Introducing Uncertainty Into Explanaible AI Methods

Szymon Bobek[1][0000−0002−6350−8405] and Grzegorz J. Nalepa[1][0000−0002−8182−4225]

Jagiellonian Human-Centered Artificial Intelligence Laboratory (JAHCAI) and Institute of Applied Computer Science, Jagiellonian University, 31-007 Kraków, Poland
{szymon.bobek,grzegorz.j.nalepa}@uj.edu.pl

Abstract. Learning from uncertain or incomplete data is one of the major challenges in building artificial intelligence systems. However, the research in this area is more focused on the impact of uncertainty on the algorithms performance or robustness, rather than on human understanding of the model and the explainability of the system. In this paper we present our work in the field of knowledge discovery from uncertain data and show its potential usage for the purpose of improving system interpretability by generating Local Uncertain Explanations (LUX) for machine learning models. We present a method that allows to propagate uncertainty of data into the explanation model, providing more insight into the certainty of the decision making process and certainty of explanations of these decisions. We demonstrate the method on synthetic, reproducible dataset and compare it to the most popular explanation frameworks.

Keywords: machine learning · rules · uncertainty · decision trees · explainability

1 Introduction

Introducing uncertainty into the machine learning (ML) process is an important research topic in the field of knowledge discovery across different areas of applications. It gained special importance in pervasive and mobile systems, where contextual information is delivered by different, possibly distributed providers. It is also an important field of study in the area of data analysis of sensor data, e.g. from industrial machines. Such sensors may not be always available, and produce uncertain, vague or ambiguous information (e.g. noisy readings, missing values, anomalous events). In an intelligent systems that use such information for knowledge discovery and decision support, capability of learning and reasoning under different types of uncertainty is a fundamental requirement.

A lot of research was devoted to providing robust methods for handling uncertainty in machine learning algorithms starting from Quinlan's C4.5 algorithm [12] for dealing with missing values in training sets and ending on more recent advances on uncertainty management in knowledge discovery from data streams [9]. Most of this research focuses primarily on the efficiency of algorithms in terms of accuracy or resources. However, due to the UE GDPR regulations, the understanding of the model becomes one of the fundamental requirement for every artificial intelligence system [8].

Explainability is not a new concept in the field of artificial intelligence [15]. However, it has been most extensively developed over the last decade due to the huge successes of black-box ML models such as deep neural networks in sensitive application contexts like medicine, industry 4.0 etc. Although a variety methods were developed over the years to support explainability of ML models such as Lime [13], Shap [11], Anchor [14] the quality of their explanations depends highly on the quality of the model predictions (e.g. accuracy). Yet, the information on the accuracy is not transferred in an explicit way to the explanation itself, leaving the final assessment of the explanation quality to the user.

In this paper we present an extension to our work in the area of semi-automatic knowledge discovery from data streams with uncertain or missing class labels [2,3,4], that aims at exploiting the semantic knowledge representation and uncertainty handling for the purpose of explainability improvement. We based our method on decision tree generation algorithm that uses modified information gain split criterion, which takes into account uncertainty of data. We show how our research can be used to increase explainability and understandability (intelligibility) of intelligent systems. In particular we show how uncertain decision trees can be used to build models that approximate arbitrary machine learning model locally and provide rule-based explanation for each decision made by the ML model.

In our approach we focus on a robust model-agnostic solution. First of all, our goal was to provide a method for building models which will be human understandable. Secondly, we wanted to create an algorithm that will not only inform the user about possible uncertainty in decision process, but could also inform other system components (or an expert) about the impact that the uncertainty may have on the model performance. The former could be used in user-centric solutions to trigger mediation with the user and request human assistance in upgrading or modifying the model.

The rest of this paper is organized as follows. The algorithm for building uncertain decision trees is presented in Section 2. In Section 3 we discuss the interpretability of models generated with our algorithm. Application of these models to generating local uncertain explanations is given in Section 4. We demonstrate our solution and present a comparison with existing explanability frameworks in Section 5. Finally, a short summary of our work is presented in Section 6.

2 Uncertain decision trees

In this section we describe the underlying mechanism that allows to build LUX models. The mechanism is based on the uID3 decision tree generation algorithm proposed by us in [2] and extended here for the purpose of explanation generation.

