Masterial
Extraction de règles à partir de réseaux de neurones

SULEJMANI Armen, EVRAERD Gauthier, PAVY Quentin, BOUX Corentin

M2 CNS SA 2023-2024

Table des matières

1 Introduction

De nos jours, avec l'apparition de modèles de plus en plus précis, il devient impératif de comprendre et d'expliquer les décisions prises par ces modèles d'intelligence artificielle afin d'assurer la transparence, la responsabilité et la confiance dans leur utilisation. Les modèles de prédiction, en particulier ceux basés sur des réseaux neuronaux profonds, opèrent souvent comme des boîtes noires, rendant difficile pour les utilisateurs et les parties prenantes de comprendre le raisonnement derrière leurs prédictions. On les appelle des boîtes noires dû au faites qu'on ne sait absolument pas pour quelles raisons le modèle a fait telle ou telle prédiction. On lui donne des données en entrée et on récupère une sortie seulement. L'explication des décisions des modèles de prédiction permet non seulement de dévoiler leur processus de prise de décision, mais aussi de détecter et de corriger d'éventuels biais ou erreurs. L'absence d'explications pour les décisions de ces boîtes noires induise une limitation de l'adoption de ces modèles dans des contextes socialement sensibles et critiques en matière de sécurité. Cela a forcé les chercheurs à proposer différentes méthodes pour expliquer les décisions de ces modèles.

2 Différentes méthodes d'explication

2.1 Explication globale et locale

Avant d'expliquer le fonctionnement de différentes méthodes d'explication. Il semble bon de faire la différence entre les explications globales et locales.

Une interprétation globale vise à comprendre le comportement du modèle sur l'ensemble du jeu de données. Le problème est que la frontière de décision d'un modèle du point de vue global est très souvent très complexe (Figure 1). Elle ne peut souvent pas être expliqué par un formalisme simple comme un arbre de décision ou une régression linéaire. Il faut utiliser d'autres modèles pour expliquer son comportement.

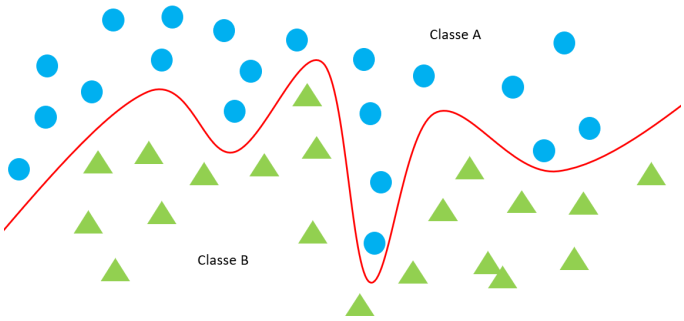


FIGURE 1 – Frontière de décision globale

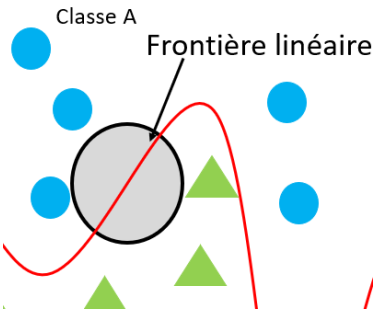


FIGURE 2 – Frontière de décision locale

Une explication locale vise quant a elle à comprendre le comportement du modèle pour une entrée spécifique du jeu de données. Cela peut servir à savoir pourquoi un modèle a classé tel individu dans telle classe ou quelles sont les caractéristiques qui ont influencé sa décision. Contrairement au point de vue global, quand on s'intéresse localement à la frontière de décision (Figure 2), elle est très simple. Elle peut être assimilée à une fonction linéaire le plus souvent. On peut donc utiliser des modèles d'explications plus simples comme un arbre de décision pour expliquer la frontière de décision.

2.2 Méthode globale

2.2.1 GLocalX

La méthode GLocalX proposée dans l'article "GlocalX - From Local to Global Explanations of Black Box AI Models"[3] en 2021, propose une méthode agnostique permettant d'obtenir une explication globale d'un modèle à partir d'explication locale de ce même modèle. Le fonctionnement de GLocalX repose sur la fusion d'explication locales en fonction de leur similarité et du Critère d'Information Bayésien (BIC) de l'explication fusionnée.

GLocalX utilise un ensemble d'explication locale à fusionner pour renvoyer un ensemble d'explication globale. Ces explications sont représentées sous la forme suivante :

$$e = \langle r = P \rightarrow y \rangle$$

Une explication e est une règle de décision r qui décrit la raison de la décision $y = b(x)$, où $b(x)$ est la prédiction renvoyée par la boite noire pour une instance x de l'ensemble de données, $P = \{p_1, ..., p_s\}$ est un ensemble de prémisses sous une forme conjonctive. Un exemple d'explication pour une demande de prêt serait :

$$e = \langle r = \{age \geq 25, job = unemployed, amount \leq 10k\} \rightarrow deny \rangle$$

La règle de décision que l'on retrouve dans l'explication ci-dessus, signifie que si la personne est âgée d'au moins 25 ans, qu'elle est sans emploi et qu'elle demande un prêt d'un montant supérieur à 10k alors le prêt sera refusé.
Le fonctionnement de GLocalX est séparé en deux algorithmes différents.

Algorithm 1 GLocalX($\mathbb{E}, \alpha$)

Input: $\mathbb{E}$ explanation theories, α filter threshold
Output: E explanation theory

$\mathrm{E} \leftarrow \emptyset$
repeat
  $\mathbb{Q} \leftarrow$ SORT($\mathbb{E}$) ▷ sort pairs of theories by similarity
  merged $\leftarrow$ `False`
  $X' \leftarrow$ batch(X)
  while $\neg$ *merged* $\wedge\ \mathbb{Q} \neq \emptyset$ **do**
    $E_i, E_j \leftarrow$ POP($\mathbb{Q}$) ▷ select most similar theories
    $E_{i+j} \leftarrow$ MERGE(E_i, E_j, X') ▷ merge theories
    if BIC(E_{i+j}) $\leq$ BIC($E_i \cup E_j$) **then** ▷ verify improvement
      merged $\leftarrow$ `True`
      break
  if *merged* **then** ▷ merge occurred
    $\mathbb{E} \leftarrow$ UPDATE(E_i, E_j, E_{i+j}) ▷ update hierarchy
until $|\ \mathrm{E}\ | > 1 \wedge$ *merged* ▷ until the merge is successful
$E \leftarrow$ FILTER(E, α) ▷ Filter final theory
return E

Algorithm 2 MERGE(E_i, E_j, X)

Input: E_i, E_j explanation theories, X batch
Output: $E_{(i+j)}$ explanation theory

$\mathrm{E} \leftarrow E_i \cup E_j$
for $x \in X$ **do**
  $C_i \leftarrow$ COVERED(x, E_j) ▷ retrieve rules in E_i covering x
  $C_j \leftarrow$ COVERED(x, E_j) ▷ retrieve rules in E_j covering x
  $C_= \leftarrow$ NON-CONFLICTING(x, C_i, C_j) ▷ non-conflicting rules in C_i, C_j and covering x
  $C_{\neq} \leftarrow$ CONFLICTING(x, C_i, C_j) ▷ non-conflicting rules in C_i, C_j covering x
  $\mathrm{E} \leftarrow \mathrm{E} \setminus (E^i \cup E^j)$
  $E_= \leftarrow$ JOIN($C_=$)
  $E_{\neq} \leftarrow$ CUT($C_{\neq}$, X)
  $\mathrm{E} \leftarrow \mathrm{E} \cup E_= \cup E_{\neq}$
return E

L'algorithme 1 GLocalX prend en entrée un ensemble d'explication locale $\mathbb{E}$ et un filtre α sur la qualité minimale de l'ensemble d'explications à retourner.
L'algorithme de GLocalX commence par trier l'ensemble d'explication par groupe de deux en fonction de leur similarité. La similarité entre deux explications est calculée comme l'indice de Jaccard de leur couverture sur un ensemble d'instance de l'ensemble de données X :

$$similarity_X(E_i, E_j) = \frac{|coverage(E_i, X) \cap coverage(E_j, X)|}{|coverage(E_i, X) \cup coverage(E_j, X)|}$$

Une explication e couvre une instance x si les prémisses P de r sont satisfaites par x. Nous élargissons cette notion aux ensembles d'explication E tel qu'on considère que l'ensemble d'explication E couvre une instance x si au moins une explication $e \in E$ couvre x. $coverage(E, X)$ renverra l'ensemble d'instances de X couverte par l'ensemble d'explication E. Inversement, on considérera que $covered(x, E)$ renverra l'ensemble d'explication de E qui couvre l'instance x.
Une fois avoir trié l'ensemble d'explication locale par groupe de deux, nous allons récupérer une partie de l'ensemble de données qui servira à la fusion des explications.
Ensuite, nous allons récupérer le groupe d'explication avec la similarité la plus élevée et exécuter l'algorithme 2 Merge sur ce groupe avec la partie de donnée X.

Dans l'algorithme Merge, pour chaque instance x de l'ensemble de données X, nous allons récupérer les ensembles d'explication C_i et C_j des ensembles d'explications E_i et E_j qui couvre l'instance x. Ensuite, nous allons séparer ces ensembles en deux : un ensemble d'explications sans conflit et un ensemble d'explications avec conflit.
L'algorithme de Merge prend en compte deux fonctions différentes, JOIN et CUT. Ces deux fonctions ont des comportements diamétralement opposés : JOIN va généraliser les explications et CUT va spécialiser ces explications en les séparant, les rendant plus fidèles à l'ensemble de données. Dans la suite de cette explication, nous considérons les opérateurs join($\oplus$) et cut($\ominus$).

