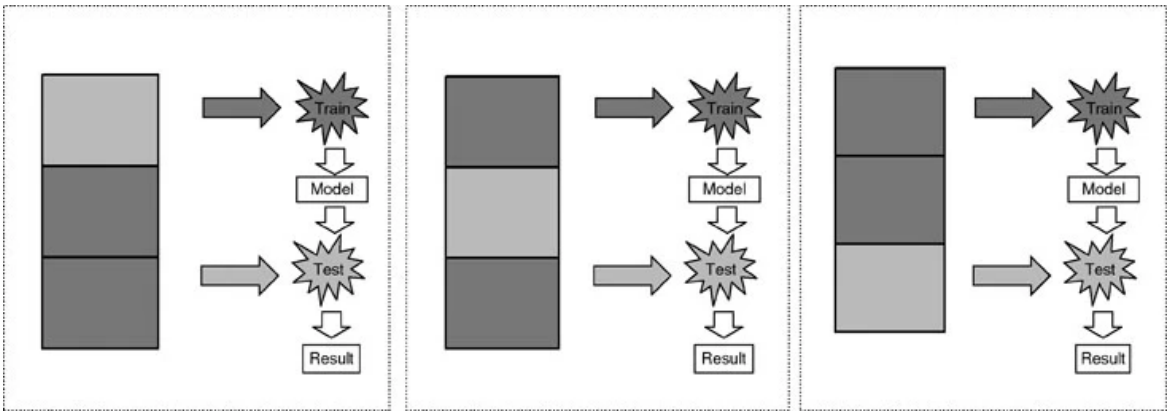


Figure 2.4 Graphical overview of the k-fold cross-validation methodology, taken from [107].

third evaluation set can also be used for selection of hyper parameters settings. The training set is larger than the test set, usually being around 80% of the original dataset, with the test set being the remaining 20%. These subsets are often created using stratified sampling procedures on the target class, to ensure homogeneous distribution between the sets.

Alternatively to hold-out, k-fold cross-validation [48, 114] is widely used to achieve consistent results in the evaluation of models. k-fold cross validation, when compared to the hold-out technique, usually gives a better and less biased indication of how well the model will perform on unseen data. That is, because during hold-out, the data used for testing the model will not be used for training, which could be detrimental in the case that the numerosity of dataset is low, or that some of those instances are actually fundamental for the model to understand outlier cases.

During every iteration of the procedure, the k-fold cross-validation technique splits the original dataset randomly in k groups, k being a parameter chosen by the user - usually 5 or 10. An higher number of folds can give better estimation, up to being equal to the number of instances in the dataset. This latter case is called Leave-One-Out cross-validation, that comes at the cost of higher computation costs and because of that is often unfeasible.

After splitting the data, a model is trained on k-1 folds and evaluated on the remaining one, as presented in Figure 2.4. The procedure is repeated k times, and the resulting performance indicators are then averaged.

The performance can estimated using a variety of different indicators, which are all computed on the basis of how many samples were correctly classified in the test set. The *confusion matrix* is a matrix C, in which $C_{i,j}$ is equal to the number of data instances that are in group i based on the ground truth, and are classified by the model in group j. In a binary classification problem, the number of true negatives is $C_{0,0}$, true positives are $C_{1,1}$, false negatives are $C_{1,0}$ and false positives are $C_{0,1}$, as shown in Table 2.1.

Table 2.1 A representation of a binary confusion matrix, which can be used to calculate various classification performance indicators.

	Predicted Class	
Ground Truth	0	1
0	True Negatives (TN)	False Positives (TP)
1	False Negatives (FN)	True Positives (TP)

The most commonly used indicator is the accuracy, the fraction of correctly classified instances, as in Eq. 2.2.

$$Accuracy = \frac{TP + TN}{TP + TN + FT + TN} \tag{2.2}$$

However, the accuracy suffers from different sources of bias. For example, if the numerosity of the target classes are greatly unequal, accuracy can be very high, despite the model does not actually performing well. To overcome this issue, more robust metrics can be used, such as precision, which is intuitively the ability of the classifier to avoid classifying a negative instance as positive, as in Eq. 2.3; and the recall, which quantifies the capacity of the classifier to correctly label the positive instances as such, as in Eq. 2.4.

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

Finally, we define the F-measure as in Eq.2.5. When $\beta = 1$, the measure is the harmonic mean of precision and recall, and is called F_1 score.

$$F_\beta = \frac{(1 + \beta^2) \cdot Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall} \tag{2.5}$$

All the indicators proposed above are bounded by $[0, 1]$, with 0 being the lowest performance where all the data instances are misclassified, and 1 being the highest where all data instances are correctly classified.

3

eXplainable AI

There is a growing interest in the use of AI algorithms in many real-life scenarios and applications. However, many AI algorithms - as in the case of machine learning - rarely provide explanations or justifications to allow users understanding what the system really learnt, and this might affect the reliability of the algorithms' outcomes when these are used for taking decisions. In order to engender trust in AI, humans must understand what an AI system is trying to achieve, and which criteria guided its decision. A way to overcome this problem requests the underlying AI process must synthesise explanations that are transparent and comprehensible to the final user, so that she/he can consider the outcome generated by the system as *believable*[1] taking decisions accordingly. Not surprisingly, an aspect that still plays a key role in machine learning relies on the quality of the data used for training the model. In essence, we may argue that the well-known principle *"garbage-in, garbage-out"* that characterises the data quality research field, also applies to machine learning ,and AI in general, that is used to evaluated data quality on big data (see, e.g. [85, 4, 7, 86]) and perform cleaning tasks as well (see, e.g. [84, 83, 12]).

Given the success and spread of AI systems, all these concerns are becoming quite relevant enabling a wide branch of AI to emerge, with the aim of making AI algorithms explainable for getting an improved trustability and transparency (*aka* Explainable AI (XAI)).

[1] Here the term believability is inherited from the definition of [127] intended as "the extent to which data are accepted or regarded as true, real and credible".

Defining XAI is not simple. Many different definitions exist, but many of them lack important aspects or are too general. An interesting one is given by [6]:

Definition 3.0.1 (Explainable AI)**.** *"Given an audience, an explainable Artificial Intelligence is one that produces details or reasons to make its functioning clear or easy to understand."*

Though some research on explainable AI had already been published before DARPA's program that launched a call for XAI in 2016 (see, e.g., [118, 97]) XAI [57, 8, 90], effectively encouraged a large number of researchers to take up this challenge. In the last couple of years, several publications have appeared that investigate how to explain the different areas of AI, such as machine learning [60, 108], robotics and autonomous systems [59], constraint reasoning [41], sentiment analysis [21, 20], and AI planning [40], just to cite a few. Furthermore, as recently argued in [8], a key element of an AI system relies on the ability to explain its decisions, recommendations, predictions or actions as well as the process through which they are made. Hence, explanation is closely related to the concept of interpretability: systems are interpretable if their operations can be understood by a human, either through introspection or through a synthesised explanation.

3.1 Different Kinds of Explanations

Explanations are not all equal. There are different dimensions of explanations, as explained in [18]:

- **Explanation Timing**: *ante-hoc* if the model is created as interpretable since its inception, or *post-hoc* if interpretability is added after training;
- **Explanation Scope**: *local* if the explanation is valid only for a single instance amd (optionally) its neighbors, *global* if it is valid for the whole model;
- **Explanation Applicability**; *model specific* if the explanation can only be synthesised for certain types of models, usually dependent on the type of algorithm used during training, or *model agnostic* if the explanation can be synthesised with any kind of model;
- **Reliance on Data**: *data independent* if no additional data is required to synthesise explanations, or *data dependent* if a number of instances is required.

We focus on explanations that are, according to the above dimensions, post-hoc, global, model agnostic and data dependent.

3.1.1 Contrastive Explanations

As recently clarified by [89], the motivation behind contrastive explanations is that people usually explain the reason for an event in relation - or in contrast - to some other event that did not occur, rather than explaining the causes for that event. This means a human explanation is more likely to answer a question like "Why P rather than Q?" instead of "Why P?" even though Q is often implicit in the context. This is called *contrastive explanation*. P is known as the target event and Q is a counterfactual contrast case that did not occur, even if the Q is implicit in the question Furthermore, people usually ask for explanations about events or outcomes that they consider abnormal or unexpected from their own point of view. However, people usually expect to see a specific event to happen while another occurs, making the observed event the fact and the expected event the foil. These kinds of contrastive explanations were classified in [121], with a variation in their names based on whether they are related to a single object, between multiple objects, and an object over time, namely P-contrast, O-contrast and T-contrast. Specifically, we consider **T-contrast** explanation, that [89] expressed as *"Why does object a have property P at time t, but property Q at time $t^{'}$?"*.

In such a scenario, one more variation that one might consider is *M-contrast*, in which the variation is caused by different classification models. The difference in the two classification models could be caused by (i) a difference in the learning functions, either by changes in the hyperparameters used by the classifiers or different types of models altogether; or (ii) a difference in the data used for training, either by retraining a model with newly acquired data instances, or using different substrata of the same dataset. Previous works provide explanations regarding properties of single or multiple objects but do not aim to explain the difference between two models. We answer the question *"Why does object A has property P under model l, but property Q under model $l^{'}$?"* in which the *foil* consists of the properties of the model, i.e., the **M-contrast** case discussed above.

Many works in the field of XAI address contrastive questions, explanations and counterfactual reasoning. In [60], the authors propose a textual counterfactual method which explains the *fact* based on the missing features of the expected *foil*. A model capable of interacting with users aiming at providing a contrastive explanation was introduced in [117]. In [33], authors present a model-agnostic method for structured data which can create contrastive explanations based on the features which should be absent and present in a specific fact, while [100] proposes an algorithm that uses contrastive local explanations to generate boolean clauses, which in turn are used to create a new dataset, to train a simple global transparent model. In recent years, explaining in particular transformers models received much attention. In [26], authors use Layer-wise Relevance Propagation to compute scores

for each attention head and synthesise class-specific visualizations, while [128] proposes an interface that allows users to explore the connections between attention weights in a graphical form.

All the works mentioned above provide explanations regarding properties of single or multiple objects without considering the effect of time, while in contrast, our method can address the *time* element *globally*, as the cause of the altering properties. To do so, we answer the question *"Why does object A have property P at time t, but property Q at time t'?"* in which the *foil* is not an expected and hypothetical event, but an event which occurred and is compatible with the fact, i.e., the **T-contrast** case discussed above. Very recently [75] proposed GraphShapley to compute meta-explanations for T-contrast questions, but different from our work, their solution only works with graphical model inference.

3.1.2 Contrastive and Counterfactual

Another type of explanation is called *counterfactual explanation*, that is defined in [55] as:

Definition 3.1.1 (Counterfactual Explanation)**.** Given a classifier ψ that outputs the decision $y = \psi(x)$ for an instance x, a counterfactual explanation consists of an instance x' such that the decision for ψ on x' is different from y, i.e., $\psi(x') \neq y$, and such that the difference between x and x' is minimal.

In other words, a counterfactual explanation describes a causal situation, and answers what kind of changes can lead the classifier to a different classification outcome. The changes made to the data instance should be *minimized*; even though the concept of minimality is not formalised by the authors of the definition, as its meaning may differ depending on the application domain. For example, in the domain of word embeddings, we could describe a minimal change as switching a word with the most similar alternative. In a tabular data classification with binary attributes, a minimal change could be switching a single feature from 0 to 1.

A canonical example of a counterfactual explanation is a bank customer asking for a loan, who is then rejected. The explanation describes what attributes should have been different in order for the loan to be accepted, e.g. the yearly income should have been higher. Counterfactual explanations are not exclusive to a switch of the target class, but can also be applied to an increased prediction probability, e.g. the probability of the loan being accepted increases by 5%. [99] defines counterfactual as a three step process:

- *Abduction*: condition the latent exogenous variables used by the data generation process, which led to a specific outcome;
- *Intervention*: force a change on an observable variable in the causal history;

- *Prediction*: under the latent conditions of the abduction step that led to the factual outcome, synthesise a new counterfactual example by propagating the intervention through the data generating process.

[55] claims that, practically speaking, in XAI applications there is no difference between counterfactual and contrastive explanations, since the purpose of them both is to find the change in features that would affect the classification outcome. Difference being, one would do so by altering the data instance, while the other by comparing it with another one that belongs to an alternative class.

3.2 Global Explanations via Surrogate Models

The classifier ψ can be seen as a black box whose behaviour is hidden to the end-user, requiring explanations. To connect our approach to the literature, we formalise a *model explanation problem* on top of the definition provided by [56].

Definition 3.2.1 (Explanation Problem)**.** Let ψ be a black box classifier and let $\mathcal{D}$ be a set of documents classified through ψ, the *model explanation problem* consists of finding an explanation $e \in \mathcal{E}$ through a global interpretable model $p_g = surr(\psi, \mathcal{D})$ synthesised from the black box ψ and the documents $\mathcal{D}$ by means of some surrogate fitting $surr(\cdot, \cdot)$. Then, an explainable model is obtained through the global interpretable predictor p_g, that is $exp(p_g)$ for some explanation logic $exp(\cdot)$ which reasons over p_g.

Following [18], a function $surr(\cdot, \cdot)$ should solve the model fitting problem of approximating the black box classifier to a suitable interpretable classifier.

Definition 3.2.2 (White Box Surrogate Model Fitting)**.** Let ψ be a black box text classifier as in Definition 2.1.1, and p_g a white box, interpretable model, as in Definition 2.1.2, the surrogate model fitting the problem consists of approximating the black box classifier to a suitable interpretable classifier by solving:

$$p_g* = \arg\max_{p_g \in I} \frac{1}{X} \sum_{x \in X} S(p_g(x), \psi(x)) \quad s.t. \Omega(p_g) \leq \Gamma \tag{3.1}$$

where I represents a set of possible white box models to be chosen as surrogate, and S is the fidelity of p_g, i.e., a measure of how well the predictions of p_g complies with the predictions of the black box model ψ. In the global case, the surrogate model p_g approximates

ψ over the whole training set X taken from $\mathcal{D}$ or a subset of it which represents sufficiently well the distribution of the predictions of ψ.

With respect to [18], we add a constraint on the complexity of the surrogate model in order to keep it simple enough to be understandable and interpretable while maximising the fidelity score. $\Omega(p_g) : \mathcal{R}^n \rightarrow \mathcal{R}$ is a measure of the complexity of the model and Γ a bounding parameter, where n is the number of the considered complexity dimensions. For instance, for a decision tree $\Omega(p_g)$ could be the number of leaf nodes, determined from its maximum depth and width, while for a logistic regression, it could be the number of non-zero coefficients. To solve the fidelity interpretability trade-off, another common approach is to transform the fidelity maximisation problem into a loss minimisation problem, adding a term $\Omega(p_g)$ accounting for the complexity of the model, as [108] did in their model LIME. However, our primary scope is to have the highest fidelity possible, as long as it is interpretable. Therefore we let the user decide whether a surrogate model is interpretable or not by setting an appropriate upper bound of complexity Γ.

Finally, our methodology relies on the definition of explanation provided by [108] to implement the function $exp()$, that considers an explanation any *"textual or a visual artefact that provides a qualitative understanding of the relationship between the instance's components (e.g. words in a text, patches in an image) and the model's prediction"*.

3.2.1 Evaluation Metrics

Despite the recent boom of XAI systems, there is a little consensus on how to evaluate and benchmark explanations [94, 93], mainly because of the wide range of disciplines they span, from humanities and social science to artificial intelligence, and their different outputs. However, researchers tend to agree [105] that good explanations should satisfy consider the *consistency* of the surrogate against the underlying black-box model. In post-hoc explainability, the model has already been trained and its prediction accuracy is fixed. Thus we should evaluate how the surrogate model is consistent with the underlying black box model [93]. A widely used measure of consistency of post-hoc explanations is the *fidelity* [93, 18]. It is a measure of how well the predictions of the surrogate comply with the predictions of the black box model, as in Eq. 3.2.

$$Fidelity = Indicator(\psi(D), p_g(D)) \tag{3.2}$$

The indicator can differ, but the most commonly used ones are the accuracy, precision, recall and F_1 score described in Sec. 2.2. The difference being, the indicators are not

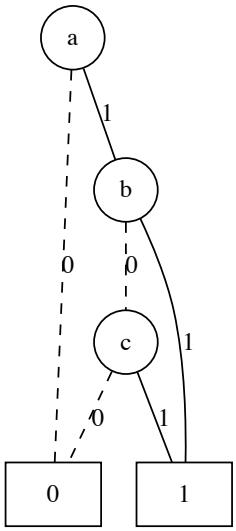


Figure 3.1 A simple example of BDD for the function $(a \wedge b) \vee (a \wedge c)$, visualized. Solid lines represent presence of a feature, while dashed lines represent its absence.

calculated between the ground truth and the black box predictions, instead between the black box predictions and the corresponding ones of the white box surrogate model.

3.3 Binary Decision Diagrams

A BDD [1, 16] is a rooted, Directed Acyclic Graph (DAG) used to represent a Boolean function. [16] defines it as:

Definition 3.3.1 (Binary Decision Diagram)**.** A function graph is a rooted, directed graph with vertex set V containing two types of vertices. A nonterminal vertex v has as attributes an argument index $index(v) \in 1, ..., n$ and two children $low(v), high(v) \in V$. A terminal vertex v has as attribute a value $value(v) \in 0, 1$. Furthermore, for any nonterminal vertex v, if $low(v)$ is also nonterminal, then we must have $index(v) < index(low(v))$. Similarly, if $high(v)$ is nonterminal, then we must have $index(v) < index(high(u))$.

Each non-terminal node (the decision node) represents a variable of the formula, and the terminal nodes are either true (1) or false (0). As the *binary* in the name suggests, each non-sink node, or *internal node*, has an out-degree of two (two outgoing edges). Each one of those represents the value assigned to the corresponding variable, 1 or 0. The BDD in Figure 3.1 represents the function $f = (a \wedge b) \vee (a \wedge c)$. Each path in the graph can be seen as a sequence of assignments for the variables in the internal nodes. As an example, in Figure 3.1, the right-most path represents the assignment $[a : 1, b : 1]$.

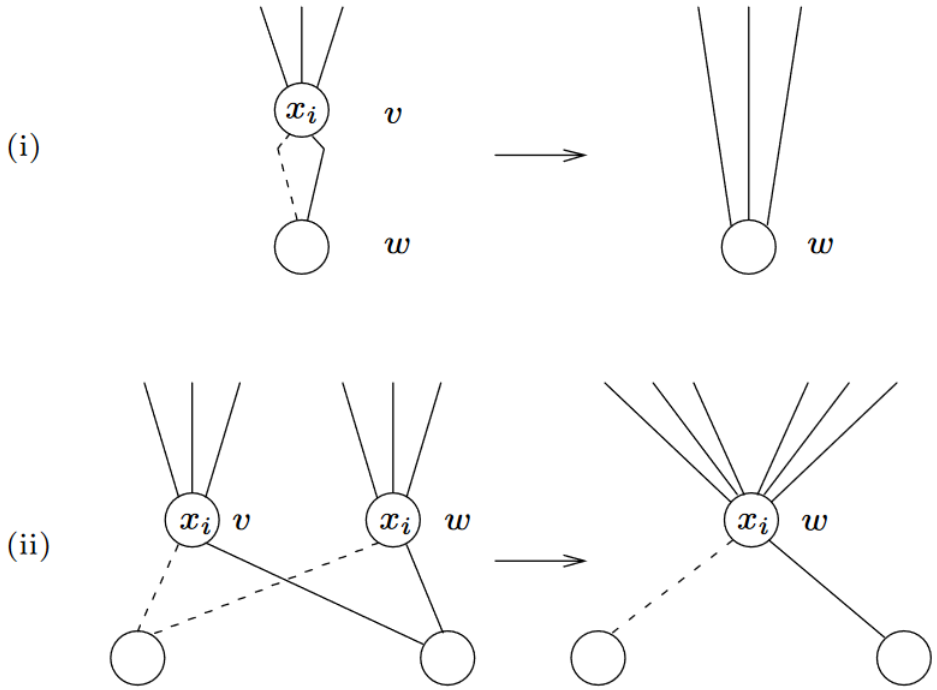


Figure 3.2 The S-deletion rule (i) and the merging rule (ii), taken from [35].

A Reduced Order Binary Decision Diagram (ROBDD) is BDD's canonical form, meaning that given an identical ordering of input variables, equivalent Boolean functions will always reduce to the same ROBDD. It also means that two functions are identical iff their ROBDDs are identical, and unsatisfiable functions are reduced to zero. A formal definition is given by [16]:

Definition 3.3.2 (Reduced Order Binary Decision Diagram)**.** A function graph G is reduced if it contains no vertex v with $low(v) = high(v)$, nor does it contain distinct vertices v and v' such that the sub graphs rooted by v and v' are isomorphic.

The reduction algorithm for BDDs is based on two reduction rules, the *S-deletion rule* (Shannon deletion rule) and the *merging rule*. A representation of the process is shown in Figure 3.3. In [16] it is proven that those rules are sufficient to obtain the corresponding ROBDD for every kind of function. The S-deletion rule applies to nodes where both outgoing edges lead to the same node. This means that the node in question can be immediately deleted without changing the functioning of the formula. The merging rule instead can be applied if two nodes with the same label have the same 0-successor and the same 1-successor. For clarity, examples of those rules are shown graphically in Fig 3.2. In short, ROBDDs are created by merging isomorphic sub graphs and eliminating all nodes having two isomorphic

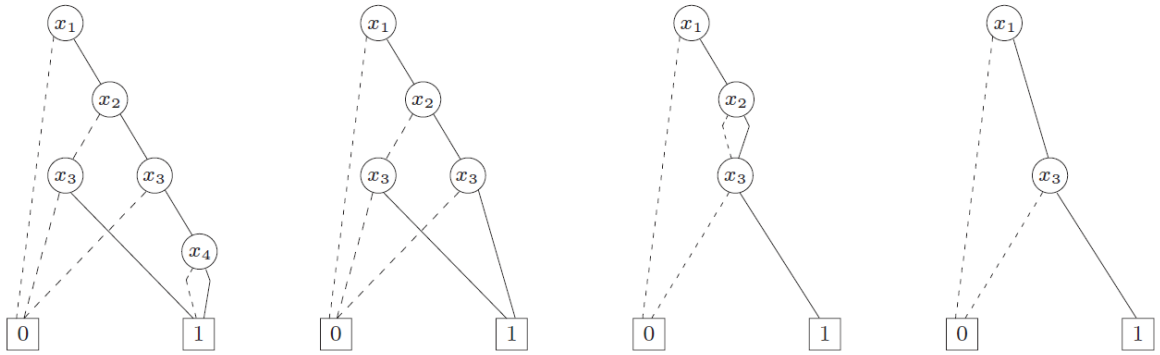


Figure 3.3 Step-by-step example of the ROBDD reduction algorithm, taken from [35].

children. The above rules can be applied in any order, since the result is always the same. However, to be most efficient, they need to be applied bottom-up.