The uID3 algorithm is based on a heuristic that uses the modified information gain split criterion that includes uncertainty of training data into the calculation. This allows to apply it to a variety of algorithms that are based on it, such as classic ID3 algorithm, or more complex, incremental versions such as VFDT [7] or CVFDT [10].

The classic information gain formula for the attribute A and a training set X is defined as follows:

$$Gain(A) = H(X) - \sum_{v \in Domain(A)} \frac{|X_v|}{|X|} H(X_v) \tag{1}$$

Where X_v is a subset of X, such that for every $x \in X$ value of $A = v$. The entropy for the training set X is defined as follows:

$$H(X) = -\sum_{v \in Domain(C)} p(v) \log_2 p(v) \tag{2}$$

Where $Domain(C)$ is a set of all classes in X and $p(v)$ is a ratio of the number of elements of class v to all the elements in X.

In case of the uncertain data, the $p(v)$ from the equation (2) has to be defined as a probability of observing element of class v in the dataset X. This probability will be denoted further as capital $P_{total}(C = v)$. Similarly, a fraction $\frac{|X_v|}{|X|}$ from equation (1) has to be redefined as a probability of observing value v of attribute A in the dataset X. This probability will be referred later as $P_{total}(A = v)$ and is defined as a probability of observing a value v_j^i of an attribute A_i in the set X_t that contains k training instances. This can be defined as follows:

$$P_{total}(A_i = v_j^i) = \frac{1}{k} \sum_{X_t \ni P_{j=1\ldots n}} P_j(A_i = v_j^i) \tag{3}$$

Similarly $P_{total}(C = v_j)$ can be defined, which represents probability of observing class label in a set. Having that, the uncertain information gain measure can be defined as shown in the equation (4).

$$Gain^U(A) = H^U(X) - \sum_{v \in Domain(A)} P_{total}(A = v) H^U(X_v) \tag{4}$$

Where the H^U is the uncertain entropy measure defined as:

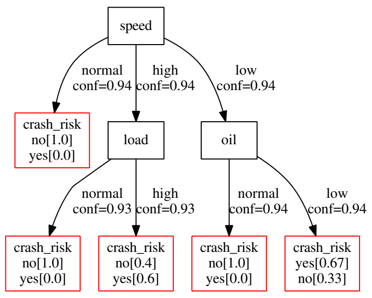
$$H^U(X) = -\sum_{v \in Domain(C)} P_{total}(C = v) \log_2 P_{total}(C = v) \tag{5}$$

Uncertain information gain and entropy represented by the equations (4) and (5) are used to build decision tree. The complete procedure of generating the uncertain tree is presented in the algorithm 1. In such a tree, every branch connecting two nodes contains statistics about the accuracy of data used to grow the subtree, as shown in Figure 1a.

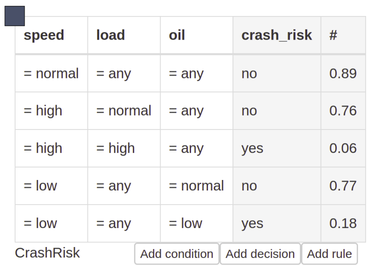
Data used to generate this tree is a special format of ARFF (the format of the WEKA ML tool), called uARFF. The example of such file used to generate the tree from Figure 1a is presented in Listing 1.1. It presents a dataset for classifier using certain parameters of a machine to predict its malfunctions. Some of the data is uncertain, and denoted as

Algorithm 1: uID3 algorithm to grow a decision tree from uncertain data

```
Input: data $X$; set of attributes $A$
Output: uncertain decision tree $uT$
if Homogeneous($X$) then
    return MajorityClass($X$)
end if
$R$ ←Best split using $Gain^U(X)$
split $X$ into subsets $X_i$ according to Domain($R$);
for each $i$ do
    if $X_i \neq \emptyset$ then
        $uT_i$ ←uID3($X_i, A$)
    else
        $uT_i$ is a leaf labeled with MajorityClass($X$)
    end if
end for
return a root $R$ of the decision tree
```


speed
normal conf=0.94
high conf=0.94
low conf=0.94
crash_risk no[1.0] yes[0.0]
load
oil
normal conf=0.93
high conf=0.93
normal conf=0.94
low conf=0.94
crash_risk no[1.0] yes[0.0]
crash_risk no[0.4] yes[0.6]
crash_risk no[1.0] yes[0.0]
crash_risk yes[0.67] no[0.33]

