teex: A toolbox for the evaluation of explanations

Jesus Antonanzas, Yunzhe Jia, Eibe Frank, Albert Bifet, Bernhard Pfahringer

Neurocomputing 555 (2023) 126642

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Original software publication

teex: A toolbox for the evaluation of explanations

Jesús M. Antoñanzas [a,c,*], Yunzhe Jia [a], Eibe Frank [a,b], Albert Bifet [a,d], Bernhard Pfahringer [a,b]

[a] AI Institute, University of Waikato, Hamilton, New Zealand
[b] Department of Computer Science, University of Waikato, Hamilton, New Zealand
[c] Department of Physics, Universitat Politècnica de Catalunya, Barcelona, Spain
[d] LTCI, Télécom Paris, Institut Polytechnique de Paris, Palaiseau, France

ARTICLE INFO

Communicated by S. Wang

Keywords:
Explainable AI
Explanation evaluation
Python

ABSTRACT

We present `teex`, a Python **t**oolbox for the **e**valuation of **ex**planations. `teex` focuses on the evaluation of local explanations of the predictions of machine learning models by comparing them to ground-truth explanations. It supports several types of explanations: feature importance vectors, saliency maps, decision rules, and word importance maps. A collection of evaluation metrics is provided for each type. Real-world datasets and generators of synthetic data with ground-truth explanations are also contained within the library. `teex` contributes to research on explainable AI by providing tested, streamlined, user-friendly tools to compute quality metrics for the evaluation of explanation methods. Source code and a basic overview can be found at github.com/chus-chus/teex, and tutorials and full API documentation are at teex.readthedocs.io.

Code metadata

Table 1 contains metadata for the code.

1. Introduction

Explainable Artificial Intelligence (*XAI*) is the field dedicated to making AI models human-understandable. An important part of this field are explainer methods, which, by generating explanations, give users a general overview of a model's functioning (global explanation) or the reasoning behind a single prediction (local explanation). `teex` is a tool designed to evaluate explainer methods in `XAI`, particularly those that generate local explanations for classifications made by machine learning models (such as LIME [1]). `teex` provides an extensible collection of metrics that enable comparison between post-hoc and ground-truth local explanations. It also provides built-in support for multiple explanation types—saliency maps, decision rules, feature importance vectors, and word importance vectors—while aiming to be extensible in this regard. Although its use is not strictly bound to the availability of ground-truth explanations (e.g., it can be used to compare explanations generated by different methods), `teex` contains multiple, easy-to-access real-world and artificial datasets [2–5] with ground-truth explanations to enable benchmark comparisons 1. In the case of the real-world data included, expert annotations are provided as ground-truth explanations. To enable integration with related software, we provide wrappers for extraction and usage of local explanations from popular Python XAI libraries.

`teex` supports the usage of *XAI* evaluation methods in a (1) general, (2) extensible, and (3) simple way:

- By allowing evaluation of the most frequently used explanation types in a model- and explainer-independent manner.
- By clearly encapsulating functionality: evaluation and data generation methods exist within distinct modules, inside a sub-package for each explanation type. APIs are standardized between all modules and the architectural structure is clearly laid out.
- By providing single-line evaluation APIs (as shown in the example below) and comprehensive documentation, including tutorials and use cases. This enables seamless integration with evaluation pipelines.

```
from teex.saliencyMap.data import Kahikatea
from teex.featureImportance.eval import feature_importance_scores

X, y, exps = Kahikatea()[:] # download and unpack data
predExplanations = get_explanations(model, X)
metrics = ['fscore', 'cs', 'auc']
feature_importance_scores(exps, predExplanations, metrics)
# >>> [0.8, 0.7, 0.7]
```

1.1. Related software

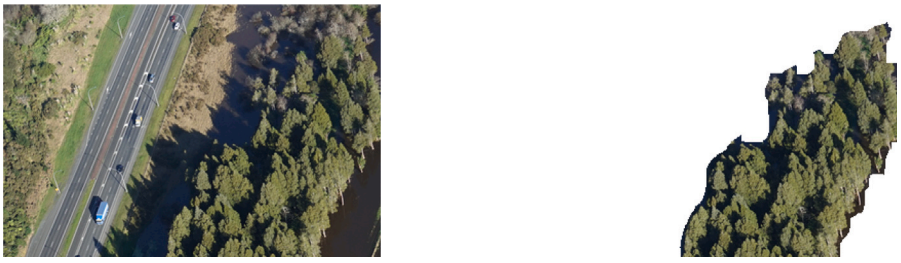
Evaluating the quality of explanations is a hard problem, mainly because there is no standardized set of metrics or methods to do so. In

