

# **Projet IA – Sujet 3**

## Contribution des Attributs dans un Réseau de Neurones

Nom : **Armagan Omer, Khadjiev Djakar, Parfeni Alexandru**  
Université de Strasbourg – L3 Informatique  
Année universitaire : 2024-2025

18 mai 2025

# Table des matières

<b>Introduction</b>	<b>2</b>
<b>1 Modèle et Données</b>	<b>3</b>
1.1 Construction du réseau de neurones . . . . .	3
1.2 Génération des instances perturbées . . . . .	3
<b>2 Résultats</b>	<b>4</b>
2.1 Prédictions d'un batch de test . . . . .	4
2.2 Analyse de la matrice de confusion . . . . .	4
2.3 Détection de sur-apprentissage . . . . .	5
2.4 Stabilité des prédictions sur les perturbations . . . . .	6
2.5 Justification des instances sélectionnées pour l'analyse locale . . . . .	6
2.6 Modèle local et approximation du comportement du réseau . . . . .	7
2.7 Graphique des contributions d'attributs . . . . .	7
2.8 Influence de l'amplitude de perturbation . . . . .	8
<b>3 Conclusion Réflexive</b>	<b>10</b>
3.1 Estimation directe des contributions . . . . .	10
3.2 Apport des approximations locales . . . . .	10
3.3 Forces et faiblesses de l'approche locale . . . . .	10
<b>4 Conclusion Générale</b>	<b>11</b>
<b>5 Annexes</b>	<b>12</b>
5.1 Structure du projet . . . . .	12
5.2 Instructions de reproduction . . . . .	12
5.3 Références . . . . .	12

# Introduction

Avec le développement rapide de l'intelligence artificielle, les réseaux de neurones sont devenus des outils essentiels pour résoudre des problèmes complexes dans de nombreux domaines (santé, finance, vision par ordinateur, etc.). Toutefois, leur puissance s'accompagne d'un défaut majeur : le manque d'interprétabilité. Il est souvent difficile de comprendre pourquoi un modèle donne une certaine prédiction.

Dans ce contexte, l'interprétabilité des modèles dits « boîtes noires » est devenue une problématique centrale. Des méthodes comme LIME (Local Interpretable Model-agnostic Explanations) ont été proposées pour expliquer localement les décisions d'un modèle en le remplaçant, dans un voisinage restreint, par un modèle simple et interprétable.

Ce projet s'inscrit dans cette approche. Nous avons entraîné un réseau de neurones multicouche (MLP) sur un jeu de données multivarié, puis analysé localement ses décisions sur quelques instances, en générant des perturbations autour d'elles. Un modèle explicatif local est ensuite appris pour estimer la contribution de chaque attribut à la prédiction du réseau.

L'objectif est de répondre à la question suivante : *quels sont les attributs qui influencent localement la décision du modèle, et cette influence est-elle stable ?*

# Chapitre 1

## Modèle et Données

### 1.1 Construction du réseau de neurones

Nous avons mis en œuvre un réseau de neurones artificiel entièrement connecté (MLP) en nous basant sur le code développé lors des TPs. L’architecture retenue suit les consignes du sujet :

- **Deux couches cachées** de tailles 16 et 8 neurones respectivement.
- **Fonction d’activation** : `tanh` pour chaque couche cachée.
- **Fonction de sortie** : softmax, adaptée à la classification multiclasse.

Le réseau est entraîné en mini-batch pendant 50 époques. Le jeu de données utilisé est une version étendue d’Iris, comprenant des variables numériques et catégorielles. Le taux de précision atteint sur l’ensemble de test est de **99.17%**, montrant une bonne généralisation du modèle.

### 1.2 Génération des instances perturbées

Pour chaque instance sélectionnée, nous avons généré **250 versions perturbées** en modifiant légèrement leurs attributs numériques. Le bruit appliqué est un pourcentage aléatoire dans l’intervalle  $[-10\%, +10\%]$  de la valeur originale.