(a) Decision tree generated with uncertain data



speed	load	oil	crash_risk	#
= normal	= any	= any	no	0.89
= high	= normal	= any	no	0.76
= high	= high	= any	yes	0.06
= low	= any	= normal	no	0.77
= low	= any	= low	yes	0.18

(b) XTT2 rule-based format of the decision tree

Fig. 1: Decision tree and a corresponding decision table

an alternative divided by semicolon, with probabilities or confidence in square brackets.

Listing 1.1: uARFF file format

```
@relation machine.symbolic

@attribute speed {high, normal, low}
@attribute temperature {high, normal, low}
@attribute load {high, normal}
@attribute oil {low, normal}
@attribute crash_risk {yes, no}

@data
high[0.5];low[0.3];normal[0.2],high,high,low,yes
high,high,high,low,yes
normal[0.2];high[0.7];low[0.1],high,high,normal,no
low,normal,high,normal,no
low,low,normal,normal[0.3],no
low,low,normal,low,yes
normal,low,normal,low,no
high,normal,high,normal[0.6];low[0.4],yes
high,low,normal[0.4],normal,no
low,normal,normal,normal,no
high,normal,normal,low,no
normal,normal,high,low[0.4],no
normal,high,normal,normal,no
low,normal,high,low,yes
```

In [2], we presented an evaluation of this method on highly distorted dataset for predicting human emotional condition based on the physiological readings. As shown in Table 1, our method was not worse than approaches that included only most probable class. However, the main advantage was the ability to quantify decision accuracy not only by the statistics in leaves but also by the certainty of data used to generate the tree. We argue that this helps in increasing transparency of both learning and decision making, improving interpretability of the model. This topic will be discussed in the next section.

Algorithm	uID3	ZeroR	J48	HoeffdingTree	NaiveBayes	RandomForest
Accuracy	49.71	21	48.96	46.42	49.55	42.91

Table 1: Evaluation summary of an uncertain decision tree generator [2]

3 Interpretability of uncertain decision trees

The learning algorithm presented in previous section does not improve drastically the accuracy of the classification [2,4]. However, the additional information that is stored in the model may be used to give user a deeper insight into the decision and learning processes. It provides more compact, and efficient way of encoding uncertain knowledge than more sophisticated methods.

Specifically, the $P_{total}(A = v)$ is included for each branch. Such information is useful while translating decision tree into rule-based knowledge representation. In such a translation uncertain branches can be verified by the user or skipped, keeping the size of the knowledge model small.

The translation from the uncertain decision tree into rule representation is straightforward. Every branch of the tree is considered one rule. Additionally, the information about the branch uncertainty is translated as a certainty factor [6] of a particular rule. The total uncertainty of the branch is calculated as a product of all the probabilities associated with edges that form the branch. Let assume that there is a branch B, which includes n attributes $A_1, A_2, \ldots, A_n$, and n edges associated with the values of these attributes. Therefore, the total branch uncertainty is defined as follows:

$$U(B) = \prod P_{total}(A_i = v_i) \cdot Acc(B) \tag{6}$$

Where Acc is the accuracy of the classification of the branch B denoted by the leaf node.

An example of the rule set for the uncertain decision tree presented in Figure 1a is given in Figure 1b. Our XTT2 rule representation [6] uses certainty factors algebra for representing uncertainty. This allows for direct modification of confidence levels of rules by the user as it does not require any knowledge in the area of probability theory.

In the Figure 1b the first rule is a translation of the first branch of the tree that was given in Figure 1a, according to equation (6)[1]. Probabilities are expressed by a number

[1] The model was created with HWeD editor we developed, available on-line on `https://heartdroid.re/hwed/`

$p \in [0;1]$, while certainty factors algebra operates on the range $[-1;1]$. Therefore a simple transformation from probability space to certainty factors space was given: $cf = p \cdot 2 - 1$. It is worth noting, that the probability theory has different foundations than certainty factors algebra, and such transformation is performed by us only to capture a simple intuition for what the value of probability may mean in the certainty factors space. This transformation does not have any formal basis though.