The size of the resulting ROBDD depends on the size of the function represented, as well as the chosen variable ordering. The latter is not a trivial task; in fact, finding the optimal variable ordering is a NP-hard problem, and the number of nodes can be linear or exponential depending on the variable ordering. In [129] it is proven that the number of variable orderings that lead to a polynomial number of nodes in an ROBDD for a certain function is exponentially small. It follows that the choice of variable order must not be randomly selected.

The problem of finding a variable order that leads to minimum ROBDD size has been tackled by many researchers, such as [44] using dynamic programming. The run time of this algorithm is exponential, so only functions with a very limited number of features can be computed. Since finding an optimal solution is not feasible for features with a large number of features, heuristics are often used to find a surrogate solution. [46] proposes a heuristic algorithm that, given a circuit, performs a depth-first search starting at the outputs, and arranges the variables in the order in which they are found during the search. Other heuristics have been proposed over the years, such as [61, 19, 47, 45].

ROBDDs have some multiple interesting properties, especially useful in our context:

i They provide compact representations of Boolean expressions, and there are efficient algorithms for performing all kinds of logical operations on ROBDDs;

ii For any function $f : \mathbb{B}^n \to \mathbb{B}$ there is precisely one ROBDD representing it, and this allows testing whether it is true or false in constant time;

iii On each path of the ROBDD a variable can occur as the label at most once.

From here on, BDD will always be used to refer to Reduced Ordered Binary Decision Diagram, ROBDD.

In connection with XAI, recently [115] used BDDs for a model-specific local explanation of Naïve Bayes classifiers. While the work of [115] is relevant and confirms BDDs can encode the behaviour of a machine learning model, our approach differs in many respects, as (i) we use global explanations instead of local ones, i.e. explaining the general decision-making process of the model instead of focusing on a single instance, (ii) our approach is model-agnostic while [115] is specific to Bayesian network classifiers, and (iii) we focus on time and model-related explanations, rather than explaining classification predictions.

Part II

Contrastive Explanations

4

eXplainable AI for Contrastive eXplanations

ContrXT, as proposed by [78], aims at explaining the differences in the behaviour of two distinct *text* classifiers by encoding of the *differences* in the logic of ψ_1 and ψ_2 through BDDs to produce T-contrast explanations. We extend the symbolic reasoning approach proposed by ContrXT, to performing a pairwise comparison of two models, namely a *left* and a *right* (or, in short, *l* and *r*), in spite of the time. An example might be the comparison of the same algorithm trained on different substrata of a dataset to evaluate the effects of data on the underlying learning function, as well as two distinct training algorithms trained on the same data to assess how their logic differs, regardless of their accuracy. Furthermore, the use of symbolic reasoning might be applied to continuous data through discretization to boolean or ordinal features, as happens in the field of AI planning and control using symbolic model checking (see, e.g. [17]). In this section, we describe the way ContrXT can deal with tabular datasets and their encoding.

If $D_{l,r}$ are datasets comprising tabular instances, then every feature must be discretized: that is, every numeric variable must be partitioned into a number of sub-ranges, each such sub-range treated as a category. Decision Tree algorithms, which are used to train the surrogates in step (B), cannot handle continuous attributes directly [69].

We follow to the definition of [27] to formalise the meaning of *discretization process*: let A be a continuous attribute, and let the domain of A be the interval $[a, b]$. A partition π_a on $[a, b]$ is defined as the following set of k subintervals: $\pi_a = [a_0, a_1), [a_1, a_2) ..., [a_{k-1}, a_k]$,

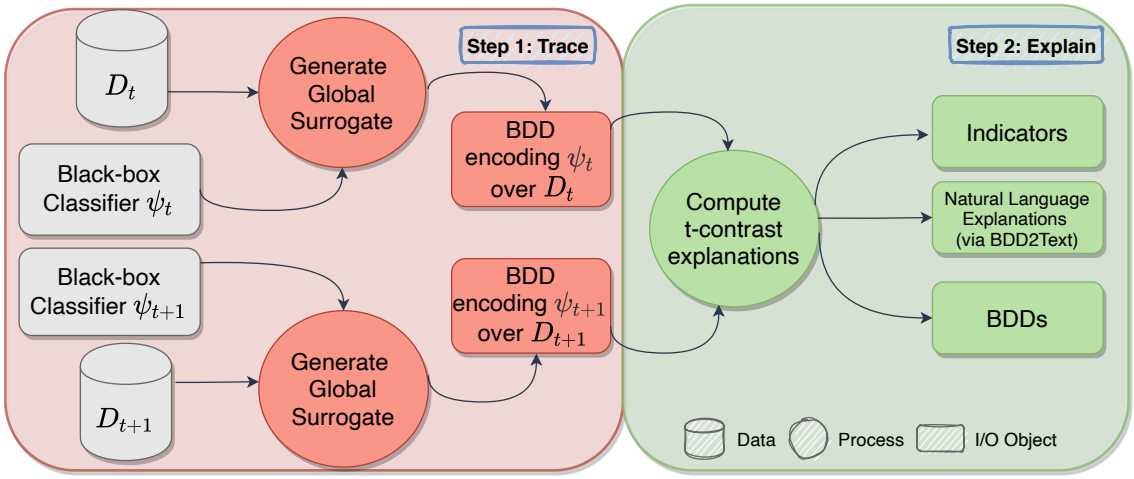


Figure 4.1 Graphical overview of how the algorithm works.

where $a_0 = a$, $a_{i-1} < a_i$ for $i = 1, 2...k$, and $a_k = b$. Thus, discretization is the process that produces a partition π_a on $[a, b]$.

The reason why `ContrXT` does not automatically perform this step is that the discretization method depends both on the problem to be learned as well as on the choice of the black box learning algorithms. No discretization method always out-performs the others [123], despite a wide variety of processes being available [69]. In the presence of a domain characterised by continuous functions, discretizing the continuous functions according to the domain characteristics has proved to be a good strategy that produces effective results (see, e.g. [17]).

Even if the discretization of the attributes often does not significantly change the accuracy of a classifier [34], it is recommended that this step is performed by the machine learning engineers responsible for the training of the black box models $\psi_{l,r}$. Rules with discrete values are normally shorter and more understandable, and for explanation purpose, it is recommended that the created intervals are reasonable in the subject domain.

4.1 Synthesising Contrastive Explanations

The proposed approach includes two steps, (1) *Trace* and (2) *Explain*. Figure 4.1 shows a graphical representation of the process. The first step aims at tracing the logic of a given black box model ψ while working on a dataset D. As one might note, to trace the differences in the classification logic between two black box models over datasets D_1 and D_2, Step 1 is performed twice: one for D_1 and, in parallel for D_2. In essence, Step 1 generates the classifiers' patterns through a global interpretable predictor (i.e., surrogate model) as explained in Sec. 3.2, then it transforms such a global interpretable predictor

Procedure 1 Contrastive Explanations via BDDs

Require: ψ_1, ψ_2, as in Def.2.1.1 ; $\mathcal{D}_1, \mathcal{D}_2$ docs ; $step, threshold$

$Indicator \leftarrow \{\}$; BDD $\leftarrow \varnothing$ //The explanation set $\mathcal{E}$ of Definition 3.2.1
$coverage \leftarrow step$; //incremental step. Default 10%
$J_{\mathcal{D}_1} \leftarrow 0$; $J_{\mathcal{D}_2} \leftarrow 0$ //Jaccard values init;
$b_{1,prev} \leftarrow \varnothing$; $b_{2,prev} \leftarrow \varnothing$;
repeat
  $\mathcal{D}_{1s} \leftarrow Sampling(D_1, coverage)$
  $\mathcal{D}_{2s} \leftarrow Sampling(D_2, coverage)$
  for all $c \in C$ **do**
    $b_1^c \leftarrow Step1\_Trace(\psi_1, \mathcal{D}_{1s}, c)$ //BDDs of ψ over D_{1s} on c
    $b_2^c \leftarrow Step1\_Trace(\psi_2, \mathcal{D}_{2s}, c)$ //BDDs of ψ over D_{2s} on c
    $J_{\mathcal{D}_1^c} \leftarrow jaccard(b_{1,prev}^c, b_1^c)$ //Eq.4.10
    $J_{\mathcal{D}_2^c} \leftarrow jaccard(b_{2,prev}^c, b_2^c)$ //Eq.4.10
  end for
  $J_{\mathcal{D}_1} \leftarrow mean(J_{\mathcal{D}_1^c})$; $J_{\mathcal{D}_2} \leftarrow mean(J_{\mathcal{D}_2^c})$
  $coverage \leftarrow coverage + step$
  $b_{1,prev} \leftarrow b_1$; $b_{2,prev} \leftarrow b_2$
until $(J_{\mathcal{D}_1} \wedge J_{\mathcal{D}_2} \geq threshold) \vee (coverage > 100\%)$
for all $c \in C$ **do**
  $< Indic^c, \text{BDD}^c > \leftarrow Step2\_Explain(b_1^c, b_2^c)$
  $Indic \leftarrow Indic \cup Indic^c$; BDD $\leftarrow$ BDD $\cup$ BDD^c
end for
return $Indic$, BDD

into the corresponding BDD. Step 2 takes as input the BDDs - that formalise the logic of the two classifiers - to compute the BDDs encoding the differences between the two. The pseudo-code is shown in Procedure 1.

4.1.1 Step 1: Trace the Black box

Step 1 Trace

Require: ψ classifier (Def.2.1.1), $\mathcal{D}$ documents and c class

$\Psi_{\mathcal{D},c} \leftarrow \psi(\mathcal{D}, c)$ //Get the predictions over $\mathcal{D}$ for class c
$p_g^c \leftarrow surr(\Psi_{\mathcal{D}_t,c}, \mathcal{D})$ //Global predictor p_g for c on $\mathcal{D}$ as in Definition 3.2.2
$b^c \leftarrow generate\_BDD(p_g^c)$ //Trace ψ over $\mathcal{D}$ through ROBDD
return b^c //ROBDD of ψ over D on c

The main goal of the first step is to build BDDs for *each category*. BDDs are synthesised directly from the global interpretable predictor, encoding a compact representation of the

logic used by the predictor over each class of the dataset processed. To allow the procedure to scale on big datasets, Procedure 1 starts by iteratively sampling each dataset used during training, with uniform incremental steps. In essence, lines (5-17) perform a sampling to build the smallest sample that approximates the distribution of the original dataset appropriately. In larger datasets, due to memory and time constraints, sampling may indeed be the most feasible option for creating surrogate trees. The sampling procedure stops when either the BDDs features remain mostly unchanged from the previous incremental step (using Jaccard similarity, see Equation 4.10), or the whole dataset is covered.

Focusing on *Step 1: Trace*, line (1) uses the black box model ψ to perform single-label classification of the instances of $\mathcal{D}$. Then, line (2) implements the $surr(\cdot,\cdot)$ function of Definition 3.2.1, to produce a rule-based predictor p_g^c for a class c from which, given an instance $d \in D$, it derives the criteria responsible for the classification of d by p_g^c. Line (3) synthesises the corresponding BDD from p_g^c. In Step 2, we manipulate the BDDs generated to provide explanations.

4.1.2 Step 2: Explain through Binary Decision Diagrams and Indicators

Step 2 implements the explanation logic $exp(\cdot)$ as in Definition 3.2.1. In essence, Step 2 manipulates the BDDs generated from Step 1 to explain how ψ_1 and ψ_2 differ (i) *quantitatively* by calculating the distance metric defined below (*aka*, Indicators[1]), and (ii) *qualitatively* by generating the BDDs of the added/deleted patterns over multiple datasets D_{t_i}, as we show in Step 2 pseudo-code. As this is a key idea of our algorithm, we formalise the following.

Definition 4.1.1 (Contrastive explanations through BDDs)**.** Given two formulas which represent the classification paths of a class in two time steps, $f_1 : \mathbb{B}^n \to \mathbb{B}$ and $f_2 : \mathbb{B}^m \to \mathbb{B}$, we define:

$$f_1 \oslash f_2 = \neg f_1 \wedge f_2 \tag{4.1}$$

$$f_1 \oslash f_2 = f_1 \wedge \neg f_2 \tag{4.2}$$

$$f_l \ominus f_r = f_l \wedge f_r \tag{4.3}$$

[1]Indicators usually refer to a type of performance measurement to evaluate the performance of a system/service or of a particular activity (such as projects, programs, products and other initiatives) in which it engages.

The goal of the operator $\oslash$ ($\oslash$) is to obtain a boolean formula that is true iff a variable assignment that satisfies (falsifies) f_1 is falsified (satisfied) in f_2 given f_1 (f_2). The operator $\ominus$ is true iff a variables assigment remains satisfied in both f_1 and f_2. Let b_1 and b_2 be two BDDs generated from f_1 and f_2 respectively, we synthesise the following BDDs:

$$b_{\oslash}^{b_1,b_2} = b_1 \oslash b_2 \tag{4.4}$$

$$b_{\oslash}^{b_1,b_2} = b_1 \oslash b_2 \tag{4.5}$$

$$b_{\ominus}^{b_l,b_r} = b_l \ominus b_r \tag{4.6}$$

where $b_{\oslash}$ ($b_{\oslash}$) is the BDD that encodes the reduced ordered classification paths that are falsified (satisfied) by b_1 and satisfied (falsified) by b_2; $b_{\ominus}$ encodes the paths that are satisfied in both b_1 and b_2.

Step 2 Explain

Require: b_1^c, b_2^c BDDs for class c on $\mathcal{D}_1$ and $\mathcal{D}_2$
1: $b_{\oslash}^c, b_{\oslash}^c, b_{\ominus}^c \leftarrow b_l^c \oslash b_r^c, b_l^c \oslash b_r^c, b_l^c \ominus b_r^c$ //Apply Equation 4.4,4.5,4.6
2: $k_{\oslash}^c, k_{\oslash}^c, k_{\ominus}^c \leftarrow Add(b_{\oslash}^c), Del(b_{\oslash}^c), Still(b_{\ominus}^c)$ //Apply Eq.4.7,4.8,4.9
3: $Indicator^c \leftarrow < k_{\oslash}^c, k_{\oslash}^c, k_{\ominus}^c >$
4: $BDD^c \leftarrow < b_{\oslash}^c, b_{\oslash}^c, b_{\ominus}^c >$
5: **return** $Indicator^c$, BDD^c

We also denote as:

- $var(b)$ the variables of b;
- $sat(b_{\oslash}^{b_l,b_r})$ all the true (satisfied) paths of $b_{\oslash}^{b_l,b_r}$ removing $var(b_l) \setminus var(b_r)$;
- $sat(b_{\oslash}^{b_l,b_r})$ all the true (satisfied) paths of $b_{\oslash}^{b_l,b_r}$ removing $var(b_r) \setminus var(b_l)$;
- $sat(b_{\ominus}^{b_l,b_r})$ all the true (satisfied) paths of $b_{\ominus}^{b_l,b_r}$.

All three of $b_{\oslash}^{b_l,b_r}$, $b_{\oslash}^{b_l,b_r}$ and $b_{\ominus}^{b_l,b_r}$ encode the differences (or similarities) in the logic used by b_1 and b_2 in terms of feature presence (i.e., classification paths). We recall that conjunction, disjunction and negation operations between BDDs can be performed in polynomial-time as for boolean functions, as well as the equivalence between BDDs [16]).

Indeed, $b_{\oslash}^{b_1,b_2}$ ($b_{\oslash}^{b_1,b_2}$) can be queried to answer a contrastive question like *"Why does a path on b_1 had a true (false) value, but is false (true) in b_2?"*. $b_{\oslash}^{b_l,b_r}$ can be queried as *"Which satisfied paths remained unchanged in both b_l and b_r?"*. Clearly, features discarded (added) by b_2 are removed from paths of $b_{\oslash}^{b_1,b_2}$ ($b_{\oslash}^{b_1,b_2}$) as they are used by ψ_1. We also exclude the features that b_2 no longer uses from $sat(b_{\oslash}^{b_1,b_2})$, and the features added by b_2

from $sat(b_{\oslash}^{b_1,b_2})$. After using the operators ($\oslash, \oslash, \ominus$), we check for zero occurrence rules, and remove them if they are never satisfied in the original datasets d_1 and d_2.

We can now define indicators that estimate the differences between b_1 and b_2 in terms of classification paths to true added and deleted by b_2 given b_1.

$$Add(b_{\oslash}^{b_1,b_2}) = \frac{|sat(b_{\oslash}^{b_1,b_2})|}{|sat(b_{\oslash}^{b_1,b_2})| + |sat(b_{\oslash}^{b_1,b_2}) + |sat(b_{\ominus}^{b_l,b_r})|} \tag{4.7}$$

$$Del(b_{\oslash}^{b_1,b_2}) = \frac{|sat(b_{\oslash}^{b_1,b_2})|}{|sat(b_{\oslash}^{b_1,b_2})| + |sat(b_{\oslash}^{b_1,b_2}) + |sat(b_{\ominus}^{b_l,b_r})|} \tag{4.8}$$

$$Still(b_{\ominus}^{b_l,b_r}) = \frac{|sat(b_{\ominus}^{b_l,b_r})|}{|sat(b_{\oslash}^{b_l,b_r})| + |sat(b_{\oslash}^{b_l,b_r})| + |sat(b_{\ominus}^{b_l,b_r})|} \tag{4.9}$$

Finally, the similarity in terms of variables between b_1 and b_2 is computed with the well known Jaccard similarity metric as:

$$Jaccard(b_1, b_2) = \frac{|var(b_1) \cap var(b_2)|}{|var(b_1) \cup var(b_2)|} \tag{4.10}$$

4.1.3 Natural Language Explanations

As the reader might note, reading BDDs might be impracticable even for experts. [22] argues that in many XAI-based systems, not enough attention has been paid to the *"last mile"*, the presentation of explanations to end-users. Though the main contribution of our tool relies on encoding the logic that differentiates the behaviour of two classifiers (Eq. 4.7, 4.8, 4.9), we enriched it with a *BDD2Text* module that exhibits the added/deleted paths derived from $b_{\oslash}$ and $b_{\oslash}$ to final users, using natural language. To do so, we follow the *six NLG tasks* described by [49] that allow us generating natural language explanations. According to Gatt and Krahmer, generating a natural language text from data (which can be in various forms like tabular, textual, vocal or visual) requires to follow some broad steps:

- Deciding about the information and concepts that are going to be included in the final text and (Content determination);
- The order of Information (Text Structuring and Sentence Aggregation);
- Choosing the right words which are needed in order to show the chosen content from the previous step (Lexicalisation);
- Synthesising descriptions of entities that enable the audience to identify entities in their specific context (Referring Expression Generation);

- Applying grammar rules in order to generate a text which is correct both syntactically and morphologically (Linguistic Realisation).

The principal text generation steps of *BDD2Text* module are: (i) redundancy reduction by identifying common criteria and aggregating those with a significant amount of shared criteria using Frequent Itemsets technique (see, e.g. [106]); (ii) utilising dynamic templates similar to [109]; (iii) as a post-processing step, similar to [71] which uses colours and font-size to increase the soundness of the explanations, we use ASCII escape codes to improve the visual aspect of the final results by adding colours that show the presence and absence of features.

As our input data provides both the required information to be included and their order, the first three tasks of “content determination”, “text structuring” and "sentence aggregation" are not done by *BDD2Text*. To perform the remaining three tasks described above, we first reduced the redundancy of text by identifying common criteria and aggregate those with a significant amount of shared criteria using Frequent Itemsets technique, and to combine all phrases together we performed "linguistic realisation" by utilising dynamic templates similar to [109, 3, 98]. As the post-processing steps, similar to [71] which uses colours and font-size to increase the soundness of the explanations, we use ASCII escape codes to improve the visual aspect of the final results by adding colours that show the presence and absence of features.

4.2 Working Example

The following example should help in clarifying the matter. Let us consider two binary classifiers ψ_1 and ψ_2 as in Definition 2.1.1 trained on D_1 and D_2 respectively, as in Step 1. The output of Step 1 is two BDDs, namely b_1 and b_2, encoding the (reduced) classification paths synthesised from ψ_1 and ψ_2 on D_1 and D_2 (Figure 4.2a, and 4.2b). For the sake of simplicity, we can assume b_1 and b_2 encode two distinct boolean formulae like $f_1 = (a \wedge b) \vee (a \wedge c)$ and $f_2 = \neg b \wedge (c \vee d)$. Then, the BDD computed as $b_{\oslash}^{b_1,b_2}$ would represent all the reduced paths (i.e., classification rules) that ψ_2 used to classify instances of D_2 that have not been applied by ψ_1 while classifying D_1 (Figure 4.2c). Conversely, $b_{\oslash}^{b_1,b_2}$ would represent all the paths that ψ_2 applied to classify instances of D_2 but ψ_1 did not use for classifying items of D_1 (Figure 4.2d). From those BDDs we can compute:

- $sat(b_{\oslash}^{b_1,b_2})$ as all the true paths in $b_{\oslash}^{b_1,b_2}$ that were either false paths in b_1 or not present in b_1. Looking at Figure 4.2c, $sat(b_{\oslash}^{b_1,b_2}) = [b:0, c:1 | b:0, c:0, d:1]$. This means ψ_2 added (i) a true path in case of $b:0$ and $c:1$, in spite of the value of a, and (ii) a new true path $[b:0, c:0, d:1]$ that does not depend on the value of a anymore;

- $sat(b_{\ominus}^{b_1,b_2})$ as all the true paths in $b_{\ominus}^{b_1,b_2}$ that were either false paths in b_2 or not present in b_2. Looking at Figure 4.2d, $sat(b_{\ominus}^{b_1,b_2}) = [a : 1, b : 1]$. This means ψ_2 removed the path leading to true instances classified through ψ_1.

Therefore, the *Add* (*Del*) as in Equation 4.4 (4.5) is the ratio of added (deleted) paths over the total number of paths which have been added or deleted. In our example, ψ_2 mainly changed three paths: 2 out of 3 paths were added as true paths whilst the remaining one was removed.