Join cherche à généraliser un ensemble d'explications sans conflit en rendant leur prémisse plus générale. Selon deux explications P et Q, l'opérateur join($\oplus$) est défini comme :

$$P \oplus Q = \{P_1 + Q_1, ..., P_m + Q_m\}$$

Tel que :

$$P_i + Q_i = \begin{cases} P_i \cup Q_i & \text{intersection non nulle} \\ (min\{P_i \cup Q_i\}, max\{P_i \cup Q_i\}) & \text{intersection vide} \\ \emptyset & P_i \text{ ou } Q_i \text{ est vide} \end{cases}$$

Par exemple, si l'on considère les deux explications suivantes :

$$e_1 = \{age \geq 50, job = \text{office clerk}\} \rightarrow deny$$
$$e_2 = \{age \geq 40\} \rightarrow deny$$

L'algorithme de MERGE appliquera l'opérateur JOIN et renverra :

$$e'_1 = \{age \geq 40\} \rightarrow deny$$

La caractéristique *age* qui est partagé entre les deux explications e_1 et e_2 n'a pas une intersection vide donc l'explication fusionnée aura l'union des deux explications pour cette caractéristique tandis que la caractéristique job qui n'apparaît que dans l'explication e_1 n'apparaîtra donc pas dans l'explication résultante.

CUT cherche à séparer un ensemble d'explications avec conflit afin d'obtenir des explications plus fidèles. Selon deux explications P et Q, l'opérateur cut($\ominus$) est défini comme :

$$P \ominus Q = \{P_1 - Q_1, ..., P_m - Q_m\}$$

Tel que :

$$P_i - Q_i = \begin{cases} \{P_i, \emptyset\} & Q_i \text{ vide} \\ \{P_i, Q_i \backslash P_i\} & \text{sinon} \end{cases}$$

On remarquera que, contrairement à l'opérateur JOIN, l'opérateur CUT n'est pas symétrique. C'est pour cela que lors de la sélection des explications, la première sera celle avec la fidélité la plus élevée.
Si l'on considère les deux explications suivantes :

$$e_1 = \{age \geq 25, job = \text{unemployed}, amount \geq 10k\} \rightarrow deny$$
$$e_2 = \{age \geq 20, job = \text{manager}, amount > 8k\} \rightarrow accept$$

L'algorithme de MERGE appliquera l'opérateur CUT et renverra :

$$e_1 = \{age \geq 25, job = \text{unemployed}, amount \geq 10k\} \rightarrow deny$$
$$e'_1 = \{age \in [20, 25], job = \text{manager}, amount \in [8k, 10k] \rightarrow accept$$

On remarquera que l'explication e_1 est retournée telle qu'elle tandis que les prémisses de l'explication e_2 sont plus limitées. La caractéristique *age* est alors limité pour enlever l'intersection avec l'explication e_1 tout comme pour la caractéristique *amount*. La caractéristique job reste inchangée par rapport à l'explication e_2.

Une fois ces deux opérations effectuées, nous renvoyons l'ensemble d'explications obtenues. Cet ensemble est alors récupérer dans l'algorithme GLocalX pour ensuite vérifier le Critère d'Information Bayésien (BIC) de ce nouvel ensemble d'explication et va le comparer à celui de l'union des deux ensembles utilisés pour la fusion. Si le Critère d'Information Bayésien de l'ensemble d'explication E_{i+j} résultant de l'algorithme Merge est inférieur à celui de l'ensemble $E_i \cup E_j$, alors on remplace ce nouvel ensemble d'explication E_{i+j} à la place des ensembles E_i et E_j. On répétera ce processus jusqu'à ce qu'il ne soit plus possible de faire de fusions. Enfin, nous effectuons un tri sur les explications obtenues en fonctions de leur fidélité à l'ensemble de données afin de réduire le nombre d'explications retourné à la fin de l'algorithme.

2.3 Méthode locale

2.3.1 Anchors

La méthode Anchors, introduite sous le titre "High-Precision Model-Agnostic Explanations" [2],propose une approche robuste et précise pour expliquer les prédictions de modèles, indépendamment de leur complexité. Présentée pour la première fois dans la publication "Anchors : High-Precision Model-Agnostic Explanations" (2018), cette méthode se distingue par sa capacité à générer des explications de grande précision adaptées à divers ensembles de données et modèles de prédiction.

Le mécanisme de fonctionnement d'Anchors repose sur la définition de "points d'ancrage" (anchors) au sein de l'espace des caractéristiques. Ces points d'ancrage sont des conditions simples et compréhensibles qui, lorsqu'elles sont remplies, garantissent une prédiction du modèle. Anchors cherche à découvrir ces points d'ancrage en minimisant une fonction objective spécifique, conduisant ainsi à des explications compréhensibles et de haute précision.

Une caractéristique distinctive d'Anchors est son objectif de fournir des explications explicites sous la forme de règles simples, rendant les raisons derrière les décisions du modèle accessibles et interprétables. Cela permet aux utilisateurs, même sans connaissances avancée en apprentissage, de comprendre les facteurs déterminants dans les prédictions du modèle.

L'objectif est d'expliquer le comportement d'un modèle :

$$f = X \to Y$$

où X est l'espace des caractéristiques et Y est l'espace des prédictions, pour une instance spécifique $x \in X$. La méthode Anchors utilise un type de règles "if-then" qui sont utilisées pour représenter les points d'ancrage. Ces règles sont formulées de manière à être compréhensibles et interprétables, permettant ainsi d'expliquer les décisions du modèle de manière explicite.

La Figure 3 représente graphiquement les ancres pour les instances + et - sans dévoiler explicitement les caractéristiques spécifiques utilisées par l'algorithme d'ancrage. L'idée principale est que les ancres sont des ensembles de prédicats (règles) qui, lorsqu'elles sont vraies pour une instance, garantissent que le modèle donnera la même prédiction de classe que pour l'instance d'origine. Les caractéristiques spécifiques utilisées pour définir ces ancres peuvent varier d'une instance à l'autre en fonction de la complexité du modèle et des données.

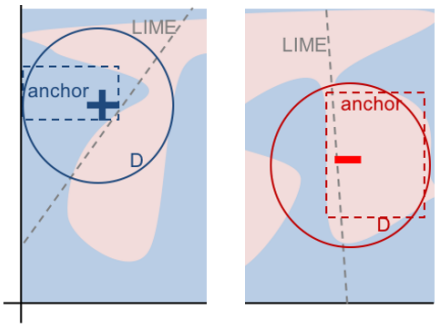


FIGURE 3 – Deux zones d'un modèle complexe, mettant en évidence des instances spécifiques

Supposons que nous ayons un modèle qui prédit l'avis d'une critique de film. Une instance (donc ici critique) donnée pourrait être :

"J'ai adoré ce film ! L'intrigue était captivante, les acteurs étaient excellents."

Un prédicat d'ancrage simple pourrait être basé sur la présence du mot "adoré" dans la critique :

"Si le mot 'adoré' est présent dans la critique, alors la prédiction sera positive." ou "if 'adoré' then 'positif'"

Dans cette situation, si le mot "adoré" est présent dans la critique, le modèle fixe sa prédiction comme positive. Ce prédicat représente une règle simple liée à une caractéristique spécifique associée à une prédiction positive. Dans des cas plus complexes, les prédicats peuvent impliquer plusieurs caractéristiques pour décrire les conditions dans lesquelles le modèle effectue certaines prédictions.

La plupart des méthodes d'interprétabilité locale agnostiques au modèle fonctionnent en perturbant l'instance x selon une "distribution de perturbation" D. Cette distribution est utilisée pour générer des variations de l'instance d'origine, permettant ainsi d'explorer localement le comportement du modèle.

Il nous faut comprendre les règles d'interprétation : Soit A une règle (ensemble de prédicats) qui agit sur une représentation interprétable de l'instance. Dans le contexte d'un modèle de traitement du langage naturel, $A(x)$ pourrait évaluer à 1 si tous les prédicats caractéristiques de A sont vérifiés pour x. L'objectif de A est d'assurer une interprétation compréhensible pour les utilisateurs, même si le modèle utilise une représentation alternative de l'entrée.

Avec cette base nous pouvons donner une définition d'une Anchor A, est une règle qui est activée ($A(x) = 1$) et qui est une condition suffisante pour la prédiction $f(x)$ avec une probabilité élevée. Formellement, A est une anchor si la dissimulation conditionnelle $\mathbb{E}_{D(z|A)}$ de z sous A vérifie $[1[f(x) = f(z)]] \geq \tau, \quad A(x) = 1$, où τ est un seuil.
Nous avons donc :

$$\mathbb{E}_{D(z|A)}\left[1[f(x) = f(z)]\right] \geq \tau, \quad A(x) = 1 \tag{2.1}$$

Anchor peut être adapté sur du texte, des images et des données tabulaires.
Ici il est sur des données tabulaires différentes. L'apprentissage supervisée est très utilisé pour classifier des données tabulaires avec des caractéristiques. Les points d'ancrage, issus des prédictions de modèles compliqués, fournissent des informations utiles et claires de manière simple.
Par exemple, dans la prédiction du revenu annuel dépassant 50 000 $, l'état matrimonial est souvent déterminant. Ces points d'ancrage éclairent certains aspects des modèles, bien que des zones non couvertes nécessitent une exploration plus approfondie. Nous pouvons voir cela sur les règles plus longues des deux autres datasets.

