M2 MALIA - XAI Projects

Angelo Lamure-Fontanini & Mohammed Ali El Adlouni

Fevrier 2024

Résumé

Dans ce document, on présente les différents projets effectués dans le cadre du cours sur l'explicabilité (XAI). On commencera avec les méthodes SHAP et DICE, que l'on appliquera sur un dataset de catalogue d'exoplanètes pour prédire l'habitabilité de celles-ci. On continuera avec Grad-CAM, une approche permettant d'expliquer les décisions des réseaux de neurones convolutifs en mettant en évidence les zones d'une image les plus influentes pour la classification. Ensuite, nous explorerons l'interprétation des modèles de type BERT à travers différentes méthodes d'attribution de gradient, notamment Layer Integrated Gradients (LIG) et Layer Conductance, appliquées à une tâche de Question-Answering.

Nous poursuivrons avec Layerwise Relevance Propagation (LRP), une approche qui permet d'analyser la contribution des neurones aux décisions d'un réseau en rétro-propageant la pertinence des prédictions. Enfin, nous conclurons avec une étude sur l'explicabilité des réseaux de neurones graphiques (GNNs), où nous mettrons en évidence les structures les plus déterminantes dans les décisions du modèle grâce à la méthode LRP appliquée aux graphes.

Ce document vise ainsi à fournir une comparaison de différentes approches d'explicabilité, en mettant en avant leurs forces, leurs limites et leurs applications spécifiques à différents types de données et d'architectures de modèles.

Table des matières

1 Projet 1 - SHAP

1.1 Méthode

La méthode SHAP (pour SHapley Additive exPlanations) [4] est une méthode d'explicabilité des modèles d'IA reposant sur la théorie des jeux coopératifs. Pour une prédiction donnée $f(x)$, elle assigne à chaque caractéristique z_i une valeur d'importance ϕ_i telle que :

$$f(z) = \phi_0 + \sum_{i=1}^{M} \phi_i z_i$$

où ϕ_0 est une constante représentant la valeur de base du modèle. Ces valeurs ϕ_i sont déterminées selon les valeurs de Shapley :

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f(S \cup \{i\}) - f(S) \right]$$

Avec cette approche, on peut garantir trois propriétés :

1. Exactitude au niveau locale : la somme des contributions ϕ_i doit reconstruire $f(z)$
2. Manque d'influence : si une caractéristique z_i n'affecte pas $f(z)$, $\phi_i = 0$
3. Cohérance : si f évolue de façon à ce que la contribution de z_i augmente ou reste constante pour toute entrée, alors ϕ_i ne doit pas diminuer.

Ainsi, SHAP fournit une explication additive et cohérente de l'impact de chaque caractéristique sur la prédiction du modèle.

1.2 Dataset *Habitable Worlds Catalog* pour SHAP et DiCE

Nouveau dataset : Pour SHAP (ainsi que pour DiCE juste après), on utilisera un dataset qui prend en entrée des observations d'exoplanètes [3], avec une multitude de variables, qui sont des caractéristiques de l'exoplanète (tel que l'estimation de sa masse, de son rayon, de la température de surface, etc.) ou du système exoplanétaire dans lequel elle se trouve (le flux reçu de l'étoile, etc.). La variable cible étant l'habitabilité de la planète (dans la variable `P_HABITABLE`). On a

Pré-traitement : Avant d'appliquer SHAP, on va expliquer quelques étapes de pré-traitement :

- La variable `P_HABITABLE` peut avoir trois valeurs : 0, 1 et 2. 0 signifie que l'exoplanète n'est pas habitable : ce sont nos cas "Non-Habitable". Pour 1 et 2, ce sont deux "variente d'habitabilité" : 1 pour *conservative* et 2 pour *optimistic*. Les deux sont considérés comme potentiellement habitables, mais seulement avec 2 niveaux différents de probabilité. Pour notre étude, on mettra les deux à 1, pour former nos cas "Habitable".
- La variable `P_TEMP_SURF`, qui correspond à la température de surface de l'exoplanète (en K [Kelvin]), contient énormément de valeurs manquantes (ce qui est plutôt normal, le calcul d'une température de surface a besoin de beaucoup de données). Notre dataset contenant énormément d'observations (surtout compte tenu de notre étude, qui n'est pas concentrée sur la performance), on va tout simplement garder les observations qui ont cette variable renseignée. On fait ça ici particulière car c'est une variable très importante pour la prédiction.

— On retire la variable `P_HABZONE_OPT`. On a retiré cette variable car il n'est pas dit comment elle est calculée : il y a un risque que cette variable utilise la connaissance de l'habitabilité pour donner ou non la valeur de la zone habitable optimale. Dans le doute, on retire cette variable. À noter que même si ce n'est pas le cas, ce n'est pas intéressant de connaître l'impact de cette variable dans notre problématique d'habitabilité, étant au mieux juste un calcul de la zone qui maximise la probabilité d'habitabilité.

On a au final 2441 observations, dont 70 "Habitable".

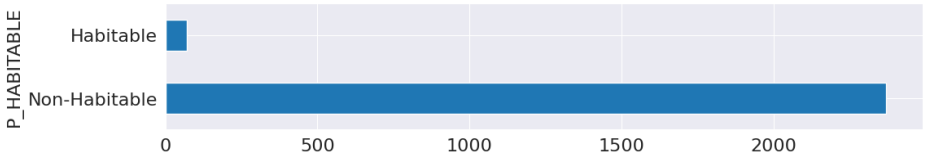

FIGURE 1 – Histogramme observation Habitable & Non-Habitable

1.3 Application de SHAP

Avant d'utiliser SHAP, il faut créer un modèle de machine learning. On utilise ici un XGBoost, puis l'entraîner sur 80% des données. On prend ensuite les 20 % de Test. On obtient la table de confusion dans la Figure 2.

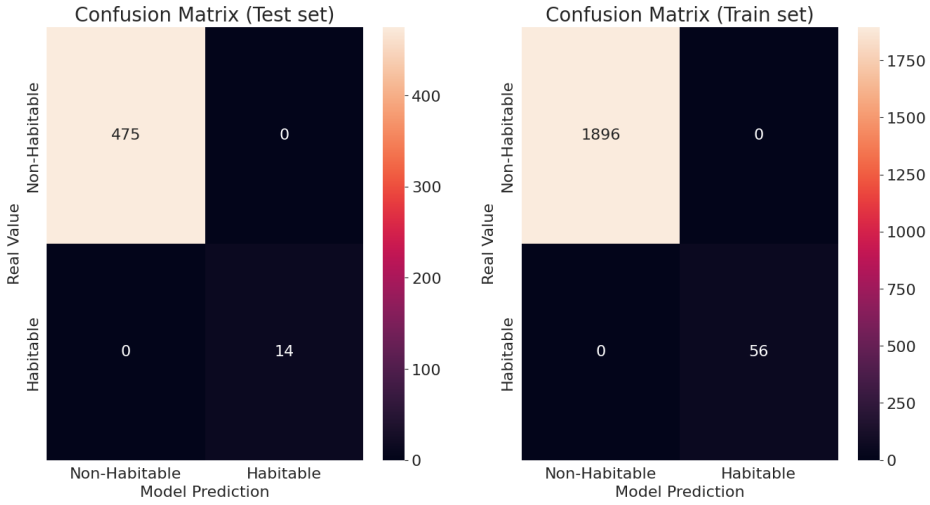

FIGURE 2 – Prediction avec notre modèle XGBoost classifier

On peut voir que l'on a la totalité des observations test (et train) qui sont bien classées. On peut maintenant utiliser SHAP pour interpréter les prédictions. On va analyser cela sous deux axes : une interprétation locale (pour des observations en particulier) et une interprétation globale.

Interprétation Local : On peut prendre une observation du test, qui est habitable, et appliquer la fonction `force_plot` du package `shap` sur notre modèle :

FIGURE 3 – Explicabilité avec SHAP d'une observation test habitable

Ici, le **7.55** est le $ln(\frac{P(Y=1)}{1-P(Y=1)})$. Plus cette valeur est grande (et supérieur à zéro), plus la prédiction "est habitable". On a en rouge les variable qui ont contribué à prédire l'observation comme Habitable, et en bleu les contribution à Non-Habitable. Par exemple, on faire les remarque suivante dans notre exemple :

- Il n'y a presque pas de contribution "bleu", il y a donc très peu de variable qui contribue à prédire Non-Habitable
- C'est la variable `P_FLUX`, le flux reçu par l'exoplanète de l'étoile du système, qui contribut le plus pour prédire Habitable.
- On a ensuite `P_ESI` (l'*Earth Similarity Index*, on reparle plus tard de cette variable) et `P_TEMP_EQUIL` (la température d'équilibre de l'exoplanète) qui contribue beaucoup à la prédiction Habitable.

