

Des explications garanties pour l'IA

Démonstration

Jeudi 22 octobre 2026

IA^2 2026 – Institut d'Automne en Intelligence Artificielle
Le Touquet-Paris-Plage, France
CRIL, Université d'Artois & CNRS

Gilles AUDEMARD
Steve BELLART
Louenas BOUNIA
Mathis KONARSKI
Jean-Marie LAGNIEZ
Pierre MARQUIS
Mehdi SABIRI
Nicolas SZCZEPANSKI

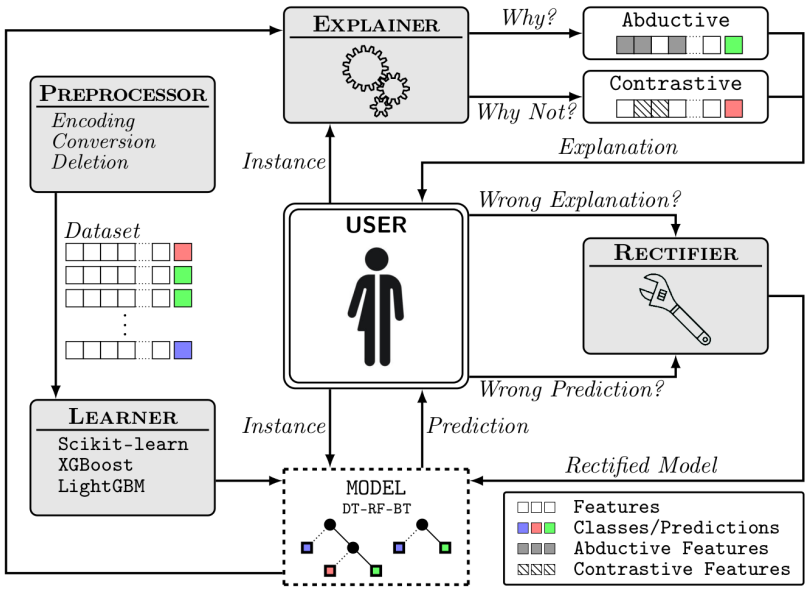








PyXAI en un coup d'œil



Expliquer une instance à un utilisateur avec PyXAI

- Bibliothèque **Python** open source du CRIL
- Explique les **prédictions** des modèles à base d'arbres : arbres de décision, forêts aléatoires, arbres boostés
- **Classification** et **régression**
- Explications **formelles** : calculées par raisonnement logique sur le modèle lui-même

Chaque explication est **garantie** : elle est vraie pour le modèle, sans approximation

Pourquoi ? Pourquoi pas ?

Pourquoi ? (explication abductive)

Pourquoi l'instance *x* a-t-elle été classée ainsi ?
En régression : pourquoi la valeur prédite tombe-t-elle dans un intervalle donné ?

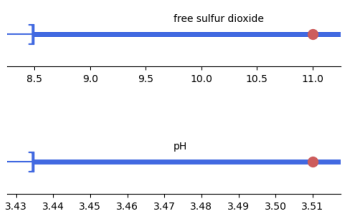
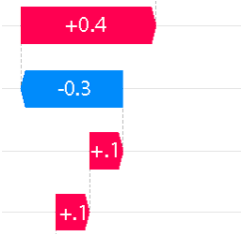
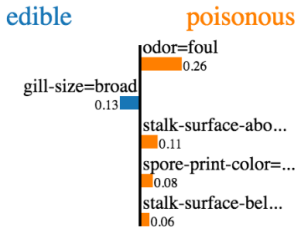
Pourquoi pas ? (explication contrastive)

Pourquoi *x* n'a-t-elle pas été classée comme l'utilisateur l'attendait ?

Modèle	scikit-learn	XGBoost	LightGBM
Arbre de décision	`DecisionTreeClassifier`		
Forêt aléatoire	`RandomForestClassifier`		
Arbres boostés		`XGBClassifier`, `XGBRegressor`	`LGBMRegressor`

Modèles pris en charge

PyXAI, SHAP et LIME

	PyXAI	SHAP	LIME
Approche	Formelle	Agnostique	Agnostique
Modèles	Arbres	Tous	Tous
Résultat	Formule logique	Valeurs de Shapley	Poids locaux
Propriété	**Abductive**	Non abductive	Non abductive
Explication			

SHAP et LIME ne donnent **aucune garantie** : leurs explications peuvent être fausses

Une explication abductive, c'est quoi ?