	If	Predict
adult	No capital gain or loss, never married	$\leq$ 50K
	Country is US, married, work hours $>$ 45	$>$ 50K
recdv	No priors, no prison violations and crime not against property	Not rearrested
	Male, black, 1 to 5 priors, not married, and crime not against property	Re-arrested
lending	FICO score $\leq$ 649	Bad Loan
	649 $\leq$ FICO score $\leq$ 699 and \$5,400 $\leq$ loan amount $\leq$ \$10,000	Good Loan

FIGURE 4 – Anchors générés pour données tabulaires

Dans l'article, les auteurs reprennent la définition du problème présentée dans l'équation 2.1 : étant donné un classificateur boîte noire f, une instance x, une distribution D et le niveau de précision souhaité τ, un ancrage A est un ensemble de prédicats de caractéristiques sur x qui atteint une précision $prec(A) \geq \tau$, où

$$prec(A) = \mathbb{E}_{D(z|A)}\left[1[f(x) = f(z)]\right] \geq \tau, \quad A(x) = 1 \tag{2.2}$$

Dans le cadre d'une distribution arbitraire D et d'un modèle boîte noire f, le calcul de cette précision est complexe. Ainsi, les auteurs proposent une définition probabiliste : les ancres visent à respecter la contrainte de précision avec une forte probabilité. En d'autres termes, au lieu de garantir une précision exacte, l'objectif est d'obtenir une précision satisfaisante avec une probabilité élevée, une approche plus réalisable dans un contexte probabiliste.

$$P(\text{prec}(A) \geq \tau) \geq 1 - \delta \tag{2.3}$$

Lorsque plusieurs ancres répondent à ce critère, on choisit celles qui couvrent une plus grande partie des données d'entrée, ce qu'on appelle la couverture (coverage). La couverture d'une ancre mesure la probabilité qu'elle s'applique à des échantillons provenant de la distribution D.

Algorithm 1 Identifying the *Best* Candidate for Greedy

function GenerateCands($\mathcal{A}, c$)
  $\mathcal{A}_r = \emptyset$
  for all $A \in \mathcal{A}$; $a_i \in x, a_i \notin A$ **do**
    if $\text{cov}(A \wedge a_i) > c$ **then** {Only high-coverage}
      $\mathcal{A}_r \leftarrow \mathcal{A}_r \cup (A \wedge a_i)$ {Add as potential anchor}
  return $\mathcal{A}_r$ {Candidate anchors for next round}

function BestCand($\mathcal{A}, \mathcal{D}, \epsilon, \delta$)
  initialize prec, prec_{ub}, prec_{lb} estimates $\forall A \in \mathcal{A}$
  $A \leftarrow \arg\max_A \text{prec}(A)$
  $A' \leftarrow \arg\max_{A' \neq A} \text{prec}_{ub}(A', \delta)$ {δ implicit below}
  while $\text{prec}_{ub}(A') - \text{prec}_{lb}(A) > \epsilon$ **do**
    sample $z \sim \mathcal{D}(z|A), z' \sim \mathcal{D}(z'|A')$ {Sample more}
    update prec, prec_{ub}, prec_{lb} for A and A'
    $A \leftarrow \arg\max_A \text{prec}(A)$
    $A' \leftarrow \arg\max_{A' \neq A} \text{prec}_{ub}(A')$
  return A

Algorithm 2 Outline of the Beam Search

function BeamSearch($f, x, \mathcal{D}, \tau$)
  hyperparameters B, ϵ, δ
  $A^* \leftarrow$ **null**, $\mathcal{A}_0 \leftarrow \emptyset$ {Set of candidate rules}
  loop
    $\mathcal{A}_t \leftarrow \text{GenerateCands}(\mathcal{A}_{t-1}, \text{cov}(A^*))$
    $\mathcal{A}_t \leftarrow \text{B-BestCand}(\mathcal{A}_t, \mathcal{D}, B, \delta, \epsilon)$ {LUCB}
    if $\mathcal{A}_t = \emptyset$ **then break loop**
    for all $A \in \mathcal{A}_t$ s.t. $\text{prec}_{lb}(A, \delta) > \tau$ **do**
      if $\text{cov}(A) > \text{cov}(A^*)$ **then** $A^* \leftarrow A$
  return A^*

Anchor est défini par deux algorithmes. L'algorithme 1 est conçu pour extraire des points d'ancrage expliquant les prédictions des modèles complexes. Voici une synthèse de son fonctionnement :

Le processus commence par initialiser un ensemble d'ancres A avec une règle vide applicable à toutes les instances. Ensuite, il procède à une recherche itérative de candidats ancrés en étendant l'ensemble A avec des prédicats dont la valeur est supérieure à c. Ces extensions sont sélectionnées en fonction de leur capacité à améliorer la couverture des ancres, garantissant ainsi des explications potentiellement qui donne plus d'information grâce à la fonction *GenerateCands*.

La sélection de la meilleure ancre candidate est réalisée par la fonction *BestCand*. Cette fonction prend en compte à la fois la précision supérieure estimée et la couverture pour choisir la candidate qui fournit le plus d'informations.

Le processus de recherche est orchestré par l'algorithme 2 avec la fonction *BeamSearch*, qui coordonne la génération de candidates *GenerateCands* et la sélection de la meilleure candidate *BestCand*. La recherche se poursuit jusqu'à ce qu'une ancre candidate atteigne le seuil de précision défini avec τ. L'ancre candidate finale A^* est alors renvoyée.

La méthode *BeamSearch* pour la construction d'ancres étend l'approche gloutonne afin de surmonter ses limitations. Alors que la méthode gloutonne est optimale pour des ancres courtes, elle ne peut maintenir qu'une règle à la fois et ne prend pas directement en compte la couverture des ancres. *BeamSearch* remédie à ces limitations en maintenant un ensemble de règles candidates tout en guidant la recherche vers l'ancre présentant la plus grande couverture. L'algorithme utilise une adaptation de l'algorithme *LUCB* pour sélectionner les meilleures règles candidates, fournissant une approximation précise avec une probabilité élevée.
LUCB est un algorithme qui aide à choisir les meilleures règles candidates en estimant leur performance tout en tenant compte de l'incertitude associée.
Structuré de manière similaire à l'approche gloutonne, mais avec un ensemble de B candidats actuels, l'algorithme 2 sélectionne l'ancre présentant la plus grande couverture parmi les multiples rencontrées. Cela optimise directement la fonction objectif et permet un élagage efficace de l'espace de recherche. Ainsi, *BeamSearch* a davantage de chances de retourner une ancre avec une couverture plus élevée que l'approche gloutonne, justifiant son utilisation dans toutes les instances et expériences.

2.3.2 LORE

La méthode LORE (Local Rule-Based Explanations) proposé en 2018 dans l'article : Local Rule-Based Explanations of Black Box Decision Systems[1], propose une nouvelle méthode agnostique d'explication locale de modèle. Cette méthode peut être utilisée sur n'importe quel ensemble de données et modèles. Le fonctionnement de cette méthode est le suivant. LORE commence par ajuster un modèle de prédiction interprétable localement sur un voisinage d'un individu de l'ensemble de données généré par un algorithme génétique. Ensuite, il crée un modèle de prédiction interprétable local sous la forme d'un arbre de décision et en dérive une explication significative comprenant : une règle de décision, qui explique les raisons de la décision, et un ensemble de règles contrefactuelles, suggérant les modifications dans les caractéristiques de l'instance qui conduisent à un résultat différent.

Le résultat retourné par la méthode est une paire d'objet définie sous cette forme :

$$e = \langle r = p \rightarrow y, \Phi \rangle$$

Le premier composant $r = p \rightarrow y$ est la règle de décision décrivant la décision du modèle. Le second composant Φ est un ensemble de règles contrefactuelles. Ces règles représentent les changements à effectuer dans les valeurs des caractéristiques de l'individu pour changer la décision du modèle.

Pour cet exemple de demande de prêt : $x = (age = 22), (job = none), (amount = 10k), (car = no)$, on obtient le résultat suivant :

$$e = \langle r = age \leq 25, job = none, amount > 5k \rightarrow deny, \phi = \{(\{age > 25, amount \leq 5k\} \rightarrow grant), (\{job = clerk, car = yes\} \rightarrow grant)\}\rangle$$

La décision est due à l'âge inférieur à 25, l'absence d'un travail et un montant supérieur à 5k. Pour que cet individu obtienne le prêt, il faut que son âge soit supérieur à 25 et un montant inférieur ou qu'il ait un travail et possède une voiture.

Algorithm 1 LORE(x,b)

Input : x - instance to explain, b - black box, N - # of neighbors
Output : e - explanation of x

$G \leftarrow 10$; $pc \leftarrow 0.5$; $pm \leftarrow 0.2$;
$Z_{=} \leftarrow GeneticNeigh(x, fitness^{x}_{=}, b, N/2, G, pc, pm)$;
$Z_{\neq} \leftarrow GeneticNeigh(x, fitness^{x}_{\neq}, b, N/2, G, pc, pm)$;
$Z \leftarrow Z_{=} \cup Z_{\neq}$;
$c \leftarrow BuildTree(Z)$;
$r = (p \rightarrow y) \leftarrow ExtractRule(c, x)$;
$\Phi \leftarrow ExtractCounterfactuals(c, r, x)$;
return $e = \langle r, \Phi \rangle$;

Algorithm 2 GeneticNeigh(x, fitness, b, N, G, pc, pm)

Input : x - instance to explain, b - black box, fitness - fitness function, N - population size, G - # of generations, pc - crossover probability, pm - mutation probability
Output : Z - neighbors of x

1: $P_0 \leftarrow \{x | \forall 1...N\}$;$i \leftarrow 0$;
2: $evaluate(P_0, fitness, b)$;
3: **while** i < G **do**
4: $P_{i+1} \leftarrow select(P_i)$;
5: $P'_{i+1} \leftarrow crossover(P_{i+1}, pc)$;
6: $P"_{i+1} \leftarrow mutate(P'_{i+1}, pm)$;
7: $evaluate(P"_{i+1}, fitness, b)$;
8: $P"_{i+1} \leftarrow P'_{i+1}$; $i \leftarrow i + 1$;
9: **end while**
10: $Z \leftarrow P_i$; **return** Z ;

On commence par identifier un ensemble d'instances Z, avec des caractéristiques proches de celles de x, capable de reproduire le comportement décisionnel local de la boîte noire b. Puisque l'objectif est d'apprendre un modèle de prédiction, le voisinage doit être suffisamment flexible pour inclure des instances avec des valeurs de décision différentes. C'est-à-dire que $Z = Z_{=} \cup Z_{\neq}$, où les instances $z \in Z_{=}$ sont telles que $b(z) = b(x)$, et les instances $z \in Z_{\neq}$, sont telles que $b(z) \neq b(x)$. Pour générer l'ensemble d'instances Z, l'algorithme 2 est utilisé.