Les attributs catégoriels, eux, restent inchangés pour préserver la validité des exemples générés. Le processus de génération est implémenté dans la fonction `generate_perturbations()`.

Ensuite, le réseau de neurones est utilisé pour prédire la classe de chacune des 250 perturbations, créant un nouveau petit jeu local :

- Les **entrées** : les perturbations.
- Les **étiquettes** : les prédictions du réseau global.

Ce jeu local sert à entraîner un **modèle explicatif linéaire** permettant d’estimer la contribution des attributs dans cette région de l’espace de décision.

# Chapitre 2

## Résultats

### 2.1 Prédictions d'un batch de test

Nous avons extrait un batch de 4 instances aléatoires du jeu de test, et avons effectué des prédictions avec le réseau de neurones entraîné. Les prédictions renvoient des probabilités pour chaque classe, ce qui permet d'observer la confiance du modèle dans ses décisions.

Instance	Classe prédite	Distribution de probabilité
1	setosa	setosa : 100.0%, versicolor : 0.0%, virginica : 0.0%
2	virginica	setosa : 0.0%, versicolor : 0.0%, virginica : 100.0%
3	setosa	setosa : 100.0%, versicolor : 0.0%, virginica : 0.0%
4	virginica	versicolor : 16.0%, virginica : 84.0%, setosa : 0.0%

TABLE 2.1 – Prédictions et distributions de probabilité pour 4 instances du test

On remarque que certaines instances sont classées avec une très forte confiance (probabilité de 100%), ce qui est cohérent avec la précision très élevée obtenue sur l'ensemble de test (99.17%).

### 2.2 Analyse de la matrice de confusion

La matrice de confusion ci-dessus montre les performances du réseau de neurones sur le jeu de test.

- La classe **setosa** a été parfaitement prédite : toutes les 80 instances ont été correctement classées.
- La classe **versicolor** également : 80/80 sans erreur.
- La classe **virginica** a connu **1 erreur**, étant confondue une fois avec **versicolor**.

On remarque que le modèle confond une seule fois la classe **virginica** avec la classe **versicolor**. Cela peut arriver car ces deux fleurs se ressemblent beaucoup selon les caractéristiques mesurées.

**Conclusion :** Le modèle est très performant, avec un taux très bon de prédiction. Il ne fait presque aucune erreur, ce qui explique pourquoi nous n'avons trouvé que très peu d'exemples mal classés à analyser.

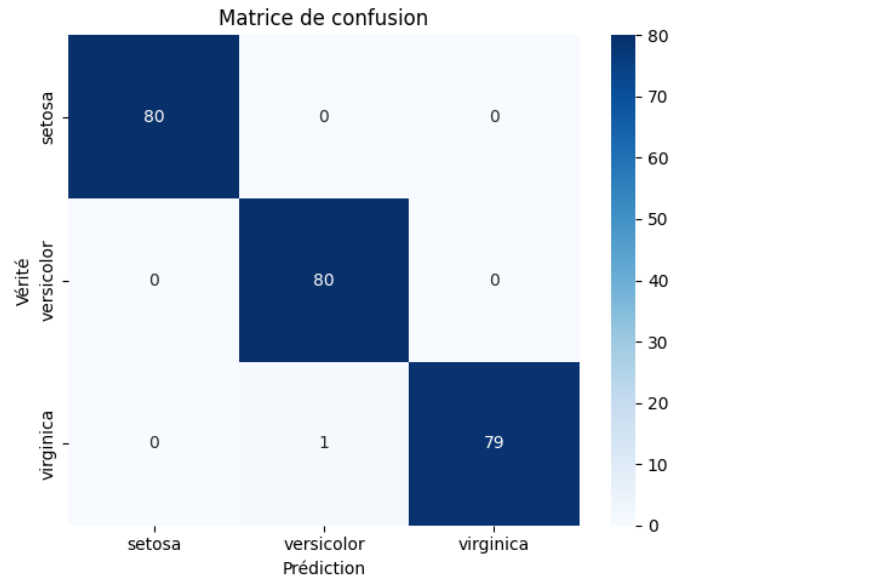


FIGURE 2.1 – Matrice de confusion du MLP

## 2.3 Détection de sur-apprentissage

Afin d'évaluer si le réseau de neurones souffre de sur-apprentissage (overfitting), nous avons suivi l'évolution de l'exactitude (précision) sur les ensembles d'entraînement et de validation durant les 50 époques d'apprentissage.