Une demande de prêt, décrite par trois variables :

- A_1 : le revenu (numérique)
- A_2 : la situation : salariée, sans emploi ou indépendante (catégorielle)
- A_3 : mariée ou non (booléenne)

Pourquoi Alice $= (2\,200, \text{salariée}, \text{mariée})$ obtient-elle son prêt ?

Méthode	**Explication**
PyXAI	$A_1 > 2\,000$
Agnostique	A_1 : 0,2 ; A_2 : 0,7 ; A_3 : 0,7

- **PyXAI** : $A_1 > 2\,000$ est **abductive**, tous les cas $(A_1 > 2\,000, *, *)$ ont la même prédiction
- **Agnostique** : (salariée, mariée) n'est **pas abductive**, car $(1\,900, \text{salarié}, \text{marié})$ est un **contre-exemple**

Des contre-exemples, souvent

Une explication est **incorrecte** s'il existe un **contre-exemple** : un cas qui la respecte, mais que le modèle classe autrement.

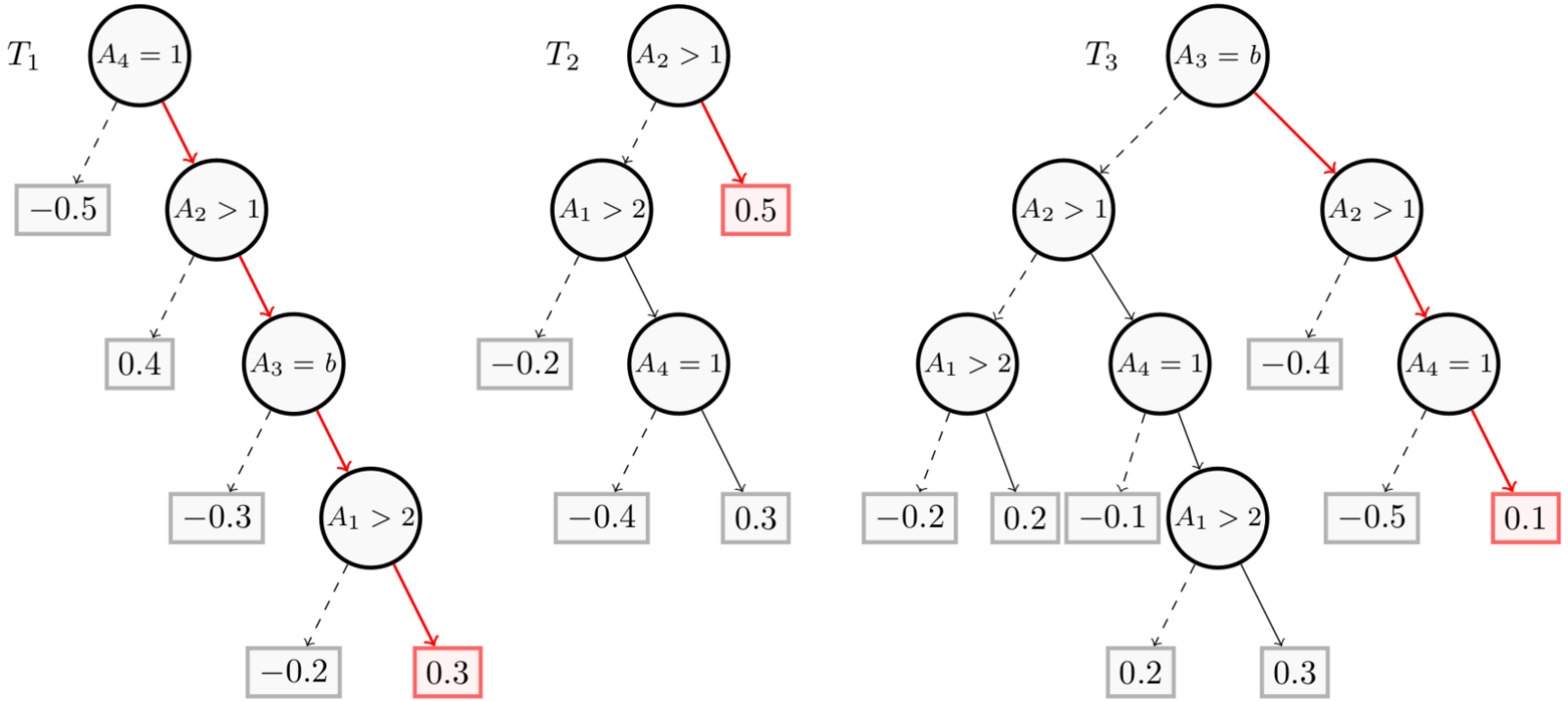
Jeu de données	LIME	Anchor	SHAP
adult	61,3 %	80,5 %	70,7 %
lending	24,0 %	3,0 %	17,0 %
rcdv	94,1 %	99,4 %	85,9 %
compas	71,9 %	84,4 %	60,4 %
german	85,3 %	99,7 %	63,0 %