L'algorithme génétique commence par initialiser une population P_0 avec N copies de l'instance x à expliquer. Ensuite, il entre dans la boucle d'évolution qui commence par la sélection de la population P_{i+1} ayant le score de fitness le plus élevé. L'objectif de l'algorithme étant de maximiser les fonctions de fitness suivantes :

$$fitness^x_=(z) = I_{b(x)=b(z)} + (1 - d(x, z)) - I_{x=z}$$
$$fitness^x_{\neq}(z) = I_{b(x)\neq b(z)} + (1 - d(x, z)) - I_{x=z}$$

où $I_{true} = 1$ et $I_{false} = 0$. La fonction de distance prend en compte la présence de caractéristiques catégoriques et continues. La fonction utilisé :

$$d(x, z) = \frac{h}{m} \cdot SimpleMatch(x, z) + \frac{m-h}{m} \cdot NormEuclid(x, z)$$

où h est le nombre de caractéristiques catégorique et $m - h$ le nombre de continue. Ensuite, une opération de crossover est appliqué sur une proportion de P_{i+1} selon la probabilité pc, les individus résultants et intacts sont placés dans P'_{i+1}. C'est un two-point crossover qui est utilisé. Il sélectionne deux parents et deux caractéristiques de au hasard, puis échange les valeurs des caractéristiques des parents.

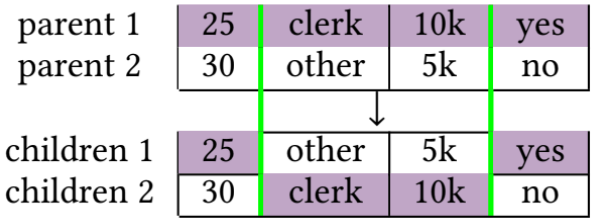


FIGURE 5 – Crossover

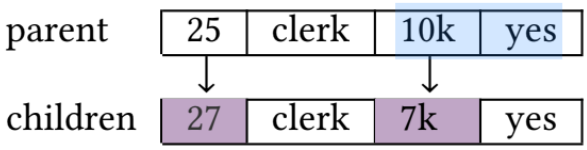


FIGURE 6 – Mutation

Ensuite, une proportion de P'_{i+1}, déterminée par la probabilité pm, est mutée et placée dans $P"_{i+1}$. Les individus non mutés sont également ajoutés à $P"_{i+1}$. La mutation consiste à remplacer les valeurs des caractéristiques au hasard selon la distribution empirique d'une caractéristique. Les individus dans $P"_{i+1}$ sont évalués selon la fonction de fitness, et la boucle d'évolution continue jusqu'à ce que G générations soient complétées. Enfin, les meilleurs individus selon la fonction de fitness sont renvoyés. L'algorithme revient à maximiser la fonction de fitness sur une population. Il est exécuté deux fois, une fois en utilisant la fonction de fitness $fitness^x_=$ pour dériver les instances du voisinage $Z_=$ avec la même décision que x, et une fois en utilisant $fitness^x_{\neq}$ pour dériver les instances du voisinage Z, avec une décision différente de x.

Après avoir obtenu le voisinage Z de x, la deuxième étape (Algorithme 1) consiste à construire un modèle interprétable c entraîné sur les instances $z \in Z$ classé selon la décision de la boîte noire b(z). LORE considère des classificateurs d'arbres de décision pour les raisons suivantes : les règles de décision peuvent naturellement être déduites d'un chemin de la racine à une feuille dans un arbre de décision ; et les règles contrefactuelles peuvent être extraites par un raisonnement symbolique sur un arbre de décision.

Pour calculer les règles contrefactuelles, il faut d'abord récupérer tous les chemins dans l'arbre de décision menant à une décision contraire que celle prédit pour x. LORE ne garde que les règles contrefactuelles contenant le minimum de condition non satisfaite par x.

2.3.3 Comparaison LORE/Anchors

Nous allons maintenant comparer les méthodes LORE et Anchors. Pour commencer, LORE peut générer des règles contrefactuelles qui montrent comment une instance pourrait être modifiée pour changer la prédiction du modèle. Anchors ne génère pas explicitement de contre-exemples, il fournit des règles qui indiquent les conditions sous lesquelles la prédiction du modèle est stable.

Ensuite, LORE possède un "coverage" plus élevé que Anchors qui donne souvent des règles spécifiques à des sous-régions de l'espace des caractéristiques. LORE tend à plus généraliser son explication et à fournir donc des règles qui s'appliquent à un ensemble plus large d'instances.
De plus, grâce aux arbres de décision LORE peut gérer les caractéristiques continues sans nécessiter de discrétisation contrairement à Anchors qui requiert la discrétisation des caractéristiques continues, ce qui peut introduire une perte de précision.

Il est intéressant de mettre en avant que LORE est généralement plus stable face aux perturbations des données d'entrée, ce qui signifie que de petites variations dans les données ne devraient pas entraîner des changements drastiques dans les explications. Anchors peut être sensible aux perturbations des données d'entrée, et de petits changements peuvent influencer les règles générées.
Pour finir, la méthode Anchors peut être utilisée sur des données différentes (tabulaires, textes, images) contrairement à LORE qui peut seulement être utilisé sur des données tabulaires.

LORE	Anchors
Précision un peu plus faible	Précision plus haute
Séparation des caractéristiques continues	Discrétisation des caractéristiques continues
Règles contrefactuelles	Pas de règles contrefactuelles explicites
Coverage plus élevé	Coverage plus faible
Plus stable	Moins stable
Seulement sur des données tabulaires	Possible aussi sur des images et du texte

Table 1 – Récapitulatif de la comparaison LORE/ANCHORS

2.4 Présentation des métriques d'évaluation

Afin de pouvoir évaluer nos méthodes et mettre en avant les avantages et inconvénients de chacune d'entre elles, nous nous sommes basés sur l'article "A Quantitative Evaluation of Global, Rule-Based Explanations of Post-Hoc, Model Agnostic Methods"[4]. Pour cela, cet article propose 8 différentes métriques que nous allons vous présenter :

— **Completeness** : elle mesure la proportion d'instances dans l'ensemble de données qui sont couvertes par les règles extraites. Une valeur élevée de cette métrique indique que les règles extraites sont capables de fournir des explications pour la plupart des instances dans l'ensemble de données.

$$\frac{c}{N} \quad avec \quad c = nombre\ d'instances\ couverte \quad et \quad N = nombre\ d'instances\ total$$

— **Correctness** : elle mesure la proportion d'instances dans l'ensemble de données qui sont correctement classées par les règles extraites. Une valeur élevée de cette métrique indique que les règles extraites sont capables de fournir des explications précises pour la plupart des instances dans l'ensemble de données.

$$\frac{r}{N} \quad avec \quad r = nombre\ d'instances\ correctement\ classifier$$

— **Fidelity** : elle mesure la proportion d'instances dans l'ensemble de données pour lesquelles les prédictions du modèle et les prédictions des règles extraites sont identiques. Une valeur élevée de cette métrique indique que les règles extraites sont capables de fournir des explications fidèles aux prédictions du modèle pour la plupart des instances dans l'ensemble de données.

$$\frac{f}{N} \quad avec \quad f = nombre\ d'instances\ ou\ les\ prèdiction\ du\ modèle\ et\ des\ règle\ sont\ identique$$

— **Robustness** : elle mesure la capacité des règles extraites à résister aux perturbations mineures des instances dans l'ensemble de données. Une valeur élevée de cette métrique indique que les règles extraites sont capables de fournir des explications stables pour les instances dans l'ensemble de données, même en présence de bruit ou de perturbations mineures.

$$\frac{\sum_{n=1}^{N} f(x_n) - f(x_n + \delta)}{N} \quad avec \quad \delta = la\ perturbation\ et\ (f(x_n)) = prèdiction\ du\ model$$

— **Number of rules** : elle mesure le nombre total de règles extraites. Une valeur faible de cette métrique indique que l'ensemble des règles extraites est plus simples et faciles à comprendre.

$$|R| = nombre\ de\ règles$$

— **Average rule length** : elle mesure la longueur moyenne des règles extraites. Une valeur faible de cette métrique indique que les règles extraites sont courtes et faciles à comprendre.

$$\frac{\sum_{i=1}^{R} a_i}{|R|} \quad avec \quad a_i = nombre\ d'antècèdents\ de\ la\ règle\ i$$

— **Fraction of classes** : elle mesure la capacité des règles extraites à couvrir toutes les classes de l'ensemble de données. Une valeur élevée de cette métrique indique que les règles extraites sont capables de fournir des explications pour toutes les classes de l'ensemble de données.

$$\frac{1}{|C|} \sum_{c' <= C} 1(\exists r = (s, c) \in R | c = c') \quad avec \quad R = liste\ des\ règle \quad |C| = nombre\ de\ classe$$

— **Fraction overlap** : mesure le degré de chevauchement entre les règles extraites dans un ensemble de règles. Une valeur faible de cette métrique indique que les règles extraites sont cohérentes et non redondantes.

$$\frac{2}{|R|(|R|-1)} \sum_{r_i, r_j, i <= j} \frac{overlap(r_i, r_j)}{N} avec \quad overlap\ renvoie\ le\ nombre\ 'antècèdents\ sur\ la\ meme\ classe\ et$$

$$dont\ les\ condition\ sont\ remplis\ sur\ une\ instance$$

3 Implémentions et utilisation des méthodes

3.1 Modèle MLP

Afin de tirer parti des différentes méthodes présentées précédemment, nous avons besoin d'un modèle de prédiction entraîné sur divers jeux de données. Le modèle que nous avons utilisé est un réseau de neurones MLP (Fully Connected).