Such a notation allows the user to verify potentially incorrect rules (e.g. rule three in Figure 1b) by deleting, adding or modifying the certainty factors. Additionally, it allows the system itself to identify the parts of the model that may be corrupted by uncertain data as the knowledge about the uncertain dataset is retained in the model.

4 Local Uncertain Explanations (LUX)

The uncertain decision tree defined previously can be used as stand alone model that solves ML classification tasks. However, in order to exploit its properties, the knowledge about certainty of readings or class labels are required. This knowledge is not always available in real-life setting, but can be obtained as an output from other machine learning models, as the predictions generated by such models are in most of the cases returned with some level of certainty that can be considered an approximation of probability.

Such an observation makes our uID3 algorithm perfectly fit the requirements of the local explanation models, which aim at building simple (and therefore interpretable) model on a fraction of data, which forms a local neighborhood of the instance in consideration. In this section the detailed information on how such an explanation can be generates will be given.

4.1 Building the local model

The main goal of the local, interpretable model L is to approximate the model M in surrounding to some instance $x^{(i)} \in X$ in order to provide explanation to the decision of the model M. This assumes that locally, the decision boundary of the original model M is simple enough to be approximated with simpler, yet interpretable model L. Under this assumption we get that $L(x^{(i)}) = M(x^{(i)})$ holds in a neighborhood N of $x^{(i)}$ and hence the impact of the features of L are similar in M as well. Such an impact may be considered a simple explanation of a decision of the model M for instance $x^{(i)}$.

For the sake of the discussion let us assume that we want to provide an explanation of a decision of any model M that can be trained on dataset X. An example of such a setting was given in Fig. 2, where two different models (SVM and XGBoost) were trained on the same training set. The figure shows decision boundaries for both of the classifiers and the instance (marked red) for which we would like to obtain an explanation.

The main goal of the explanation mechanism is not to provide a correct solution to the classification problem, but explain the decision of the model M. Hence, the local model L should approximate the model M even in cases where the latter one is wrong. This leads to the conclusion that the approximated model should be trained with target labels acquired directly from the model M. However, the predictions from the model M are uncertain, as the model itself is an approximation of an unknown function.

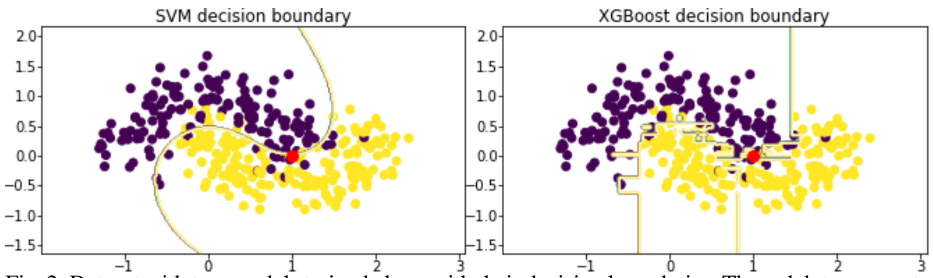


Fig. 2: Dataset with two models trained along with their decision boundaries. The red dot represents an instance for which an explanation is needed.

Figure 3 shows the uncertainty of a prediction of a SVM and XGBoost classifiers. Such an uncertainty should not only be taken into consideration while training local approximation model L but also should be transferred to the final explanations. This will allow for better assessment of the quality of the explanation by an expert.

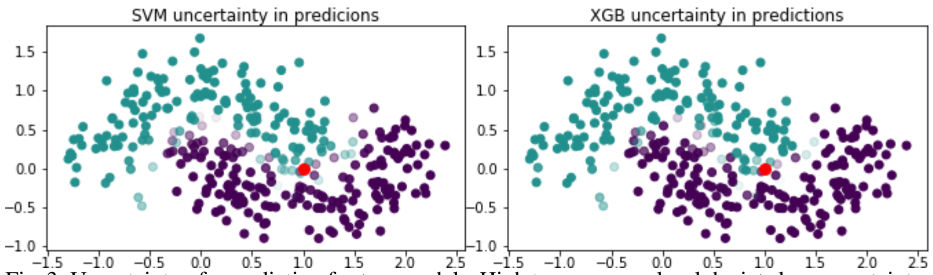


Fig. 3: Uncertainty of a prediction for two models. High transparency level depicts low uncertainty.