On peut voir un exemple d'une prédiction Non-Habitable :

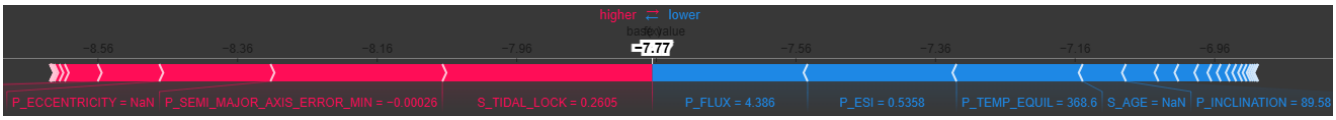

FIGURE 4 – Explicabilité avec SHAP d'une observation test Non-Habitable

On a ici bien un score négatif (-7.77), donc bien prédit Non-Habitable, avec nos trois variables qui contribuent maintenant à la prédiction Non-Habitable. Ce sont peut-être des variables qui sont les plus influentes.

Interprétation globale : On peut justement maintenant, avec la fonction `summary_plot` de `shap`, avoir une interprétation globale des variables dans l'attribution des prédiction Habitable / Non-Habitable, dans la Figure 5.

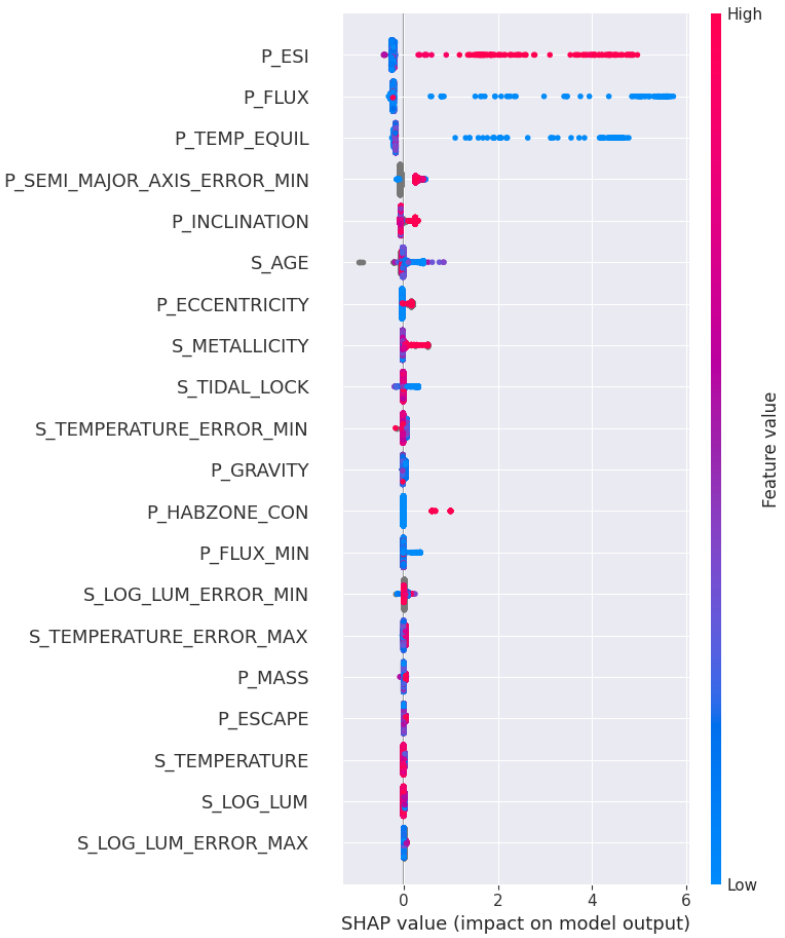

FIGURE 5 – Explicabilité globale avec SHAP

On remarque que ce sont les variables `P_ESI`, `P_FLUX` et `P_TEMP_EQUIL`, que l'on avait déjà incriminées avant, qui contribuent le plus pour prédire l'habitabilité ! Cependant, une dernière remarque que l'on peut faire, c'est sur cette variable `P_ESI`, qui est l'*Earth Similarity Index* : c'est un index qui mesure la ressemblance d'un objet céleste (ici une exoplanète) à la Terre. On remarque donc ici que ce sont les valeurs élevées de ESI qui contribuent fortement à une prédiction habitable. Il est important de noter que cet index est indépendant du critère d'habitabilité, au niveau du calcul qui est fait, mais on voit qu'il est fortement corrélé. On peut donc tout refaire sans cette variable et voir ce que l'on a.

1.4 Application additionnelle en retirant la variable *ESI*

Sans ESI, on a les mêmes prédictions qu'avec, on a 100 % de bonnes prédictions. On a dans la Figure 6 les contributions pour la même observation que dans la Figure 3.

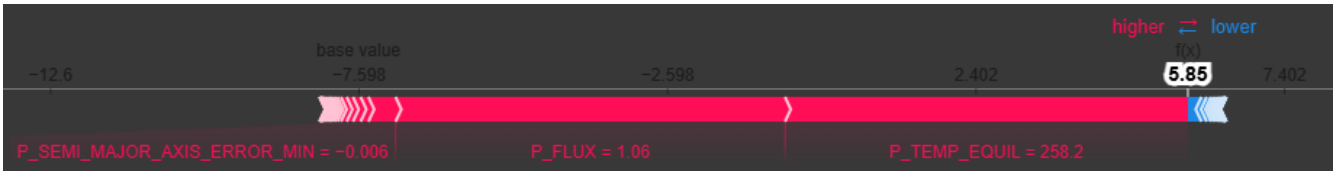

FIGURE 6 – Explicabilité avec SHAP d'une observation test habitable, sans avoir utiliser la variable *ESI*

Points communs, on n'a presque pas de contributions Non-Habitable. On a toujours nos deux variables `P_TEMP_EQUIL` et `P_FLUX` qui contribuent beaucoup à la prédiction Habitable. Cependant, on peut quand même noter que le score de prédiction (5.85) est plus faible que la précédente, à cause du manque de la variable *ESI*. On peut finir par refaire l'interprétation globale sans la variable *ESI* dans la Figure **??**.

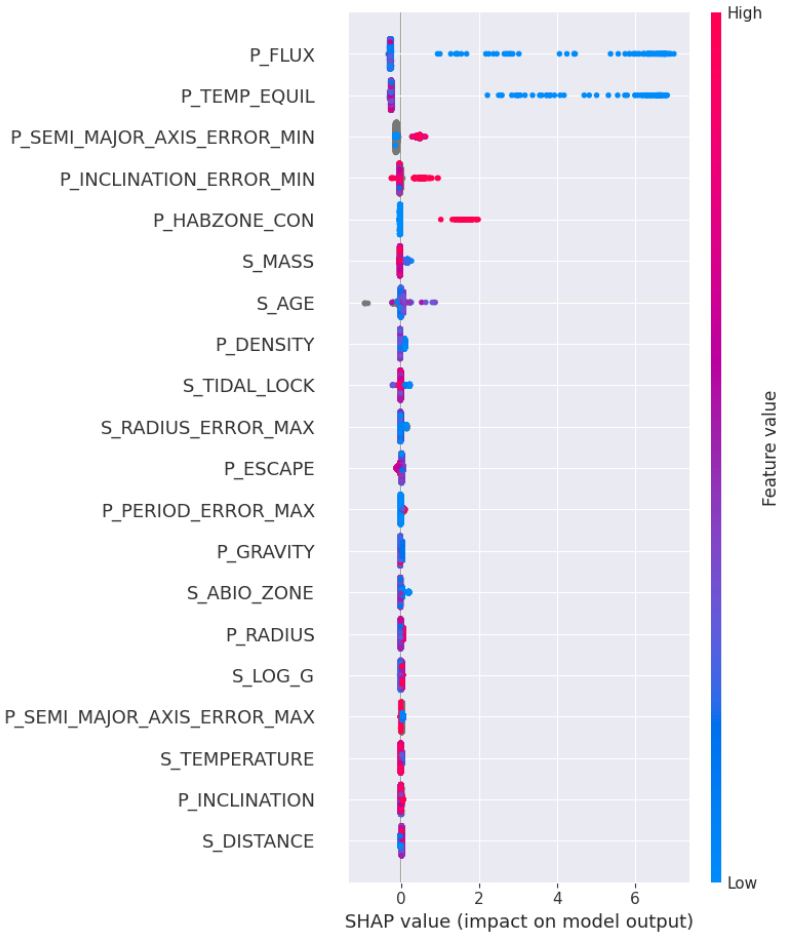

FIGURE 7 – Explicabilité globale avec SHAP, sans la variable *ESI*

On voit que c'est toujours `P_FLUX` et `P_TEMP_EQUIL` qui sont les plus forts contribuables. De plus, on peut remarquer que dans ce cas, les valeurs du SHAP sont plus importantes que précédemment.