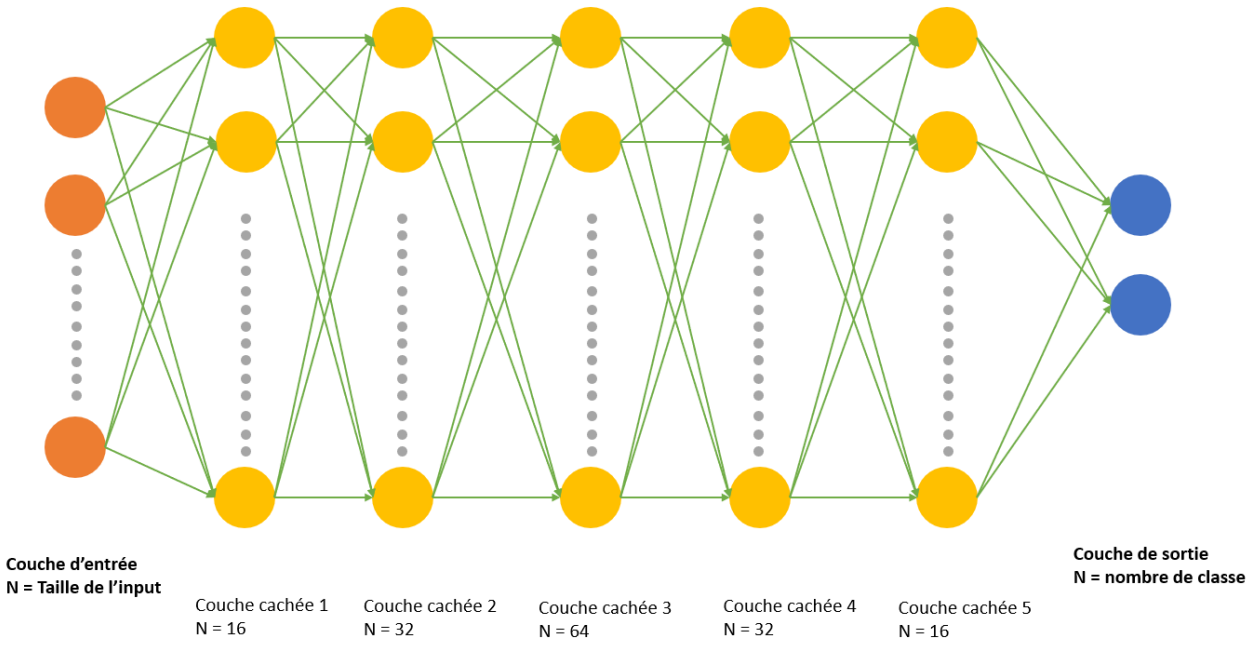


FIGURE 7 – Réseau de neurone MLP

Notre réseau est constitué de cinq couches cachées avec la configuration suivante : 16, 32, 64, 32 et 16 neurones. Nous utilisons une fonction d'activation linéaire sur toutes les couches cachées. La taille de la couche d'entrée correspond à la dimension des données d'entrée, tandis que la taille de la couche de sortie correspond au nombre de classes à prédire. Par exemple, pour le jeu de données sur le diabète, la taille de l'entrée est de 8 et celle de la sortie est de 2.

Pour l'entraînement, nous avons mis en place une procédure de "GridSearch" pour optimiser les hyper-paramètres tels que le learning rate, le batch size, et le dropout sur 250 epochs. Nos données ont été réparties en trois ensembles distincts : jeu d'entraînement, de validation et de test. Le modèle a été entraîné sur le jeu d'entraînement, et à chaque epoch, nous avons évalué sa performance sur l'ensemble de validation. Pour évaluer la précision finale de notre modèle, nous l'avons testé sur l'ensemble de test.

Nous avons entraîné notre modèle sur quatre jeux de données différents : Diabète, Breast-Cancer, Heart et Covid-19.

	Nombre d'instance	Nombre de caractéristique
Diabètes	768	8
Breast-Cancer	569	30
Heart	303	13
Covid-19	20 000	21

Table 2 – Caractéristiques des Datasets

	Diabètes	Breast-cancer	Heart	Covid-19
Précision	78.81 %	95.65 %	83.61 %	74.25 %

Table 3 – Précisions obtenus sur l'ensemble de test pour les jeux de données : Diabètes, Breast cancer et Heart

Afin de valider les performances de notre modèle, nous avons comparé nos résultats avec ceux d'un modèle XGBoost. C'est un type de modèle réputé pour avoir de bonne performance sur des données tabulaire. Pour que la comparaison soit correcte, nous effectuons les mêmes transformations sur les données et utilisons le même ensemble d'entraînement et de test que pour notre modèle.

	Diabètes	Breast-cancer	Heart	Covid-19
Notre modèle	78.81 %	95.65 %	83.61 %	74.25 %
XGBoost	83.12 %	96.49 %	80.33 %	72.80 %

Table 4 – Comparaisons de la précision entre notre modèle et XGBoost

Nous pouvons observer que notre modèle fait aussi bien pour le jeu de données breast-cancer. De plus, il est plus précis sur ses prédictions sur les jeux de données Heart et Covid-19. Malheureusement, il est bien moins performant sur l'ensemble de données du Diabètes.

3.2 Résultats

Pour les méthodes locales (Anchors et LORE), nous allons utiliser le même individu à expliquer.

$$x = \{Pregnancie = 4, Glucose = 76, BloodPressure = 62, SkinThickness = 0,$$
$$Insulin = 0, BMI = 34, DiabetesPedigreeFunction = 0.391, Age = 25\}$$

3.2.1 LORE

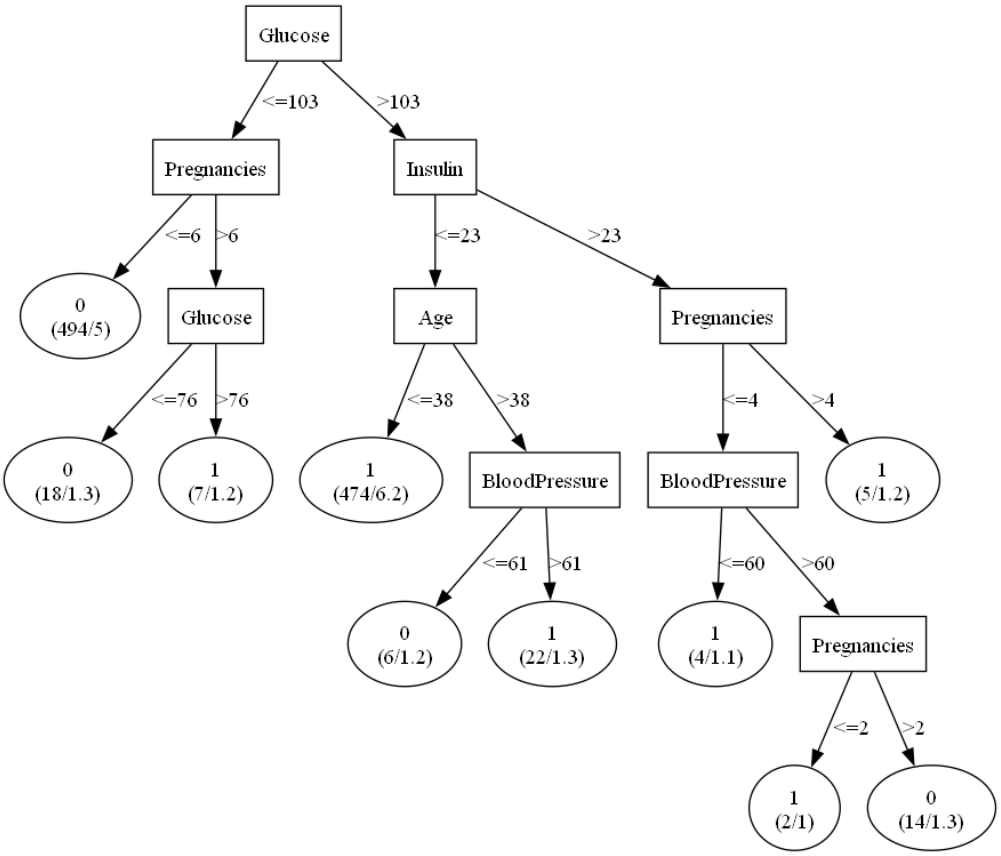


FIGURE 8 – Arbre de décision obtenue par la méthode LORE sur le "neighborhood" de x

La règle que nous obtenons pour expliquer la décision de notre modèle sur l'individu x :

$$r = \{Glucose \leq 103, Pregnancies \leq 6\} \rightarrow No\_Diabete$$

Les règles contrefactuelles expliquant les caractéristiques à changer pour modifier la prédiction de notre modèle sur l'individu x :

$$\phi = \{(\{76 < Glucose \leq 103, Pregnancies > 6\} \rightarrow Diabetes), (\{Glucose > 103, Insulin \leq 23, Age \leq 38\} \rightarrow$$
$$Diabetes), (\{Glucose > 103, Insulin > 23, Pregnancies > 4\} \rightarrow Diabetes)\}\rangle$$

3.2.2 Anchors

La règle que Anchors nous donne pour expliquer la décision de notre modèle pour l'individu x :

$$Glucose \leq 117.5 \text{ AND } Age \leq 29$$

4 Évaluation des méthodes

4.1 Résultats

Une fois nos différentes méthodes d'extraction de règles implémentées, nous avons pu les utiliser afin de générer les règles sur chacun de nos datasets. Pour GlocalX, nous avons utilisé les ensembles de règles générés par Anchor et Lore afin de créer les ensembles de règles GlocalX-Anchors et GlocalX-lore. Nous avons également utilisé GlocalX avec une fusion des ensembles d'Anchors et de Lore afin de générer l'ensemble de règles GlocalX-anchor-lore. Ensuite, nous avons calculé les métriques de l'article "A Quantitative Evaluation of Global, Rule-Based Explanations of Post-Hoc, Model Agnostic Methods"[4] présentées précédemment.