We exploit the fact that the local model L is trained with uncertain labels to use uID3 mechanism. In order do build the local model, we first select a neighborhood N of a training instance $x^{(i)}$ in consideration. The neighborhood is created in a stratified way in order to assure existence of both positive and negative training examples. Hence, the neighborhood N of size K is defined in Equation (7).

$$N(x^{(i)}, K) = \left\{ x^{(k)} \in X : d(x^{(i)}, x^{(k)}) \leq D_i^{(K)} \right\} \tag{7}$$

Where $D_i^{(K)}$ is K-th element from a tuple D_i defined for all m instances from a training set as:

$$D_i = \left\{ d(x^{(i)}, x^{(1)}), d(x^{(i)}, x^{(2)}), \ldots, d(x^{(i)}, x^{(m)}) \right\}$$

Where D is sorted in ascending order, and $d(x^{(i)}, x^{(j)})$ is a distance between instances i-th and j-th. Figure 4 depicts the neighborhood for the instance $x^{(i)} = (1.0, 0.0)$ (marked red).

Finally, the uID3 algorithm is run on the training dataset formed by $N(x^{(i)}, K)$ and the LUX model is created. Figure 5a presents a fragment of training set obtained

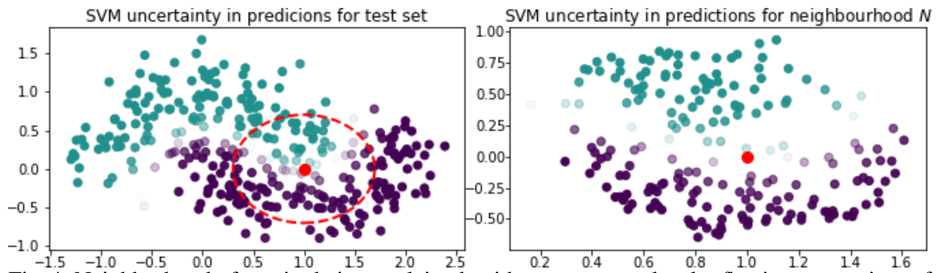


Fig. 4: Neighborhood of a point being explained, with transparency level reflecting uncertainty of the prediction. Right figure provides the full dataset, while the left figure only the neighborhood selection.

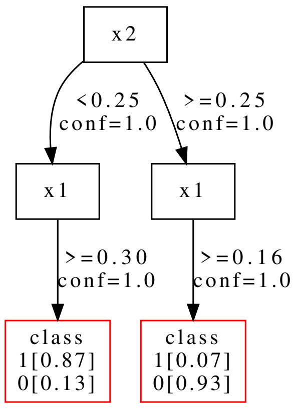
from neighborhood of a point $x^{(i)} = (1.0, 0.0)$. Figure 5b on the other hand depicts the uncertain decision tree obtained by running uID3 algorithm on the dataset. In the next section we discuss how the final explanation is obtained from the LUX model and how it can be combined with external systems to increase its intelligibility.

```
@relation lux

@attribute x1 @REAL
@attribute x2 @REAL
@attribute class {1,0}

@data
0.94,0.01,1[0.48]
0.87,-0.04,1[0.64]
1.02,-0.16,1[0.78]
1.14,0.08,1[0.37]
1.01,-0.21,1[0.83]
1.10,-0.19,1[0.81]
0.80,-0.13,1[0.81]
0.91,-0.23,1[0.87]
0.77,-0.12,1[0.83]
1.01,-0.28,1[0.89]
0.97,-0.28,1[0.89]
...
```

(a) Training set

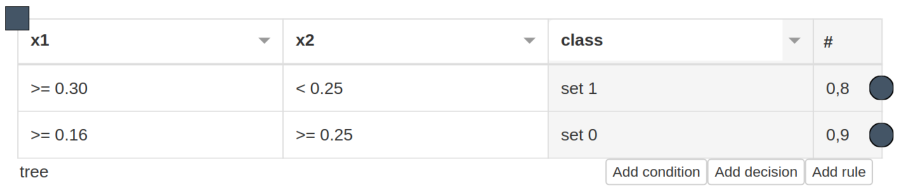


(b) LUX model

Fig. 5: Training set obtained by sampling neighborhood of a point $x^{(i)}$ (a) and LUX model (b) generated form the data using uID3 algorithm.