Part d'explications incorrectes (Ignatiev, Narodytska et Marques-Silva, 2019)

PyXAI : des explications **correctes par construction** (garantie formelle)

Raison directe

Arbres boostés $F = \{T_1, T_2, T_3\}$ sur 4 variables et l'instance $x = (A_1 = 4, A_2 = 3, A_3 = b, A_4 = 1)$



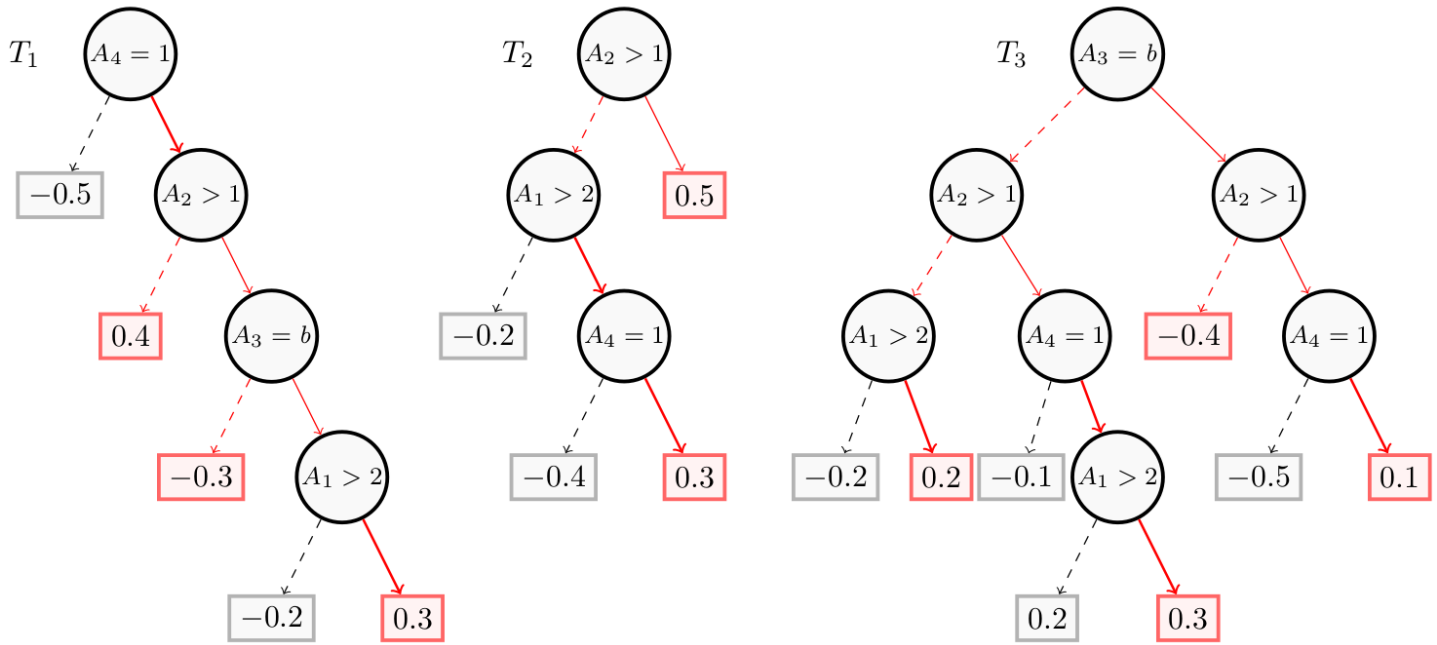
$w(T_1, x) = 0{,}3$, $w(T_2, x) = 0{,}5$ et $w(T_3, x) = 0{,}1$, donc
$W(F, x) = 0{,}9 > 0$: x est classée **1**

Raison directe : les conditions des chemins suivis par x (en rouge)
$(A_1 > 2, A_2 > 1, A_3 = b, A_4 = 1)$

Raison suffisante

Une **raison suffisante** est une explication abductive **minimale** :
aucune de ses parties n'est encore une explication abductive

$t = (A_1 > 2, A_4 = 1)$ est une raison suffisante pour x



Raison suffisante : la vérification

Toute instance x' qui respecte t entre dans l'un des 4 cas ci-dessous, selon les deux variables libres A_2 et A_3 :