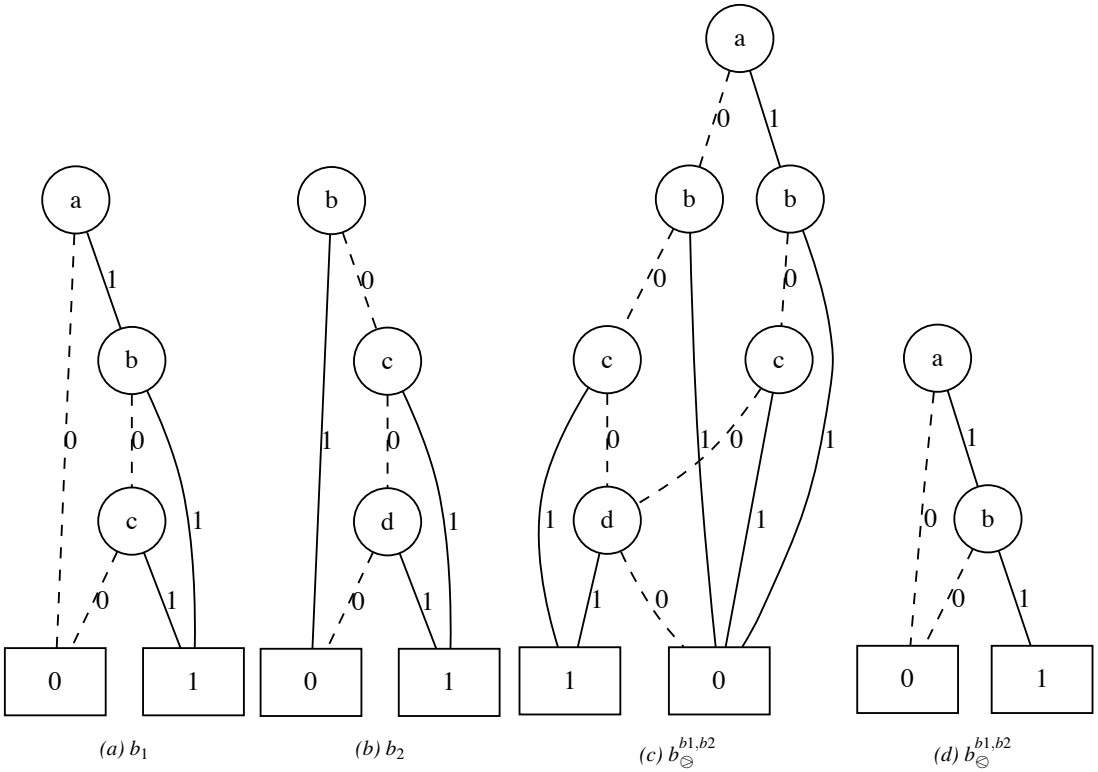


Figure 4.2 ROBDDs for the working example. Each node indicates a feature. A solid line means that the feature is present, while a dashed line that the feature is not present.

4.3 Time Complexity

The complexity of a single algorithm iteration, as in Procedure 1, depends on (i) the cost of *Trace* and (ii) the cost of *Explain* steps, multiplied by the number of classes of the domain (i.e., $|C|$). The cost of *Trace*, in turn, depends on the time needed for generating the surrogate plus the cost of generating the ROBDD from it. The former, in case of building a balanced Decision Tree (DT), is $O(n_{samples} \cdot n_{features} \cdot \log n_{samples})$ while the latter is bounded by the cost of reducing the BDD. Remember that the ROBDD encodes the logic of the surrogate,

that mimics the original model by solving the Equation 3.1. This means the surrogate, and in turn, the ROBDD is composed by fewer features, i.e. $m_{features} << n_{features}$, where $m_{features}$ is the number of features of the ROBDD. Hence, following [16] the cost of building such a ROBDD is $O(m_{features} \cdot \log m_{features})$. Since $n_{samples} >> n_{features} >> m_{features}$, the cost of Trace is bounded by $O(n_{samples} \cdot n_{features} \cdot \log n_{samples})$. On the other side, the *Explain* step depends on the cost of computing Equation 4.4, 4.5, 4.7, and 4.8, that requires (i) to manipulate two ROBDDs to compute Definition 4.4, 4.5, and (ii) to compute $sat(\cdot, \cdot)$ as argument of Equation 4.7, 4.8 over them. According to [16], the cost of the *apply* operator, needed for computing Equation 4.4, 4.5 is $O(m^{b_1}_{features} \cdot m^{b_2}_{features})$ while the cost of computing $sat(\cdot, \cdot)$ is $O(m_{features})$. Then, the cost of the *Explain* step is bounded by $O(m^{b_1}_{features} \cdot m^{b_2}_{features})$. Recalling $m_{features} << n_{features}$, the cost of a single iteration is bounded by the cost of generating the surrogate, that leads the cost of running the algorithm over all C classes to $O(|C| \cdot n_{samples} \cdot n_{features} \cdot \log n_{samples})$.

5

Experimental Results for Text

5.1 Datasets and Settings for Reproducibility

20newsgroups is a well-established benchmark used in [65] to build a reproducible text classifier, and in [108], to evaluate LIME's effectiveness in providing local explanations. It is composed of 18,446 newsgroup documents, partitioned evenly across 20 different topics (classes). The inputs D_1 and D_2 are organised by date, and their size is 11,314 and 7,532 news records, respectively, whilst the number of classes is 20 classes (i.e., multiclass).

In terms of running time, `ContrXT` never required more than 2 minutes to run on *20newsgroup*.

`ContrXT` is model-agnostic, as it can deal with any supervised machine learning algorithm.. We evaluated `ContrXT` in terms of approximation quality (i.e., the fidelity of the surrogate) to the input model to be explained. To this end, we ran `ContrXT` over different classifiers, trained through the most used algorithms, such as Linear Regression (LR), Random Forest (RF), Support Vector Machine (SVM) with RBF, Naive Bayes (NB), Bidirectional Gated Recurrent Unit (bi-GRU) [28] with a single hidden layer of dimension 100 and dropout, and Bidirectional Encoder Representations from Transformers (BERT) [32] *(bert-base-uncased)* with a sequence classification layer on top. Results are shown in Table 5.1. Hyperparameter selection was automated with a grid search. For each model, the best performing parameters were selected using 5-fold cross validation. The following set of hyper parameters were selected: (LR) {L2 penalty, no penalty}, (RF) max depth $\in \{5, 10, 20, 100\}\times$ number of estimators $\in \{50, 100\}$, (SVM) $C \in \{1, 10, 100\}\times$ gamma $\in$

{auto, scale}, (NB) $\alpha \in \{0.1, 0.01, 0.001\}$, (bi-GRU) epochs $\in \{5, 10, 20, 40\}\times$ batch size $\in \{8, 16, 32, 64\}$ optimiser $\in \{Adam, RMSprop\}$, (BERT) learning rate $\in \{5e-5, 2e-5\}\times$ eps $\in \{1e-8, 1e-5\}\times$ optimiser $\in \{Adam, AdamW\}$. To select the surrogate algorithms to implement within `ContrXT`, we considered and evaluated *all* the global surrogate models surveyed by [18], presenting the state of the art. Clearly, approaches whose outcome is limited to the feature importance values (e.g., SP-LIME [108] and k-LIME [58]) fall outside the goal of `ContrXT`, therefore, they were not considered. Unfortunately, we found that many papers surveyed by [18] do not provide the code freely available, thus they were discarded (e.g., [9, 66, 112]). Compared to the models providing code and using decision rules, `ContrXT` relies on decision trees to build the surrogate, though it might be extended to include additional surrogate algorithms in the future.

5.2 Evaluation on Benchmark

Table 5.1 `ContrXT` on 20newgroups (D_{t_1}, D_{t_2} from [65]) varying the ML algorithm. • is the best surrogate.

ML Algo	Model F1 weighted		Surrogate Fidelity F1 weighted	
	D_{t_1}	D_{t_2}	D_{t_1}	D_{t_2}
LR	0.88	0.83	0.76 ($\pm$0.06)	0.78 ($\pm$0.07)
RF	0.78	0.74	0.77 ($\pm$0.06)	0.79 ($\pm$0.07)
SVM	0.89	0.84	0.76 ($\pm$0.06)	0.78 ($\pm$0.06)
NB	0.91	0.87	0.76 ($\pm$0.06)	0.78 ($\pm$0.06)
bi-GRU	0.79	0.70	0.77 ($\pm$0.06)	0.78 ($\pm$0.06)
BERT	0.84	0.72	**0.78 ($\pm$0.05)** •	**0.83 ($\pm$0.06)** •

5.2.1 Analysis of Results

`ContrXT` provides to users (i) BDDs, (ii) indicators that estimate the change in the classification criteria, and (iii) BDD2Text to allow users reading the changes through text.

Indicators estimate the differences among the classification paths of the two BDDs through the *ADD* and *DEL* values (see Equation 4.7 and 4.8). To compare *ADD* and *DEL* across classes, we compute the *ADD_Global* (*DEL_Global*) as the number of paths to true in $b_{\oplus}$ ($b_{\ominus}$) over the corresponding maximum among all the $b^c_{\oplus}$ ($b^c_{\ominus}$) with $c \in C$. In the case of a multiclass classifier, as for 20newsgroup, `ContrXT` suggests focusing on classes that went through major alterations from ψ_1 to ψ_2, distinguishing between three groups according to their *ADD* and *DEL* values being above or below the 75th percentile.

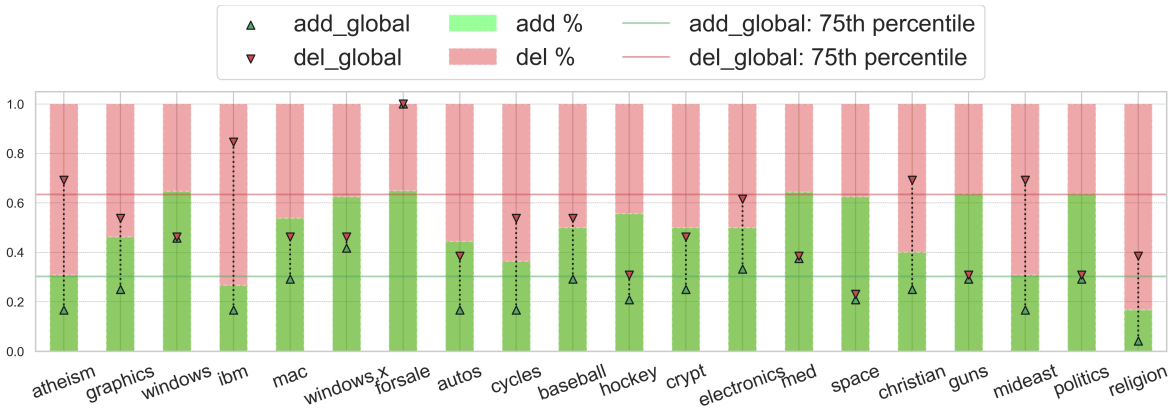


Figure 5.1 Indicators for the changes in classification paths from t_1 to t_2 for each 20newsgroup class, using a DT surrogate to explain a BERT classifier. On the x-axis we present the classification classes and on the y-axis the ADD/DEL indicators presented in Section 4.1.2.

Group 1 (55%) contains classes below both 75th percentile thresholds, showing the classifier did not change its criteria significantly;

Group 2 (40%) contains classes that have either *ADD_Global* or *DEL_Global* above the third quartile. This means there are some classes for which the classifier either added or deleted a number of criteria above the threshold;

Group 3 (5%) contains classes having both *ADD_Global* and *DEL_Global* values above the threshold. Those classes might have classification criteria which very differently with respect to the past.

One might inspect how the classification changes from ψ_1 to ψ_2 for each class, i.e., which are the paths leading to class c in b_1 (before) that lead to other classes at time b_2 (now) (*added paths*) and those who lead to c in b_2 that were leading to other classes in b_1 (*deleted paths*). Some examples are provided in Figure 5.2, showing classes having higher/lower of ADD and DEL values as in Figure 5.1. Specifically, one might concentrate on the class *atheism* that is provided in Figure 5.2a, as for this class, the number of deleted paths is higher than the added ones (Figure 5.1). Since a path (or classification rule) is a combination of criteria (or splitting rules) from the root to a leaf of the BDDs ($b_{\oslash}$ and $b_{\oslash}$), different paths can share one or more criteria. For example, the presence of the word *bill* leads the ψ_2 classifier to assign the label *atheism* to a specific record whilst the presence of such a feature was not a criterion for the ψ_1 classifier. Conversely, the BDD2Text demonstrates that ψ_1 used the feature *keith* to assign the label, whilst ψ_2 discarded this rule. Actually, both terms refer to the name of the posts' authors: *Bill*'s posts are only contained within D_{t_2} whilst *Keith*'s ones are more frequent in D_{t_1} rather than D_{t_2}. On the other side, ψ_2 discarded the rule *having political_atheist* that was sufficient for ψ_1 for classifying the instance.

BDD2Text allows the user to discover that regardless of BDD2Text allows discovering that though the accuracy of ψ_1 and ψ_2 is high, the underlying learning functions (i) learned terms that might be discarded during the preprocessing (e.g., the name of the posts' authors to decide whether a post is about atheism), (ii) ψ_2 persists in relying on those terms, which are changed after retraining (using *bill* instead of *keith*), and (iii) having *political_atheist* is no longer enough to classify in the class.

The example of Figure 5.2a sheds light on the goal of `ContrXT` : providing to the final user a way to investigate why ψ_2 classified documents to a different class with respect to ψ_1, as well as monitoring future changes. The generated explanations help the user in answering the driving questions we draw in Chapter 1, that are (Q1) observing the classification logic of a machine learning-based system over multiple training phases (Q2) for understanding why the newly trained model is classifying data differently after retraining on D_{t_2} through natural language (Q3).

ADD/DEL is not correlated with the Accuracy of the models. All classifiers perform well in terms of F1-score (see Table 5.1). To assess the presence of a correlation between *ADD*/*DEL* and the change in performance of the classifiers in terms of F1-score, we compute the Spearman's ρ between the *ADD* of every class and its change in F1-score between the two classifiers, and the equivalent for the *DEL*. The correlation values are not significant, $p = 0.21$, $\rho = -0.14$ for *ADD* and $p = 0.21$, $\rho = 0.14$ for *DEL*[1]. This confirms that *ADD* and *DEL* are not related to the F1-score of the trained model. Instead, they estimate its behaviour change handling new data, considering which classification paths have been added or deleted with respect to the past.

5.3 Evaluation through Human Subjects

User Study Design. We designed a study to assess if - and to what extent - final users can understand and describe what differs in the classifiers' behaviour by looking at BDD2Text outputs. We recruited 15[2] participants from *prolific.co* [96], an online service which provides participants for research studies. Prolific has been preferred to Amazon MTurk as the latter only has the US and Indian participants, lacking European ones. Participants were asked to look at four BDD2Text textual explanation and to select one (or more) statements according to the meaning they catch from BDD2Text. For example, a participant is asked to select among the following statements as they emerge from Figure 5.2a.

[1] Notice the two ρ values are mutual as $ADD + DEL = 1$

[2] Each participant is compensated £10

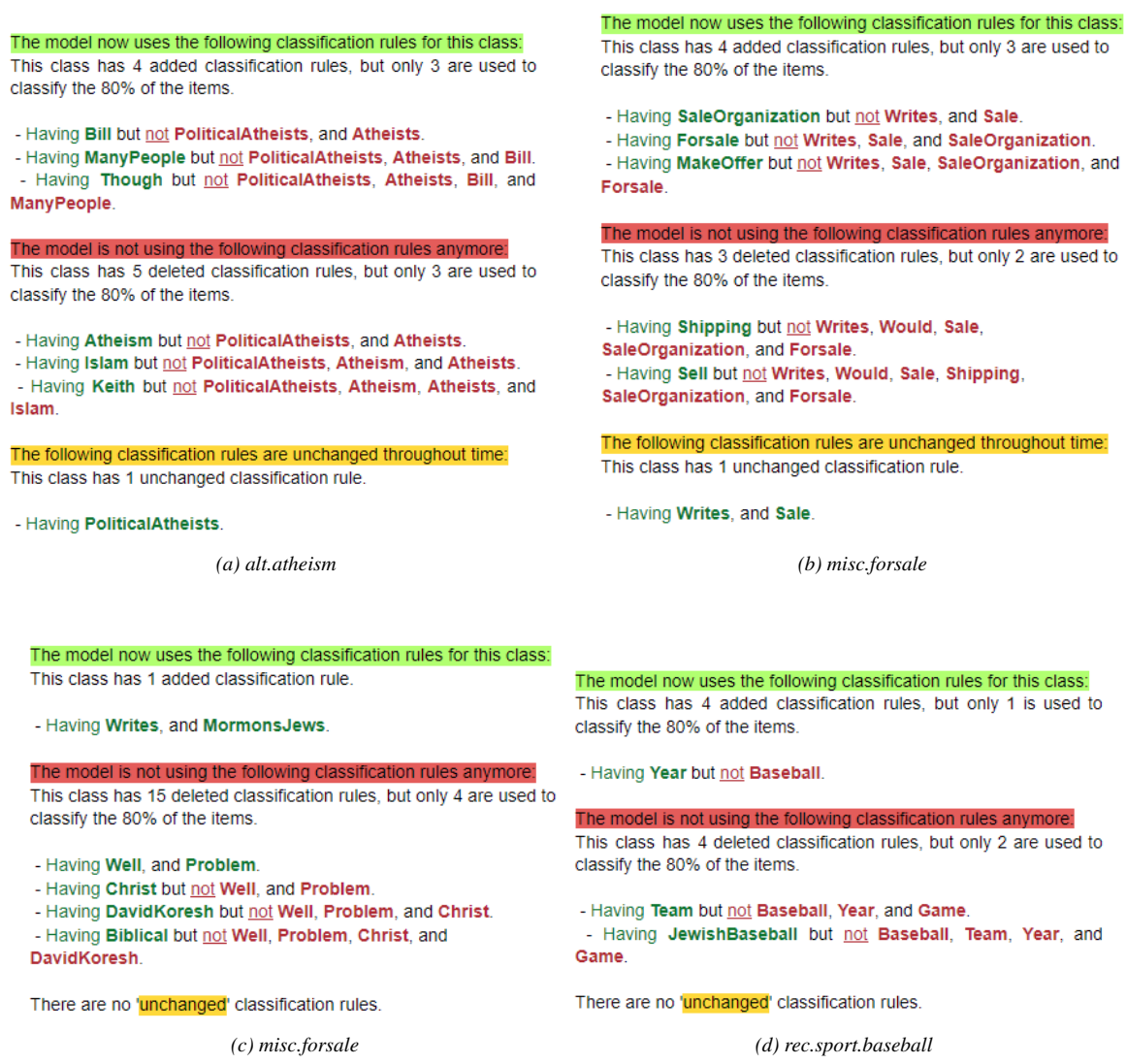


Figure 5.2 The `ContrXT` output in the multiclass case using the BERT model of Table 5.1 for (a) alt.atheism; (b) misc.forsale; (c) rec.sport.baseball; (d) talk.religion classes, using the BERT model of Table 5.1.

(A) Not having neither *"atheists"* nor *"political_atheists"* alone indicates a classification rule that applies in both t_1 and t_2;

(B) Together with other criteria, having *"keith"* can indicate both added and deleted paths

(C) Having *"bill"* but neither "atheists" nor *"political_atheists"* has been used in t_2 but not in t_1.

In the example above, only (C) can be derived from Figure 5.2a whilst (A) and (B) should be considered as erroneous, as they are not reflected in Figure 5.2a. Results showed that the participants understood the BDD2Text format and answered with an 89% accuracy on

average, and an F1-score of 87%. For the sake of transparency, the complete survey, along with the results of the survey and the transaction ids have been made publicly available on the github repo of `ContrXT`.

Do Subjects agree on the evaluation? Finally, we computed Krippendorff's alpha coefficient, which is a statistical measure of the extent of agreement among users. We reached an alpha value of 0.7, which Krippendorff considers as acceptable to positively assess the subjects consensus [70].

6

Experimental Results for Tabular

6.1 Datasets and Settings for Reproducibility.

Experiments have been carried out on eight different tabular benchmark datasets. Basic information about each and citations are provided in Table 6.1. We divide the experiments in two kinds of M-contrast: (i) *varying the learning function* while keeping the underlying data unchanged, and (ii) *varying the training data* while keeping the learning function unchanged. In the case of varying data, each dataset has been split in two, D_l and D_r based on the value of one attribute. The decision of which attribute and its value depends on each dataset:

Adult [68] The Adult dataset aims to predict whether a person earns more than $50k yearly based on the available socio-economic attributes. The split was performed on the person's age being above or below median.

Bank Marketing [91] Bank marketing's goal is to predict if a bank client will subscribe a term deposit. Split on two different marketing campaigns.

Wisconsin Breast Cancer [87] is a popular benchmark for cancer prognosis. Split on breast position (left/right).

Compas In Compas, each defendant is classified as a low, medium or high risk individual, given its criminal history and demographics. Split on age above or below median.

Cover [10] Cover classifies forest cover type from cartographic variables. Split on two different wilderness areas.

Occupancy [23] Occupancy is a binary classification dataset that estimates room occupancy from light, humidity, temperature and C02 values. Split on time of the day - between 8 am and 8 pm, or the opposite.

OnlineShoppers [110] OnlineShoppers collects user website sessions and aims to predict the intent to purchase an item. Split on day of the week, whether it is the weekend or not.

TD [103] TD classifies whether a patient is affected by thyroid disease. Split on age above or below median.

Table 6.1 Statistics and references about the datasets used for the experimental study.

Dataset	# Instances	# Classes	URL Source	Bib
Adult	48,000	2	UCI ML Repository	[68]
Bank Marketing	45,000	2	UCI ML Repository	[91]
Breast Cancer	286	2	UCI ML Repository	[87]
Compas	37,000	3	ProPublica Data Store	N/A
Cover	581,000	4	UCI ML Repository	[10]
Occupancy	20,000	2	UCI ML Repository	[23]
Online Shoppers	12,000	2	UCI ML Repository	[110]
TD	7,000	3	UCI ML Repository	[103]

`ContrXT` is model-agnostic, validated on its ability to replicate and explain any black box model. To this end, we ran `ContrXT` over different classifiers, trained through the most used algorithms, such as linear regression (LR), random forest (RF), support vector machines with RBF (SVM), Naive Bayes (NB) and eXtreme Gradient Boosting (XGB). Hyperparameter selection was automated with a grid search. For each model, the best performing parameters were selected using 5-fold cross validation. The following set of hyper parameters were selected: (LR) {L2 penalty, no penalty}, (RF) max depth $\in \{5, 10, 20, 100\}\times$ number of estimators $\in \{50, 100\}$, (SVM) $C \in \{1, 10, 100\}\times$ gamma $\in$ {auto, scale}, (NB) $\alpha \in \{0.1, 0.01, 0.001\}$, (XGB) default hyperparameters. After training the black box models, we have chosen the best performing one for each dataset, shown in Table 6.2. We also note the performances on each split with the F1 weighted score, which are in line with what is presented in the literature.

Due to `ContrXT`'s requirement of having only binary or ordinal variables as inputs, categorical attributes were represented with an one-hot encoding. In the case of continuous variables, a discretization step need to be applied beforehand. This has been implemented as a one-hot encoding in quantiles, with the number of quantiles being different for each dataset.

6.2 Consistency: Fidelity of Surrogate Models

Table 6.2 `ContrXT` *experimental results on tabular datasets, varying the underlying data. Performance in terms of F1 weighted score (F1w) on each dataset (left, right) of the best performing ML algorithm and its surrogates. Mean and standard deviation of surrogate models are calculated over each class of the dataset.*

Dataset	Best Algo	Model F1w		Surrogate Fidelity F1w	
		l	*r*	*l*	*r*
Adult	**RF**	0.94	0.87	0.89 (±0.00)	0.86 (±0.00)
Bank Marketing	**XGB**	0.89	0.89	0.87 (±0.00)	0.86 (±0.00)
Breast Cancer	**LR**	0.77	0.68	0.94 (±0.00)	0.87 (±0.00)
Compas	**SVC**	0.67	0.47	0.92 (±0.03)	0.86 (±0.05)
Cover	**RF**	0.86	0.85	0.87 (±0.07)	0.84 (±0.06)
Occupancy	**SVC**	0.97	0.99	0.99 (±0.00)	0.99 (±0.00)
Online Shoppers	**NB**	0.86	0.87	0.94 (±0.00)	0.95 (±0.00)
TD	**XGB**	0.92	0.90	0.93 (±0.01)	0.95 (±0.01)

Tabs. 6.2 and 6.3 show the average fidelity of the surrogates built by `ContrXT`, in the form of F1 weighted score; the first during the experiments varying the data, the latter varying the learning function. Hyperparameters for each model used in Table 6.3 noted in footnote [1]. This confirms the ability to consistently replicate the decisions of the black box model, as the fidelity is in all cases high, with up to 99% in the case of models trained on simple datasets such as the case of Occupancy.