4.2 Generating explanation

Generating explanations from the LUX model is straightforward and is obtained by feeding LUX model with instance $x^{(i)}$. The branch that is activated during the classification forms the rule that defines an explanation. Figure 6 shows an XTT2 table generated for SVM classifier with uID3 toolkit[2]. The first rule is the one that was triggered by the inference process, and thus is considered an explanation of a decision of the original SVM model. The float number in the last column marked with # denotes the certainty of the rule.



x1	x2	class	#
>= 0.30	< 0.25	set 1	0,8
>= 0.16	>= 0.25	set 0	0,9

Fig. 6: XTT2 table generated as an explanation to the LUX model given in Figure 5b

It is worth noting that the explanation is a valid XTT2 table, that can be combined with the larger rule-based system and process with HeaRTDroid inference engine [5]. This is especially useful in settings where there exists some kind of domain knowledge that cannot be detected on such a small fraction of data, defined with N. Such a knowledge can be encoded with rules and serve as additional guards of correctness of the explanation. For instance if a LUX system forms an explanation rule that based on two attributes *temperature* and *pressure* that a water *boils*, one can easily develop additional rules that will put more strict constraints on both *pressure* and *temperature*, as there exist a law of physics that forbids some of the values coexist.

The next section provides more insight into the validity of the explanation generated by the LUX in comparison to most popular frameworks such as Lime, Shap, Anchor.

5 Evaluation

In this section we present a comparison of LUX explanations and selected explanation mechanism. We will focus on two aspects of the explanation: qualitative and quantitative. In qualitative explanation we would like to emphasize the way the explanation is presented to the user, how much information it presents to the user, and how this information can be used in the system to perform more advanced reasoning. In quantitative comparison we focus on measurable aspect of explanations as a whole, such as consistency and stability. For that purpose we will use our InXAI toolkit[3].

5.1 Qualitative comparison

Lime. Lime presents its explanation in a visual form given in Figure 7. Negative (blue) bars indicate class 0, while values (orange) indicate class 1. Values of the bars represent

[2] See: `https://github.com/sbobek/udt`.
[3] See `https://github.com/sbobek/inxai`.

the importance of each feature in making the prediction. The way to interpret the importance is by applying them to the prediction probabilities obtained from the original model. For example, if we remove the variable x2, we expect the classifier to predict class 0 with probability 0.52 - 0.14 = 0.38.

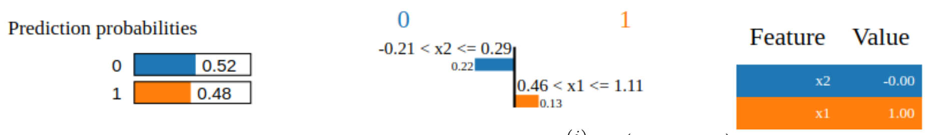


Fig. 7: Lime explanation for instance $x^{(i)} = (1.0, -0.0)$

This information itself brings an insight on the performance of the main model M and its prediction confidence, however it does not include any information on the confidence of the explanation itself. Although the results are available also as numerical values, they are not formalized in any form of logical rules, nor executable model to provide further inference.

Shap This framework provides a lot more interactive methods for visualizing explanations, however the quality of the explanation is not included within. The explanation presented in Figure 8 shows how features contribute to push the model output from the base value (the average model output over the training dataset) to the model output. Features pushing the prediction higher are shown in red, those pushing the prediction lower are in blue. Similarly to Lime, there is no information on the confidence of the explanation. The raw values are returned in the same way as in the case of Lime.