2 Projet 2 - DICE

2.1 Méthode DICE

La méthode DICE (pour Diverse Counterfactual Explanations) est, comme SHAP, une méthode d'explicabilité des modèles de machine learning. Ici, cette méthode est basée sur la génération d'explications contrefactuelles. Avec DICE, on va chercher à savoir quelles sont les variables d'entrée à modifier pour obtenir l'autre prédiction. Contrairement à SHAP qui va nous montrer la contribution de chaque variable, DICE va générer des observations alternatives à une observation donnée qui conduit à la prédiction inverse.

2.2 Application du dataset HWC avec DICE

On va reprendre le dataset *Habitable Words Catalog* pour le tester avec DICE. Pour que nos résultats soit interpretable, on va garder le moins de variable possible. On gardera (on met les unité entre crochet pour pouvoir expliquer ensuite) :

- `P_MASS` : Masse de l'exoplanète (en nombre de masse terrestre)
- `P_RADIUS` : Rayon de l'exoplanète (en nombre de rayon terrestre)
- `P_FLUX` : Flux reçu de l'étoile du système (en nombre de fois du flux reçu par la Terre du Soleil)
- `S_DISTANCE` : Distance étoile-exoplanète (en UA, distance Terre-Soleil)
- `P_TEMP_SURF` : Température de surface de l'exoplanète (en K)
- `S_TEMPERATURE` : Température de surface de l'étoile du système (en K)
- `P_HABITABLE` : Pour la variable à prédire comme pour SHAP

On utilise un modèle de Random Forest (avec 100 estimateurs) pour prédire l'habitabilité. Comme pour la partie précédente, on prend 80 % pour le train et 20 % pour le test. On a 489 observations dans le test. Tout d'abord, on vérifie l'accuracy du modèle sur nos tests : on obtient 100 %. On peut maintenant utiliser DICE. On va faire deux analyses : une application sur une observation test qui est habitable, pour voir ce qu'il faudrait pour qu'elle devienne non habitable, et une deuxième application sur, au contraire, une observation non habitable.

DICE sur observation habitable : On recense dans la Table 1 les valeurs et les propositions de DICE pour ce cas.

Table 1 – Application de DICE sur une observation habitable

	P_MASS	P_RADIUS	P_FLUX	S_DISTANCE	P_TEMP_SURF	S_TEMPERATURE	P_HABITABLE
Obs.	5.2	2.13	0.270	7.618	198.9	3728	1
DICE	5.2	2.13	0.270	7.618	**5214.9**	3728	0
DICE	**355.1**	2.13	0.270	7.618	**2308.9**	3728	0

Dans la première proposition de DICE, on voit qu'il a juste modifié la température de surface. En effet, alors que l'on avait à la base $198.9K$ (soit $-76C$, ce qui est un peu froid mais pas non plus extrèmement froid en moyenne), on a maintenant $5214.9K$, soit près de $5000C$, ce qui effectivement rendrait non-habitable.

Dans la deuxième proposition, il a modifié la température de surface mais aussi la masse de la planète. Et effectivement, on peut faire un calcul rapide pour voir ce que cela change au niveau de

l'habitabilité, en calculant ce que cela implique pour la valeur de la gravité à la surface. On peut utiliser la formule suivante :

$$g_{planet} = G \cdot \frac{M_{planet}}{R^2_{planet}}$$

Avec G la constante gravitionnelle $= 6.674 \cdot 10^{-11} m^3 kg^{-1} s^{-2}$.

Pour la Terre, on peut recalculer la valeur connue pour celle-ci :

$$g_{Terre} = G \cdot \frac{M_{Terre}}{R^2_{Terre}} = 6.674 \cdot 10^{-11} \frac{5.974 \cdot 10^{24}}{(6.371 * 10^6)^2} = 9.8 \; m \cdot s^{-2}$$

Pour notre planète, on avait à la base :

$$g_{obs} = G \cdot \frac{5.2 M_{Terre}}{(2.13 R_{Terre})^2} = 11.3 \; m \cdot s^{-2}$$

Ce qui est un peu plus élevé, mais reste supportable. Si on calcule la gravité de la planète modifiée par DICE :

$$g_{obs,DICE} = G \cdot \frac{355.1 M_{Terre}}{(2.13 R_{Terre})^2} = 768.8 \; m \cdot s^{-2}$$

Cette valeur est très élevée (à titre de comparaison, celle du soleil est d'environ 274 $m \cdot s^{-2}$), et cela nous écraserait sur place. On peut donc dire que cette exoplanète deviendrait effectivement non habitable. On peut maintenant tester DICE sur une observation non habitable.

DICE sur observation non habitable : On recense dans la Table 2 les valeurs et les propositions de DICE pour ce cas.

TABLE 2 – Application de DICE sur une observation non habitable

	P_MASS	P_RADIUS	P_FLUX	S_DISTANCE	P_TEMP_SURF	S_TEMPERATURE	P_HABITABLE
Obs.	1.24	1.07	917.12	623.9	1583.2	6112	0
DICE	1.24	1.07	917.12	623.9	**316.4**	6112	1
DICE	1.24	1.07	917.12	**66.0**	**280.2**	6112	1

On a ici un peu le cas inverse de précédemment. Dans le premier cas DICE, il diminue la température jusqu'à $316,4K$ (soit environ 41 degrés Celsius), ce qui devient plutôt habitable. Ensuite, pour la deuxième proposition, il diminue encore la température (ce qui devient un peu froid pour le coup, environ 5 degrés C), mais il diminue aussi la distance étoile-exoplanète, ce qui a une certaine logique, on approche la planète de son étoile et on diminue un peu la température de surface. Cependant, DICE ici ne prend pas en compte les corrélations entre les variables, car ici évidemment, certaines variables sont liées les unes aux autres, et c'est ça ici la limite de ces interprétations.

3 Projet 3 - GradCam

3.1 Méthode GradCam

Grad-CAM (pour Gradient-weighted Class Activation Mapping) [9] est une méthode permettant d'expliquer les prédictions des CNN (Convolutional Neural Network) en mettant en évidence les zones d'une image qui sont les plus impliquées dans ladite prédiction. Avec Grad-CAM, on va pouvoir produire des cartes de chaleur (heatmaps) qui indiquent quelles régions de l'image ont le plus contribué à la décision du modèle pour la prédiction.

Grad-CAM va donc analyser les couches de convolution du réseau de neurones et va identifier quelles sont les zones activent le plus le modèle pour une prédiction donnée.

3.2 Application de Grad-CAM

On va ici appliquer Grad-CAM à l'image de la Figure 8

FIGURE 8 – Image d'origine, avec ici un lapin à gauche et un chien à droite

Le but va être de voir si l'on peut prédire le lapin ET le chien, et si oui, quelles zones le modèle utilise pour chacun des deux. La première étape est de prédire ce qui est représenté dans l'image. Pour cela, on va utiliser un CNN, avec $tensorflow$ et $kiras$. On présente le top 5 prédictions dans la Table 3.

TABLE 3 – Meilleure prédiction de notre modèle pour l'image Figure 8

N°	Label	Probabilité (%)
1	`golden_retriever`	21.2
2	`Great_Pyrenees`	8.1
3	`wood_rabbit`	7.2
4	`Labrador_retriever`	2.0
5	`kuvasz`	1.9

On voit que l'on a notre chien en prédiction **N°1**. Ensuite, les prédictions 2, 4 et 5 sont d'autres races de chien. En **N°3** on a "wood rabbit". En cherchant un peu, on trouve que ça pourrait être le surnom d'une race de lapin le *New England cottontail* [5]. C'est soit ça soit une prédiction litérale d'un lapin en bois. Que ce soit l'un ou l'autre, le modèle a quand même détecté une sorte de lapin !