* Corresponding author at: AI Institute, University of Waikato, Hamilton, New Zealand.
E-mail address: jmariaan@waikato.ac.nz (J.M. Antoñanzas).

https://doi.org/10.1016/j.neucom.2023.126642
Received 27 March 2023; Received in revised form 16 July 2023; Accepted 24 July 2023
Available online 3 August 2023

Table 1
Code metadata (mandatory)

Nr.	Code metadata description	Please fill in this column
C1	Current code version	v1.1.3
C2	Permanent link to code/repository used for this code version	https://github.com/chus-chus/teex
C3	Permanent link to Reproducible Capsule	https://github.com/chus-chus/teex/releases/tag/v1.1.3
C4	Legal Code License	MIT
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	Python
C7	Compilation requirements, operating environments & dependencies	*tqdm, numpy, scikit-learn, scipy, sympy, Pillow*
C8	If available Link to developer documentation/manual	https://teex.readthedocs.io
C9	Support email for questions	jmariaan@waikato.ac.nz, alvin.jia@waikato.ac.nz



(a) `Kahikatea` sample. (b) `Kahikatea` g.t.

Fig. 1. Example observation from the `Kahikatea` dataset.

Table 2
Comparison of libraries that include functionality to evaluate explanations.

	Evaluation-centric?	Model-independent?	Explainer-independent?	*XAI* Datasets?
Captum	No	No	No	No
AIX360	No	No	No	No
Torchray	No	No	No	No
Quantus	**Yes**	No	No	No
teex	**Yes**	**Yes**	**Yes**	**Yes**

particular, when no ground-truth explanations are available, evaluation is bound to indirect metrics related to the underlying model's behavior, usually measuring fidelity, sensitivity, complexity, or other aspects. While this form of evaluation is valid, it is desirable to streamline automatic evaluation against ground truths. Although this approach requires data with expert annotations, it is straightforward to use and understand, additionally being model- and explainer-independent. It also provides a way to establish whether a model produces correct classifications for the right reasons.

We believe that providing a tool that implements this approach to evaluating explanations is an important step for the community. There are libraries solely specializing in generating explanations (`Alibi` [6], `dalex` [7], `iNNvestigate` [8], `zennit` [9]) and libraries that include some evaluation metrics (`Captum` [10], `AIX360` [11], `TorchRay` [12]), but there is relatively little comprehensive tool support for the streamlined evaluation of *XAI* techniques. The only other dedicated library (`Quantus` [13]) does not focus on evaluating against ground-truth explanations. Important features of these libraries are compared in Table 2.

2. Software description

To evaluate the explanation quality, the required elements are demonstrated in Fig. 3, where e is the explanation generated by an explanation method. e explains the prediction y made by a black box model b for a given input x. To evaluate the quality of e, there needs to be a ground truth explanation $\hat{e}$. Now, given an evaluation function Q, we can compute $Q(e, \hat{e})$. See Fig. 2 for a concrete example of the evaluation process.

`teex` makes the evaluation process convenient by:

- Providing a collection of metrics Q that are commonly used in the literature for evaluating explanations.
- Providing easy access to $\hat{e}$ for a collection of machine learning datasets, where the ground truth explanations for individual instances are available. This information is difficult to collect in practice and is not available in many traditional datasets.

2.1. Datasets

We provide datasets with ground-truth explanations for four explanation representations, including both real datasets and synthetic ones. All datasets share the same user API. In particular, `teex` includes, as of now:

- Image data with saliency maps as explanations.
- Text data with word importance as explanations.
- Tabular data with rules as explanations.
- Tabular data with feature importance as explanations.