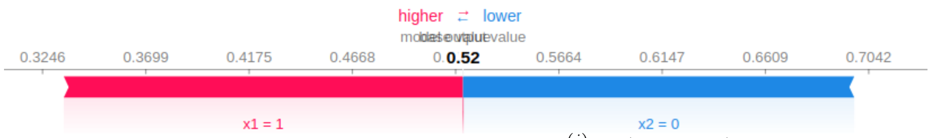


Fig. 8: Shap explanation for instance $x^{(i)} = (1.0, -0.0)$

Anchor Opposite to the former two, Anchor produces only a textual representation of an explanation. It is given to the user in a form of *anchor* which define an area in the space of features that is dominated by samples of a given class. The example of an explanation generated with Anchor is given in Listing 1.2.

Listing 1.2: Anchor eplanation for instance $x^{(i)} = (1.0, -0.0)$

```
Anchor: x2 <= 0.29 AND x1 > -0.17
Precision: 0.96
Coverage: 0.45
```

Although the explanation is presented as a rule and it can be executed within a framework to obtain prediction, it is not possible to export the rule to a format acceptable by rule-based systems in a straightforward way.

LUX In comparison to the former explanation mechanism, our solution provides both: a visual representation of explanation (either in a form of a decision tree or a XTT2 table), and information about the confidence of the explanation, which is unique with respect to previous frameworks. Furthermore, the XTT2 table presented in Figure 6 is a visual representation of a HMR+ language that can be directly executed with HeaRTDroid inference engine we developed as a part of larger explanation system that integrates also domain and expert knowledge [5].

5.2 Quantitative comparison

In this section we focus on comaprison of the frameworks in more quantitative way, providing means of assessing their quality in an automate way. For this purpose the InXAI framework will be used. We excluded Anchor from this comparison, as it does not provide feature importance information which is required to compute appropriate statistics.

Consistency measures how explanations generated for predictions of different ML models are similar to each other. Therefore, it is more related to stability of ML models with respect to decision making rather than to explanation mechanisms directly. Figure 9 (left) presents Consistency measures obtained for the dataset presented in Figure 2 and SVM and XGBoost classifiers.

It can be observed that the overall consistency of LUX is better than in other frameworks, meaning that the explanations are not that much sensitive to different models. However, the spread in the values of consistency is high, meaning that there exists regions in dataset, where different models yield different explanations. This reflects the points that are located near decision boundary.

Stability (or robustness) assures generation of similar explanations for similar input. To obtain a numerical value to this property, modified notion of Lipschitz continuity has been proposed in [1]. Figure 9 (right) presents Consistency measures obtained for the dataset presented in Figure 2 and SVM classifier.

Similarly as in case of the consistency, the Stability of LUX is generally better, although the spread is much larger than in two remaining frameworks. It means that there are regions in the dataset that yield different explanation for neighborhood points. This is caused by regions that contains points of different classes that are mixed. It can also be the case of too large neighborhood selected for the LUX model.

To summarize, both qualitative and quantitative evaluation shows the advantage of LUX explanations in cases where the interpretability and accountability of the explanation is crucial. Although the stability and consistency measures are better on average in LUX, the large spread of these values will be a subject of further investigation, especially in consideration to neighborhood selection for LUX training.

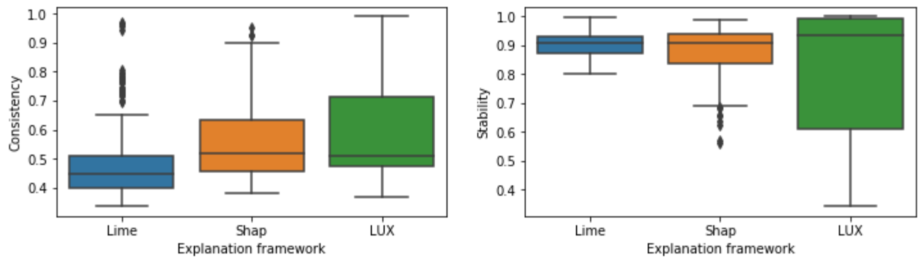


Fig. 9: Consistency and stability plots for explanation frameworks

6 Summary

In this paper we presented LUX, an algorithm for building decision trees from uncertain data and using it for generating local uncertain explanation of an arbitrary machine learning model. Such an approach transfers uncertainty from the machine learning model to the explanation in an explicit way, which helps in assessing the quality of the explanation. We demonstrated our solution on an exemplary dataset and compared it with the most popular frameworks.