On va donc maintenant construire une petite heatmap pour visualiser les zones pour les prédictions 1 et 3, tracer dans la Figure 9.

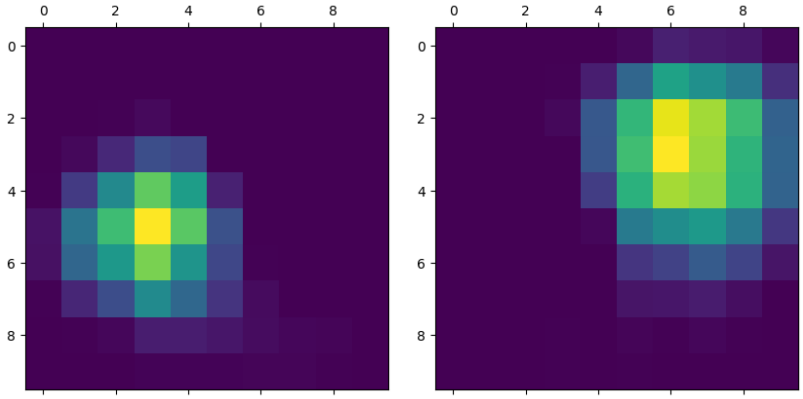

FIGURE 9 – Heatmaps des zones activées pour la prédiction 3 (le *wood rabbit*) à gauche et pour la prédiction 1 (le *golden retriever*) à droite

On peut ensuite mettre les heatmaps sur les images dans la Figure 10

FIGURE 10 – Heatmaps des zones activées pour la prédiction 3 (le *wood rabbit*) à gauche et pour la prédiction 1 (le *golden retriever*) à droite

On peut voir ici que notre modèle s'est bien servi de la zones avec le lapin pour prédire *wood rabbit* et de la zone avec le chien pour prédire *golden retriever*, tout est rassurant !

4 Projet 4 - Interprétation de BERT

4.1 Introduction

BERT (Bidirectional Encoder Representations from Transformers) est un modèle de type Transformer qui apprend des représentations contextuelles des mots grâce à un pré-entraînement sur de grandes quantités de texte. Il repose sur un mécanisme d'attention permettant de capturer les relations entre les mots indépendamment de leur position dans la phrase. Chaque token est encodé sous forme d'un vecteur haute dimension, facilitant la compréhension contextuelle des phrases.

Dans ce projet, nous explorons différentes techniques permettant d'interpréter les prédictions de BERT lorsqu'il est utilisé pour des tâches de Question-Answering. Nous nous concentrons sur deux méthodes d'interprétabilité basées sur les gradients : Layer Integrated Gradients (LIG) et Layer Conductance.

Nous utilisons la bibliothèque Captum pour mettre en œuvre ces méthodes et comprendre le comportement du modèle.

4.2 Modèles Fine-tunés pour Question-Answering

Nous utilisons plusieurs versions de BERT, chacune fine-tunée sur un dataset différent de Question-Answering :

- **bert-base-uncased** : Version de base de BERT sans distinction entre majuscules et minuscules.
- **digitalepidemiologylab/covid-twitter-bert-v2** : Modèle pré-entraîné sur des tweets liés au COVID-19.
- **deepset/bert-base-cased-squad2** : Modèle pré-entraîné et fine-tuné sur SQuAD v2.
- **bert-large-uncased-whole-word-masking-finetuned-squad** : Version plus grande de BERT fine-tunée sur SQuAD (24 couches).

Modèle	**Réponse Prédite**
BERT Base	*Réponse incorrecte, avec inversion des tokens de début et de fin.*
BERT Large (fine-tuné sur SQuAD)	*using self - attention mechanisms*
deepset/bert-base-cased-squad2	*self - attention mechanisms*
Covid-Twitter-BERT	*does a transform ##er model process text ? [SEP] a transform ##er model processes text using self - attention mechanisms , which allow it to capture relationships between words regardless*

TABLE 4 – Comparaison des réponses des modèles fine-tunés sur différentes bases de données pour la question : *how does a transformer model process text ?*

Cette figure illustre comment chaque modèle réagit à une même question, mettant en évidence les différences d'interprétation dues aux datasets d'entraînement.

4.3 Interprétabilité des Modèles : Approches par Attribution

Dans ce projet, nous nous focalisons sur deux méthodes principales : **Layer Integrated Gradients (LIG)** et **Layer Conductance**, qui permettent d'analyser l'importance des mots et des couches dans la prise de décision d'un modèle de type BERT.

4.3.1 Layer Integrated Gradients (LIG)

Principe théorique et formulation mathématique Layer Integrated Gradients (LIG) est une extension de la méthode **Integrated Gradients**, appliquée à des couches spécifiques du réseau de neurones. L'importance d'une entrée x est calculée en comparant les gradients du modèle pour cette entrée avec une entrée de référence x_0 le long d'un chemin intégral.

La formule est définie comme suit :

$$IG(x) = (x - x_0) \times \int_{\alpha=0}^{1} \frac{\partial F(x_0 + \alpha(x - x_0))}{\partial x} d\alpha \tag{1}$$

Où :
- x est l'entrée actuelle du modèle.
- x_0 est une entrée de référence (ex. une entrée neutre).
- α est un facteur de pondération variant de 0 à 1.
- F est la fonction de sortie du modèle.

Cette équation calcule la somme des gradients le long du chemin linéaire entre l'entrée de référence et l'entrée réelle, permettant ainsi d'identifier l'importance de chaque composant dans la décision du modèle.

Résultats et Interprétation Nous avons appliqué LIG à un modèle BERT fine-tuné sur le jeu de données **SQuAD** pour l'interprétation de questions-réponses. La figure 11 illustre les scores d'attribution obtenus.

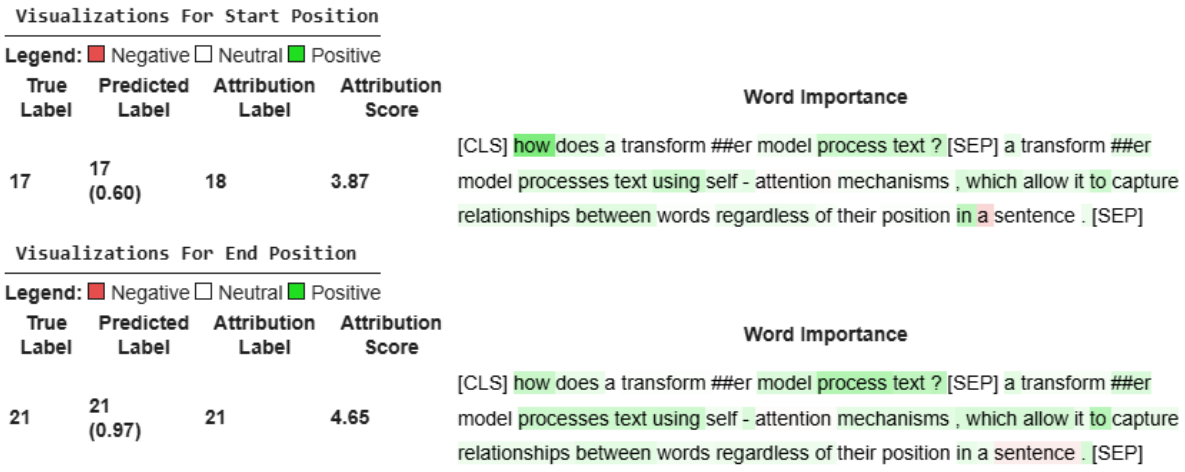

FIGURE 11 – Visualisation des attributions des tokens avec LIG.

Pour le token de début, nous pouvons observer que les mots interrogatifs "How", ceux directement liés à la réponse attendue, et ceux qui se trouvent just avant le token de début "self" reçoivent une attribution plus élevée ce qui permet d'interpréter que ces tokens là permettent au modèle de

choisir le token "self" comme token de départ.

En ce qui concerne le token de fin, on remarque que les tokens à la fin de la question, le token de fin "mechanism" ainsi que les 4 tokens d'après contribuent le plus à la prédiction. Cela est plutôt logique et cohérent car pour choisir le token de fin, le modèle a besoin de comprendre le contexte de la question ainsi que les tokens qui se retrouvent vers la fin de la réponse attendue.

Pour le moment, nous avons considéré un seul type d'embedding, à savoir la liste des tokens donnés en entrée. Cependant, nous allons à présent prendre en compte trois embeddings distincts et voir l'influence de chacun d'entre eux :

— L'embedding des mots (word embedding).
— L'embedding de l'ordre des mots (position embedding).
— L'embedding du type de token (token type embedding).