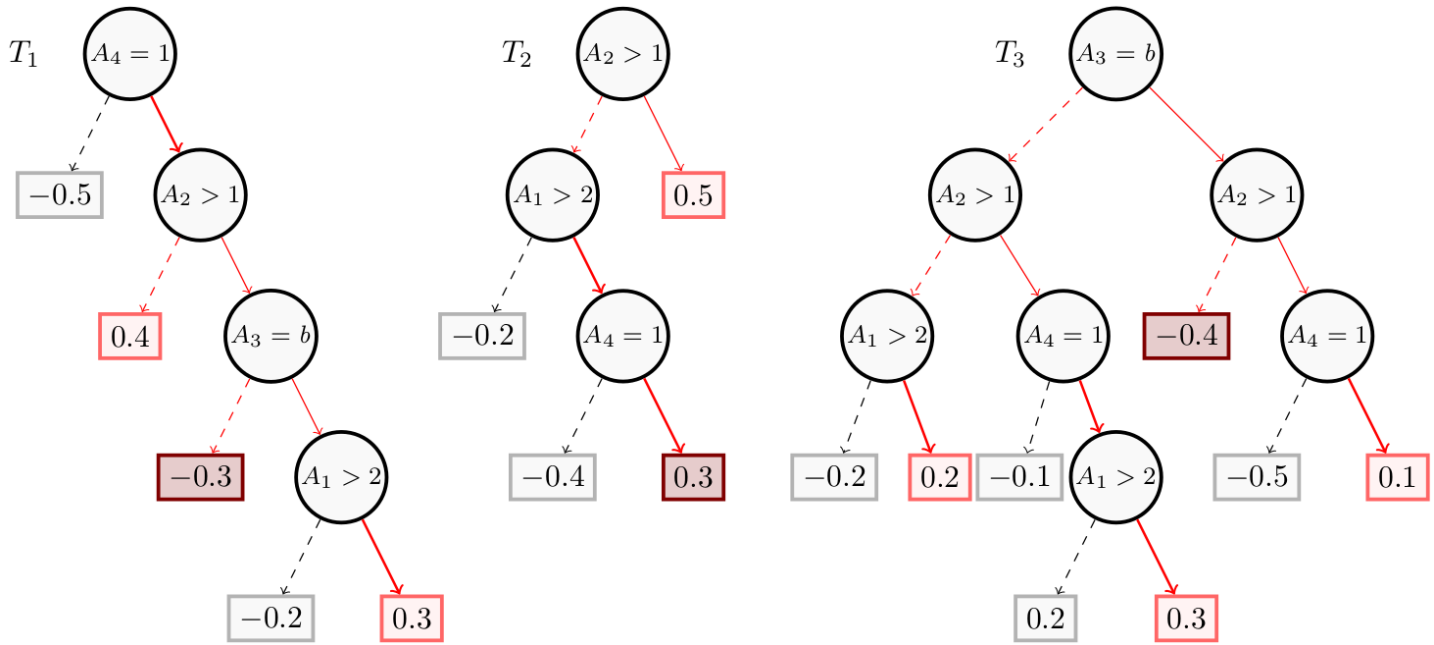
$A_2 > 1$	$A_3 = b$	$w(T_1, x')$	$w(T_2, x')$	$w(T_3, x')$	$W(F, x')$
non	non	0,4	0,3	0,2	**0,9**
non	oui	0,4	0,3	$-0{,}4$	**0,3**
oui	non	$-0{,}3$	0,5	0,3	**0,5**
oui	oui	0,3	0,5	0,1	**0,9**