The fidelity does not correlate with the discretization selected The hyperparameters of the surrogate model have a significant effect on fidelity. The maximum depth of the decision tree is the most significant one, as a more complex model has higher fidelity, but is also harder to explain. We have found a positive correlation between the two using an OLS linear regression (p=0.002). The second parameter we are interested in is the discretization method; in this case by modifying the number of quantiles in which the continuous variables are split. We have tested 2, 4 or 10 quantiles for each dataset. In this case, the fidelity does not seem to have a significant correlation with the number of quantiles in a OLS regression (p=0.069). We deduct that the discretization of continuous variables are not to be chosen automatically, it is dependent on the dataset and requires expert knowledge to be chosen. For example, for

[1] *(a)* C: 1, gamma: scale, kernel: rb, *(b)* C: 10, gamma: auto, kernel: rb, *(c)* C: 10, gamma: scale, kernel: rb, *(d)* C: 100, gamma: auto, kernel: rb, *(e)* alpha: 0.001, *(f)* alpha: 0.01, *(g)* alpha: 0.1, *(h)* max depth: 10, n estimators: 100, *(i)* max depth: 20, n estimators: 100, *(j)* max depth: 20, n estimators: 50, *(k)* penalty: l2, *(l)* Default hyperparameters

an "age" attribute, a specific domain could split the variable between children, middle aged people and elderly, while a different one might require a different approach.

```
The model now uses the following classification rules for this class:
This class has 1 added classification rule.

 - Having Light3Q but not Light4Q, Humidity2Q, and Temperature3Q.

There are no 'deleted' classification rules.

The following classification rules are unchanged throughout time:
This class has 1 unchanged classification rule.

 - Having Light4Q.
```

Figure 6.1 Global Natural Language Explanation for the Occupancy dataset, using tree surrogates.

6.3 Evaluation on Benchmark

In order to highlight the usefulness of ContrXT, in this section we show its output on the benchmark dataset *Occupancy* [23], varying the underlying data. The dataset deals with the prediction of occupancy in an office room using data from light, temperature, humidity and CO_2 sensors.

We split the original dataset in two: D_l comprises data obtained during daytime, between 8 AM and 8 PM, while D_r contains the remaining data obtained during nighttime. The accuracy of our trained black box models is high for both classifiers, as shown in tab 6.1, and, even after the numerical variables being discretised, it is comparable to the best performing model shown in the original paper - a Linear Discriminant Analysis (LDA) with 99.33% accuracy. The fidelities of the surrogate models are also very high, as the classifier is able to estimate the occupancy with few variables and a simple CART surrogate can mimic the original black box almost perfectly.

Our main objective by using ContrXT is to understand the differences between the two classifiers. The best way to do that is to inspect figure 6.1, which shows the output of the *Natural Language Explanation* from ContrXT . We can inspect the differences between the classification paths of the BDD generated by the two classifiers: one path has not changed between ψ_l and ψ_r; an high level of light, in the 4th quartile, means that the room is well lit and is the best indicator for showing whether it is occupied or not. There is also one added path in ψ_r: having the light variable in the 3rd quartile now leads to a positive classification, which was not true in ψ_l. We comment this by pointing out that during daytime the light in

this 3rd quartile would not have been sufficient to positively classify a data instance, but it is so during nighttime.

6.3.1 Dealing with Ordinal Attributes and Different Surrogate Models

```
The model now uses the following classification rules for this class:
This class has 1 added classification rule.

 - Having Co2 <= 2* Quantile, and Light = 3* Quantile.

There are no 'deleted' classification rules.

The following classification rules are unchanged throughout time:
This class has 3 unchanged classification rules, but only 2 are used to
classify the 80% of the items.

 - Having Light = 3* Quantile but not Co2 <= 2* Quantile.
 - If there are not Co2 <= 2* Quantile, and Light <= 2* Quantile.
```

Figure 6.2 Global Natural Language Explanation for the Occupancy dataset, using the RuleFit surrogate model and encoding the continuous variables with an ordinal discretisation.

ContrXT was only able to deal with the boolean presence or absence of a feature - which is adequate in the context of a text classifier, where features are words. In a tabular dataset, however, every continuous variable requires discretisation into an ordinal partition, and subsequently a one-hot encoding to transform it into multiple boolean attributes. For example, if the variable *Age* is split in four quartiles, ContrXT could find added decision paths such as *"the target class is now positive if the age is not in the third quartile"*; lacking any information about the other quartiles, as seen in Figure 6.1.

By not applying the one-hot encoding and keeping the attribute as an ordinal feature, ContrXT is now able to give more insightful explanations on continuous data. Using the same example of the *Occupancy* dataset, in Figure6.2 it is shown how the classification rules now have ordering. The new rules are not identical to Figure 6.1 as the underlying data encoding is different, however, it seems that the overall logic is similar, light being the common feature in both.

In this example we have also used the RuleFit [43] algorithm, that is available only for tabular classifiers, as it does not support the large number of features of text classifiers. If required, we remind that it is also possible to implement additional types of surrogate models.

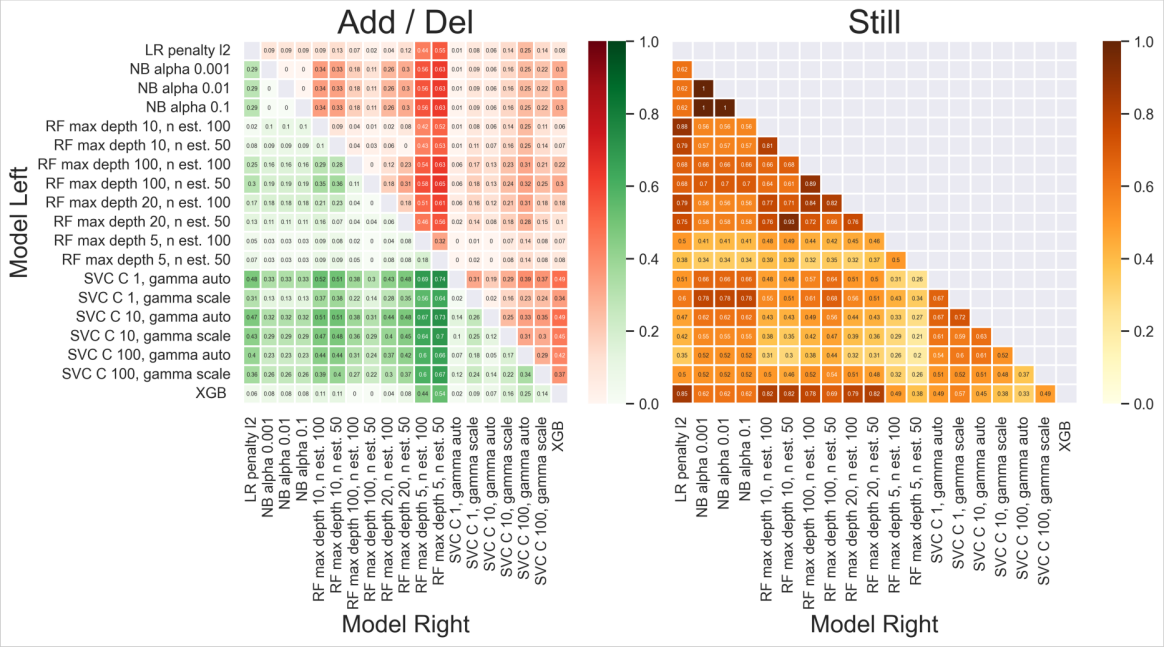


Figure 6.3 Heatmap showing a pairwise evaluation of the Add, Del, Still indicators with `ContrXT`, varying the Machine Learning model and hyperparameters used.

6.3.2 Get Rule Examples

The NLE shows the differences between the two models. However, a user might also wish to see example instances in the datasets where these rules apply.

To do so, `ContrXT` provides the *get_rule_examples* function, which requires the user to specify a rule to be applied and the number of examples to show. `ContrXT` applies the rule to D_l and D_r, specifying the number of document classified by that rule and provides some examples, highlighting the portion in which the rule applies, as in Figure 6.5.

Notice this function is also useful to check the consistency of a specific rule, that is, for an *add* rule, its prevalence should be higher in D_l, for a *del* rule the opposite, while for a *still* rule the should be roughly equivalent in both D_l and D_r.

6.3.3 Indicators

`ContrXT` also summarises quantitatively the changes through Add/Del indicators as presented in step (D). Since *Add_Global* and *Del_Global* are applicable only in the multiclass case, in Figure 6.4 we show the results for the *Cover* dataset, a multiclass example. One can inspect which classes have changed the most, focusing on the ones that went through major alterations.

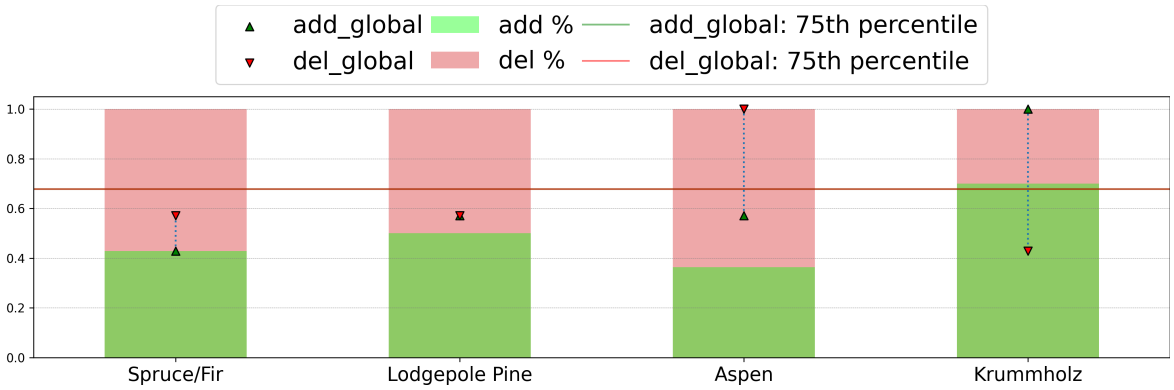


Figure 6.4 Indicators for the changes in classification paths from D_l to D_r for each class in the Cover dataset, using a DT surrogate to explain the change in two Random Forest classifiers. On the x-axis we present the classification classes and on the y-axis the Add/Del indicators Step (E).

6.3.4 `ContrXT` helps understanding the coherence between machine learning models

Different machine learning models might have similar performances on the same data, but this does not guarantee their decision function are similar as well. For instance, in Table 6.3 we can observe that for the dataset Adult the worst machine learning models present a an F1-weighted of 0.744, but this does not mean that at least 74.4% of the instances have been classified with the same rules by all the classifiers. In this section, we want to show how `ContrXT` can be used to understand how much the behaviour of two machine learning classifiers is similar, beyond their performances. To this aim, we trained 19 machine learning models on the same dataset, Adult, and we perform a pairwise evaluation of them using `ContrXT`. In Figure 6.3 we present two heatmaps. In the first one, we show the Add of the model on the row in green in the lower triangular matrix and its Del in red in the upper triangular. In the second matrix, which is symmetric, we show the Still. As we can see, those value vary considerably across the different models. For instance, when we train on the same dataset two NB classifiers only changing the value of alpha from 0.1 to 0.01, we would expect the model will behave in a similar way. Indeed, the still is 1 (i.e., the rules found by `ContrXT` for the two models are the the same) and consequently there are no added or deleted rules. This also shows that if two models are coherent, the explanations generated from `ContrXT` are coherent as well. For other models, the behaviour is completely different. For instance, the NB models add 29% of the rules w.r.t the logistic regression, and delete 9% of the rules. As a consequence, the Still between those models is 0.62. In some cases the still is even lower than 0.5, with a minimum of 0.2 between the RF with 50 estimators and max depth of 5 and the SCV with C=100, meaning than training those two models on the same

Rule **ADD**: {Light3Q: 1, Light4Q: 0, Humidity2Q: 0, Temperature3Q: 0}

Overall, the rule appeared 736 times in D_{left}.
Out of these, 18 (2.5%) belong to the positive class.

	Temperature2Q	Temperature3Q	Temperature4Q	Humidity2Q	Humidity3Q	Humidity4Q	Light2Q	Light3Q	Light4Q
135	0	0	0	0	1	0	0	1	0
140	0	0	0	0	0	1	0	1	0
199	1	0	0	0	0	1	0	1	0

In D_{right}, the rule appeared 109 times.
Out of these, 36 (33%) belong to the positive class.

	Temperature2Q	Temperature3Q	Temperature4Q	Humidity2Q	Humidity3Q	Humidity4Q	Light2Q	Light3Q	Light4Q
38	0	0	0	0	1	0	0	1	0
157	1	0	0	0	0	0	0	1	0
422	1	0	0	0	0	1	0	1	0
1013	0	0	0	0	1	0	0	1	0
1086	1	0	0	0	0	0	0	1	0

Figure 6.5 `ContrXT` *shows examples in which a rule applies in the Occupancy dataset.*

dataset, even if they have similar performances, they might learn largely different decision functions. This confirms the fact that in some cases the similarity of the performances is not enough to evaluate the similarity between different models, and that the model contrastive explanations generated by `ContrXT` capture those difference and might help humans to better understand the difference between classification models.

Table 6.3 `ContrXT` *experimental results, varying the underlying learning function. Performance in terms of F1 weighted score (F1w) on each dataset (left, right) of each ML algorithm and its surrogates.*

Dataset	Model Contr		F1w		Surr. F1w	
	l	*r*	*l*	*r*	*l*	*r*
Adult	LR^{k}	RF^{j}	.845	.839	.90 (±0)	.85 (±0)
		SVC^{a}		.744		.89 (±0)
		NB^{g}		.793		.92 (±0)
		XGB^{l}		.843		.86 (±.01)
	RF^{j}	SVC^{a}	.839	.744	.85 (±0)	.89 (±0)
		NB^{g}		.793		.92 (±0)
		XGB^{l}		.843		.86 (±.01)
	SVC^{a}	NB^{g}	.744	.793	.89 (±0)	.92 (±0)
		XGB^{l}		.843		.86 (±.01)
	NB^{g}	XGB^{l}	.793	.843	.92 (±0)	.86 (±.01)
BankMarketing	LR^{k}	RF^{i}	.920	.920	.96 (±.01)	.87 (±.02)
		SVC^{a}		.781		.78 (±.02)
		NB^{e}		.893		.94 (±0)
		XGB^{l}		.913		.89 (±0)
	RF^{i}	SVC^{a}	.920	.781	.87 (±.02)	.78 (±.02)
		NB^{e}		.893		.94 (±0)
		XGB^{l}		.913		.89 (±0)
	SVC^{a}	NB^{e}	.781	.893	.78 (±.02)	.94 (±0)
		XGB^{l}		.913		.89 (±0)
	NB^{e}	XGB^{l}	.893	.913	.94 (±0)	.89 (±0)
BreastCancer	LR^{k}	RF^{i}	.752	.748	.86 (±0)	.69 (±.08)
		SVC^{d}		.734		.71 (±.05)
		NB^{g}		.749		.95 (±.07)
		XGB^{l}		.702		.65 (±.03)
	RF^{i}	SVC^{d}	.748	.734	.69 (±.08)	.71 (±.05)
		NB^{g}		.749		.95 (±.07)
		XGB^{l}		.702		.65 (±.03)
	SVC^{d}	NB^{g}	.734	.749	.71 (±.05)	.95 (±.07)
		XGB^{l}		.702		.65 (±.03)
	NB^{g}	XGB^{l}	.749	.702	.95 (±.07)	.65 (±.03)
Compas	LR^{k}	RF^{h}	.669	.647	.87 (±.07)	.86 (±.06)
		SVC^{a}		.582		.85 (±.06)
		NB^{g}		.658		.91 (±.03)
		XGB^{l}		.628		.84 (±.05)
	RF^{h}	SVC^{a}	.647	.582	.86 (±.06)	.85 (±.06)
		NB^{g}		.658		.91 (±.03)
		XGB^{l}		.628		.84 (±.05)
	SVC^{a}	NB^{g}	.582	.658	.85 (±.06)	.91 (±.03)
		XGB^{l}		.628		.84 (±.05)
	NB^{g}	XGB^{l}	.658	.628	.91 (±.03)	.84 (±.05)

Table 6.3 `ContrXT` experimental results, varying the underlying learning function (continued).

Dataset	Model Contr l	Model Contr r	F1w l	F1w r	Surr. F1w l	Surr. F1w r
Cover	LR^k	RF^i	.719	.758	.92 (±.10)	.78 (±.18)
		SVC^a		.740		.66 (±.38)
		NB^g		.666		.88 (±.09)
		XGB^l		.760		.90 (±.12)
	RF^i	SVC^a	.758	.740	.86 (±.12)	.66 (±.38)
		NB^g		.666		.84 (±.12)
		XGB^l		.760		.78 (±.15)
	SVC^a	NB^g	.740	.666	.66 (±.38)	.92 (±.04)
		XGB^l		.760		.89 (±.09)
	NB^g	XGB^l	.666	.760	.84 (±.12)	.78 (±.15)
Occupancy	LR^k	RF^j	.976	.980	1.00 (±0)	.99 (±0)
		SVC^b		.980		.99 (±0)
		NB^g		.951		.99 (±0)
		XGB^l		.981		.99 (±0)
	RF^j	SVC^b	.980	.980	.99 (±0)	.99 (±0)
		NB^g		.951		.99 (±0)
		XGB^l		.981		.99 (±0)
	SVC^b	NB^g	.980	.951	.99 (±0)	.99 (±0)
		XGB^l		.981		.99 (±0)
	NB^g	XGB^l	.951	.981	.99 (±0)	.99 (±0)
OnlineShoppers	LR^k	RF^i	.888	.892	.93 (±0)	.85 (±.01)
		SVC^b		.893		.96 (±0)
		NB^f		.855		.95 (±0)
		XGB^l		.886		.90 (±.01)
	RF^i	SVC^b	.892	.893	.85 (±.01)	.96 (±0)
		NB^f		.855		.95 (±0)
		XGB^l		.886		.90 (±.01)
	SVC^b	NB^f	.893	.855	.96 (±0)	.95 (±0)
		XGB^l		.886		.90 (±.01)
	NB^f	XGB^l	.855	.886	.95 (±0)	.90 (±.01)
TD	LR^k	RF^j	.941	.934	.93 (±.01)	.81 (±.07)
		SVC^c		.928		.89 (±.07)
		NB^g		.935		.93 (±.01)
		XGB^l		.928		.82 (±.06)
	RF^j	SVC^c	.934	.928	.81 (±.07)	.89 (±.07)
		NB^g		.935		.93 (±.01)
		XGB^l		.928		.82 (±.06)
	SVC^c	NB^g	.928	.935	.89 (±.07)	.93 (±.01)
		XGB^l		.928		.82 (±.06)
	NB^g	XGB^l	.935	.928	.93 (±.01)	.82 (±.06)

7
Building the Tool

The tool has been implemented as a Python library, and can be installed with the standard pip package installer.

The input parameters that users needs to specify are as follows:

- **(Required)** the *feature data* (can be training or test, labelled or unlabelled). Text data must be in the form of a list of documents, while tabular data as a Pandas dataframe;
- **(Required)** the corresponding *labels* predicted by the classifier;
- **(Optional)** a *save path* where to save all the results, as .csv files and images. You can choose with different parameters whether you want to save the fidelities as .csv files, the surrogate model paths, and the *BDD* represented as images
- **(Optional)** whether you want to perform an *hyperparameter selection* on the surrogate models, and if so, you can specify the grid search parameters;
- **(Optional)** the *surrogate algorithm* to use (*CART* is the default);
- **(Optional)** the *coverage* of the dataset to be used (100% as default), otherwise a sampling procedure is used (as in Procedure 1, lines 5-16);
- **(Optional)** the Γ value (as in Definition 3.2.2) as a measure of complexity of the surrogate, e.g. the max number of decision rules (leaf nodes) in case of DT.

The tool automatically checks if the couple *(data type, surrogate algorithm* is viable. For example, using the Rulefit algorithm with text data is not recommended, because of the long computation times due to large number of features, and the user will be warned.

7.1 Implementation

The tool is implemented in two main classes, called *Trace* and *Explain*, as described by Sec. 4.1.1 and 4.1.2. They contain the main logic described in the algorithm. In this section, we will explain in detail their technical implementation in Python.

7.1.1 Trace

The *Trace* class deals with generating surrogate models and the corresponding BDDs for each category from input data, as in Sec. 4.1.1. First, it assigns internally input parameters, such as the grid search hyperparameter settings, what kind of output data and images to save and the path where they should be located, the surrogate to use and the type of data (text or tabular). The training data and labels are saved in the *DataManager* class, a support class. Depending on the type of data, a *TextDataManager* or *TabularDataManager* class can be instantiated. Both inherit the same functions from *DataManager*, but the implementation is not equal, due to the intrinsic differences in the data types. Two instances of *DataManager* are instantiated during the *Trace* step, one for each dataset D. This class performs some useful functions:

- First, it checks if the column names are valid. Since they are used as feature names in the BDDs, they should not contain some special characters which could interfere with the implementation in Python. The pyEDA [1] library is later used for synthesising BDDs, and it does not accept some type of character in the feature names, e.g. the question mark. When using textual data, this check is performed on the training corpus, since the words in the documents will become BDD features;
- Then, it performs a sampling of the dataset if requested by the user. The percentage of data is an input parameter, defaulted to 1, meaning no sampling will be performed without being explicitly asked to do so;
- It organises the data by assigning the training data and black box labels for each $c \in C$ in a dictionary. If the data is composed of text documents, they must also be transformed to a one-hot encoded sparse matrix.

The tool uses the scikit-learn [101] or Rulefit [43] libraries for generating surrogates. Those are implemented respectively in the *SklearnSurrogate* and *RulefitSurrogate* classes, both inheriting from the base class *GenericSurrogate*. In the future it is possible to eas-

[1] https://github.com/cjdrake/pyeda

ily extend the tool to add other surrogate algorithms, by adding new children classes to *GenericSurrogate*. The main functions to implement are:

- Hyperparameters selection using 5-fold cross validation, as explained in Sec. 2.2. The grid search parameters can be chosen by the user;
- Fit the surrogate model to the available data and black box predictions using the parameters with the highest fidelity from the previous step;
- Compute the fidelity score between the surrogate predictions and the original black box labels to evaluate the performance;
- Transform the interpretable model to a BDD string parsable by pyEDA;
- Optionally save the surrogate model for inspection, for example with a visualization.