2.1.1. Image data

We provide several image classification datasets with ground-truth saliency maps that are, e.g., suitable for the evaluation of the explanations of classifications obtained from convolutional neural networks:

- **Kahikatea**[1] contains images for Kahikatea classification. The Kahikatea is an indigenous plant in New Zealand. The data has

[1] https://zenodo.org/record/5059769#.Y-OCSnZBwQ-

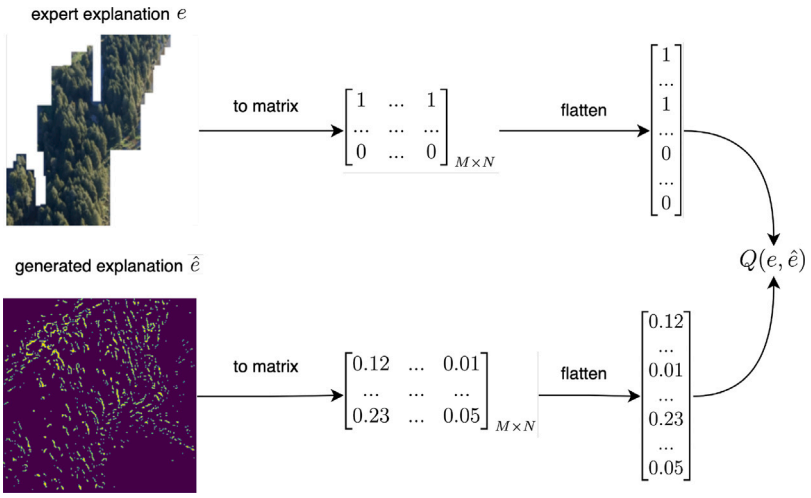


Fig. 2. Explanation evaluation procedure for a `saliency map` image explanation, given an expert explanation and an explanation generated by an external method. (1) First, the expert explanation is transformed into a binary 2D matrix, where each entry corresponds to a pixel, and is set to 1 or 0 depending on whether it contains the object or not. (2) Then, the generated explanation is transformed into a 2D matrix, where each entry is the normalized attribution (from 0 to 1) of the corresponding pixel. This matrix, depending on the quality metric that the user chooses, will need to be binarized by choosing a value threshold. (3) After this, both matrices are flattened into 1D vectors. (4) Finally, both vectors can be quantitatively compared using a selected metric Q.

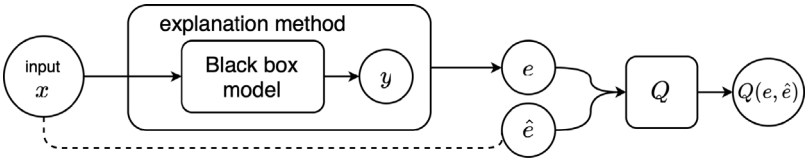


Fig. 3. Evaluating explanation e (generated with any explanation method) by contrasting it against ground truth $\hat{e}$.

human-annotated pixel-level explanations highlighting the tree pixels in individual images if Kahikatea trees are presented. It contains 519 images (232 images contain Kahikatea).

```
from teex.saliencyMap.data import Kahikatea
X, y, exps = Kahikatea()[:]
```

An example image from the Kahikatea data and a corresponding explanation can be found in Fig. 1.

- **CUB-200-2011**[2] and **Oxford-IIIT Pet**[3] are well-known dataset frequently used for evaluating the accuracy of image classification techniques and are also available in `teex`. They exhibit over 19,000 images and 230 distinct classes.

```
from teex.saliencyMap.data import CUB200
X, y, exps = CUB200()[:]
```

An example image from the CUB-200-2011 data and corresponding explanation can be found in Fig. 4.

- The included **synthetic image data** generation method, adapted from [5] can produce an arbitrary number of images with pixel-level explanations of one class. An example can be found in Fig. 5. Given some parameters, first, a pattern image (yellow pixels in Fig. 5) is generated, then the images are generated with cells randomly colored in cyan, green or blue with black background. An image is considered positive if it contains the pattern and its true explanation consists of the pixels corresponding to the pattern.

```
from teex.saliencyMap.data import SenecaSM
X, y, exps = SenecaSM()[:]
```

2.1.2. Text data

For evaluation on text data, we provide, for now, a subset of the **20NewsGroup**[4] dataset with the word importance as explanations for individual articles:

```
from teex.wordImportance.data import Newsgroup
X, y, exps = Newsgroup()[:]
X[3]
>> b"...Subject: Re: Hi Volt from battery\nNntp-Posting-Host:..."
dataGen.classMap[y[3]]
>> 'electronics'
exp[3]
>> {'volt': 1.0, 'battery': 1.0, 'batteries': 0.5,
  'electronics': 1.0,'flash': 0.5}
```

2.1.3. Tabular data

Evaluating explanations on tabular data is another important task. We provide synthetic tabular data generation methods with two types

[2] https://www.vision.caltech.edu/datasets/cub_200_2011/
[3] https://www.robots.ox.ac.uk/vgg/data/pets/
[4] https://www.kaggle.com/datasets/crawford/20-newsgroups

(a) CUB-200-2011 sample. (b) CUB-200-2011 g.t.