Dans tous les cas $W(F, x') > 0$: la prédiction reste **1**

Mais calculer des raisons suffisantes est **coûteux**

Raison spécifique à l'arbre

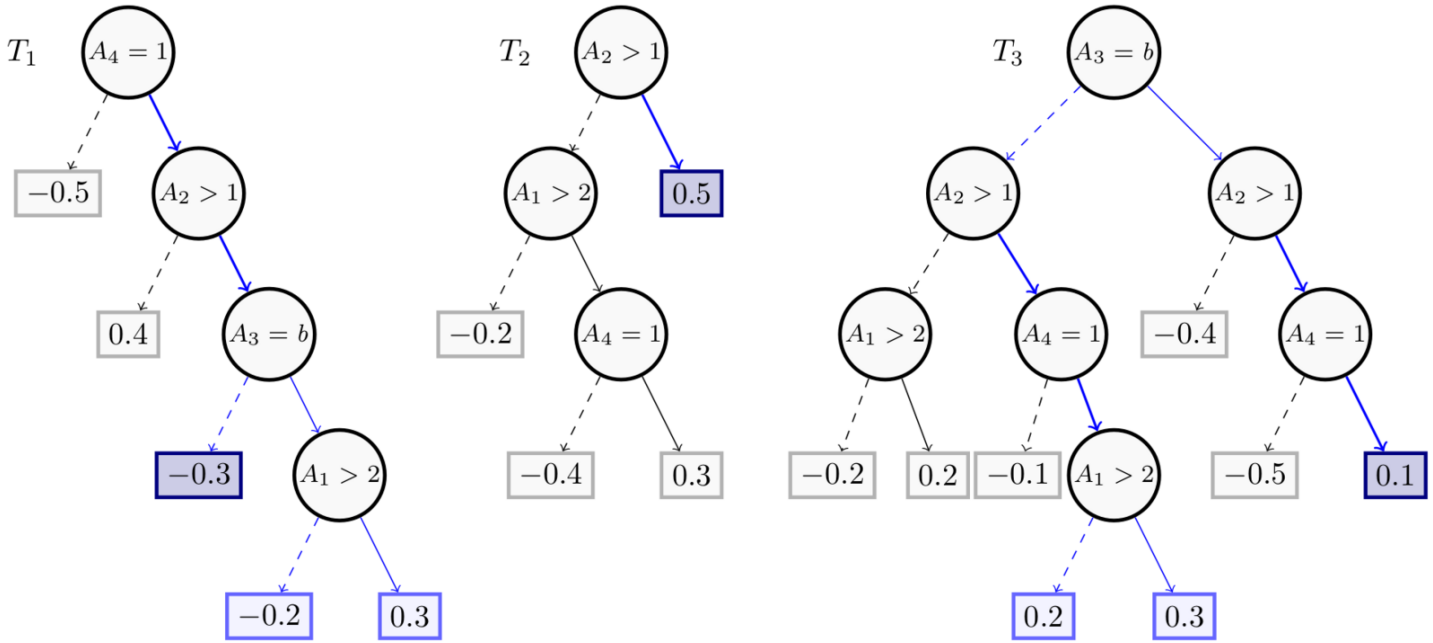
Pour chaque arbre, on prend la **pire feuille** qu'une instance respectant t peut atteindre (en rouge foncé)



$t = (A_1 > 2, A_4 = 1)$ n'est **pas** spécifique à l'arbre :
$w_\downarrow(t, T_1) + w_\downarrow(t, T_2) + w_\downarrow(t, T_3) = -0{,}3 + 0{,}3 - 0{,}4 = -0{,}4 < 0$

Raison spécifique à l'arbre

$t' = (A_2 > 1, A_4 = 1)$ est spécifique à l'arbre :
$-0{,}3 + 0{,}5 + 0{,}1 = 0{,}3 > 0$



Bien plus **faciles à calculer** que les raisons suffisantes, et toujours **abductives**

Quelles explications pour quels modèles ?

Explication	Idée	AD	FA	AB
Directe	Les conditions du chemin suivi	✓	✓	✓
Suffisante	Abductive et minimale	✓	✓	
Majoritaire	Vraie pour une majorité d'arbres		✓	
Spécifique à l'arbre	Abductive et rapide à calculer			✓
Préférée	Suffisante, selon les préférences de l'utilisateur	✓	✓	
Contrastive	**Pourquoi pas ?** Ce qu'il faut changer	✓	✓	✓

AD : arbre de décision, FA : forêt aléatoire, AB : arbres boostés

Chaque explication peut être **vérifiée** :
`is_sufficient_reason`, `is_contrastive_reason`, ...

PyXAI en action

```
from pyxai import Learning, Explaining

learner = Learning.Scikitlearn("iris.csv",
                               problem_type=Learning.CLASSIFICATION)
model = learner.evaluate(splitting_method=Learning.HOLD_OUT,
                         model_type=Learning.DT)
instance, prediction = learner.get_instances(model, n=1,
                                             is_correct=True)

explainer = Explaining.initialize(model, instance)

# pourquoi ?
reason = explainer.sufficient_reason(n=1)
print(explainer.to_features(reason))

# pourquoi pas ?
contrastive = explainer.contrastive_reason()
print(explainer.to_features(contrastive, contrastive=True))
```

Quelques lignes pour apprendre un modèle, choisir une instance et l'expliquer

Importer son propre modèle

```
from sklearn.ensemble import RandomForestClassifier
from pyxai import Learning, Explaining

model_rf = RandomForestClassifier().fit(X, y)    # X : tableau NumPy

learner, model = Learning.ModelIO.import_models(
    model_rf, instances_type="tabular")
instance, prediction = learner.get_instances(
    dataset=df, model=model, n=1)

explainer = Explaining.initialize(model, instance)
print(explainer.to_features(explainer.majoritary_reason()))
```

scikit-learn, XGBoost et LightGBM : **pas besoin de réapprendre** le modèle

Théories : les connaissances du domaine

Une **théorie** décrit des connaissances sur les données. PyXAI s'en sert pour **écarter les explications impossibles** et **simplifier** celles qu'il produit.