The implementation differs based on the surrogate used, for example the transformation of the surrogate to a *BDD* string using scikit-learn decision trees is shown in Listing 7.1.

```
def surrogate_to_bdd_string(self):
    '''Transform a scikit-learn surrogate decision tree to BDD string
    using depth first search.
    '''
    stack = []
    self.bdd = []

    def _tree_recurse(node):
        if self.tree.feature[node] == TREE_UNDEFINED:
            # Leaf node, base case
            value = np.argmax(self.tree.value[node][0])
            if value == 1:
                path = ' & '.join(stack[:])
                self.bdd.append(path)
                self.paths[path] = self.tree.n_node_samples[node]
            return

        # Recursion case
        name = self.feature_names[self.tree.feature[node]]
        stack.append(f'~{name}')

        # Recurse to the left children
        _tree_recurse(self.tree.children_left[node])

        stack.pop()
        stack.append(name)

        # Recurse to the right children
        _tree_recurse(self.tree.children_right[node])
```

```
29
30          stack.pop()
31
32      # Start with the root of the tree
33      _tree_recurse(0)
34      self.bdd = ' | '.join(self.bdd)
```

Code 7.1 Transforming a scikit-learn decision tree to a BDD string.

The algorithm traverses the tree using depth-first search to write a pyEDA valid string. A valid string is in the form of "a & ~ b | a & b & ~ c", where & defines logical AND, | defines logical OR, ~ defines logical NOT, and the literals represent feature names. Line 4 and 5 instantiate two lists, the first being a stack that keeps track of which features are visited in the currently evaluated classification paths, the second is a list that records already visited paths. The procedure begins at the root of the tree, node 0, and uses recursion to navigate each classification path until its end, first to the left child of the current node and then to the right. Every downward traversal adds a feature to the stack, moving to left child represents a negation on the feature, while the right child includes a positive feature. The base case of the recursion corresponds with a leaf, that is implemented in scikit-learn as a node where the feature is equal to the TREE_UNDEFINED value. Since it represents the end of a classification path, the features that have been collected are joined by the & symbol and are appended to the list of paths. Only the paths that result in the positive class are added to form a valid pyEDA string. Finally, after traversing the whole tree, the list of paths are all joined with the | operator.

To recap, the *Trace* class is responsible for creating a surrogate for each $c \in C$, using the surrogate models and the data managers, converting them to pyEDA strings as described above and saving the results for the following *Explain phase*. Table 7.1 shows a potential output of the *Trace* step.

Table 7.1 An example output of the Trace step. Some columns are not shown for shortness: dataset, runtime (seconds), percentage of data, hyperparameter settings of the surrogate. additional fidelity indicators e.g. accuracy score.

Class ID	BDD String	F1 Score	Recall	Precision
Atheism	~God & Atheists | God & PoliticalAtheists	0.969	0.944	0.995
Graphics	Image & Format | Graphics & ~Image	0.989	0.981	0.998
Windows	~Windows & Dos | Windows & Microsoft	0.949	0.91	0.991
Hardware	~Pc & Motherboard | ~Drive & ~Card & Pc	0.978	0.962	0.995
Mac	Mac & Apple | ~Mac & Centris	0.97	0.95	0.99

7.1.2 Explain

The *Explain* step can be executed after *Trace*, as explained in Sec. 4.1.2. It deals with manipulating the BDDs generated to explain how ψ_1 and ψ_2 differ/similar.

The pyEDA library is a Python library for electronic design automation that features symbolic Boolean algebra with a selection of function representations, such as logic expressions, truth tables with three output states (0, 1, "don't care"), and BDDs. It is used to perform the logical operations *Add*, *Del* and *Still* described in Eq.4.7, 4.8 and 4.9. Listing 7.2 shows their simple implementation, where $s1$ and $s2$ are two strings as shown in Table 7.1:

```
from pyeda.inter import expr, expr2bdd

f = expr2bdd(expr(s1))
g = expr2bdd(expr(s2))

f_add = ~f & g
f_del = f & ~g
f_still = f & g

sat_add = len(list(f_add.satisfy_all()))
sat_del = len(list(f_del.satisfy_all()))
sat_still = len(list(f_still.satisfy_all()))
```

Code 7.2 Using PyEDA to manipulate BDDs.

The operations above are performed for each $c \in C$. Table 7.2 shows an example of the indicators output of the *Explain* step. Other metrics are saved, such as the number of features in each BDD, their union and Jaccard similarity as in Eq. 4.10. The Add, Del and Still BDDs can be visualized using the Graphviz software, an example of this output is seen in Figure 3.1.

Table 7.2 An example output of the Explain step. Some columns are not shown for shortness: number of features (nodes) in each BDD, their union and Jaccard similarity, runtime (seconds).

Class ID	Add	Del	Still	Sat Add	Sat Del	Sat Still
Atheism	0,667	0,167	0,167	4	1	1
Graphics	0,333	0,333	0,333	2	2	2
Windows	0,2	0,2	0,6	1	1	3
Hardware	0,333	0,333	0,333	2	2	2
Mac	0,333	0,667	0	1	2	0

Each path of the resulting BDDs is a rule that can be used to find the number of occurrences matching it in the whole dataset, in the subset of the expected class and it can be used

Rule: {'Developer': 1, 'TechLead': 0, 'Project': 0}
Overall, the rule appeared 1349 times.
Out of these, 1326 (98.3%) belong to the class Software Developers.

Some example instances:
- **Developer** Python PythonDeveloper
- [...] BackendDeveloperLondon Remote£75 Remote£75000 **Developer** LondonFully Fully
- **Developer** NetDeveloper Net
- **Developer** C++ C++Developer
- **Developer** JuniorDeveloper Junior

Figure 7.1 Showing examples of a particular rule in the data.

to find some relevant examples for further inspection. Figure 7.1 shows examples that match the rule *"Developer & ~TeachLead & ~Project"*. The user can choose the number of rules to show, up to the maximum of instances that match the rule in the dataset. This function can highlight the consistency of a specific rule.

Finally, Natural Language Explanations (NLE) exhibits the added/deleted paths derived from $b_{\oslash}$, $b_{\oslash}$ and $b_{\ominus}$ to final users through natural language. Our methodology uses the last four steps of *six NLG tasks* described by [49], responsible for *microplanning* and *realisation*. In our case, the structured output of BDDs obviates the necessity of *document planning* which is covered by the first two steps.

The explanation is composed of three main parts, corresponding to Add, Del and Still paths. Content of each part is generated by parsing the BDDs, extracting features, aggregating them using Frequent Itemsets technique [106] to reduce the redundancy, inserting the related parts in the predefined sentences [109] and finally, adding colours and styling to increase the comprehensibility of the explanations by the final users. A post-processing step is also implemented in the NLE, which aggregates redundant variables - e.g., if a rule states that a certain variable has to be at the same time less than 5 and less than 4, then only the more inclusive constraint of being less than 5 is shown to the user. This improves the readability of the explanation and decreases its length.

7.2 Installation

`ContrXT` is available on PyPi[2]. Simply run:

```
pip install contrxt
```

[2]https://pypi.org/project/contrxt/

Or clone the Github repository and run:

```
pip install .
```

The PyEDA package is required but has not been added to the dependencies. This is due to installation errors on Windows. If you are on Linux or Mac, you should be able to install it by running:

```
pip3 install pyeda
```

However, if you are on Windows, we found that the best way to install is through Christophe Gohlke's pythonlibs[3] page. For further information, please consult the official PyEDA installation documentation[4].

To produce the PDF files, a Graphviz installation is also required. Full documentation on how to install Graphviz on any platform is available at https://graphviz.org/download/.

Basic unit tests are provided. To run them, after installation, execute the following command while in the main directory:

```
python -m unittest discover
```

7.3 eXplainable AI as a Service

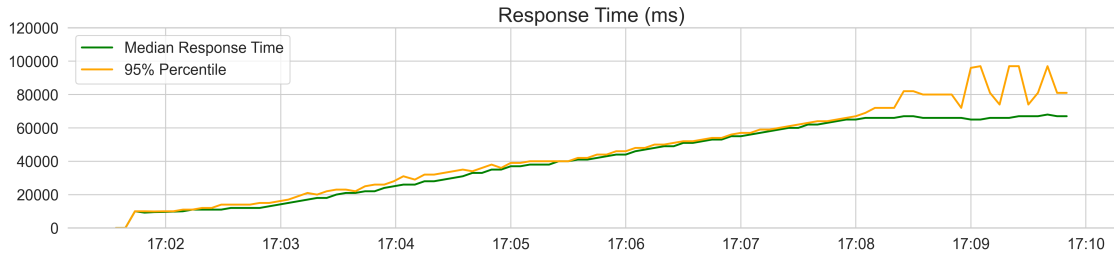
The usability of the tool by non-technical users plays a crucial role in the design of our approach. For this reason We provide REST API [39] to generate explanations for any text classifier, similar to [24] which provide an API for XAI Planning.

Our API enables users to get the outcome of `ContrXT`, i.e. Indicators and BDD2Text explanations without installing and configuring `ContrXT` locally. As for the `ContrXT` tool, the required input from user are the training data and the predicted labels by the classifier of their choice. The detail of such input is described below.

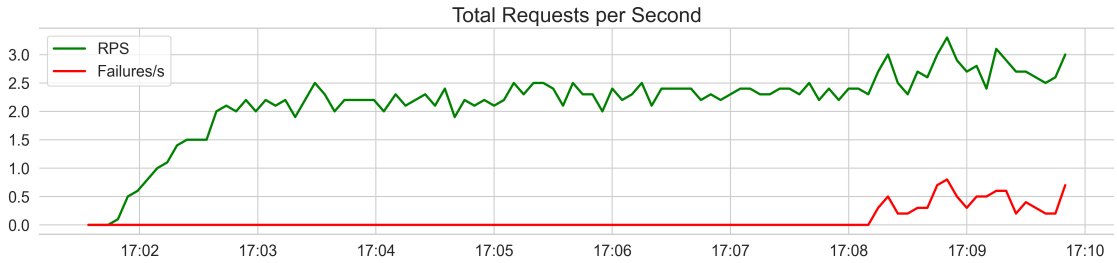
The API is written using Python and the Flask library [54]. Users are required to upload two csv files for time 1 and 2 (see Chapter 7) for which the schema is shown in the following JSON.

```
schema = {
    "type" : "csv",
    "columns" : {
        "corpus" : {"type" : "string"},
        "category" : {"type" : "string"},
```

[3] https://www.lfd.uci.edu/~gohlke/pythonlibs/#pyeda
[4] https://pyeda.readthedocs.io/en/latest/install.html



(a) MRT (median response time , green)



(b) The Request per Seconds (throughput, green) and the number of failures (requests reached the 5 min timeout, red)

Figure 7.2 Load testing provided by Locust.io: (a) MRT (median response time); (b) The Request per Seconds and the number of failures

```
        "predicted" : {"type" : "string"},
    },
}
```

Code 7.3 API schema

To validate this schema we use the *jsonschema* package[5]. According to Code 7.3, each csv is expected to have three columns respectively for corpus (texts to be classified), label (the true label for each text) and predicted (the outcome of the classifier). The API can be invoked using a few lines code shown in Code 7.4.

```
import requests, io
from zipfile import ZipFile
files = {
    'time_1': open(t1_csv_path, 'rb'),
    'time_2': open(t2_csv_path, 'rb')
}
r = requests.post('[see the URL and port on github repo]',files=files
    )
result = ZipFile(io.BytesIO(r.content))
```

Code 7.4 Complete Python code to call `ContrXT` API

[5]https://python-jsonschema.readthedocs.io/en/stable

7.3.1 Load Testing

A load testing has been performed using Locust[6] to measure the quality of service of the API deployed and accessible to the community[7]. Specifically, we followed [82] to determine the number of users/requests our API web server can tolerate in order to guarantee an acceptable response time (set to 5 minutes) while increasing the throughput, i.e., requests per second.

According to [82], a load testing is valid if the virtual users' behaviour is similar to those of actual users. Then, to allow for reproducibility, we restricted our load testing to add a virtual user every 10 seconds, executing the whole procedure for the *20newsgroups* dataset for each[8]. The results show our architecture has a throughput of 2.55 users per second (see Figure 7.2. Above this value, the API service keeps working, putting additional requests into a queue.

[6] locust.io

[7] see the github repository for connection details

[8] Time needed to upload/download datasets and to generate PDF versions of the BDDs are not considered

Part III

Applications of XAI to Labour Market Intelligence

Over the past several years, the growth of web services has been making available a massive amount of structured and semi-structured data in different domains. An example is the web labour market, with a huge number of Online Job Advertisements. An Online Job Advertisement (OJA, *aka*, job offers, job vacancy) is a document containing a *title* - that shortly summarises the job position - and a *full description*, usually used to advertise the skills a candidate should hold. available through web portals and online applications. The problem of processing and extracting insights from OJAs is gaining researchers' interest in the recent years, as it allows modelling and understanding complex labour market phenomena (see, e.g. [132, 131, 13, 30]).

In this chapter, we discuss applications applying AI and contrastive explanations to OJAs that aim to explain the evolution of the labour market and help experts understand the undergoing changes.

8

Contrastive Explanations in a Real-life Scenario: The Case of Online Job Ads

8.1 The practical significance of applying AI to Job Ads

In recent years, the European labour demand conveyed through specialised web portals and services has grown exponentially. This also contributed to introducing the term *"Labour Market Intelligence"* (LMI), which refers to the use and design of AI algorithms and frameworks to analyse labour market data for supporting decision making (see, e.g., [51, 119, 64, 11, 53, 132]).

Nowadays, the problem of monitoring, analysing, and understanding labour market changes (i) timely and (ii) at a very fine-grained geographical level has become practically significant in our daily lives. Recently, machine learning has been applied to compute the effect of robotisation within occupations in the US labour market [42] as well as to analyse skill relevance in the US standard taxonomy O*Net [2], just to cite a few. In 2016 the EU Cedefop agency - aimed at supporting the development of European Vocational Education and Training - has launched a European tender for realising a machine-learning-based system able to collect and classify Web job adverts from all 28 EU country members using the ESCO hierarchy for reasoning over the 32 languages of the Union [25], see [11, 13]). From a statistical perspective, in 2016, the EU and Eurostat launched the ESSnet Big Data project [37], involving 22 EU member states to integrate big data in the regular production of

official statistics. In the late 2020, EUROSTAT and Cedefop have joined forces announcing a call for tender [38] aimed at establishing results from [25] fostering AI and Statistics to build up the European Hub of Online Job Ads.

As one might note, the use of classified OJAs and skills, in turn, enables several third-party research studies to understand and explain complex labour market phenomena. To give few recent examples, in [30] we used OJAs for estimating the impact of AI in job automation and measuring the impact of digital/soft skills within occupations; In [50, 51] we used classified OJAs to identify new emerging occupations. In [50] we used classified OJAs to build a recommendation system of skills for citizens, while in [50] we implemented the first graph database of the European labour market to be explored through graph-traversal queries.

In May 2020, the EU Cedefop Agency has been started using those OJAs to build an index named Cov19R that identifies workers with a higher risk of COVID-19 exposure, who need greater social distancing, affecting their current and future job performance capacity[1].

All these initiatives and research studies elucidate the importance of explaining the rationale behind the classification process for decision-makers. Specifically, a crucial aspect is related to the explaining of the rationale *over time*, that is, explaining if - and to what extent - a retrained classifier is still working coherently with respect to the past, plays a crucial role to guarantee the trustworthiness of the analyses.

8.2 Analysis of `ContrXT` Results on OJA Dataset

We applied `ContrXT` on an OJA dataset composed of $100,000$ documents, partitioned evenly across the 50 most popular ESCO codes (i.e., multiclass). The size of the inputs D_1 and D_2 is organised by date, with D_1 belonging to 2016 and D_2 to 2018, that is 50,000 records for each dataset.

As the indicators described in Section 5.2, we distinguish the classes in three groups.

Group 1 (30%) contains classes below both thresholds, showing the classifier did not change its criteria significantly;

Group 2 (22%) contains classes that have either *ADD_Global* or *DEL_Global* above the third quartile. This means there are some classes for which the classifier either added or deleted a number of criteria above the threshold;

[1] https://tinyurl.com/cedefop-covid

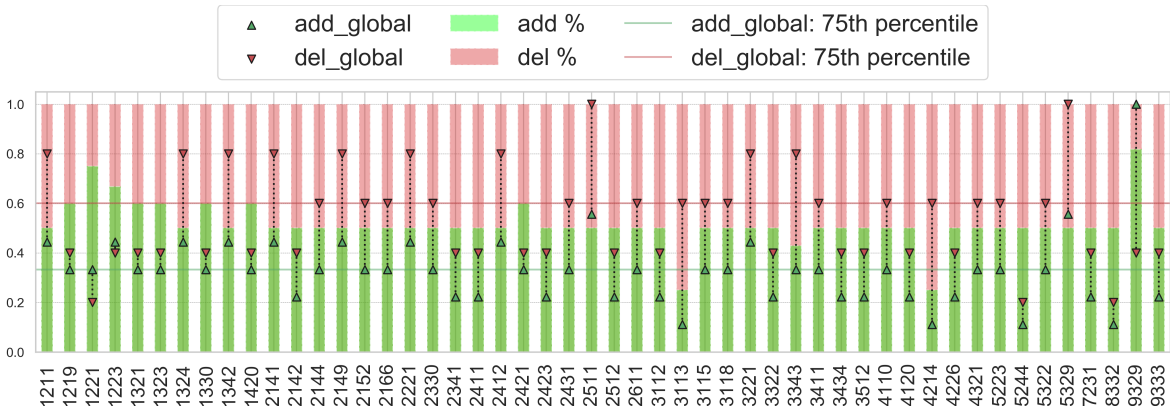


Figure 8.1 Indicators for the changes in classification paths from t_1 to t_2 for each OJA class, using a DT surrogate to explain a Random Forest classifier. On the x-axis, we present the classification classes and on the y-axis, the add/del indicators are presented in Section4.1.2. To access the full name of the concept, add the prefix http://data.europa.eu/esco/isco/C to the class code.

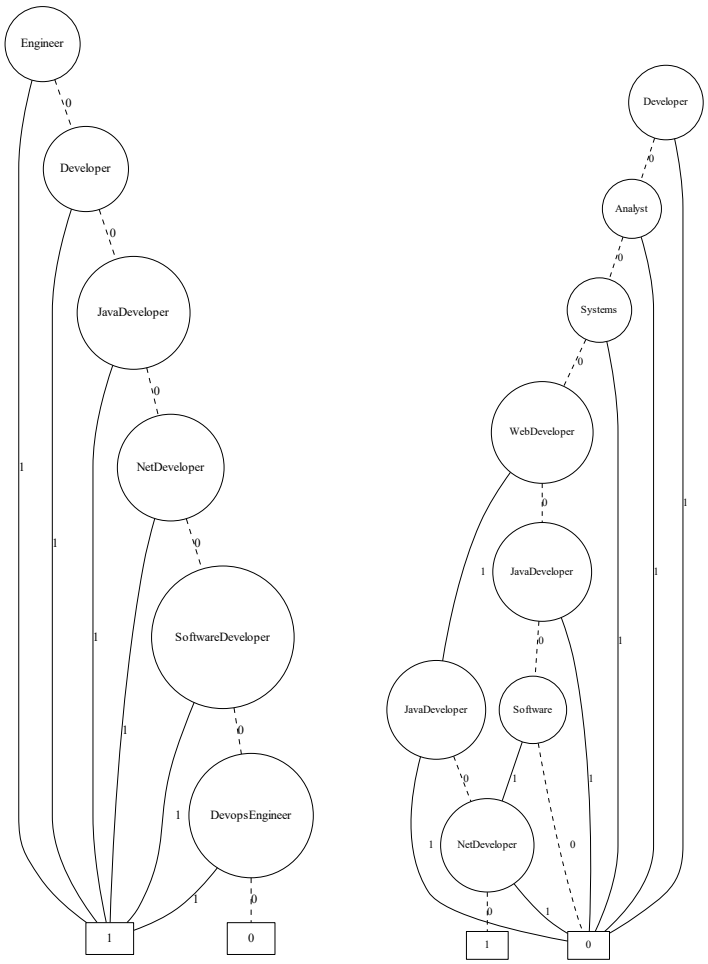
Group 3 (48%) contains classes having both *ADD_Global* and *DEL_Global* values above the threshold. Those classes might have been classified very differently with respect to the past.

Table 8.1 `ContrXT` on OJA dataset varying the ML-algorithm. • is the best surrogate.

ML Algo	Model F1 weighted		Surrogate Fidelity F1 weighted	
	D_{t_1}	D_{t_2}	D_{t_1}	D_{t_2}
LR	0.79	0.78	0.86 (±0.06)	0.85 (±0.06)
RF	0.76	0.77	**0.89 (±0.07)** •	**0.87 (±0.06)** •
SVM	0.79	0.82	0.84 (±0.08)	0.82 (±0.07)
NB	0.73	0.73	0.84 (±0.06)	0.83 (±0.06)
bi-GRU	0.80	0.86	0.86 (±0.06)	0.82 (±0.07)
BERT	0.81	0.88	0.85 (±0.06)	0.83 (±0.07)

Similarly to Sec. 5.2, the ADD/DEL is not correlated with the accuracy of the models. All classifiers perform well in terms of F1-score (see Table 8.1). To assess the presence of a correlation between *ADD*/*DEL* and the change in performance of the classifiers in terms of F1-score, we compute the Spearman's ρ between the *ADD* of every class and its change in F1-score between the two classifiers. The correlation values are not significant, $p = 0.14$, $\rho = -0.10$ for *ADD* and $p = 0.14$, $\rho = -0.10$ for *DEL*[2]. This confirms that *ADD* and *DEL* are not related to the F1-score of the trained model. Instead, they estimate its behaviour change handling new data, considering which classification paths have been added or deleted regarding the past.

[2]Notice the two ρ values are mutual as $ADD + DEL = 1$



(a) $b_{\trianglerighteq}^{b_1,b_2}$ *for 2512: Software developers.* *(b)* $b_{\trianglelefteq}^{b_1,b_2}$ *for 2512: Software developers.*

Figure 8.2 ADD and DEL BDDs for the class vs class case. The ADD/DEL BDD for the class 2512 - Software Developers, reported in figure, in the class vs class case are equivalent to the DEL/ADD ROBDDs for the other class, 2511 - Systems Analysts

BDD2Text. Also, in the case of OJA, the BDD2Text module enables one to identify the underlying changes in classification paths between ψ_1 and ψ_2.

Class vs Class. For the class vs class case, we chose the classes *2511, systems analysts*, and *2512, software developers*, and we compare them to each other. We show both ADD and DEL BDDs for these in Figure 8.2. Given the symmetry of the problem, the ADD of one class is equivalent to the DEL of the other and vice versa.