Fig. 4. Example observation from the CUB-200-2011 dataset.

(a) Synthetic sample. (b) Synthetic g.t.

Fig. 5. Example observation from the Synthetic image dataset generation method.

of explanations – decision rules and feature importance – based on different underlying transparent models.

The example of feature importance is similar to the above example of word importance for text data. That is, an observation is a numerical list with an associated class, and its corresponding explanation is a list of numerical importances for each feature, bounded from -1 (inversely correlated with the observation's class) to 1 (positively correlated). The data points are sampled from normal distributions, labeled by thresholding randomly generated linear functions, and explained using their gradients. The observations are the same in the case of the synthetic decision rule data, but the explanations contain conditional rules for the classification of the observation into a class, instead of just importances. The data points are sampled from normally distributed clusters, and their explanations are generated by parsing decision trees trained on them. Below is an example of what the decision rule explanation looks like. See the original Ref. [5] for an in-depth explanation of these methods.

```
from teex.decisionRule.data import SenecaDR
X, y, exps = SenecaDR(nSamples=1000, nFeatures=3)[:]
print(f'Observation: {X[0]} \nLabel: {y[0]} \nExplanation: {exps[0]}')
>> Observation: [1.25824083 1.37756901 0.4123272 ]
>> Label: 0
>> Explanation: IF 0.111 < 'c', -0.015 < 'a', 0.901 < 'b' <= 2.31
>>            THEN 'Class' = 0

```

2.2. Metrics

Here we present an overview of the current quality metrics included in teex.

2.2.1. Feature importance

- Cosine Similarity [14]. If the explanations are vectors of feature importance, regardless of whether the values are binary or in the range $[0, 1]$, we can measure the explanation quality using Cosine Similarity:

$$Q(e, \hat{e}) = cosine(e, \hat{e}) = \frac{\|e \cdot \hat{e}\|}{\|\hat{e}\| \cdot \|e\|} \tag{1}$$

  where $e \cdot \hat{e}$ is the dot product, and $\|e\|$ is the L2-norm of e. The closer the metric is to 1, the greater the explanation quality of e.
- Precision, Recall, F_1 score. For these metrics, both ground truth and prediction are binarized according to a user-defined threshold. Once this has been done, the explanation quality can be measured by the well-known precision, recall, and F_1 score metrics.

$$Q(e, \hat{e}) = Precision(e, \hat{e}) = \frac{|e \cdot \hat{e}|}{|e|} \tag{2}$$

  Precision measures how many selected features are truly important: its value is 1 if all features with non-zero importance in the generated explanation are also non-zero in the ground truth explanation.

$$Q(e, \hat{e}) = Recall(e, \hat{e}) = \frac{|e \cdot \hat{e}|}{|\hat{e}|} \tag{3}$$

  Recall measures how many truly important features are selected. Its value is 1 if all features with non-zero importance in the ground truth explanation are also non-zero in the generated explanation. Note that this can be easily achieved with an explanation that assigns non-zero importance to all features.

$$Q(e, \hat{e}) = F_1(e, \hat{e}) = \frac{2 \cdot Precision(e, \hat{e}) \cdot Recall(e, \hat{e})}{Precision(e, \hat{e}) + Recall(e, \hat{e})} \tag{4}$$

  The closer the F_1 score, the harmonic mean of precision and recall, is to 1, the greater the explanation quality of e.
- AUC: The area under the ROC curve provides an alternative to the above metrics. In the case of this metric, only the ground truth is binarized and ground truth importance scores are used to obtain the ranking for the calculation of the area under the curve.

2.2.2. Saliency maps

In the context of image classification, a saliency map explanation e for a prediction $f(x)$ is represented as a two dimensional array of the same size, where each entry in e is a real number and provides the attribution of the corresponding pixel in x. For the evaluation of saliency maps, we provide the same metrics as in the case of feature importance. In this case, each pixel in an image is considered to be a feature: an saliency map explanation of size (M, N) is flattened into a feature importance vector of length $M \times N$.