Table 5 – Métrique d'Anchor

	Covid-19	breast-cancer	diabetes	heart
Completeness	0.2150	0.3215	0.1382	0.1794
Correctness	0.5	0.3796	0.3398	0.4785
Fidelity	0.7705	0.3567	0.3398	0.4983
Robustness	0.9999	0.9894	0.9817	0.9141
Number of rules	140	114	154	61
Average rule length	2.8642	1.1578	2.1623	2.1803
Fraction of classes	0.3809	0.3333	1	0.8461
Fraction overlap	0.6718	0.1867	0.1708	0.0617

Table 6 – Métrique de lore

	Covid-19	breast-cancer	diabetes	heart
Completeness	0.1599	0.2105	0.2258	0.1873
Correctness	0.4999	0.5975	0.6002	0.5049
Fidelity	0.7552	0.6203	0.6028	0.5181
Robustness	0.9999	0.9859	0.9791	0.9504
Number of rules	140	114	154	61
Average rule length	3.7214	2.6140	3.6493	3.7704
Fraction of classes	1	0.7333	1	1
Fraction overlap	0.3066	0.0460	0.0786	0.0831

Table 7 – Métrique de GlocalX-anchor

	Covid-19	breast-cancer	diabetes	heart
Completeness	1	1	0.8932	1
Correctness	0.4999	0.6133	0.4960	0.4587
Fidelity	0.2293	0.6362	0.5221	0.4389
Robustness	0.9999	0.9894	0.8463	0.9702
Number of rules	9	8	23	6
Average rule length	2.5555	2	3.8260	2.1666
Fraction of classes	0.3333	0.2666	1	0.5384
Fraction overlap	0.6382	0.0439	0.0061	0.1355

Table 8 – Métrique de GlocalX-lore

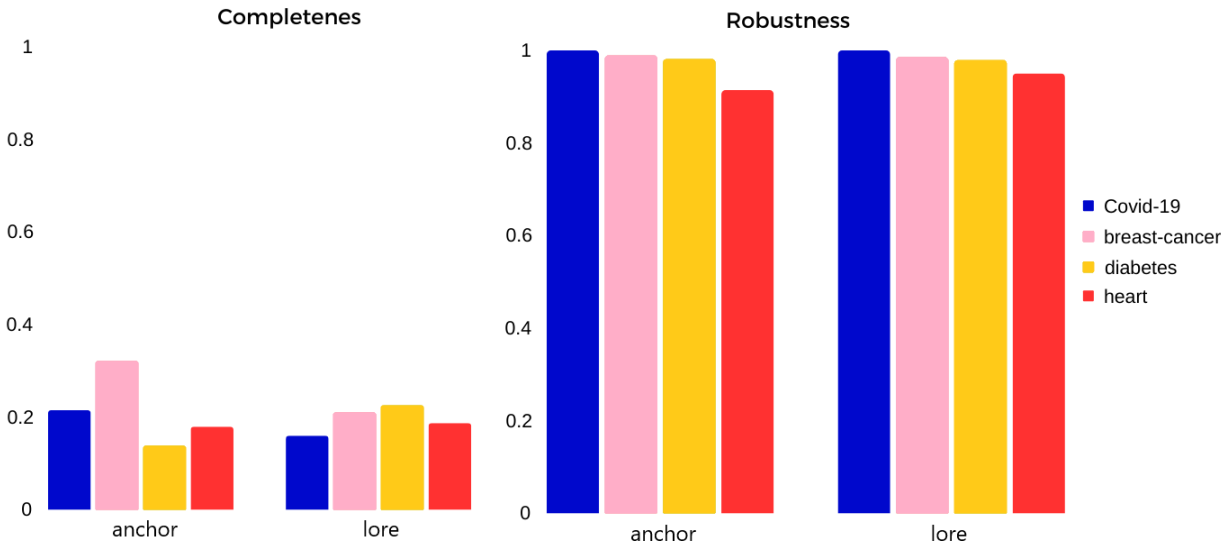
	Covid-19	breast-cancer	diabetes	heart
Completeness	1	0.9982	1	1
Correctness	0.4999	0.6115	0.6145	0.4488
Fidelity	0.2293	0.6344	0.6276	0.4290
Robustness	0.9999	09876	0.9830	0.8481
Number of rules	29	14	26	10
Average rule length	5	3.6428	4.0769	3.2
Fraction of classes	0.9523	0.3	1	0.7692
Fraction overlap	0.1646	0.0246	0.0121	0.0885

Table 9 – Métrique de GlocalX-anchor-lore

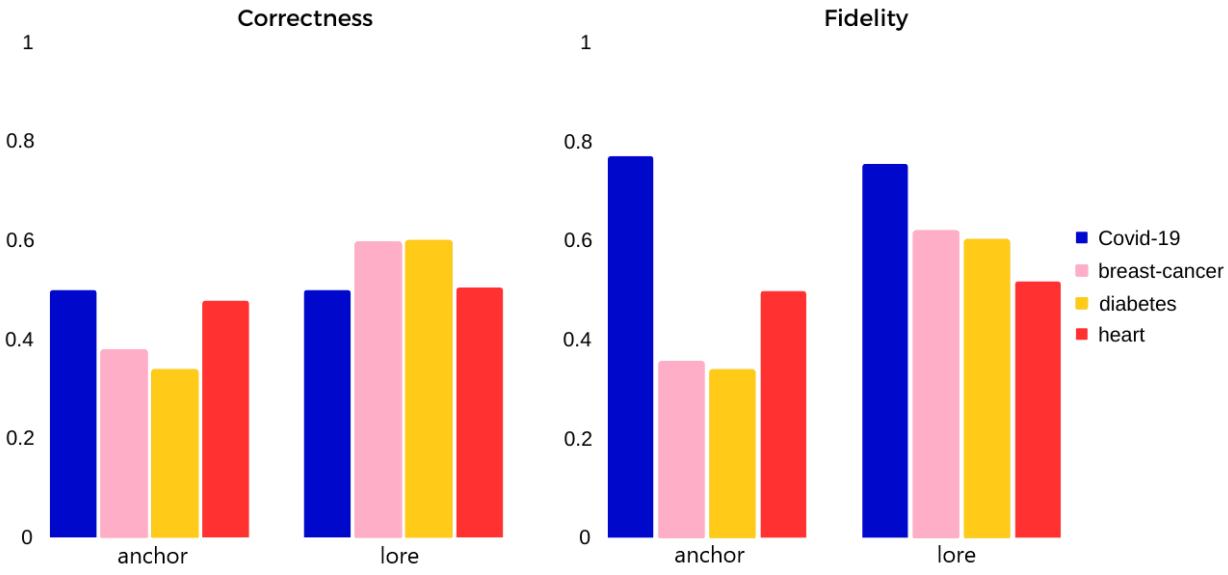
	Covid-19	breast-cancer	diabetes	heart
Completeness	1	0.7838	1	0.9933
Correctness	0.5046	0.4393	0.6497	0.4554
Fidelity	0.3055	0.4551	0.6835	0.4323
Robustness	0.9999	0.7662	0.9973	0.8745
Number of rules	66	16	40	57
Average rule length	4.4696	4.1875	4.3750	3.3157
Fraction of classes	1	0.5	1	0.8461
Fraction overlap	0.0423	0.0114	0.0067	0.0110

Maintenant que toutes les métriques sont calculées pour chaque extracteur, nous allons pouvoir comparer les méthodes entre elles. Nous allons faire cette comparaison en deux parties : la première comparera les méthodes locales (anchor et lore) entre elles, et la deuxième comparera les méthodes locales entre elles (GlocalX-anchor, GlocalX-lore, GlocalX-anchor-lore). Afin de rendre ces comparaisons plus explicites, nous utiliserons des graphiques.