Figure 8.3 shows the BDD2Text for class *2512, software developers* vs *2511, systems analysts*. From the BDD2Text, it emerges that at time t_2 having the words *engineer*, or *developer*, or *Java Developer* makes the model classify a job ad as a software developer (class 2512), rather than a system analyst (class 2511), while it does not hold for the classifier

```
The model now uses the following classification rules for this class:
This class has 6 added classification rules, but only 3 are used to classify the 80%
of the items.

 - Having Engineer.
 - Having Developer but not Engineer.
 - Having JavaDeveloper but not Engineer, and Developer.

The model is not using the following classification rules anymore:
This class has 2 deleted classification rules.

 - Having WebDeveloper but not Developer, Analyst, Systems, JavaDeveloper, and NetDeveloper.
 - Having Software but not Developer, Analyst, Systems, WebDeveloper, JavaDeveloper, and NetDeveloper.

There are no 'unchanged' classification rules.
```

Figure 8.3 The output for 2512, Software Developers vs 2511, Systems Analysts in the class vs class case using a DT to explain the RF model of Table 8.1.

at t_1, that used instead the two deleted rules that are not used at time 2. It is worth to notice that, because this is the BDD2Text of the Class vs Class case, those rules does not characterise software engineers in general, with respect to all the other occupations in the OJAs, but specifically against system analysts.

8.3 A Novel Methodology for Evaluating Occupation Classification

The following work is part of an ongoing EU project, *"Towards the European Web Intelligence Hub - European system for collection and analysis of online job advertisement data (WIH-OJA)"* [38], and has been presented during the CEDEFOP Annual Expert workshop and development of the methodology for improving occupation classification.

The workshop provides a platform for presenting the latest developments in the data production system and developing a new methodology for improving the occupation classification. One of the critical issues in the analysis of OJAs is misclassification, which can affect some specific occupations/languages. We, as CRISP (Interuniversity Research Centre for Public Services, the research group I'm currently a part of) are developing an innovative approach to identify and correct misclassification. Instead of correcting the wrong outcome, it generalizes the rules that the algorithm uses to classify occupations, going therefore at the root of the problem. As this method will involve the expertise of International Country Experts (ICEs), the workshop events will be used to explain and clarify the methodology. Moreover, as the methodology is innovative and still in the experimental stage, it will be developed and tested on a subset of countries (8) in order to collect the information needed to design a full scale system. The approach is applied in the following countries: IT, DE, CZ,

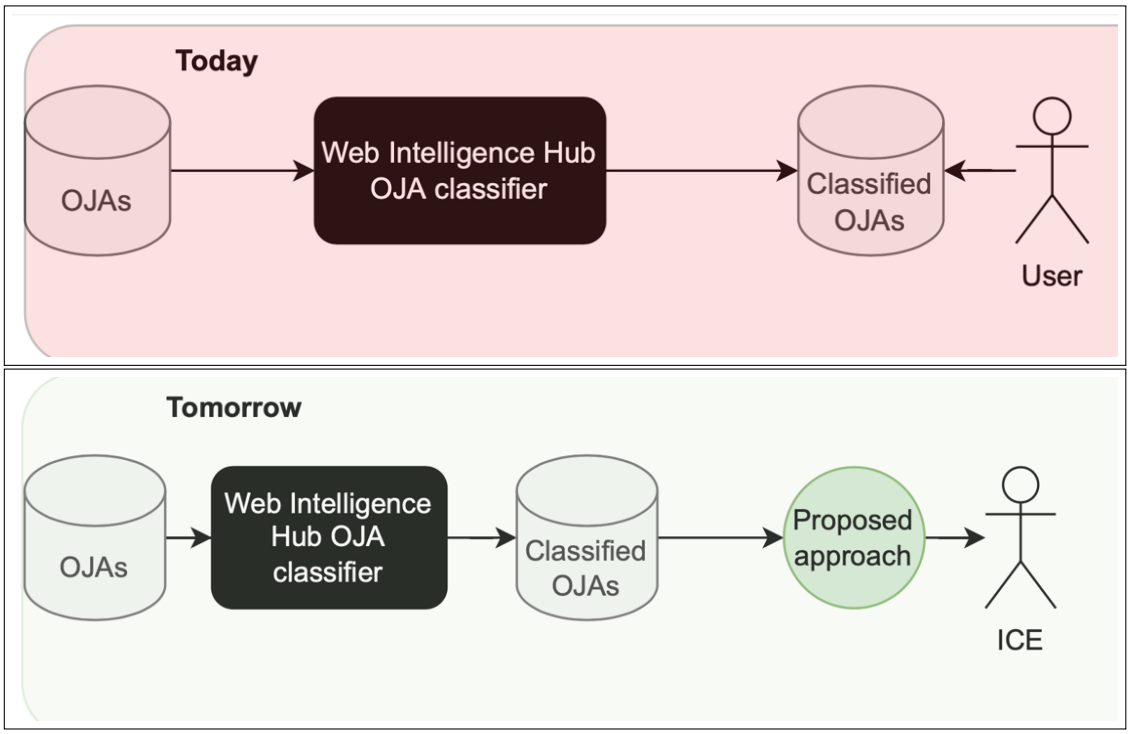


Figure 8.4 Key differences between the approach used today by WIH to classify OJA, and the proposed approach.

ES, PL, RO, UK, and HU. Therefore the workshop main aim is to present and develop a new method for understanding and validating the classification rationale and, in turn, improving provides insights to improve the occupation classification. A graphical overview of the contribution that the approach would give is depicted in Figure 8.4. We aim to answer the following questions:

- Which words – including their combination – are used by the WIH to classify job ads?
- Can we summarise the main words that led to an ESCO occupation through natural language?
- Is there a way to identify the existence of misleading classification terms?
- Can we focus on the classification rules as a whole, rather than checking OJAs manually and randomly?

We apply the techniques described in Sec. 3.2 to derive the classification rules used to classify Online Job Ads over the IV ESCO classification code. ICEs can look at the main-rules (terms and combinations of terms) used to classify OJAs rather than individual OJAs, and can check if the classifier is using words that are not related with the ESCO occupation, looking at OJAs for which the rule applies. Rules are expressed through natural language. ICEs can have a global view of the classification criteria that cover > 80% of the OJAs, rather than looking at randomly selected ones. For each rule, the ICE can look at 5

examples to check job ads title and description used by the WIH classifier for classifying that ESCO code.

To allow for comparability and reproducibility, the method will run on a audit dataset produced by CEDEFOP-Eurostat, that is built to be representative of the WIH dataset as well. The total number of OJAs is $2,290,855$, across 423 different ISCO LV4 codes. A set of ESCO occupations (IV digit) is identified in common among all languages and selected according to (i) frequencies of ESCO codes, (ii) coverage of the ESCO level I hierarchy, and (iii) sectorial characteristics of occupations as well. The list of ESCO occupations will be defined according to CEDEFOP.

Each ICE of the involved eight countries will be asked to evaluate an Excel file containing the top rules (one for each row) used by the WIH to classify OJA. For each rule (row), the ICE is expected to evaluate – on a Likert scale – if and to what extent the rule is considered sound with the corresponding ESCO occupation code. If the ICE considers a rule (or a part of) either misleading or critical, the ICE is asked to mark the rule and provide a "correction", intended as feedback to be used in further activities to improve the classification process accordingly. At the end of the evaluation process, the ICE will return the initial Excel file filled with Likert values (for each row) and comments/corrections for those rules considered as erroneous or critical.

The file provided to ICEs is comprised of the following data:

(i) *Classification Rule*: The rule summarised and applied by the WIH classifier on the Eurostat audit sample benchmark;

(ii) *Quartile*: The position of the Occupation in the sampling process;

(iii) *ISCO LV4 N. OJAs*: number of OJAs in the audit dataset classified by the WIH classifier as a certain ISCO LV4;

(iv) *Overall Rule Impact*: number of OJAs where the rule applies in the whole audit dataset;

(v) *Class Rule Impact*: number of OJAs where the rule applies in the corresponding ISCO class;

(vi) *Class % Incidence*: percentage ratio of Class Rule Impact / Overall Rule Impact. The higher the value, the more specific the rule is;

(vii) *Rule Impact (%)*: percentage ratio of Class Rule Impact / ISCO LV4 N. OJA. The higher the value, the more widespread the rule in the ISCO code. In essence, this is the importance of the rule for the class.

The outcome of the new approach is twofold:

- It will provide an estimate of the effectiveness of the new method in evaluating and improving occupation classification and in identifying misclassification errors in terms of criteria rather than examples.

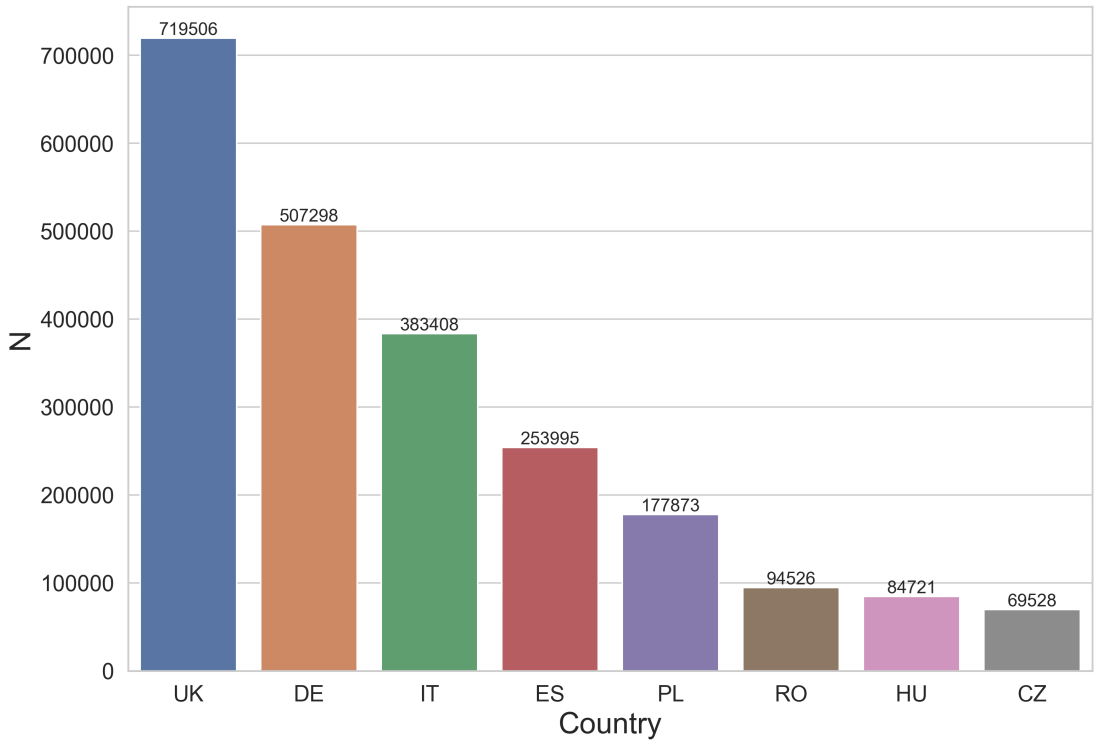


Figure 8.5 Total count of OJAs by country. All languages are considered.

- The method also allows tracking of the ICE rules validation activity, and this is helpful to perform a cost-benefit analysis to decide whether and how to extend the method to more countries and classes.

There are multiple benefits to this approach. It is rigorous, deterministic, and can be reproduced at anytime (same input will lead you to same results). It is trasparent, as it aks ICEs to concentrate – and then to validate - the rationale, rather than validating singular OJAs. Identified rules can act as a "bug-hunter" to check if further WIH improvement solved the misclassification, reducing time, ICE efforts and costs. As rules and classes are selected on the basis of frequency, the probability to encounter a misclassification is much higher if compared with the probability of looking at a randomly selected missclassified OJA.

Although the new methodology will initially be applied to only 8 countries, it will be presented and explained during the workshop so that all ICEs will grasp it in order to be ready for its scaling up. Figure 8.5 shows the distribution of the OJAs in each country.

Subsequently, for each of the defined 8 countries/languages a sample of classes of occupations will be selected. ICEs of these 8 countries will be required to perform the validation of the rules. They will then present the results of their work interacting with other ICEs. In this way, all ICEs will have a clear understanding of the methodology for improving the classification system. After having assessed the validity of the approach, it

will be extended in the future to all the other countries/occupations. Table 8.2 describes in detail the number of OJAs after class sampling, number of rules found, and the percentage of OJAs covered by a rule.

Table 8.2 Number of Online Job Advertisements considered by country, including the total, number of OJAs present in the considered classes, count and percentage of instances correctly covered by a rule.

Country	Tot. N. OJAs	Sampled N. OJAs	N. Rules	Rule Impact
UK	719,506	181,230	571	150,516 (83.0%)
DE	507,298	100,516	529	84,759 (84.3%)
IT	383,408	77,244	429	62,087 (80.4%)
ES	253,995	45,053	393	34,023 (75.5%)
PL	177,873	35,080	302	30,180 (86.0%)
RO	94,526	12,500	223	10,915 (87-3%)
HU	84,721	16,139	245	13,803 (85.5%)
CZ	69,528	13,062	201	11,465 (87.8%)

To provide examples and show the results of our approach, we will focus on United Kingdom. The total number of OJAs for UK is $719,506$ – of those, $677,835$ (94%) are in english, while the others in various different languages, which are discarded. There are 410 different ISCO LV4 codes in this dataset. However, many codes have a limited number of instances, with some having less than ten. We choose to remove all those ISCO LV4 codes that classify less than 100 OJAs, removing 61 codes. 349 (85%) codes are kept. We also remove all the occupations labeled as *"not elsewhere classified"*, for example, *"Software and applications developers and analysts not elsewhere classified"*, as they are too general and cannot be classified properly. We exclude ISCO LV1 digit 6, "Skilled agricultural, forestry and fishery workers", since those occupations are usually not recruited via online advertisements. For each ISCO LV1, with the exception of digit 6, we sample 8 different ISCO LV4 codes, for a total of 64 different ones. The code sampling is based on quartile numerosity distribution: counting the number of ISCO LV4 for each each LV1, the two most common LV4 are picked for each quartile. This is done to provide a fair sampling, not based on just the most common codes. $181,230$ OJAs are kept after these filtering and sampling steps. Figure 8.6 shows the numerosity distribution of the remaining class codes.

Figure 8.7 shows two classes as result examples. First is class *Software developers* in Figure 8.7a. The most common rules involve the terms *Developer* and *Development*, but exclude some non-software related term such as *Business Development* and *Sales Development*. Just those two rules conjoined match 1888 OJAs, or 25% of all Software developers in the dataset. It can be noted, however, that their Class % Incidence is quite low, at 43.5% and 25.8% respectively. This means that the rule is not highly specific to the class, and it is also

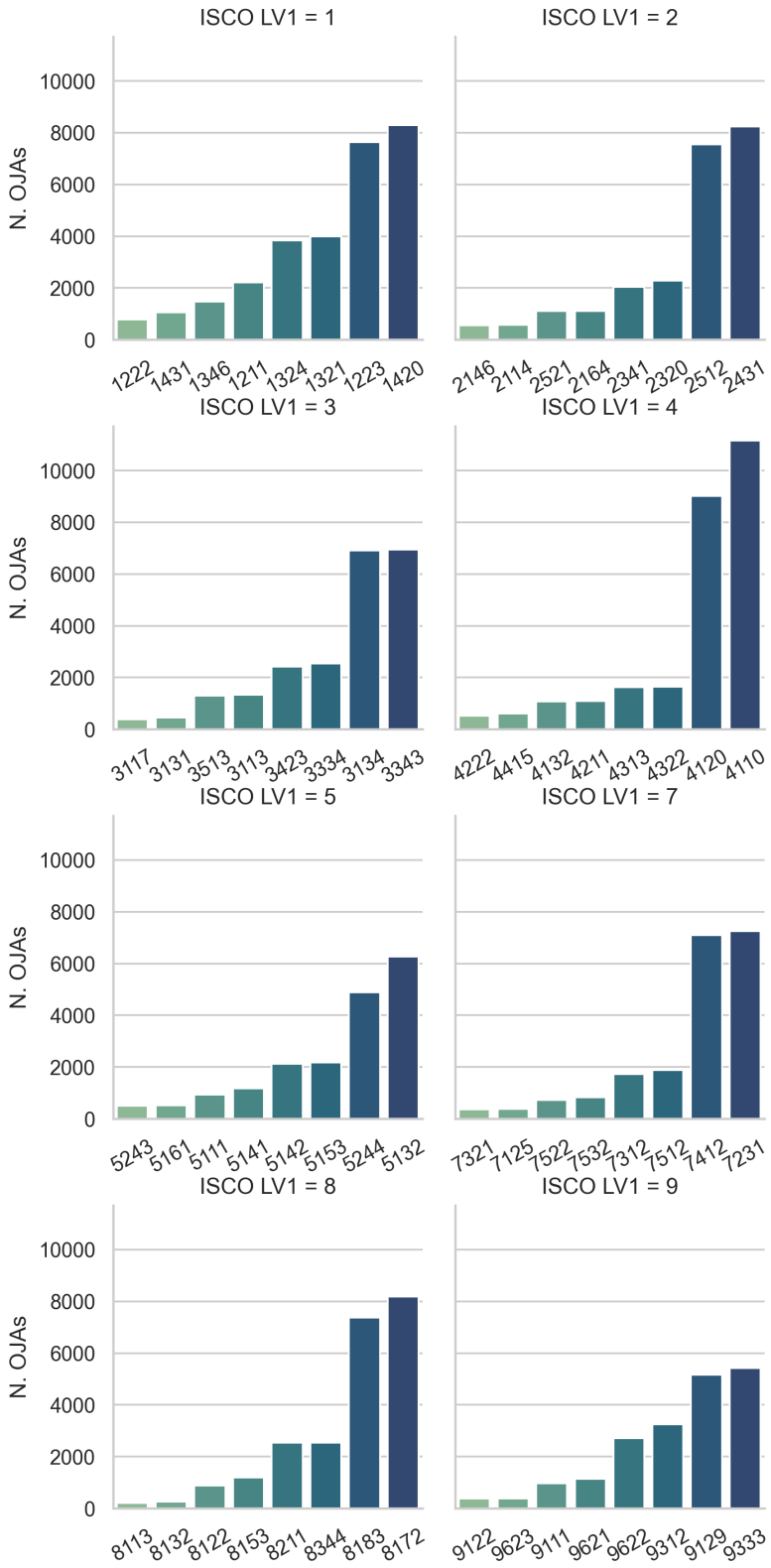


Figure 8.6 Distribution of number of United Kingdom OJAs between each ISCO LV4 class. Plots are grouped by first ISCO digit.

matched in other codes. Other rules, such as *Having Software Engineer* and *Having Senior Estimator*, are much more specific, although they are less common. It is also to be noted that some rules can highlight classification errors, such as *Having Senior Chef*. Those are considered software engineers by the WIH classifier. However, after reviewing the relative OJAs, it can be seen that many of these are clearly classification errors and should belong to another class, as the "chef" in question seems to often be required in a kitchen[3].

The second example in Figure 8.7b, *Primary school teachers*, shows another interesting set of rules. The first one, *Having Teacher*, is simple but requires many negative terms to disambiguate primary school teachers from other kinds of teachers that belong to other ISCO codes. It is also interesting to note that *Primary Teacher* is negated in the second rule despite being so close to the class label, since it is (surprisingly) a term most commonly used by "Advertising and marketing professionals". Other rules are more specific but less widespread, such as *Key Stage Teaching*, which is the phase of primary education for pupils aged 5 to 7 in England.

The ICEs are not expected to validate the accuracy of the classifier, but to inspect the classification rules to identify misleading/wrong terms, clear misclassifications and OJAs that cannot be classified only by looking at job titles. They will be asked the following questions for each derived rule: **(Q1):** Looking at a each class rule, do you think is there any term that is not related with the ESCO class? (Y/N) If so, which one and why? Specify and Comment. **(Q2)**: To what extent the class-rule is coherent with the corresponding ESCO class? 1-5 Likert scale, the higher, the better.

As this is an ongoing project, ICEs have not yet finished their evaluation. They have been already presented with the methodology, and are interested, as it is a structured approach to look at criteria rather than a randomly selected job ads sample (as they previously operated). Another interesting and planned activity will be to upgrade the WIH classifier using the expert evaluation, and then compare the current and enhanced versions using the `ContrXT` approach.

[3]Chef is also a DevOps automation software, but the OJAs in question are indeed looking for a chef in the more common sense.

Quant	ISCO L	Label	Classification Rule	ISCO LV4 N. OJA	Overall Rule Impact	Class Rule Impac	Class % Incidence	Rule Impact (%	
Q1	2512	Software developers	Having Developer - but none of Sql Developer, Senior Net Developer, and Front	7542	3028	1319	43,56	17,49	View Job Ads
Q1	2512	Software developers	Having Development - but none of Product Development, Sales Development, Business Development, Development Technical, and Development Manager	7542	2205	569	25,8	7,54	View Job Ads
Q1	2512	Software developers	Having Software Engineer	7542	615	551	89,59	7,31	View Job Ads
Q1	2512	Software developers	Having Architect - but none of Landscape, Part Architect, and Solutions Architect	7542	1125	445	39,56	5,9	View Job Ads
Q1	2512	Software developers	Having Programme - but none of Programme Coordinator, Programme Manager, and Management	7542	808	231	28,59	3,06	View Job Ads
Q1	2512	Software developers	Having Senior Associate - but none of Employment Senior	7542	438	201	45,89	2,67	View Job Ads
Q1	2512	Software developers	Having Devops	7542	330	199	60,3	2,64	View Job Ads
Q1	2512	Software developers	Having Senior Chef	7542	167	154	92,22	2,04	View Job Ads
Q1	2512	Software developers	Having Senior Estimator	7542	143	136	95,1	1,8	View Job Ads
Q1	2512	Software developers	Having Senior Practitioner	7542	139	100	71,94	1,33	View Job Ads
Q1	2512	Software developers	Having Developer, Front, and React	7542	34	7	20,59	0,09	View Job Ads

(a) ISCO LV4 2512 - Software developers.

Quant	ISCO L	Label	Classification Rule	ISCO LV4 N. OJA	Overall Rule Impact	Class Rule Impac	Class % Incidence	Rule Impact (%	
Q2	2341	Primary school teachers	Having Teacher - but none of Primary Teacher, Technology Teacher, Art Design, Food, Technology Food Teacher, Design Art Teacher, and Tourism Teacher Travel	2049	9225	1642	17,8	80,14	View Job Ads
Q2	2341	Primary school teachers	Having Primary - but none of Primary Teacher, Supervisor, Primary Care Network, and Primary Teaching	2049	965	275	28,5	13,42	View Job Ads
Q2	2341	Primary school teachers	Having English Language	2049	35	32	91,43	1,56	View Job Ads
Q2	2341	Primary school teachers	Having Ks1, and Ks1Teaching Assistant	2049	27	27	100	1,32	View Job Ads
Q2	2341	Primary school teachers	Having Key Stage Teaching	2049	18	12	66,67	0,59	View Job Ads
Q2	2341	Primary school teachers	Having Ks1, and Ks1Teachers	2049	6	6	100	0,29	View Job Ads

(b) ISCO LV4 2341 - Primary school teachers.

Figure 8.7 UK results for two different ISCO classes, in the final Excel format, as presented to the experts.