Table 3
In 3.a (left), we report the average metrics for the test set—comparing explanations extracted with each of the specified methods to the ground truth. In 3.b (right), these same explanations are compared to the ones extracted via `Integrated Gradients` (scores for Integrated Gradients are blank because we would be comparing them to themselves). Note that the explanations are binarized where necessary in order to compute *f1score*, *precision*, *recall*, *cosine similarity* and *AUC* (in 3.b, just for the explanation considered as ground truth, which would be `Integrated Gradients`). 0.5 was set as the binarization threshold.

	AUC	F1 Score	Prec.	Rec.	C.S.
Grad. SHAP	.51	.24	.3	.45	.493
GradCAM	.46	.15	.37	.37	.495
deepLIFT	.53	.21	.35	.29	.493
Guided backprop.	.47	.18	.32	.35	.491
Occlusion	.5	.23	.32	.53	.485
Int. grad.	.5	.26	.29	.48	.492

	AUC	F1 Score	Prec.	Rec.	C.S.
Grad. SHAP	.59	.5	.59	.61	.997
GradCAM	.51	.21	.47	.38	.995
deepLIFT	.75	.27	.61	.41	.996
Guided backprop.	.51	.24	.48	.36	.992
Occlusion	.5	.31	.47	.52	.972
Int. grad.	–	–	–	–	–

2.2.3. Decision rules

In the case of tabular data, `teex` can also process explanations in the form of decision rules, not just feature importance scores.

- Complete Rule Quality. Each rule explanation can be converted into a vector where, for the ith feature in the observed domain, the values of the lower and upper bounds $v_i^{(l)}, v_i^{(u)}$ are reported. For example, the explanation $x_0 > 5, x_1 < 2$ can be converted to $\{(x_0^{(l)}, 5), (x_0^{(u)}, \infty), (x_0^{(l)}, -\infty), (x_0^{(u)}, 2)\}$. Then the explanation quality can be measured as:

$$Q(e, \hat{e}) = \frac{1}{N} \sum_{i=1}^{|e|} \delta_\epsilon(e_i, \hat{e}_i) \tag{5}$$

where

$$\delta_\epsilon(e_i, \hat{e}_i) = \begin{cases} 1 & \text{if } |e_i - \hat{e}_i| \leq \epsilon \wedge |e_i| \neq \infty \wedge \hat{e}_i \neq \infty, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

  Here, ϵ is the similarity threshold, and N is the number of lower and upper bounds that are neither ∞ nor $-\infty$ in both e and $\hat{e}$. This means that the closer a rule explanation's lower and upper limits are to the real lower and upper limits, the better the explanation is. The more limits that are close, the closer the explanation is to being accurate.
- All metrics available for feature importance are also available for evaluating rules, where a transformation of the rule into a feature importance vector is performed first. The feature importance vector will have the same number of entries as the number of features in the domain. Each entry will be either 1 or 0, depending on whether the feature appears, or not, in the rule in question.

2.2.4. Word importance

For word importance, we have the same metrics as for feature importance, where the vocabulary is considered the feature space and a word importance explanation may or may not contain words from the vocabulary.

3. Experiments

To demonstrate `teex`'s usage, we present a benchmark comparison. A classification model (pre-trained *SqueezeNet* [15] from `PyTorch` [16]) has been fine-tuned on the `Kahikatea` dataset, obtaining 0.82 F1 score on the validation data and 0.65 F1 score on the test data when evaluating classification performance. From this model, we extract local explanations for the test set for 40 positively labeled observations, using the following methods: `Gradient SHAP` [17], `gradCAM` [18], `deepLIFT` [19], `Guided Backpropagation` [20], `Occlusion` [21] and `Integrated Gradients` [22]. Then, we use `teex` to quantitatively evaluate them. The results are shown in Table 3. Code is available on `teex`'s GitHub `Saliency Map` demo.