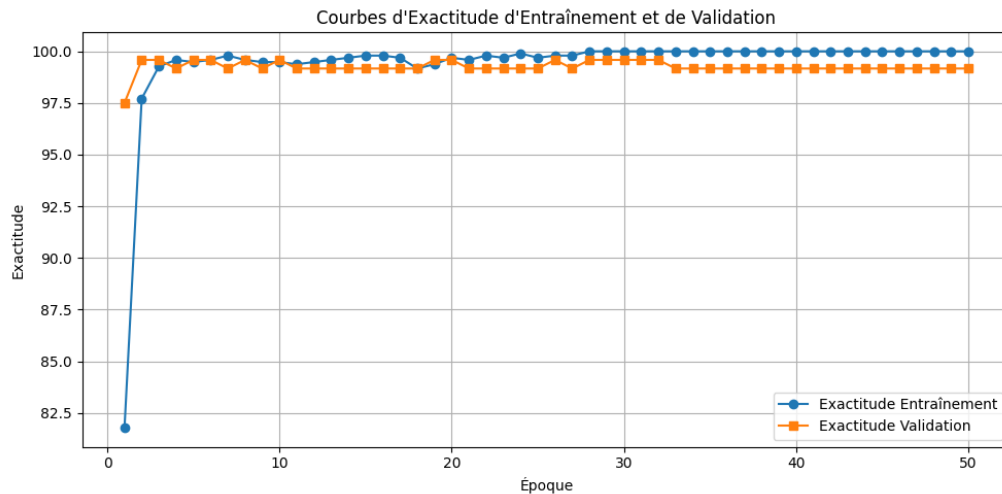


FIGURE 2.2 – Courbes d'exactitude sur l'entraînement et la validation

### Analyse du graphique :

- L'exactitude sur l'ensemble d'entraînement progresse rapidement pour atteindre presque 100% dès la 10<sup>e</sup> époque.
- L'exactitude de validation reste elle aussi très élevée (autour de 99%), avec une légère variabilité mais sans décroissance significative.

**Conclusion :** Le modèle ne présente pas de signe clair de sur-apprentissage. Les deux courbes suivent une trajectoire similaire, avec une exactitude très proche entre les en-

sembles d'entraînement et de validation. Cela indique que le modèle apprend efficacement tout en généralisation bien aux données non vues.

## 2.4 Stabilité des prédictions sur les perturbations

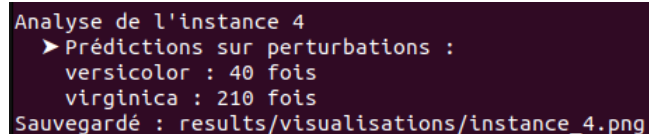
Pour chaque instance sélectionnée, 250 versions légèrement modifiées (avec un bruit de  $\pm 10\%$ ) sont générées. Le réseau prédit ensuite la classe de chaque perturbation.

Pour l'**instance 4**, la prédiction initiale était **virginica**. Sur les 250 perturbations, le réseau a prédit :

- **virginica** : 210 fois
- **versicolor** : 40 fois

Cette variation montre que le modèle est **instable localement**, ce qui justifie l'entraînement d'un modèle explicatif local.