9

Real World Projects in Need of XAI

In this chapter, we show additional works that inspired work on XAI here reported. Although they do not employ contrastive explanations (yet), they have laid the ground for pursuing eXplainable AI in this work, and inspired the ideas that led to the techniques presented in the Thesis. Because of that, we believe they are strongly related to the subject matter, and will now be briefly illustrated. Both of these real-world applications, as part of ongoing EU research projects, require explainability. The explanation techniques researched in the Thesis will be applied on those projects in the future.

9.1 NEO: A System for Identifying New Emerging Occupation from Job Ads

We propose NEO, a tool for automatically enriching the European Occupation and Skill Taxonomy (ESCO) with terms that represents new occupations extracted from million Online Job Advertisements (OJAs). NEO proposes (i) a novel metric that allows one to measure the semantic similarity between words in a taxonomy, and (ii) a set of measures that estimate the adherence of new terms to the most suited taxonomic concept, enabling the user to evaluate the suggestions. To test its effectiveness, NEO has been evaluated over 2M+ 2018 UK job ads, along with a user-study to confirm the usefulness of NEO in the taxonomy enrichment task.

9.1.1 Introduction

Unlike the automated construction of new taxonomies from scratch, which is a well-established research area [124], the augmentation of existing hierarchies is gaining in importance, given its relevance in many practical scenarios (see, e.g. [122, 77]). To date, the most adopted approach to enrich or extend standard *de-jure* taxonomies - that cannot be constructed from scratch - lean on expert panels, that identify and validate which term has to be added to a taxonomy. This process totally relies only on human knowledge, making it costly, time-consuming and error prone, besides suffering from sparse coverage. Those challenges need the development of automated methods for taxonomy enrichment.

9.1.2 Overview of NEO

NEO [51] aims at enriching the European standard labour market taxonomy ESCO [36] with new potential occupations derived from real Online Job Advertisements (OJAs). It is developed as part of the research activity of an ongoing EU grant aimed at realising the first EU real-time labour market monitoring system, by collecting and classifying OJAs over all 27+1 EU countries and 32 languages [25, 11][1]. *Novel occupations* are usually intended as *mentions* that deserve to be represented within the taxonomy, as they might represent either an emerging job (e.g., *SCRUM master*) or a new alternative label characterising an existing job (e.g., *Android developer*). This activity is crucial to allow economists and policy makers to observe up-to-date labour market dynamics using standard taxonomies as a *lingua franca*, overcoming linguistic boundaries (see, e.g. [42, 51, 30]). NEO relies on distributional semantics to extract semantic information from the OJAs, exploiting the characteristics that words occurring in similar context tend to have a similar meaning. Our approach is composed of three steps: (i) *synthesise word embeddings*, (ii) *suggest new entities*, (iii) *vote and enrich* (Figure 9.1).

(Step 1) Synthesise Word Embeddings resorts to Deep Learning to learn a vector representation of words in the corpus, preserving the semantic relationships expressed by the taxonomy itself. To select the best representing vectors, we rely on three distinct sub-tasks, that are the following: **T1.1**: train three different word embedding models (Word2Vec, GloVe, FastText), **T1.2**. construct measure of pairwise semantic similarity between taxonomic elements, namely Hierarchical Semantic Relatedness (*HSR*) [51]. Compared with *HSR*, state-of-the-art metrics for semantic similarity (see Aouicha et al. [5] for a survey) suffer of two main drawbacks. First, when a word has multiple senses, those methods compute a value of similarity for each

[1] https://tinyurl.com/skillovate

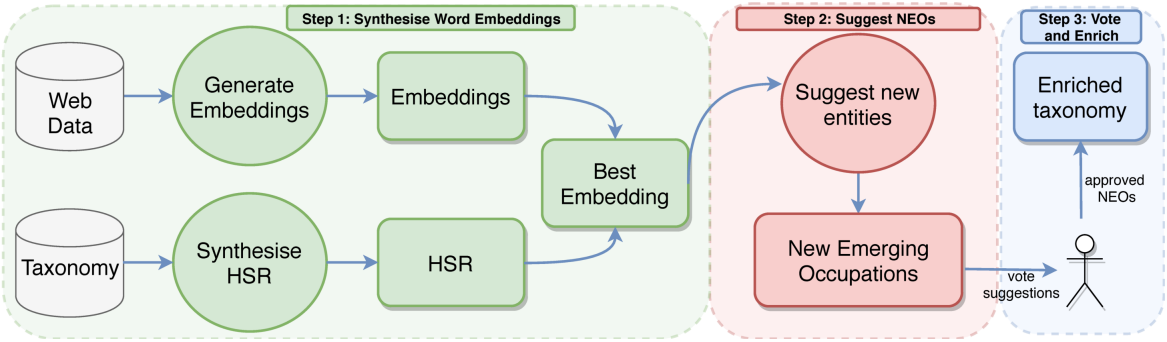


Figure 9.1 A representation of the NEO workflow highlighting the main modules. Taken from [51]

word sense and then consider only the highest, which is the self-information of the least frequent lowest common ancestor. As a consequence, more specific senses will have a higher value of similarity, but this does not reflect the use of words in advertising job positions; second, though they consider the structure of the taxonomy (i.e., the relationship between concepts) they do not take into account the number of child entities (i.e., words) belonging to those concepts. This is crucial in our case as ESCO includes generic concepts that, in turn, contain many different occupations. On the contrary, some very specific concepts can be represented by a few occupations which are highly informative. The aim of *HSR* is to overcome these limitations to work with the ESCO taxonomy. **T1.3**. Evaluate the embeddings in terms of correlation between the *HSR* and the cosine similarity between pair of terms in the taxonomy.

(Step 2) Suggest New Entities is aimed at extracting new occupation terms from the corpus of OJAs, and to suggest the most suitable concepts under which they could be added in the taxonomy $\mathcal{T}$. First we select a starting word w_0 from $\mathcal{T}$. Then we consider the top-5 mentions in the corpus of documents $\mathcal{D}$ with associated the highest *score* value $\mathcal{S}$ with w_0, where is $\mathcal{S}$ a function that quantifies how similar a new mention m is in relation to w_0. The most suitable concepts for m are identified on the basis of four measures, namely GASC (*Generality*, *Adequacy* *Specificity*, and *Comparability*, formally defined in Giabelli et al. [51]), that estimate the fitness of a concept c for a given m. *Generality* quantifies how similar a mention is to all the other unrelated concepts; *Adequacy* quantifies how much the mention is overall a good candidate as an entity of the concept, considering both Specificity and Generality; *Specificity* quantifies how similar a mention is to the related concept; *Comparability* quantifies how similar two occupations are in term of the number of skills shared between them.

(Step 3) Vote and Enrich allows validating the outcome of the previous steps - which is fully automated - by asking: (*Q1*) whether the mentions extracted from the corpus are valid

Figure 9.2 Evaluation of the ESCO concept candidate where the mention business intelligence analyst *should be added*

emerging occupations and (*Q2*) to what extent the concepts suggested as entry for a new mention are appropriate for it (by looking at skills-gap).

9.1.3 Experimental Results on 2M+ UK Job Ads

Experimental settings. The corpus, a subset of the data collected for the project [25], contains 2,119,025 OJAs published in the United Kingdom in 2018. The best performing model is the following: architecture=*fastText*, algorithm=CBOW, size=300, epochs=100, learning rate=0.1.

As a first step, the user selects the starting word w_0 among the occupations already in ESCO. Then, `NEO` prompts the 5 mentions with associated the highest *score* with w_0. The

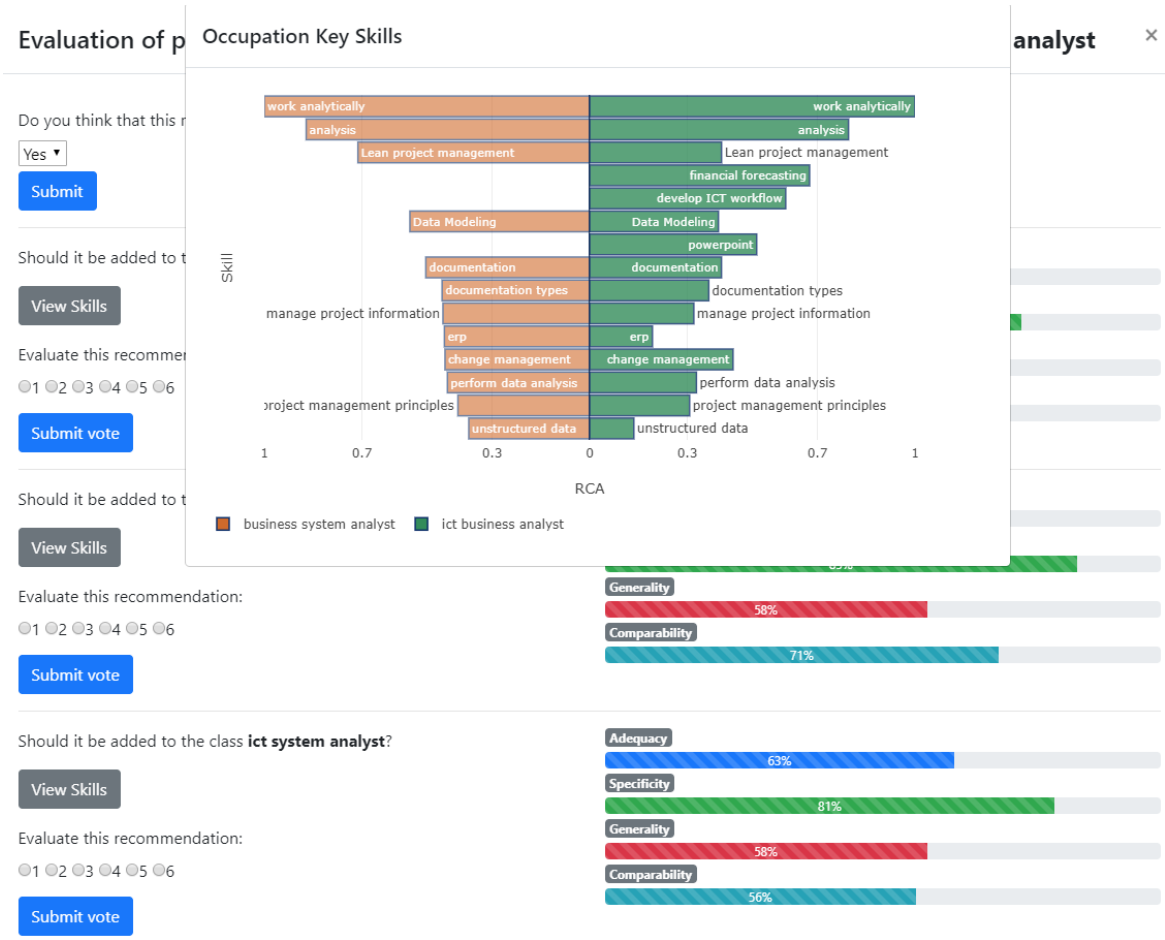


Figure 9.3 Skill relevance between the new term business intelligence analyst *and the parent ESCO concept suggested* data warehouse designer

user can therefore select a new mention m (*business intelligence analyst*)[2] to evaluate to which extent it fits as an entity of the starting word's ESCO concept, and as an entity of other two ESCO concepts selected (i.e., *data warehouse designer*. For each one of these three pairs mention $< m, concept >$ `NEO` provides the GASC measures (Figure 9.2), along with a comparison of the *rca* [2] of skills for both the mention and the concept (Figure 9.3). These skills, together with the GASC measures, support the user in evaluating if the suggested entry is appropriate as an entity of a concept. A user evaluation involving 10 final users reveladed that 88% of the mentions (43 out of 49) were successfully evaluated to be new occupations, while the correlation (Spearman's ρ and Kendall's τ) between the Likert values and GASC values is positive and statistically significant.

[2]Remember those occupations are not included in the ESCO taxonomy though requested by the market

9.1.4 Conclusion

We have shown NEO, a system developed within the research activities of an ongoing EU tender for monitoring EU labour market (see, [11, 14]).

NEO allows identifying potential new occupations as they emerge from job ads through deep learning, suggesting the suitable concept to enrich the European standard occupation taxonomy by means of (i) a novel semantic similarity metric and (ii) a set of measures that estimate the adherence of new terms to the concept. Though NEO can be used with any OJA dataset and EU language, here it has been trained on a 2M+ 2018 UK advertisements identified 49 novel occupations, 43 of which were validated as novel occupations by a panel of 10 experts as final users. Two statistical hypothesis tests confirmed the correlation between the proposed GASC metrics of NEO and the user judgements.
A demo is provided at https://tinyurl.com/NEO-aaai2021.

9.2 Skills2Job: A Recommender System that Encodes Job Offer Embeddings on Graph Databases

We propose a recommender system that, starting from a set of users skills, identifies the most suitable jobs as they emerge from a large text of Online Job Advertisements (OJAs). To this aim, we process 2.5M+ OJAs posted in three different countries (United Kingdom, France and Germany), generating several embeddings and performing an intrinsic evaluation of their quality. Besides, we compute a measure of skill importance for each occupation in each country, the Revealed Comparative Advantage (*rca*). The best vector models, together with the *rca*, are used to feed a graph database, which will serve as the keystone for the recommender system. Finally, a user study of 10 validates the effectiveness of `skills2job`, both in terms of precision and nDGC.

9.2.1 Introduction

Given the very high number of job positions and applicants on online job portals, the problem of person-job fit [102, 134] has become relevant in recent literature, both as a skill measuring system [131] and job recommendation system [133, 79]. Recommender systems in the labour market domain rely strongly on handcrafted features and expert knowledge, which make them costly, difficult to update and error-prone. For that reason, we propose `skills2job`, a knowledge poor and data driven job recommendation system, which can be adapted to different countries/industries and easily updated over time. Moreover, `skills2job` is the

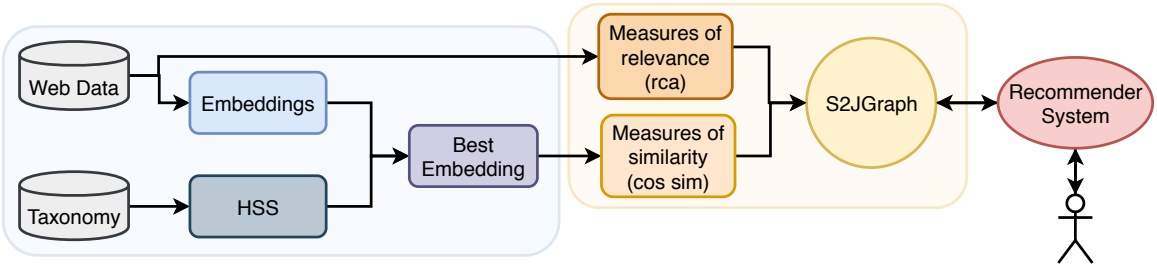


Figure 9.4 Workflow of steps for building `skills2job`

first system that organises labour market information, automatically extracted from a large corpus of OJAs, in a graph database which can be queried to recommend the most suitable occupations for an user based on its skills. `skills2job` uses ESCO (https://ec.europa.eu/esco) as a target taxonomy to organise occupations and skills, to allow querying the graph using 27 languages. `skills2job` was realised as part of the research activity of an EU project[3] (see [13]), which aims at realising the first EU real-time labour market monitor, by collecting and classifying OJAs from all 27+1 EU countries.

9.2.2 An Overview of `skills2job`

The workflow of `skills2job`, presented in Figure 9.4, can be divided in five main steps: **S.1** To extract linguistic patterns from OJAs, we train multiple embedding models through FastText, a library for representation learning which builds word embeddings considering sub-word information by representing each word as the sum of its character *n*-gram vectors; **S.2** we compute measure of pairwise semantic similarity between taxonomic elements, namely `HSS` (developed in [51] and previously called Hierarchical Semantic Relatedness (*HSR*)).

Compared with `HSS` , previous measures of semantic similarity in taxonomies (see Aouicha et al. [5] for a survey) suffer of two main limitations. First, when a word has multiple senses, those methods compute a value of similarity for each word sense and then consider only the highest, which is the self-information of the least frequent lowest common ancestor. As a consequence, more specific senses will have a higher value of similarity, but this does not reflect the use of words in advertising job positions; second, though they consider the structure of the taxonomy (i.e., the relationship between concepts) they do not take into account the number of child entities (i.e., words) belonging to those concepts. This is crucial in our case as ESCO includes generic concepts that, in turn, contain many different

[3]CEDEFOP 2014. Real-time Labour Market information on skill requirements: feasibility study and working prototype". https://goo.gl/qNjmrn

occupations. On the contrary, some very specific concepts can be represented by a few occupations which are highly informative. The aim of `HSS` is to overcome these limitations to work with the ESCO taxonomy. Since we want to encode semantic information from a semantic hierarchy built from human experts into our vector model, we adopt those values as a proxy of human judgements. **S.3** To select the embedding that better preserves taxonomic relations, we perform an intrinsic evaluation by computing the Pearson correlation of the cosine similarity between each couple of skills and their corresponding `HSS`. As one might easily imagine, the simple use of the skill frequency to compute the relevance of a skill within a given set of OJAs might be highly inaccurate. **S.4** We employ co-occurrence statistics, using a normalised count based measure of skill-relevance, the Revealed Comparative Advantage (*rca*) [2]. **S.5** The information extracted through word embeddings and *rca* is stored in our graph database, called `S2JGraph`, which is formalised as a directed labelled multi-graph and the formalisation is inspired by [50]. Note that both the *rca* and the best embedding are computed for each country, capturing the difference between the requested skills and the occupation terms as they are used in different countries.

Distinguishing between the starting country and the target country allows us to catch the differences between ICT occupations in UK, Germany, and France focusing on skills set, hence discovering the characteristics of local Labour Markets.

9.2.3 Skill Based Recommendations

Table 9.1 Example of query (ii) rcaB *matching part*

Rank	Arriving occupation	4 skills	rca_{NORM}
0.56	Web Technicians	C#	0.3996
		implement front-end website design	0.5967
		use markup languages	0.6066
		CSS	0.6264
0.2	Applications programmers	C#	0.3293
		implement front-end website design	0.143
		use markup languages	0.1832
		CSS	0.1265
0.18	Software developers	C#	0.3145
		implement front-end website design	0.1327
		use markup languages	0.1614
		CSS	0.130

Table 9.2 User evaluation results for the two methods. P@3-N indicates that a user score of at least N is considered a true positive.

	rcaB	**cosB**
P@3-3	0.823	0.763
P@3-4	0.610	0.570
nDCG	0.985	0.984

The graph database, `S2JGraph`, is used as a keystone for several recommendations tasks using the Cypher query language. Given a set of starting skills **S**, a starting occupation o_S, a starting country c_S, a target country c_A and a target skill s_T provided by the user, `skills2job` returns:
(i) The relevance of each $s \in \mathbf{S}$ for o_S in c_S;
(ii) A list of occupations **O** in c_A and for each $o_i \in \mathbf{O}$: (a) The relevance of each $s \in \mathbf{S}$ for o_i; (b) A list of skills that o_i requires different from those in **S** and relevant for o_i (namely, the *skill gap*).
(iii) A set of skills recommended to the user given **S** and s_T.

The main use case - query (ii) - recommends a series of occupations in a target labour market based on the user's skills, matching all the occupations in the target country c_A which require at least one of the starting skills in **S**. Then the query matches all the skills which are required by the target occupation with a $rca > \alpha$ and which have a *cosine similarity* with all of the starting skills in $\mathbf{S} < \beta$. These are the *skill gap*, which are relevant for the target occupation (high *rca*) and different enough from the starting skills (low *cosine similarity*). Those are skills that she/he should acquire to do that job in the target country.

An example of query (ii) is reported in Table 9.1 ($\alpha = 0.6$, $\beta = 0.7$). The starting parameters are the following: **S** =["implement front-end website design", "CSS","C#", "use markup languages"]; o_S ="Web and multimedia developers"; $c_S = UK$: $c_A = DE$; s_T = "Python".

9.2.4 User Evaluation and Conclusion Remark

The results of `skills2job` were evaluated through a user study following [67]. We asked 10 Labour Market experts belonging to the European Network on Regional Labour Market Monitoring to judge whether the starting skills are relevant for the occupations provided by the system or not, using a Likert scale. As 3 recommendations were presented for each item, we decided to use Precision@3 (P@3), assuming either a user score of at least 3 as a true positive (P@3-3), or of at least 4 (P@3-4). The normalised Discounted Cumulative

Gain (nDCG) has also been computed, which measures the usefulness of an item based on its position in the result list. The results (see Table 9.2) show a high degree of correlation between the user evaluation and the recommendation ranking.

In conclusion, `skills2job` identifies the most suited job on the basis of a set of user's skills, encoding the skill relevance as emerges from real-labour market demand. It can process any OJA dataset in any EU language. Here we used 2.5M+ advertisements processed trough distributional semantics and co-occurrence statistics, organised in a graph database. A user evaluation made by experts show the system is effective in identifying the correct job given a set of user skills.

We have been working to extend `skills2job` to all 27+1 EU Countries, enabling policy-makers to observe the labour market demand at skill level.

Part IV

Conclusions

We presented a novel model-agnostic approach to globally explain how a black box text classifier change its learning criteria with regard to the past (T-contrast) and different models (M-contrast) by manipulating BDDs. The approach has been implemented as a tool, namely `ContrXT` . It first traces the behaviour of the surrogate classifiers through BDDs; then it manipulates them to generate further BDDs that encode the logic changes of the classifiers, along with the classification paths added, deleted or unchanged, providing contrastive explanations for each category. Finally, it summarises and presents those changes through ADD/DEL/STILL indicators and natural language explanations. Both helps the user answering the questions we draw in Chapter 1, which are (Q1) observing the classification logic of a machine learning-based system over multiple training phases (Q2) to understand why the newly trained model is classifying data differently after retraining (Q3) through natural language.

The evaluations have been performed on several multiclass benchmarks, i.e., *20newsgroup* and a real-life application deployed as an Online Job Ads multiclass classifier (50 classes, see, e.g. [11, 51]) in an EU project [25]. We trained our models using different learning algorithms, showing `ContrXT` can work with the state-of-the-art learning algorithms, reaching an high F1 surrogate fidelity. The Spearman correlation test revealed the accuracy is not correlated with the ADD/DEL indicators, confirming they provide additional insights beyond the quality of the trained models.

Secondly, we extended the approach by generating model-contrastive explanations from any tabular or text classifier. Our approach allows users to understand how -and to what extent- two distinct models differ, despite their accuracy. To do so, we generalised and enhanced the methodology proposed above and in [78] to deal with (i) model-contrastive explanations, despite the time, (ii) tabular data through an ordinal or boolean discretisation process and (iii) multiple types of surrogate models. The approach has been evaluated using well-established benchmarks to allow the reproducibility of results. Finally, the approach has been implemented as a pip-python tool and an API, and it is available to the whole community.

Third, we have shown real-world applications of XAI in the LMI domain, framed within multiple ongoing EU projects, aiming to explain to international experts the constant evolution in the labour market.