Inspecting the results in Table 3.a, all explainers achieve similar scores for the various metrics when compared to the ground truth, but there are some differences in precision and recall in particular. Also, the scores are not high, which indicates that the model has not entirely learned the particular features of the Kahikatea trees. Table 3.b reflects a characteristic of our evaluation procedure: the binarization threshold that is chosen for the evaluation directly influences the results. In this case, the *cosine similarity* scores indicate that explanations are almost identical to those of `Integrated Gradients`, but the other scores do not. This is due to how the threshold (0.5) interacts with the distributions of attributions, and needs to be taken into account by the user. This is particularly true if the explanations that are being compared seem to be very similar to each other.

For this simple experiment we have used our tool to quickly obtain relevant information about the model behavior, as well as compare the performance of various explainer methods on it. This could help iterate in the model-development process and allow for the selection of the right, and best-performing explainer for our particular use-case. Combining `teex` with other explanation evaluation libraries would bring us even more benefits.

4. Summary

`teex` is a Python library comprised of tools to help researchers and end-users evaluate the quality of local explanations against ground truth explanations for labels provided by human experts (or algorithmically in the case of synthetic data). It includes a comprehensive set of quality metrics that can be applied to different explanation types and also aims to serve as a hub for datasets with ground-truth local explanations, which are notoriously hard to find. `teex` has been conceived as an effort to help make *XAI* evaluation a more streamlined, reproducible, simple, and clear procedure, with ease-of-use and flexibility in mind, and can be used in tandem with other libraries for the generation and the evaluation of explanations.

5. Future improvements

Future improvements will be focusing on expanding access to more datasets with explanations, as well as more metrics. Beyond the immediate scope of the project, it may also be possible to extend the library to other predictive tasks such as regression or clustering.

CRediT authorship contribution statement

Jesús M. Antoñanzas: Conceptualization, Methodology, Software, Validation, Data curation, Writing – original draft, Writing – review & editing. **Yunzhe Jia:** Supervision, Methodology, Validation, Writing – original draft, Writing – review & editing. **Eibe Frank:** Supervision, Writing – original draft, Writing – review & editing. **Albert Bifet:** Supervision, Writing – original draft. **Bernhard Pfahringer:** Supervision, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All data used in this paper is openly available via teex or the original, referenced, sources.

Acknowledgments

teex has been developed as part of the TAIAO project (Time-Evolving Data Science/Artificial Intelligence for Advanced Open Environmental Science), funded by the New Zealand Ministry of Business, Innovation, and Employment (MBIE).