Afin d'analyser l'influence des différents embeddings sur l'interprétation des résultats, nous avons comparé les cinq premiers tokens les plus influents pour chaque type d'embedding, à la fois pour le token d'entrée et le token de sortie.

Word(Index), Attribution	Token Type(Index), Attribution	Position(Index), Attribution
self (18), 0.95	[SEP] (10), 0.95	self (18), 0.92
using (17), 0.2	[SEP] (39), 0.2	using (17), 0.21
which (23), 0.09	self (18), 0.09	a (3), 0.11
a (3), 0.08	- (19), 0.08	which (23), 0.08
allow (24), 0.07	which (23), 0.07	allow (24), 0.08

TABLE 5 – Comparaison des attributions des embeddings pour le token d'entrée.

Attributions des embeddings pour le token d'entrée Nous observons que les mots les plus influents dans l'embedding des mots sont liés à l'action et à l'agent (ex. *self*, *using*), tandis que les attributions de type token montrent une forte importance des séparateurs et des tokens spéciaux qui séparent la question du contexte et qui marque la fin de la séquence de tokens. L'embedding de position, quant à lui, met en avant des tokens clés à des positions stratégiques dans la phras et a des résultats similaires que celui des mots.

Word(Index), Attribution	Token Type(Index), Attribution	Position(Index), Attribution
mechanisms (21), 0.82	[SEP] (10), 0.82	mechanisms (21), 0.91
using (17), 0.37	[SEP] (39), 0.37	using (17), 0.27
self (18), 0.33	mechanisms (21), 0.33	self (18), 0.19
, (22), 0.17	self (18), 0.17	text (16), 0.08
text (16), 0.12	, (22), 0.12	processes (15), 0.06

TABLE 6 – Comparaison des attributions des embeddings pour le token de sortie.

Attributions des embeddings pour le token de sortie Pour le token de sortie, nous constatons que les termes liés à la mécanique et aux actions (*mechanisms*, *using*) sont les plus influents dans les embeddings de mots. Les attributions de type token continuent de mettre en avant les tokens spéciaux, tandis que les embeddings de position renforcent l'importance des termes au centre de l'action décrite.

Ces analyses approfondies nous permettent de mieux comprendre l'influence des différentes composantes des embeddings sur la prise de décision du modèle.

4.3.2 Layer Conductance

Principe théorique et formulation mathématique Layer Conductance mesure la contribution de chaque couche à une sortie donnée. Contrairement à LIG qui s'intéresse à l'importance des mots, Layer Conductance évalue l'impact des activations intermédiaires sur la prédiction finale.

La conductance C d'une couche l est définie comme :

$$C_l = \sum_i (x_i - x_{0,i}) \times \frac{\partial F}{\partial x_i}(x_i) \tag{2}$$

Où :
- x_i est l'activation d'une unité neuronale dans la couche l.
- $x_{0,i}$ est l'activation de référence pour cette unité.
- F est la sortie du modèle.

Cette équation mesure comment chaque neurone contribue à la prédiction finale en comparant ses activations à une base de référence, aidant ainsi à identifier les couches ayant le plus grand impact sur les décisions du modèle.

Résultats et Interprétation L'analyse des attributions par couche a mis en évidence que certaines couches du modèle BERT jouent un rôle plus important dans la prise de décision que d'autres. De plus, on peut voir l'évolution de l'importance des tokens quand on passe d'une couche à une autre. La figure 12 illustre la conductance des couches pour le token d'entrée.

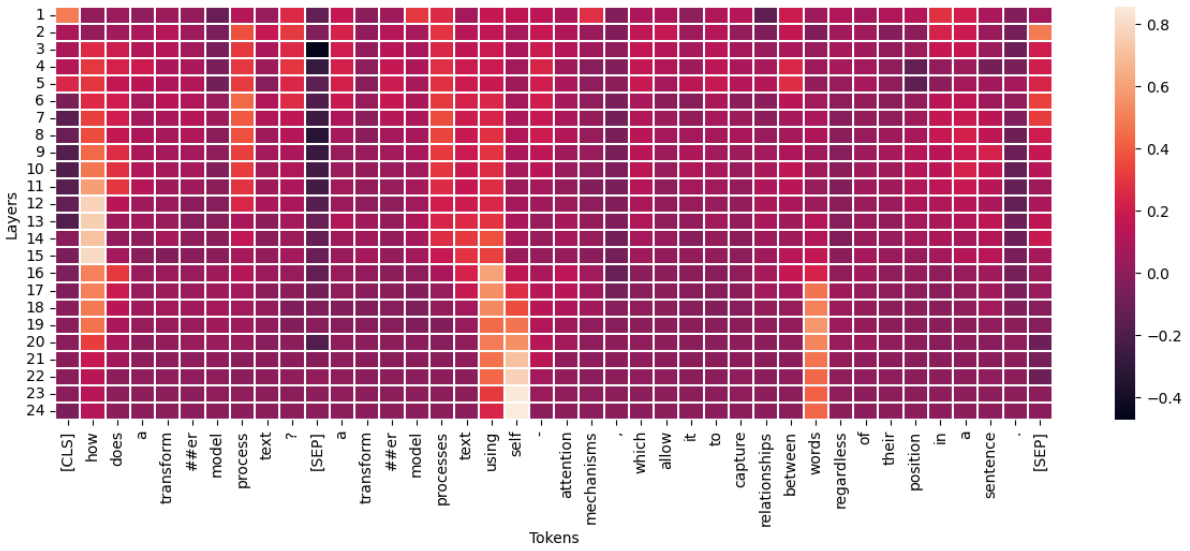

FIGURE 12 – Heatmap des attributions des couches pour le token d'entrée.

Nous observons que tous les tokens ont l'air davoir une contribution plus ou moins égale dans la plupart des couches. Cela montre que le modèle s'est basé sur la plupart des tokens pour trouver le token de début. Toutefois, on remarque l'évolution particulière de quelques tokens notamment les tokens [CLS] et [SEP] qui prennent plus d'importance vers les couches finales. Le token "How" est important dans toutes les couches mais de manière beaucoup plus prononcée dans les couches du milieu. Enfin, les tokens "using" et "self" sont les plus importants avec une augmentation de leur importance quand on s'approche vers la dernière couche.

De manière similaire, nous analysons la conductance des couches pour le token de sortie en figure 13.

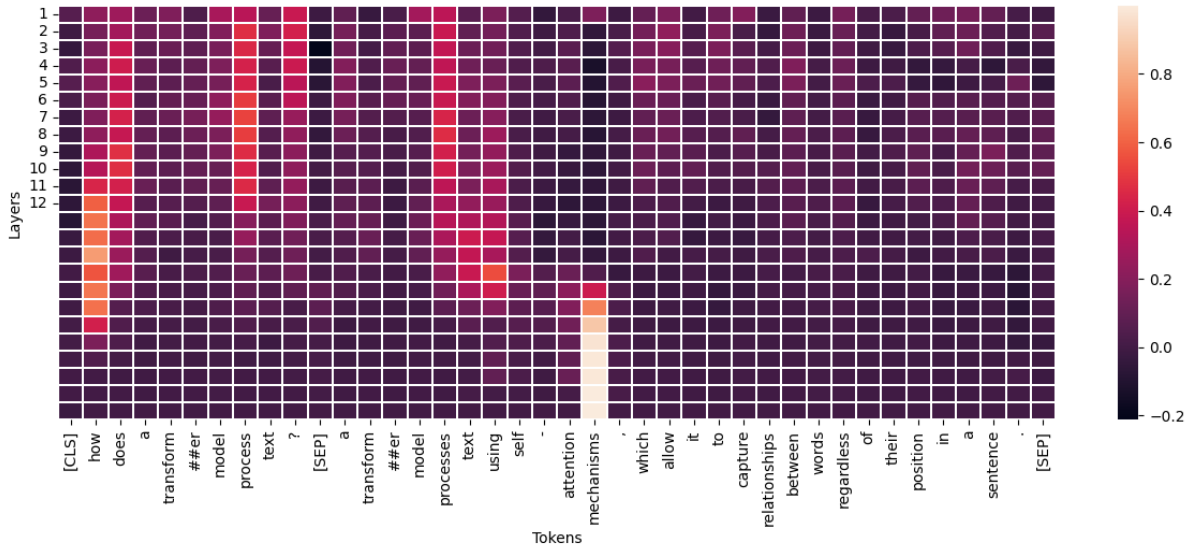

FIGURE 13 – Heatmap des attributions des couches pour le token de sortie.