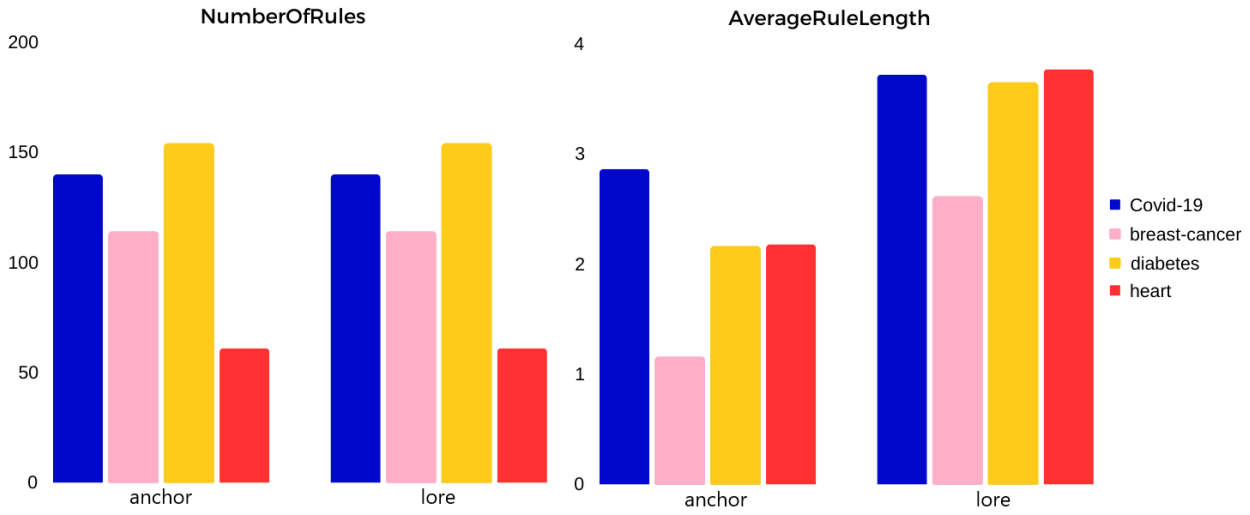
4.2 Comparaison des règles local



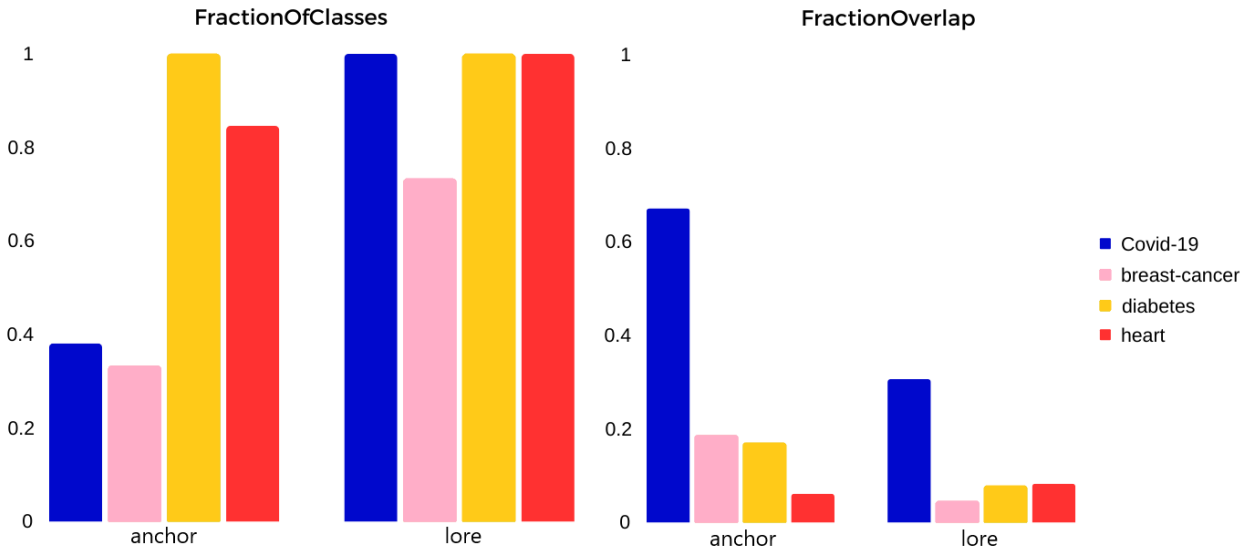
Comme Anchor et Lore sont des méthodes locales, elles disposent d'une faible Completenes. On peut également constater qu'elles ont une forte robustesse.



On peut remarquer que de manière générale, Lore a une meilleure Correctness et une meilleure Fidelity que Anchor, bien qu'Anchor ait un résultats légèrement meilleur sur le dataset "Covid-19". Cela montre que pour ces métriques, un extracteur de règles peut être plus ou moins performant en fonction du dataset.



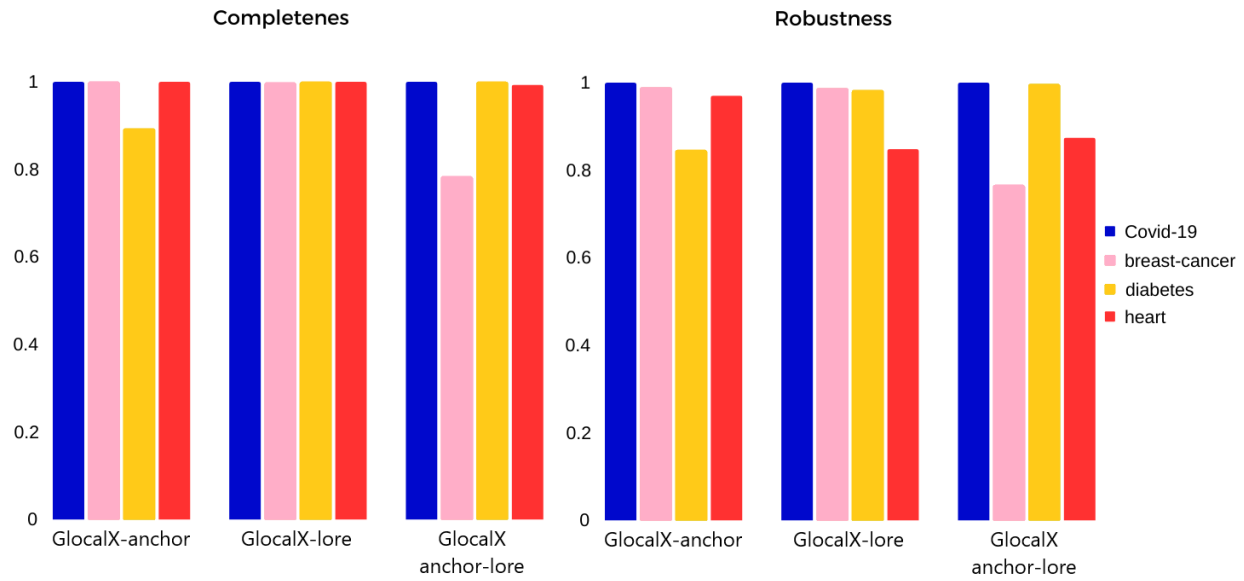
On peut constater qu'Anchor et Lore vont générer exactement le même nombre de règles. Cependant, Lore a tendance à produire des règles lus longues et donc plus complexes que celles d'Anchor.



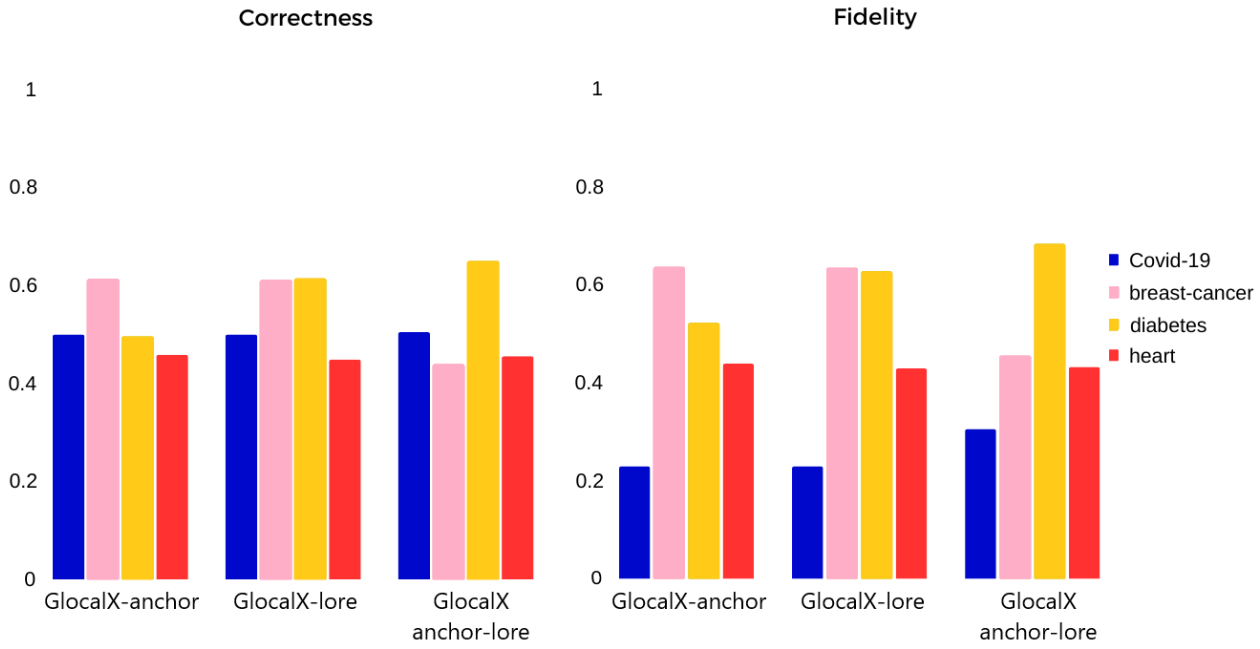
On peut voir que Lore représente mieux les différentes classes des datasets qu'Anchor, ce qui n'est pas surprenant étant donné que ce dernier avait des règles plus grandes. Cependant, malgré la taille plus élevée des règles de Lore, il a un FractionOverlap inférieur à celui d'Anchor.

Grâce à ces comparaisons, on peut se rendre compte que sur la plupart des métriques, Lore a globalement de meilleurs résultats qu'Anchor. Cependant, ce dernier a tendance à générer des règles plus longues et donc plus complexes. Les deux ont donc leurs propres avantages et peuvent avoir chacun leur utilité.