Sans théorie, PyXAI pourrait proposer « revenu ≥ 30 » tout en gardant « revenu < 20 » : c'est impossible.

```
australian_types = {
    "numerical": Learning.DEFAULT,
    "categorical": {"A4*": (1, 2, 3), "A5*": tuple(range(1, 15))},
    "binary": ["A1", "A8", "A9", "A11"],
}
explainer = Explaining.initialize(model, instance=instance,
                                  features_type=australian_types)
```

Les types peuvent aussi venir d'un fichier `.types` produit par le préprocesseur de PyXAI.

Préférences

- **Préférer** certaines explications : poids par variable, calculés (valeurs de Shapley, importance des variables, fréquence des mots) ou donnés par l'utilisateur
- **Exclure** des variables des explications, par exemple le sexe ou l'âge

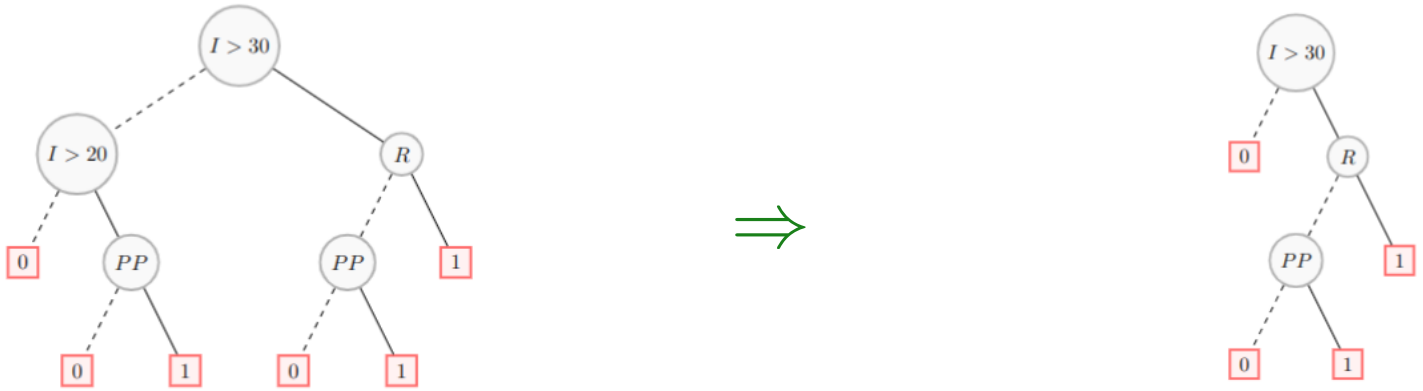
```
explainer.set_excluded_features(["sex", "age"])
reason = explainer.sufficient_reason(n=1)        # sans sex ni age
explainer.unset_excluded_features()

method = Explaining.PreferredReasonMethod.FeatureImportance
reason = explainer.preferred_sufficient_reason(method=method)
```

On Preferred Abductive Explanations for Decision Trees and Random Forests (IJCAI 2022)

Rectification

Un expert impose une règle : **revenu** $\leq 30 \Rightarrow$ **refus**. PyXAI **corrige** le modèle pour qu'il la respecte ; les autres prédictions ne changent pas.