Le token de sortie voit son importance amplifiée dans les dernières couches du modèle, qui semblent focaliser l'attention sur les éléments les plus pertinents pour la réponse finale. De plus, on remarque que le modèle ne se focalise pas sur la majorité des tokens pour trouver le token de fin et ce sur toutes les couches avec quelques exceptions. D'abord, les tokens de début et de fin de la question sont importants surtout dans les premières couches.

Enfin, la distribution des probabilités des attributions des couches pour le token de fin "mechanisms" est présentée en figure 14.

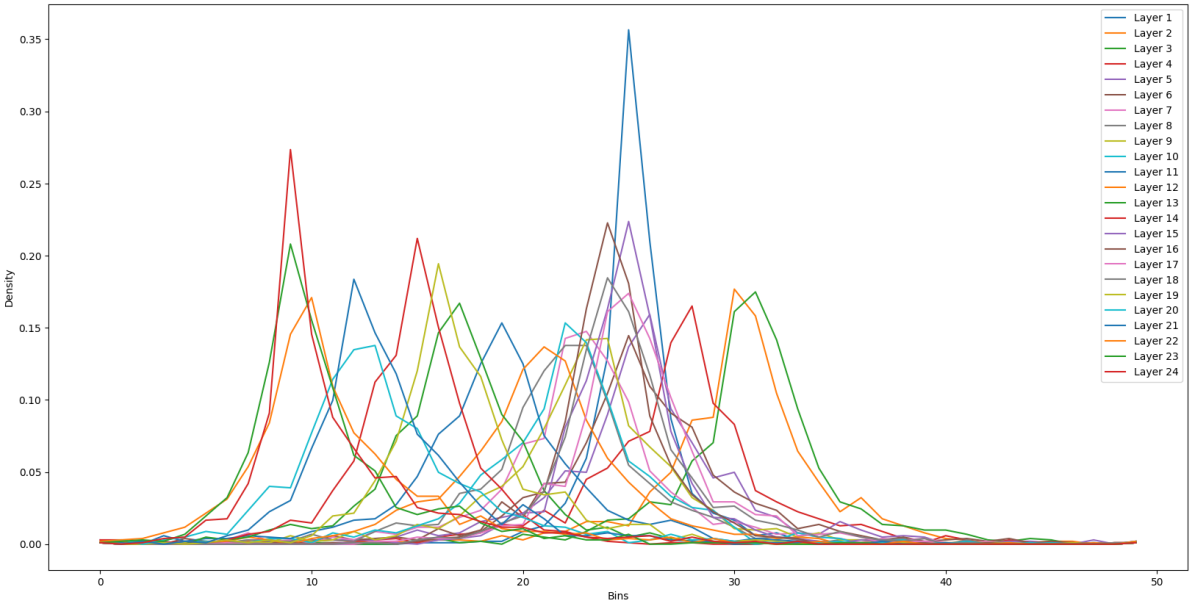

FIGURE 14 – Distribution des probabilités des attributions des couches pour le token de fin "mechanisms".

— L'axe des **x** montre les valeurs d'attribution (*importance accordée au token*).
— L'axe des **y** représente la densité (*fréquence d'apparition de ces valeurs dans la couche*).

Interprétation

— Une **courbe étroite et haute** signifie que la couche **attribue de manière cohérente** une importance similaire au token.
— Une **courbe large** indique une plus grande dispersion des attributions, suggérant que certaines parties de la couche contribuent plus que d'autres.
— Si le **pic est déplacé** vers la gauche ou la droite, cela signifie que l'importance du token *"mechanisms"* varie selon la profondeur du modèle.

5 Projet 5 - Layerwise Relevance Propagation pour l'analyse d'image

5.1 Introduction

Layerwise Relevance Propagation (LRP) se distingue par sa capacité à attribuer de manière rétroactive l'importance de chaque neurone à l'entrée du réseau. Contrairement aux gradients classiques, LRP suit un principe de redistribution de la sortie du modèle vers les couches précédentes, offrant ainsi une visualisation intuitive des zones les plus influentes d'une image pour une classification donnée.

5.2 Cadre Théorique

LRP repose sur une répartition des relevances (importances) depuis la dernière couche du réseau jusqu'à la première, en s'assurant que l'information transmise respecte deux principes fondamentaux :

- **La positivité de la relevance** : La somme des relevances attribuées aux couches intermédiaires ne doit pas générer de valeurs négatives, garantissant ainsi une interprétation cohérente.
- **Le principe de conservation** : La somme des relevances attribuées aux neurones d'une couche doit être égale à la somme des relevances de la couche suivante, ce qui assure une propagation équilibrée.

La propagation des relevances suit la règle suivante pour un neurone j de la couche l :

$$R_j^{(l)} = \sum_k \frac{z_{jk}}{\sum_{j'} z_{j'k}} R_k^{(l+1)} \tag{3}$$

où :

- $R_k^{(l+1)}$ est la relevance attribuée au neurone k de la couche $(l+1)$,
- z_{jk} est la contribution du neurone j au neurone k en fonction de ses poids synaptiques,
- La somme est effectuée sur tous les neurones de la couche suivante $(l+1)$.

Ce mécanisme garantit que l'information issue de la décision finale est redistribuée jusqu'aux entrées du modèle.

5.3 Réseau de Neurones Fully Connected

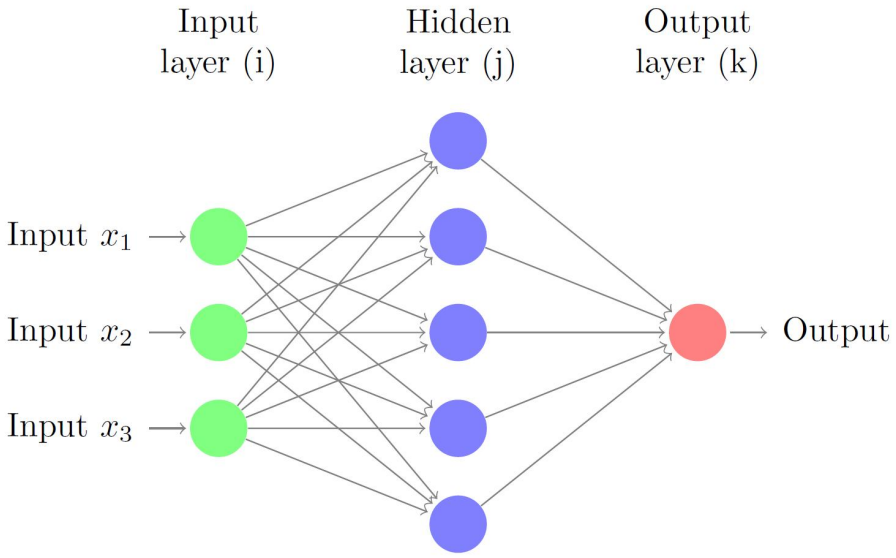

FIGURE 15 – Réseau de neurones Fully Connected utilisé pour l'analyse avec LRP.

Pour illustrer LRP, nous avons utilisé un réseau de neurones entièrement connecté (Fully Connected, FC). L'architecture du réseau est illustrée dans la figure ci-dessus. Le réseau est constitué des éléments suivants :

— Une architecture simple avec une couche à 3 entrées, une couche cachée à 5 neurones et une couche de sortie avec un unique neurone (son activation est égale à la somme des activations de la couche cachée).
— La fonction d'activation utilisée est ReLU, qui permet de conserver uniquement les valeurs positives et facilite l'interprétation.

Le calcul de l'activation du neurone de la couche de sortie est précisé ci-dessous :

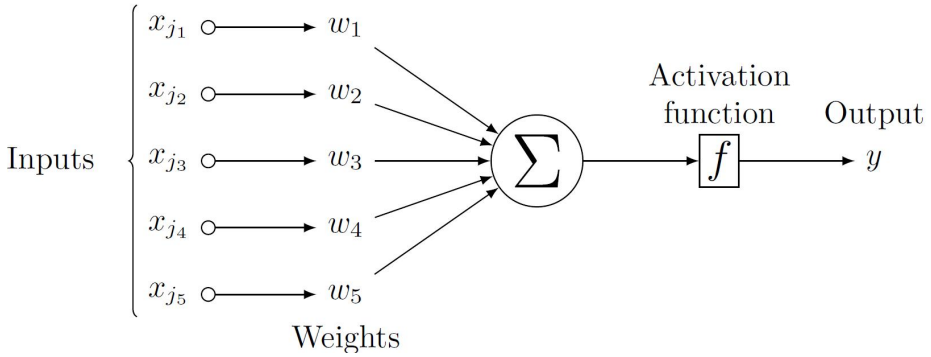

FIGURE 16 – Réseau de neurones Fully Connected utilisé pour l'analyse avec LRP.