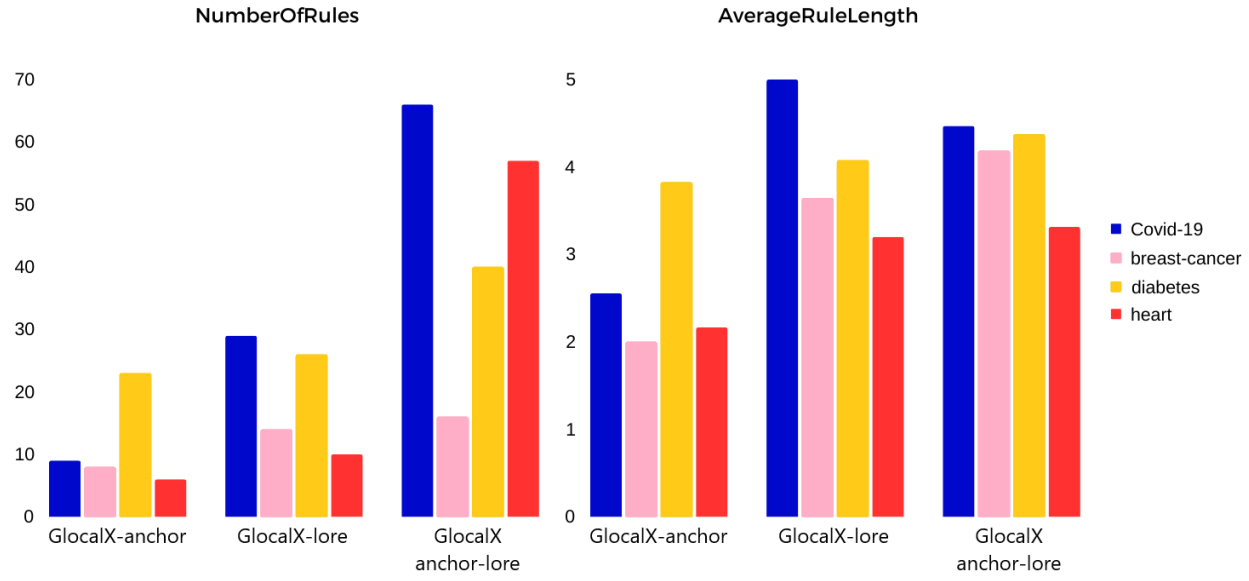
4.3 Comparaison des règles global



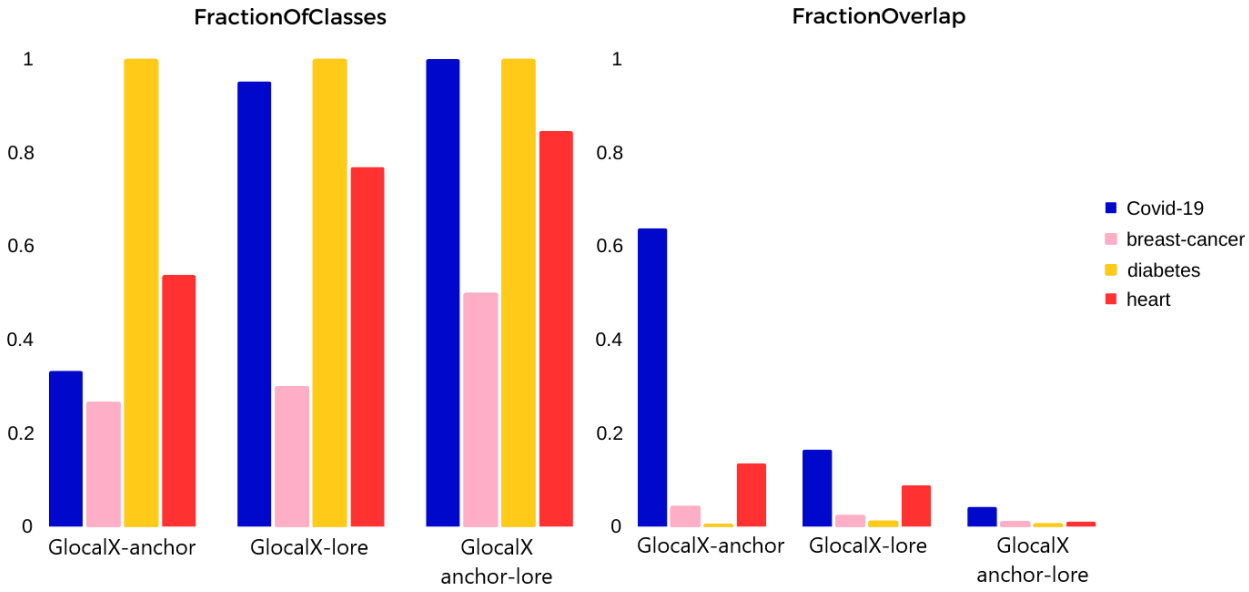
On constate que la complétude des ensembles de règles générées par GlocalX est assez élevée, même si elle est plus basse pour certains datasets, de même que la robustesse.



Les correctness de ces ensembles sont assez similaires, même si GlocalX-anchor-lore semble avoir du mal sur "breast-cancer", de même que pour la fidelity. On peut également remarquer que les méthodes GlocalX ont toutes une fidelity assez faible sur le dataset "covid-19".



On peut remarquer que les ensembles créés par GlocalX sont bien plus petits que les ensembles initiaux. Cependant, GlocalX-anchor-lore possède des ensembles plus grands que ceux générés par GlocalX-anchor et GlocalX-lore. Par contre, ils sont de taille similaire à ceux de GlocalX-lore.



La Fraction of classes des ensembles GlocalX semble vraiment varier en fonction du dataset. On peut le voir notamment avec le dataset "diabetes" qui a toujours une Fraction of classes de 1, alors que le dataset "breast cancer" dispose d'une Fraction of classes assez faible. On peut également remarquer que GlocalX-anchor-lore dispose des plus hauts résultats pour cette métrique pour chaque dataset. En ce qui concerne la fraction d'overlap, elle semble généralement assez faible, particulièrement pour GlocalX-anchor-lore. Cependant, on peut également remarquer que tous les ensembles disposent de valeurs assez élevées pour le dataset "Covid-19" par rapport aux autres datasets.

Grâce à ces comparaisons, on peut se rendre compte que, comme pour Anchor et Lore, GlocalX-lore génère des règles plus longues que pour GlocalX-anchor. Cependant, il n'a des résultats que très peu supérieurs à ceux de GlocalX-anchor. On peut donc en déduire que sur les datasets et le modèle que l'on a utilisés, GlocalX semble mieux fonctionner en utilisant les règles générées par Anchor que celles générées par Lore. De plus, on peut remarquer que GlocalX-anchor-lore semble renvoyer des résultats assez similaires à ceux des deux autres. Cependant, il semble avoir tendance à générer plus de règles et d'une taille similaire à GlocalX-lore. Néanmoins, il a également une Fraction Overlap plus petite que les autres.

5 Conclusion

En conclusion, notre projet a permis d'explorer et de comparer différentes méthodes globales et locales pour expliquer les modèles de prédiction, en particulier ceux basés sur les réseaux de neurones. Nous avons étudié la méthode GLocalX, qui est une méthode globale, et les méthodes LORE et Anchors qui sont des méthodes locales. Nos expériences sur différents jeux de données ont souligné l'importance de choisir judicieusement parmi ces méthodes en fonction du contexte d'application et des besoins spécifiques en interprétabilité. En effet, la nature de la tâche, la complexité des données et les exigences en matière de transparence peuvent influencer le choix de la méthode la plus appropriée. De plus, nous avons remarqué qu'il pouvait être pertinent en fonction des jeux de données, de combiner les explications locales de plusieurs méthodes afin de fournir une explication globale du modèle.

6 Annexe

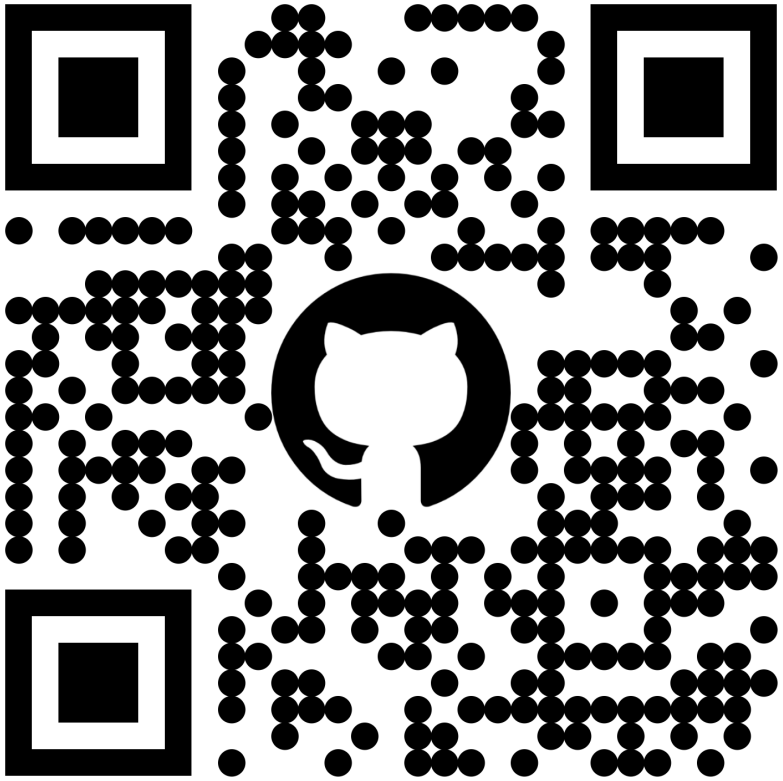


FIGURE 9 – QR-Code du projet : *https://github.com/Arm-Sl/Masterial* (Cliquable)

Références

[1] Riccardo Guidotti et al. « Local Rule-Based Explanations of Black Box Decision Systems ». In : (2018). arXiv : 1805.10820 [cs.AI]. URL : https://arxiv.org/abs/1805.10820.

[2] Marco Tulio Ribeiro, Sameer Singh et Carlos Guestrin. « Anchors: High-Precision Model-Agnostic Explanations ». In : (2018).

[3] Mattia Setzu et al. « GLocalX - From Local to Global Explanations of Black Box AI Models ». In : *Artificial Intelligence* 294 (2021), p. 103457. ISSN : 0004-3702. DOI : https://doi.org/10.1016/j.artint.2021.103457. URL : https://www.sciencedirect.com/science/article/pii/S0004370221000084.

[4] Giulia Vilone et Luca Longo. « A Quantitative Evaluation of Global, Rule-Based Explanations of Post-Hoc, Model Agnostic Methods ». In : *Frontiers in Artificial Intelligence* 4 (2021). ISSN : 2624-8212. DOI : 10.3389/frai.2021.717899. URL : https://www.frontiersin.org/articles/10.3389/frai.2021.717899.