References

[1] M.T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?": Explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, 2016, pp. 1135–1144.
[2] Y. Jia, E. Frank, B. Pfahringer, A. Bifet, N. Lim, Studying and exploiting the relationship between model accuracy and explanation quality, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2021, pp. 699–714.
[3] C. Wah, S. Branson, P. Welinder, P. Perona, S. Belongie, The Caltech-UCSD Birds-200–2011 Dataset, Tech. Rep. CNS-TR-2011-001, California Institute of Technology, 2011.
[4] O.M. Parkhi, A. Vedaldi, A. Zisserman, C.V. Jawahar, Cats and dogs, in: IEEE Conference on Computer Vision and Pattern Recognition, 2012.
[5] R. Guidotti, Evaluating local explanation methods on ground truth, Artificial Intelligence 291 (2021) 103428, http://dx.doi.org/10.1016/j.artint.2020.103428, URL https://www.sciencedirect.com/science/article/pii/S0004370220301776.
[6] J. Klaise, A.V. Looveren, G. Vacanti, A. Coca, Alibi explain: Algorithms for explaining machine learning models, J. Mach. Learn. Res. 22 (181) (2021) 1–7, URL http://jmlr.org/papers/v22/21-0017.html.
[7] H. Baniecki, W. Kretowicz, P. Piatyszek, J. Wisniewski, P. Biecek, Dalex: Responsible machine learning with interactive explainability and fairness in python, J. Mach. Learn. Res. 22 (214) (2021) 1–7, URL http://jmlr.org/papers/v22/20-1473.html.
[8] M. Alber, S. Lapuschkin, P. Seegerer, M. Hägele, K.T. Schütt, G. Montavon, W. Samek, K.-R. Müller, S. Dähne, P.-J. Kindermans, Innvestigate neural networks!, J. Mach. Learn. Res. 20 (93) (2019) 1–8, URL http://jmlr.org/papers/v20/18-540.html.
[9] C.J. Anders, D. Neumann, W. Samek, K.-R. Müller, S. Lapuschkin, Software for dataset-wide XAI: From local explanations to global insights with Zennit, CoRelAy, and ViRelAy, 2021, CoRR, arXiv:2106.13200.
[10] N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan, O. Reblitz-Richardson, Captum: A unified and generic model interpretability library for PyTorch, 2020, arXiv:2009.07896.
[11] V. Arya, R.K.E. Bellamy, P.-Y. Chen, A. Dhurandhar, M. Hind, S.C. Hoffman, S. Houde, Q.V. Liao, R. Luss, A. Mojsilović, S. Mourad, P. Pedemonte, R. Raghavendra, J. Richards, P. Sattigeri, K. Shanmugam, M. Singh, K.R. Varshney, D. Wei, Y. Zhang, One explanation does not fit all: A toolkit and taxonomy of AI explainability techniques, 2019, URL https://arxiv.org/abs/1909.03012.
[12] R. Fong, M. Patrick, A. Vedaldi, Understanding deep networks via extremal perturbations and smooth masks, 2019, CoRR arXiv:1910.08485.
[13] A. Hedström, L. Weber, D. Bareeva, F. Motzkus, W. Samek, S. Lapuschkin, M.M.-C. Höhne, Quantus: An explainable AI toolkit for responsible evaluation of neural network explanations, 2022, arXiv:2202.06861.
[14] Y. Jia, J. Bailey, K. Ramamohanarao, C. Leckie, M.E. Houle, Improving the quality of explanations with local embedding perturbations, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 875–884.
[15] F.N. Iandola, M.W. Moskewicz, K. Ashraf, S. Han, W.J. Dally, K. Keutzer, SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size, 2016, CoRR, arXiv:1602.07360.
[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An imperative style, high-performance deep learning library, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc. 2019, pp. 8024–8035, URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
[17] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc. 2017, URL https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.
[18] R.R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, D. Batra, Grad-CAM: Why did you say that? Visual explanations from deep networks via gradient-based localization, 2016, CoRR, arXiv:1610.02391.
[19] A. Shrikumar, P. Greenside, A. Kundaje, Learning important features through propagating activation differences, 2017, CoRR, arXiv:1704.02685.
[20] J.T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: The all convolutional net, 2014, arXiv preprint arXiv:1412.6806.
[21] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, 2013, CoRR, arXiv:1311.2901.
[22] M. Sundararajan, A. Taly, Q. Yan, Axiomatic attribution for deep networks, 2017, CoRR, arXiv:1703.01365.

Jesús M. Antoñanzas received a B.Sc. in Data Science and Engineering by the Polytechnic University of Catalonia and is undertaking a M.Sc. in Computer Science at the same institution. He has been involved as a research assistant with the UPC Complex Systems research group and with the Computer Science department at the University of Waikato. His main interests are machine learning and its applications.

Yunzhe Jia He is a postdoc research fellow at the University of Waikato, New Zealand. He obtained his Ph.D. degree from the University of Melbourne, Australia. Before that he received his Master's degree in Computer Science from New York University. His research area is in explainable machine learning and pattern mining.

Eibe Frank Professor Frank is a computer scientist whose primary area of interest is machine learning and its applications. He is a core developer of the WEKA machine learning software and has more than 100 publications on machine learning methods and their application to data mining, text mining, and areas of research outside computer science.

Albert Bifet is a Professor of AI, Director of the Te Ipu o te Mahara AI Institute at the University of Waikato and Co-chair of the Artificial Intelligence Researchers Association (AIRA). His research focuses on Artificial Intelligence, Big Data Science, and Machine Learning for Data Streams.

He is leading the TAIAO Environmental Data Science project, and he is co-leading the open source projects MOA Massive On-line Analysis, StreamDM for Spark Streaming and SAMOA Scalable Advanced Massive Online Analysis.

He is the co-author of a book on Machine Learning from Data Streams published at MIT Press. He served as PC Co-Chair of DSAA'2021, Co-Chair of the Industrial track of IEEE MDM 2016, ECML PKDD 2015, and as Co-Chair of KDD BigMine (2019–2012), and ACM SAC Data Streams Track (2023–2012).

Bernhard Pfahringer Bernhard is an Austrian computer scientist and AI researcher, since 2000 professor at the Computer Science Department at the University of Waikato, New Zealand. He is a co-director of the Te Ipu o te Mahara AI Institute. His research interests covers Machine Learning and Programming Languages.