5.4 Calcul des Relevances

Pour appliquer LRP, nous avons commencé par fixer les valeurs des poids entre la couche d'entrée et la couche cachée, ainsi qu'entre la couche cachée et la sortie. De même, nous avons fixé les valeurs des deux entrées du modèle.

Ensuite, nous avons effectué un forward pass à travers le réseau. La sortie obtenue du neurone final correspond directement à la relevance totale, et elle est positive en raison de l'utilisation de la fonction d'activation ReLU, qui préserve uniquement les valeurs positives. On note que les relevances des neurones de la couche cachée sont égales à leurs activations, car la fonction ReLU conserve cette relation.

À partir de cette sortie, nous avons appliqué la relation de propagation des relevances décrite dans la section théorique :

$$R_j^{(l)} = \sum_k \frac{z_{jk}}{\sum_{j'} z_{j'k}} R_k^{(l+1)} \tag{4}$$

où $R_k^{(l+1)}$ est la relevance du neurone de la couche suivante et z_{jk} représente la contribution du neurone j au neurone k de la couche suivante.

Grâce à cette équation, nous avons pu calculer les relevances attribuées aux entrées du réseau. Les histogrammes ci-dessous illustrent ces relevances :
— **À gauche** : les relevances attribuées aux **entrées du modèle**.
— **À droite** : les relevances des **neurones de la couche cachée**.

Cette approche offre une interprétabilité du modèle, en permettant d'identifier quels inputs contribuent le plus à la décision finale. Nous pouvons ainsi mieux comprendre les mécanismes sous-jacents du réseau de neurones et identifier les caractéristiques les plus déterminantes dans la classification.

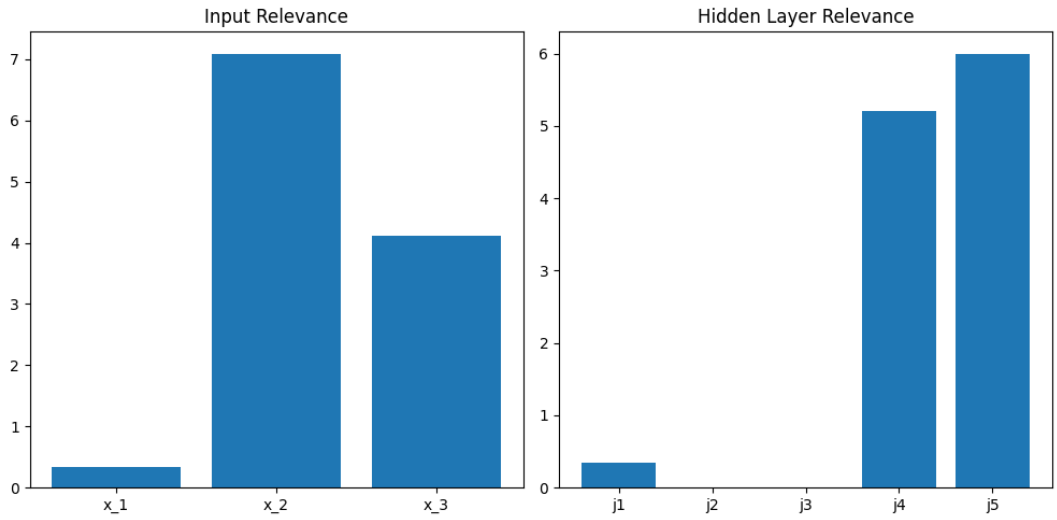

FIGURE 17 – Propagation des relevances obtenue pour le réseau Fully Connected.

5.5 Application de LRP

Pour implémenter LRP, nous avons utilisé PyTorch et défini un réseau de neurones simple pour une tâche de classification binaire sur un jeu de données synthétique (*moons dataset*). Après entraînement, nous avons appliqué LRP pour analyser l'importance de chaque feature dans la prise de décision du modèle.

Nous avons représenté les distributions des relevances sous forme de boxplots, mettant en évidence la variabilité et l'importance relative des deux features dans les prédictions du modèle.

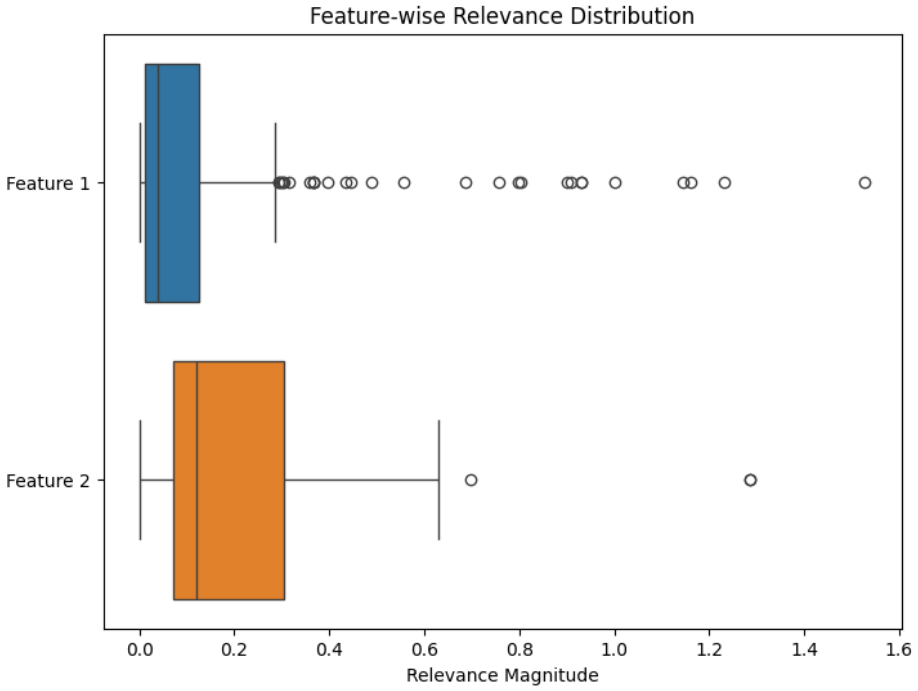

FIGURE 18 – Distribution des magnitudes de relevance pour chaque feature.

6 Projet 6 - LRP pour réseaux de neurones graphiques (GNNs)

6.1 Introduction et motivation

Contrairement à des données plus classiques (images, séries temporelles, etc.), les graphes décrivent des entités (les nœuds) et les relations qui les relient (les arêtes). Cette structure offre une grande expressivité, mais complique également l'analyse et la modélisation par des méthodes classiques de machine learning. L'objectif de ce projet est de développer et d'expliquer (XAI : eXplainable AI) les prédictions d'un réseau de neurones spécialisé pour les graphes, appelé *Graph Neural Network* (GNN).

6.2 Génération des graphes scale-free et facteur de croissance

Pour constituer un jeu de données varié, nous avons besoin de graphes de tailles identiques mais de structures différentes. Nous utilisons pour cela le **modèle de Barabási–Albert**, également appelé modèle *scale-free*. Dans ce modèle, les nœuds sont ajoutés un à un au graphe, et chaque nouveau nœud se connecte à un nombre fixe de nœuds déjà présents, selon une probabilité proportionnelle à leur degré (nombre de liens). Cette probabilité de connexion favorise les nœuds déjà fortement connectés, ce qui reproduit un phénomène réel d'« attraction des plus connectés » (on parle alors d'effet « riche devient plus riche »).

Le *facteur de croissance* (`growth factor`) détermine combien de nouvelles arêtes chaque nœud ajouté va former. Par exemple, un facteur de croissance de 2 implique que chaque nouveau nœud se connecte à deux nœuds existants. Dans notre projet, ce `growth factor` sert de **label** à prédire : chaque graphe est ainsi étiqueté par la valeur du facteur de croissance qui a servi à le générer. Nous varions plusieurs valeurs du facteur de croissance (1, 2, etc.) pour créer un jeu de données plus ou moins diversifié.