9.3 Future Research Directions

The work presented in the Thesis opens multiple novel research directions. First, we are applying `ContrXT` on a machine learning classifier that classifies job ads for all the 27+1

countries of the EU [11] framed within an EU project [25] that aims at monitoring the changes in the labour market, its lexicon and new professions [51, 77].

The tool is constantly being worked on, and updated with new features. An interesting and important development is the addition of new surrogate types, which can be easily implemented in the *Trace* phase.

`ContrXT` is also being used in another EU project, *"Towards the European Web Intelligence Hub - European system for collection and analysis of online job advertisement data (WIH-OJA)"* [38], as described in Sec. 8.3. This is an ongoing work, and International Country Experts are currently working to assess the quality of the extracted decision rules. Once the WIH classifier will be updated, the `ContrXT` *Explain* approach will also be used to assess the differences between the current classifier and its next version.

Bibliography

[1] Akers, S. B. (1978). Binary decision diagrams. *IEEE Transactions on computers*, 27(06):509–516.

[2] Alabdulkareem, A., Frank, M. R., Sun, L., AlShebli, B., Hidalgo, C., and Rahwan, I. (2018). Unpacking the polarization of workplace skills. *Science advances*, 4(7).

[3] Alonso, J. M. and Bugarín, A. (2019). Expliclas: automatic generation of explanations in natural language for weka classifiers. In *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. IEEE.

[4] Amato, F., Castiglione, A., Mercorio, F., Mezzanzanica, M., Moscato, V., Picariello, A., and Sperlì, G. (2018). Multimedia story creation on social networks. *Future Generation Computer Systems*, 86:412 – 420.

[5] Aouicha, M. B., Taieb, M. A. H., and Hamadou, A. B. (2016). Taxonomy-based information content and wordnet-wiktionary-wikipedia glosses for semantic relatedness. *Applied Intelligence*, 45(2):475–511.

[6] Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al. (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115.

[7] Bergamaschi, S., Carlini, E., Ceci, M., Furletti, B., Giannotti, F., Malerba, D., Mezzanzanica, M., Monreale, A., Pasi, G., Pedreschi, D., Perego, R., and Ruggieri, S. (2016). Big data research in italy: A perspective. *Engineering*, 2(2):163 – 170.

[8] Biran, O. and Cotton, C. (2017). Explanation and justification in machine learning: A survey. In *IJCAI-17 Workshop on Explainable AI (XAI)*, page 8.

[9] Biswas, S. K., Chakraborty, M., Purkayastha, B., Roy, P., and Thounaojam, D. M. (2017). Rule extraction from training data using neural network. *International Journal on Artificial Intelligence Tools*, 26(03):1750006.

[10] Blackard, J. A. and Dean, D. J. (1999). Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151.

[11] Boselli, R., Cesarini, M., Marrara, S., Mercorio, F., Mezzanzanica, M., Pasi, G., and Viviani, M. (2018a). Wolmis: a labor market intelligence system for classifying web job vacancies. *J. Intell. Inf. Syst.*, 51(3):477–502.

[12] Boselli, R., Cesarini, M., Mercorio, F., and Mezzanzanica, M. (2013). Inconsistency knowledge discovery for longitudinal data management: A model-based approach. In *Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data - Third International Workshop, HCI-KDD*, pages 183–194.

[13] Boselli, R., Cesarini, M., Mercorio, F., and Mezzanzanica, M. (2017). Using machine learning for labour market intelligence. In *ECML PKDD*, pages 330–342.

[14] Boselli, R., Cesarini, M., Mercorio, F., and Mezzanzanica, M. (2018b). Classifying online job advertisements through machine learning. *Future Generation Computer Systems*, 86:319–328.

[15] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (2017). *Classification and regression trees*. Routledge.

[16] Bryant, R. E. (1986). Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on*, 100(8):677–691.

[17] Burch, J. R., Clarke, E. M., McMillan, K. L., Dill, D. L., and Hwang, L.-J. (1992). Symbolic model checking: 10^{20} states and beyond. *Information and computation*, 98(2):142–170.

[18] Burkart, N. and Huber, M. F. (2021). A survey on the explainability of supervised machine learning. *JAIR*.

[19] Butler, K. M., Ross, D. E., Kapur, R., and Mercer, M. R. (1991). Heuristics to compute variable orderings for efficient manipulation of ordered binary decision diagrams. In *Proceedings of the 28th ACM/IEEE Design Automation Conference*, pages 417–420.

[20] Cambria, E., Kumar, A., Al-Ayyoub, M., and Howard, N. (2022a). Guest editorial:: Explainable artificial intelligence for sentiment analysis.

[21] Cambria, E., Liu, Q., Decherchi, S., Xing, F., and Kwok, K. (2022b). Senticnet 7: a commonsense-based neurosymbolic ai framework for explainable sentiment analysis. *Proceedings of LREC 2022*.

[22] Cambria, E., Malandri, L., Mercorio, F., Mezzanzanica, M., and Nobani, N. (2023). A survey on xai and natural language explanations. *Information Processing & Management*, 60(1):103111.

[23] Candanedo, L. M. and Feldheim, V. (2016). Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models. *Energy and Buildings*, 112:28–39.

[24] Cashmore, M., Collins, A., Krarup, B., Krivic, S., Magazzeni, D., and Smith, D. (2019). Towards explainable ai planning as a service. *arXiv preprint arXiv:1908.05059*.

[25] CEDEFOP (2016). Real-time labour market information on skill requirements: Setting up the eu system for online vacancy analysis. https://goo.gl/5FZS3E.

[26] Chefer, H., Gur, S., and Wolf, L. (2021). Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 782–791.

[27] Chmielewski, M. R. and Grzymala-Busse, J. W. (1996). Global discretization of continuous attributes as preprocessing for machine learning. *International journal of approximate reasoning*, 15(4):319–331.

[28] Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

[29] Cohen, W. W. (1995). Fast effective rule induction. In *Machine learning proceedings 1995*, pages 115–123. Elsevier.

[30] Colombo, E., Mercorio, F., and Mezzanzanica, M. (2019). AI meets labor market: exploring the link between automation and skills. *Information Economics and Policy*, 47.

[31] Craven, M. and Shavlik, J. (1995). Extracting tree-structured representations of trained networks. *Advances in neural information processing systems*, 8.

[32] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

[33] Dhurandhar, A., Chen, P.-Y., Luss, R., Tu, C.-C., Ting, P., Shanmugam, K., and Das, P. (2018). Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *NIPS*, pages 592–603.

[34] Dougherty, J., Kohavi, R., and Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In *Machine learning proceedings 1995*, pages 194–202. Elsevier.

[35] Drechsler, R. and Sieling, D. (2001). Binary decision diagrams in theory and practice. *International Journal on Software Tools for Technology Transfer*, 3(2):112–136.

[36] European Commission (2019). ESCO: European skills, competences, qualifications and occupations, available at https://ec.europa.eu/esco/portal/browse. last accessed 19/11/2020.

[37] EuroStat (2016). The essnet big data project - available at https://goo.gl/EF6GtU.

[38] EuroStat (2020). Towards the european web intelligence hub — european system for collection and analysis of online job advertisement data (wih-oja), available at https://tinyurl.com/y3xqzfhp.

[39] Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*, volume 7. University of California, Irvine Irvine.

[40] Fox, M., Long, D., and Magazzeni, D. (2017). Explainable planning. *arXiv preprint arXiv:1709.10256*.

[41] Freuder, E. C. (2017). Explaining ourselves: Human-aware constraint reasoning. In *AAAI*, pages 4858–4862.

[42] Frey, C. B. and Osborne, M. A. (2017). The future of employment: How susceptible are jobs to computerisation? *Technological Forecasting and Social Change*, 114(Supplement C).

[43] Friedman, J. H. and Popescu, B. E. (2008). Predictive learning via rule ensembles. *The annals of applied statistics*, 2(3):916–954.

[44] Friedman, S. J. and Supowit, K. J. (1987). Finding the optimal variable ordering for binary decision diagrams. In *24th ACM/IEEE Design Automation Conference*, pages 348–356. IEEE.

[45] Fujii, H., Ootomo, G., and Hori, C. (1993). Interleaving based variable ordering methods for ordered binary decision diagrams. In *Proceedings of 1993 International Conference on Computer Aided Design (ICCAD)*, pages 38–41. IEEE.

[46] Fujita, M., Fujisawa, H., and Kawato, N. (1988). Evaluation and improvement of boolean comparison method based on binary decision diagrams. In *ICCAD*, volume 88, pages 2–5. Citeseer.

[47] Fujita, M., Matsunaga, Y., and Kakuda, T. (1991). On variable ordering of binary decision diagrams for the application of multi-level logic synthesis. In *Proceedings of the European Conference on Design Automation.*, pages 50–54. IEEE.

[48] Fushiki, T. (2011). Estimation of prediction error by using k-fold cross-validation. *Statistics and Computing*, 21(2):137–146.

[49] Gatt, A. and Krahmer, E. (2018). Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *JAIR*, 61.

[50] Giabelli, A., Malandri, L., Mercorio, F., and Mezzanzanica, M. (2020a). GraphLMI: A data driven system for exploring labor market information through graph databases. *Multimedia Tools and Applications*, pages 1–30.

[51] Giabelli, A., Malandri, L., Mercorio, F., Mezzanzanica, M., and Seveso, A. (2020b). NEO: A tool for taxonomy enrichment with new emerging occupations. In *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference*, volume 12507 of *Lecture Notes in Computer Science*, pages 568–584. Springer, Springer.

[52] Giabelli, A., Malandri, L., Mercorio, F., Mezzanzanica, M., and Seveso, A. (2021a). NEO: A system for identifying new emerging occupation from job ads. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, pages 16035–16037. AAAI Press.

[53] Giabelli, A., Malandri, L., Mercorio, F., Mezzanzanica, M., and Seveso, A. (2021b). Skills2job: A recommender system that encodes job offer embeddings on graph databases. *Appl. Soft Comput.*, 101:107049.

[54] Grinberg, M. (2018). *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.".

[55] Guidotti, R. (2022). Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, pages 1–55.

[56] Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. (2018). A survey of methods for explaining black box models. *CSUR*, 51(5).

[57] Gunning, D. (2017). Explainable artificial intelligence (xai). *Defense advanced research projects agency (DARPA), and Web*, 2(2):1.

[58] Hall, P., Gill, N., Kurka, M., and Phan, W. (2017). Machine learning interpretability with h2o driverless ai. *H2O. ai. URL: http://docs. h2o. ai/driverless-ai/latest-stable/docs/booklets/MLIBooklet. pdf*.

[59] Hayes, B. and Shah, J. A. (2017). Improving robot controller transparency through autonomous policy explanation. In *12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 303–312. IEEE.

[60] Hendricks, L. A., Akata, Z., Rohrbach, M., Donahue, J., Schiele, B., and Darrell, T. (2016). Generating visual explanations. In *European Conference on Computer Vision*, pages 3–19.

[61] Herve, J., Hamid, S., Bill, L., Robert, K. B., and Alberto, S.-V. (1990). Implicit state enumeration of finite state machines using bdd's. In *in Computer-Aided Design, 1990. ICCAD-90. Digest of Technical Papers., 1990 IEEE International Conference on*, pages 130–133.

[62] Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine learning*, 11(1):63–90.

[63] Huysmans, J., Dejaeger, K., Mues, C., Vanthienen, J., and Baesens, B. (2011). An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154.

[64] Javed, F., Hoang, P., Mahoney, T., and McNair, M. (2017). Large-scale occupational skills normalization for online recruitment. In *Twenty-Ninth IAAI Conference*.

[65] Jin, P., Zhang, Y., Chen, X., and Xia, Y. (2016). Bag-of-embeddings for text classification. In *IJCAI*, pages 2824–2830.

[66] Kamruzzaman, S. (2010). Rex: An efficient rule generator. *arXiv preprint arXiv:1009.4988*.

[67] Kanakia, A., Shen, Z., Eide, D., and Wang, K. (2019). A scalable hybrid research paper recommender system for microsoft academic. In *The World Wide Web Conference*, pages 2893–2899.

[68] Kohavi, R. (1996). Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pages 202–207.

[69] Kotsiantis, S. and Kanellopoulos, D. (2006). Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering*, 32(1):47–58.

[70] Krippendorff, K. (2004). Reliability in content analysis: Some common misconceptions and recommendations. *Human communication research*, 30(3):411–433.

[71] Kulesza, T., Burnett, M., Wong, W.-K., and Stumpf, S. (2015). Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th international conference on intelligent user interfaces*, pages 126–137.

[72] Lakkaraju, H., Bach, S. H., and Leskovec, J. (2016). Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1675–1684.

[73] Lakkaraju, H., Kamar, E., Caruana, R., and Leskovec, J. (2017). Interpretable & explorable approximations of black box models. *arXiv preprint arXiv:1707.01154*.

[74] Letham, B., Rudin, C., McCormick, T. H., and Madigan, D. (2012). Building interpretable classifiers with rules using bayesian analysis. *Department of Statistics Technical Report tr609, University of Washington*, 9(3):1350–1371.

[75] Liu, Y., Chen, C., Liu, Y., Zhang, X., and Xie, S. (2020). Shapley values and meta-explanations for probabilistic graphical model inference. In *ACM-SIGKDD*, pages 945–954.

[76] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., and Zhang, G. (2018). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12).

[77] Malandri, L., Mercorio, F., Mezzanzanica, M., and Nobani, N. (2021). MEET-LM: A method for embeddings evaluation for taxonomic data in the labour market. *Computers in Industry*, 124:103341.

[78] Malandri, L., Mercorio, F., Mezzanzanica, M., Nobani, N., and Seveso, A. (2022). ContrXT: Generating contrastive explanations from any text classifier. *Information Fusion*, 81:103–115.

[79] Malinowski, J., Keim, T., Wendt, O., and Weitzel, T. (2006). Matching people and jobs: A bilateral recommendation approach. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, volume 6, pages 137c–137c. IEEE.

[80] Martens, D., Baesens, B., and Van Gestel, T. (2008). Decompositional rule extraction from support vector machines by active learning. *IEEE Transactions on Knowledge and Data Engineering*, 21(2):178–191.

[81] Martens, D., Baesens, B., Van Gestel, T., and Vanthienen, J. (2007). Comprehensible credit scoring models using rule extraction from support vector machines. *European journal of operational research*, 183(3):1466–1476.

[82] Menascé, D. A. (2002). Load testing of web sites. *IEEE internet computing*, 6(4):70–74.

[83] Mezzanzanica, M., Boselli, R., Cesarini, M., and Mercorio, F. (2011). Data quality through model checking techniques. In *Advances in Intelligent Data Analysis X - 10th International Symposium, IDA*, pages 270–281.

[84] Mezzanzanica, M., Boselli, R., Cesarini, M., and Mercorio, F. (2012). Data quality sensitivity analysis on aggregate indicators. In *International Conference on Data Technologies and Applications*, pages 97–108.

[85] Mezzanzanica, M., Boselli, R., Cesarini, M., and Mercorio, F. (2013). Automatic synthesis of data cleansing activities. In *Proceedings of the 2nd International Conference on Data Technologies and Applications*, pages 138–149.

[86] Mezzanzanica, M., Boselli, R., Cesarini, M., and Mercorio, F. (2015). A model-based approach for developing data cleansing solutions. *J. Data and Information Quality*, 5(4):13:1–13:28.

[87] Michalski, R. S., Mozetic, I., Hong, J., and Lavrac, N. (1986). The multi-purpose incremental learning system aq15 and its testing application to three medical domains. In *Proc. AAAI*, volume 1986, pages 1–041.

[88] Miller, T. (2018). Contrastive explanation: A structural-model approach. *arXiv preprint arXiv:1811.03163*.

[89] Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267.

[90] Miller, T., Howe, P., and Sonenberg, L. (2017). Explainable ai: Beware of inmates running the asylum. In *IJCAI-17 Workshop on Explainable AI (XAI)*, page 36.

[91] Moro, S., Cortez, P., and Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31.

[92] Mueller, S. T., Hoffman, R. R., Clancey, W., Emrey, A., and Klein, G. (2019). Explanation in human-ai systems: A literature meta-review, synopsis of key ideas and publications, and bibliography for explainable ai. *arXiv preprint arXiv:1902.01876*.

[93] Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., and Yu, B. (2019). Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080.

[94] Narayanan, M., Chen, E., He, J., Kim, B., Gershman, S., and Doshi-Velez, F. (2018). How do humans understand explanations from machine learning systems? an evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1802.00682*.

[95] Otero, F. E. and Freitas, A. A. (2016). Improving the interpretability of classification rules discovered by an ant colony algorithm: extended results. *Evolutionary computation*, 24(3):385–409.

[96] Palan, S. and Schitter, C. (2018). Prolific. ac—a subject pool for online experiments. *Journal of Behavioral and Experimental Finance*, 17:22–27.

[97] Papadimitriou, A., Symeonidis, P., and Manolopoulos, Y. (2012). A generalized taxonomy of explanations styles for traditional and social recommender systems. *Data Mining and Knowledge Discovery*, 24(3):555–583.

[98] Papamichail, K. N. and French, S. (2003). Explaining and justifying the advice of a decision support system: a natural language generation approach. *Expert Systems with Applications*, 24(1):35–48.

[99] Pearl, J. et al. (2000). Models, reasoning and inference. *Cambridge, UK: CambridgeUniversityPress*, 19(2).

[100] Pedapati, T., Balakrishnan, A., Shanmugam, K., and Dhurandhar, A. (2020). Learning global transparent models consistent with local contrastive explanations. *NIPS*, 33.

[101] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

[102] Qin, C., Zhu, H., Xu, T., Zhu, C., Jiang, L., Chen, E., and Xiong, H. (2018). Enhancing person-job fit for talent recruitment: An ability-aware neural network approach. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 25–34.

[103] Quinlan, J. R., Compton, P. J., Horn, K., and Lazarus, L. (1987). Inductive knowledge acquisition: a case study. In *Proceedings of the Second Australian Conference on Applications of expert systems*, pages 137–156.

[104] Quinlan, J. R. et al. (1996). Bagging, boosting, and c4. 5. In *Aaai/Iaai, vol. 1*, pages 725–730.

[105] Rader, E., Cotter, K., and Cho, J. (2018). Explanations as mechanisms for supporting algorithmic transparency. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–13.

[106] Rajaraman, A. and Ullman, J. D. (2011). *Mining of massive datasets*. Cambridge University Press.

[107] Refaeilzadeh, P., Tang, L., and Liu, H. (2009). Cross-validation. *Encyclopedia of database systems*, 5:532–538.

[108] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). Why should i trust you?: Explaining the predictions of any classifier. In *ACM-SIGKDD*, pages 1135–1144.

[109] Rosenthal, S., Selvaraj, S. P., and Veloso, M. M. (2016). Verbalization: Narration of autonomous robot experience. In *IJCAI*, volume 16, pages 862–868.

[110] Sakar, C. O., Polat, S. O., Katircioglu, M., and Kastro, Y. (2019). Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and lstm recurrent neural networks. *Neural Computing and Applications*, 31(10):6893–6908.

[111] Sebastiani, F. (2002). Machine learning in automated text categorization. *CSUR*, 34(1).

[112] Sethi, K. K., Mishra, D. K., and Mishra, B. (2012). Extended taxonomy of rule extraction techniques and assessment of kdruleex. *International Journal of Computer Applications*, 50(21).

[113] Setiono, R., Baesens, B., and Mues, C. (2008). Recursive neural network rule extraction for data with mixed attributes. *IEEE transactions on neural networks*, 19(2):299–307.

[114] Shao, J. (1993). Linear model selection by cross-validation. *Journal of the American statistical Association*, 88(422):486–494.

[115] Shih, A., Choi, A., and Darwiche, A. (2018). A symbolic approach to explaining bayesian network classifiers. In *IJCAI*, pages 5103–5111.

[116] Singh, C., Nasseri, K., Tan, Y. S., Tang, T., and Yu, B. (2021). imodels: a python package for fitting interpretable models. *Journal of Open Source Software*, 6(61):3192.

[117] Sokol, K. and Flach, P. A. (2018). Glass-box: Explaining AI decisions with counterfactual statements through conversation with a voice-enabled virtual assistant. In *IJCAI*.

[118] Swartout, W., Paris, C., and Moore, J. (1991). Explanations in knowledge systems: Design for explainable expert systems. *IEEE Expert*, 6(3):58–64.

[119] Turrell, A., Speigner, B., Djumalieva, J., Copple, D., and Thurgood, J. (2018). Using job vacancies to understand the effects of labour market mismatch on uk output and productivity.

[120] Ustun, B. and Rudin, C. (2016). Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102(3):349–391.

[121] Van Bouwel, J. and Weber, E. (2002). Remote causes, bad explanations? *JTSB*, 32(4).

[122] Vedula, N., Nicholson, P. K., Ajwani, D., Dutta, S., Sala, A., and Parthasarathy, S. (2018). Enriching taxonomies with functional domain knowledge. In *ACM SIGIR*, pages 745–754.

[123] Ventura, D. and Martinez, T. R. (1995). An empirical comparison of discretization methods. In *Proceedings of the Tenth International Symposium on Computer and Information Sciences*, pages 443–450.

[124] Wang, C., He, X., and Zhou, A. (2017). A short survey on taxonomy learning from text corpora: Issues, resources and recent advances. In *EMLP*, pages 1190–1203.

[125] Wang, F. and Rudin, C. (2015). Falling rule lists. In *Artificial intelligence and statistics*, pages 1013–1022. PMLR.

[126] Wang, J., Fujimaki, R., and Motohashi, Y. (2015). Trading interpretability for accuracy: Oblique treed sparse additive models. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1245–1254.

[127] Wang, R. Y. and Strong, D. M. (1996). Beyond accuracy: What data quality means to data consumers. *Journal of management information systems*, 12(4):5–33.

[128] Wang, Z. J., Turko, R., and Chau, D. H. (2021). Dodrio: Exploring transformer models with interactive visualization. *arXiv preprint arXiv:2103.14625*.

[129] Wegener, I. (2000). *Branching programs and binary decision diagrams: theory and applications*. SIAM.

[130] Wu, Y., Zhang, Z., Kou, G., Zhang, H., Chao, X., Li, C., Dong, Y., and Herrera, F. (2021). Distributed linguistic representations in decision making: Taxonomy, key elements and applications, and challenges in data science and explainable artificial intelligence. *Inf. Fusion*, 65:165–178.

[131] Xu, T., Zhu, H., Zhu, C., Li, P., and Xiong, H. (2018). Measuring the popularity of job skills in recruitment market: A multi-criteria approach. In *AAAI*.

[132] Zhang, D., Liu, J., Zhu, H., Liu, Y., Wang, L., Wang, P., and Xiong, H. (2019). Job2vec: Job title benchmarking with collective multi-view representation learning. In *CIKM*, pages 2763–2771.

[133] Zhang, X., Zhou, Y., Ma, Y., Chen, B.-C., Zhang, L., and Agarwal, D. (2016). Glmix: Generalized linear mixed models for large-scale response prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 363–372.

[134] Zhu, C., Zhu, H., Xiong, H., Ma, C., Xie, F., Ding, P., and Li, P. (2018). Person-job fit: Adapting the right talent for the right job with joint representation learning. *ACM Transactions on Management Information Systems (TMIS)*, 9(3):1–17.