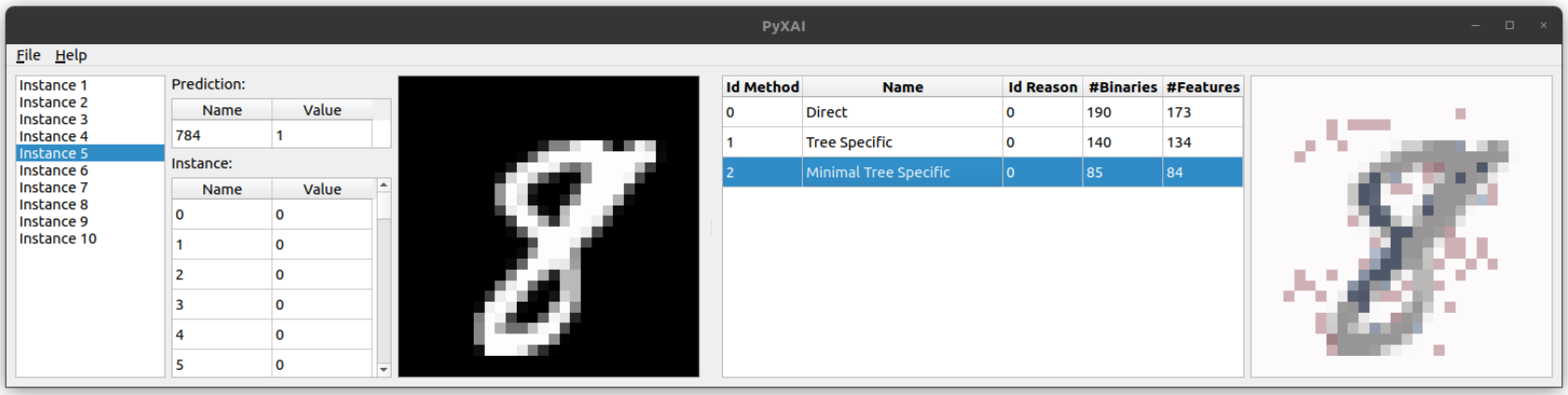
Avant Après

```
# (-1,) correspond à la condition I <= 30
model = explainer.rectify(conditions=(-1,), label=0)
```

I : revenu annuel (k€), *R* : prêt déjà remboursé, *PP* : emploi stable. Rectifying Binary Classifiers (ECAI 2023)

Interface graphique

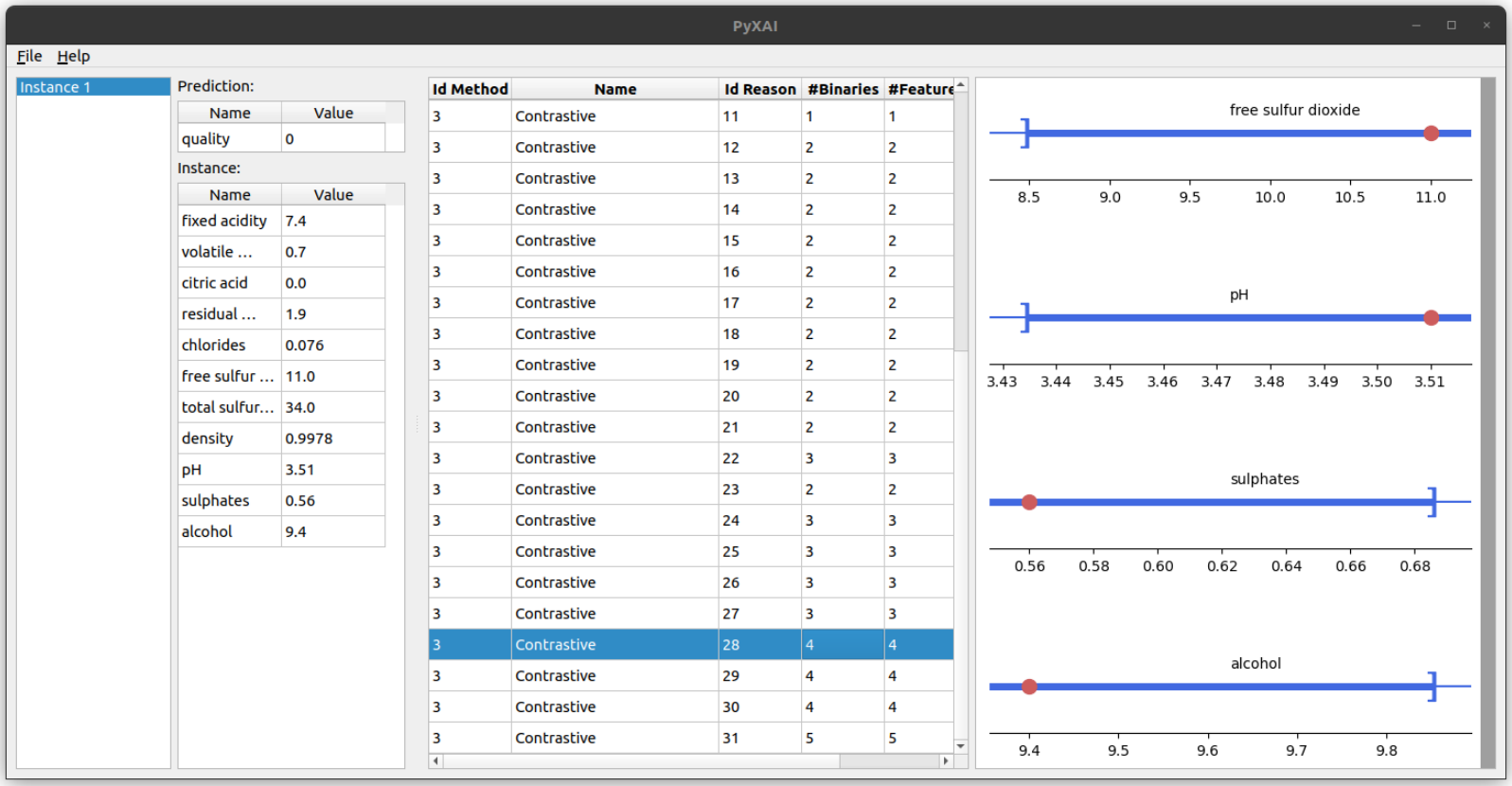
Afficher, sauvegarder et recharger des instances et leurs explications, ici sur MNIST




```
python3 -m pyxai -gui
```