Currently, our toolkit is implemented as a hybrid solution that integrates Java and Python and is available online along with the datset for reproducing experiments presented in the paper [4].

For the future work, we plan to test different approaches for selecting neighborhood for local classifier in order to improve stability and consistency of the model. In particular we would like to test similarity kernels and use values obtained from them to weight samples from neighborhood according to their applicability to the explanation.

Acknowledgements

The paper is funded from the XPM project funded by the National Science Centre, Poland under CHIST-ERA programme (NCN UMO-2020/02/Y/ST6/00070).

References

1. Alvarez-Melis, D., Jaakkola, T.S.: On the robustness of interpretability methods (2018)
2. Bobek, S., Misiak, P.: Uncertain decision tree classifier for mobile context-aware computing. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M. (eds.) Artificial Intelligence and Soft Computing - 17th International Conference, ICAISC 2018, Zakopane, Poland, June 3-7, 2018, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10842, pp. 276–287. Springer (2018). https://doi.org/10.1007/978-3-319-91262-2_25, https://doi.org/10.1007/978-3-319-91262-2_25
3. Bobek, S., Nalepa, G.J.: Uncertain context data management in dynamic mobile environments. Future Generation Computer Systems **66**(January), 110–124 (2017). https://doi.org/http://dx.doi.org/10.1016/j.future.2016.06.007, http://dx.doi.org/10.1016/j.future.2016.06.007

[4] See: https://github.com/sbobek/lux

4. Bobek, S., Nalepa, G.J.: Uncertainty handling in rule-based mobile context-aware systems. Pervasive and Mobile Computing **39**(August), 159–179 (2017). https://doi.org/http://dx.doi.org/10.1016/j.pmcj.2016.09.004, `http://dx.doi.org/10.1016/j.pmcj.2016.09.004`
5. Bobek, S., Nalepa, G.J., Ślażyński, M.: HeaRTDroid – rule engine for mobile and context-aware expert systems. Expert Systems **36**(1), e12328 (2019). https://doi.org/10.1111/exsy.12328, `https://doi.org/10.1111/exsy.12328`
6. Bobek, S., Nalepa, G.: Compact representation of conditional probability for rule-based mobile context-aware systems. In: Bikakis, A., Fodor, P., Roman, D. (eds.) Rules on the Web. From Theory to Applications. Lecture Notes in Computer Science, Springer International Publishing (2015)
7. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 71–80. KDD '00, ACM, New York, NY, USA (2000). https://doi.org/10.1145/347090.347107, `http://doi.acm.org/10.1145/347090.347107`
8. Goodman, B., Flaxman, S.: EU regulations on algorithmic decision-making and a "right to explanation" (2016), `http://arxiv.org/abs/1606.08813`, cite arxiv:1606.08813Comment: presented at 2016 ICML Workshop on Human Interpretability in Machine Learning (WHI 2016), New York, NY
9. Goyal, N., Jain, S.K.: A comparative study of different frequent pattern mining algorithm for uncertain data: A survey. In: 2016 International Conference on Computing, Communication and Automation (ICCCA). pp. 183–187 (April 2016). https://doi.org/10.1109/CCAA.2016.7813714
10. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 97–106. KDD '01, ACM, New York, NY, USA (2001). https://doi.org/10.1145/502512.502529, `http://doi.acm.org/10.1145/502512.502529`
11. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 4768–4777. NIPS'17, Curran Associates Inc. (2017)
12. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
13. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should i trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 1135–1144. KDD '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939778, `https://doi.org/10.1145/2939672.2939778`
14. Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: High-precision model-agnostic explanations. In: AAAI Publications, Thirty-Second AAAI Conference on Artificial Intelligence (2018)
15. Schank, R.C.: Explanation: A first pass. In: Kolodner, J.L., Riesbeck, C.K. (eds.) Experience, Memory, and Reasoning. pp. 139–165. Lawrence Erlbaum Associates, Hillsdale, NJ (1986)