6.3 Réseaux de neurones graphiques et LRP

Un *Graph Neural Network* (GNN) est un modèle de deep learning conçu pour traiter les données structurées en graphe. Pour expliquer les prédictions de ce GNN, on utilise la méthode de Layerwise Relevance Propagation (LRP).

- **Principe :** LRP rétro-propage la prédiction finale à travers les couches du réseau pour mesurer la contribution (positive ou négative) de chaque nœud ou sous-structure du graphe.
- **Visualisation :** Dans les figures, les contributions positives (celles qui poussent la prédiction vers une classe donnée) sont tracées en rouge, tandis que les contributions négatives (qui éloignent la décision de cette classe) sont en bleu.

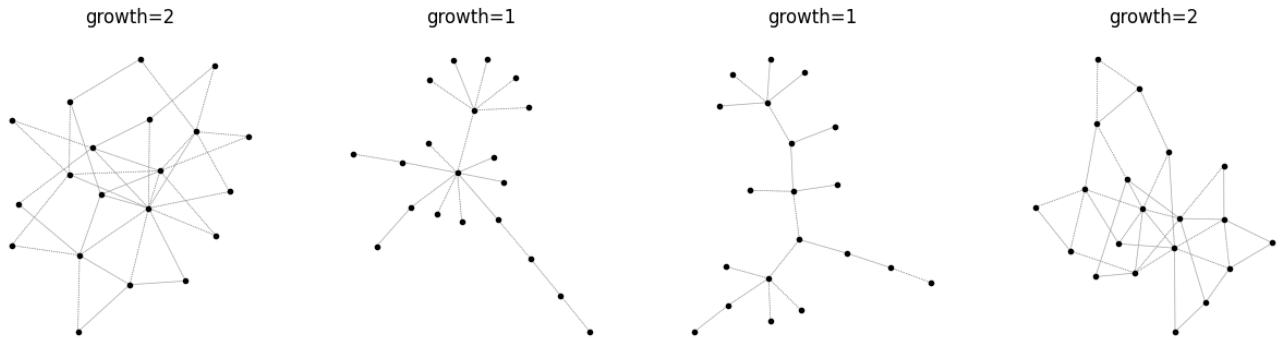

FIGURE 19 – Exemples de graphes générés avec un *facteur de croissance* (`growth`) égal à 1 ou 2. Chaque graphe présente un niveau de connectivité plus dense, car chaque nouveau nœud se connecte à 1 ou 2 nœuds déjà existants selon le modèle de Barabási–Albert.

6.4 Interprétation des résultats

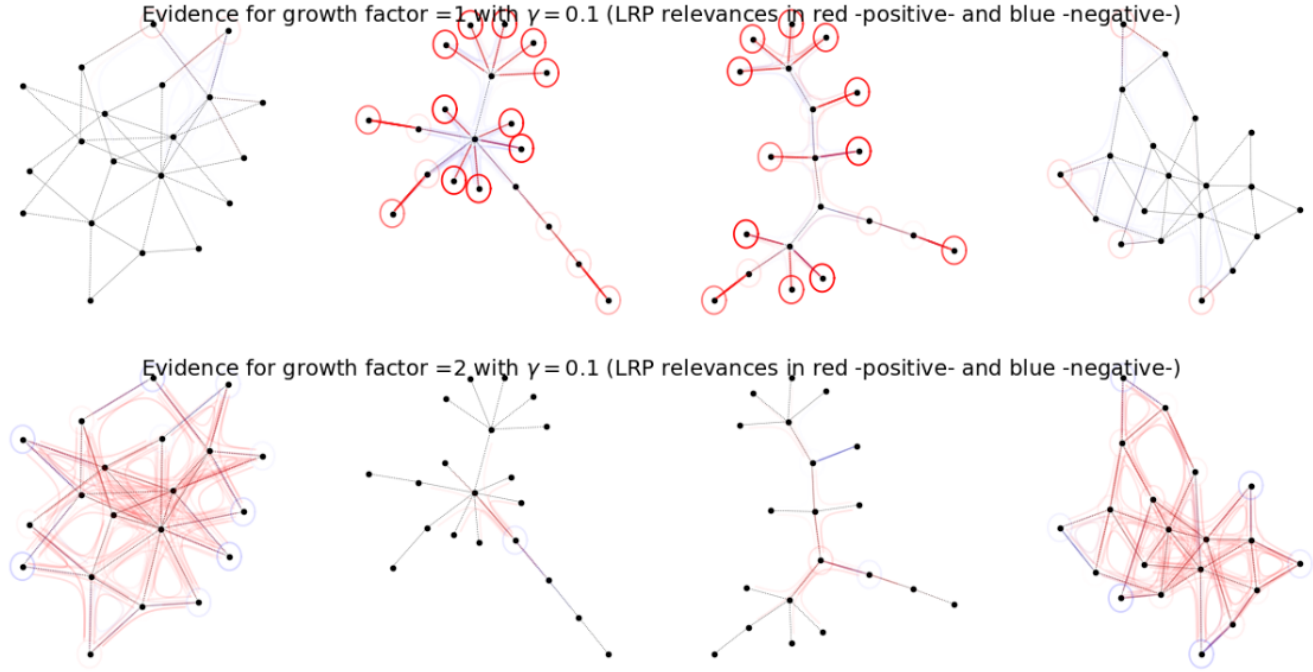

FIGURE 20 – Exemple de graphes avec les relevances LRP, où les arcs en rouge indiquent une contribution positive à la classe considérée et ceux en bleu une contribution négative.

La figure ci-dessus illustre différents graphes, chacun généré avec un facteur de croissance donné (ici 1 ou 2), puis classifié par le GNN. Sur chaque sous-figure :

- **Les nœuds et leurs arêtes** sont tracés en noir (points et lignes en pointillés).
- **Les arcs colorés (rouge ou bleu)** représentent des *chemins de longueur 3* (des triplets de nœuds) où la méthode LRP a identifié une forte pertinence.
- **Le rouge (positive relevance)** indique une contribution qui renforce l'idée que le graphe appartient à la classe considérée (par exemple, facteur de croissance 1).

— **Le bleu (negative relevance)** marque, au contraire, des sous-parties du graphe qui s'opposent à cette classe (faisant pencher la prédiction vers une autre valeur possible du facteur de croissance).

6.5 Conclusion

Dans le cas d'un graphe plutôt "en étoile", un grand nombre de contributions positives (rouges) apparaît autour des nœuds centraux, ce qui indique que la topologie en étoile est un indice fort pour l'une des classes de croissance (ici, `growth = 1`). À l'inverse, des graphes plus denses avec de multiples cycles présenteront souvent plus de zones rouges pour la classe `growth = 2` ou supérieure.

Grâce à LRP, on peut donc visualiser la logique interne du réseau : on voit où se concentrent les éléments structurels les plus déterminants pour la décision du GNN. Une telle approche est importante pour la transparence et l'interprétabilité du modèle, permettant à un utilisateur de mieux comprendre pourquoi le réseau estime qu'un graphe s'apparente à un certain facteur de croissance plutôt qu'à un autre.

Références

[1] Sebastian Bach et al. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In : *PloS one* 10.7 (2015). https://doi.org/10.1371/journal.pone.0130140, e0130140.

[2] A-L Barabási et R Albert. "Emergence of scaling in random networks". In : *Science* 286.5439 (1999), p. 509-512.

[3] *Habitable Worlds Catalog (HWC)*. URL : https://phl.upr.edu/hwc/about.

[4] Scott M Lundberg et Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In : *arXiv preprint arXiv :1705.07874* (2017). URL : https://arxiv.org/pdf/1705.07874.

[5] *New England cottontail*. URL : https://en.wikipedia.org/wiki/New_England_cottontail.

[6] Facebook AI Research. *Captum : Model Interpretability for PyTorch*. https://captum.ai/. Accessed : 2025-02-11. 2025.

[7] Lawrence K Saul et Sam T Roweis. "An introduction to locally linear embedding". In : *unpublished. Available at : http ://www. cs. toronto. edu/˜ roweis/lle/publications. html* (2000).

[8] Thomas Schnake et al. "Higher-Order Explanations of Graph Neural Networks via Relevant Walks". In : *arXiv preprint arXiv :2006.03589* (2020). https://doi.org/10.48550/arXiv.2006.03589.

[9] Ramprasaath R Selvaraju et al. *Grad-CAM : Why did you say that ?* 2017. arXiv : 1611.07450 [stat.ML]. URL : https://arxiv.org/abs/1611.07450.