Interface graphique

Explications contrastives pour un vin (jeu de données Wine Quality)



Sauvegarder et recharger

Sauvegarder les modèles, leurs hyperparamètres et les instances choisies, pour les recharger dans un autre programme.

```
models = learner.evaluate(splitting_method=Learning.K_FOLDS,
                          model_type=Learning.RF)
Learning.ModelIO.save(models, "try_save")

# dans un autre programme
learner, models = Learning.ModelIO.load("try_save")
for model in models:
    explainer = Explaining.initialize(model, instance)
    print(explainer.to_features(explainer.majoritary_reason()))
```

Pour chaque modèle : un fichier `.map` (JSON : index d'apprentissage et de test, précision, ...) et un fichier `.pkl`.

Pour suivre la démo

```
pip install pyxai

python3 -m pyxai -examples          # copie les exemples
python3 -m pyxai -explanations      # fichiers de démo pour l'interface
python3 -m pyxai -gui               # ouvre l'interface graphique
```

`pip install pyxai` (version 2.0.1, licence MIT)

Documentation

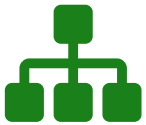
- Site : www.cril.univ-artois.fr/pyxai
  - plus de 10 notebooks Jupyter
  - référence de l'API (nouveau en v2.0)
- Code : github.com/crillab/pyxai
- PyPI : pypi.org/project/pyxai

Publications autour de PyXAI

Année	Lieu	Titre
2026	IJCAI	Computing Coverage-Based Prime Implicant Explanations for Tree-Based Models
2026	KR	A Rectification-Based Approach for Distilling Boosted Trees into Decision Trees
2025	EGC	Une interface XAI pour des modèles d'apprentissage automatique à base d'arbres
2024	ECAI	Designing an XAI Interface for Tree-Based ML Models
2024	IJCAI	On the Computation of Example-Based Abductive Explanations for Random Forests
2024	IJCAI	Deriving Provably Correct Explanations for Decision Trees: The Impact of Domain Theories
2024	IJCAI (démo)	**PyXAI: An XAI Library for Tree-Based Models**
2023	AISTATS	Computing Abductive Explanations for Boosted Trees
2023	ECAI	On Contrastive Explanations for Tree-Based Classifiers
2023	ECAI	Rectifying Binary Classifiers
2023	IJCAI	Computing Abductive Explanations for Boosted Regression Trees
2023	EGC	PyXAI : calculer en Python des explications pour des modèles d'apprentissage supervisé
2022	DKE	On the Explanatory Power of Boolean Decision Trees
2022	IJCAI	On Preferred Abductive Explanations for Decision Trees and Random Forests
2022	AAAI	Trading Complexity for Sparsity in Random Forest Explanations
2021	KR	On the Computational Intelligibility of Boolean Classifiers
2020	KR	On Tractable XAI Queries based on Compiled Representations

Auteurs : G. Audemard, S. Bellart, L. Bounia, S. Coste-Marquis, F. Koriche, J.-M. Lagniez, P. Marquis, M. Sabiri et N. Szczepanski (CRIL).

Perspectives



Réseaux de neurones

Des explications garanties au-delà des arbres



CNN

Expliquer des modèles qui travaillent sur des images



Valeurs manquantes

Expliquer quand des données sont absentes (NaN), fréquent en santé

Garder la **garantie de correction** sur des modèles et des données plus réalistes