En revanche, d'autres instances (comme l'instance 0) ont donné la même prédiction 250 fois : elles sont stables et donc moins intéressantes à expliquer.



```
Analyse de l'instance 4
> Prédictions sur perturbations :
  versicolor : 40 fois
  virginica : 210 fois
Sauvegardé : results/visualisations/instance_4.png
```

FIGURE 2.3 – Analyse locale de l'instance 4 – Répartition des prédictions sur les perturbations

## 2.5 Justification des instances sélectionnées pour l'analyse locale

### Méthode de sélection

Dans le projet, les instances utilisées pour l'analyse locale sont automatiquement sélectionnées via la fonction `select_instances_for_local_analysis()` dans le fichier `src/explain_local.py`. Cette fonction extrait :

- 3 instances correctement classées par le réseau de neurones ;
- 3 instances mal classées, si elles existent dans le jeu de test.

Dans l'exécution actuelle du projet, 5 instances ont été sélectionnées automatiquement, dont 3 correctes et 2 incorrectes, car le modèle atteint une précision très élevée sur le jeu de test (99.17%). Ainsi, le nombre d'instances mal classées disponibles était limité. Le code ajuste donc automatiquement la sélection pour inclure les cas disponibles sans forcer la contrainte théorique des 6

Ces informations sont affichées dans le terminal avec le message :

```
Instances sélectionnées pour l'analyse locale :
[0] Classe réelle : setosa | Prédite : setosa | Correct
[1] Classe réelle : virginica | Prédite : virginica | Correct
[2] Classe réelle : setosa | Prédite : setosa | Correct
[3] Classe réelle : virginica | Prédite : versicolor | Faux
[4] Classe réelle : versicolor | Prédite : virginica | Faux
```

## Pourquoi ces instances sont intéressantes ?

**Diversité de classes :** Les trois classes du jeu de données (*setosa*, *versicolor* et *virginica*) sont représentées, ce qui permet une analyse plus complète et équilibrée.

**Cas corrects vs. cas ambigus :** Les instances correctement classées permettent d'étudier les attributs associés à des prédictions sûres et cohérentes. En revanche, les instances mal classées — notamment les instances 3 et 4 — révèlent des zones de confusion où le modèle se trompe ou hésite.

**Pertinence pédagogique :** L'instance 3 est un exemple de confusion entre deux classes proches : *virginica* (réelle) est prédite comme *versicolor*. L'instance 4 est encore plus intéressante : bien qu'elle soit prédite comme *virginica*, ses perturbations montrent une forte instabilité locale (210 prédictions *virginica*, 40 *versicolor*). Cela révèle que l'instance se situe à la frontière de décision entre deux classes, ce qui la rend cruciale pour l'analyse locale et l'interprétabilité du modèle.

## 2.6 Modèle local et approximation du comportement du réseau

Pour analyser le comportement local du réseau de neurones autour d'une instance donnée, nous avons entraîné un modèle interprétable sur les 250 perturbations générées pour cette instance.

Nous nous intéressons ici à l'**instance 4**, mal classée par le modèle global. Ce choix est motivé par l'instabilité observée dans ses prédictions (210 fois *virginica*, 40 fois *versicolor*) qui indique une zone frontière entre deux classes.

Le modèle local est une régression logistique entraînée from scratch sur :

- Les **perturbations** comme entrées,
- Les **prédictions du réseau de neurones** comme étiquettes.

L'objectif est d'approximer le comportement du réseau dans cette zone locale. Le modèle linéaire ainsi appris fournit un vecteur de poids indiquant l'importance de chaque attribut dans la prise de décision locale.

## Performances et fidélité locale

Nous avons évalué la fidélité du modèle local en mesurant la proportion de cas où il prédit la même classe que le réseau de neurones sur les 250 perturbations. Dans le cas de l'instance 4, le modèle local atteint une fidélité de **environ 89%**, ce qui montre qu'il parvient à bien approximer les décisions du réseau dans cette région.

## 2.7 Graphique des contributions d'attributs

Le graphique ci-dessous représente les contributions des attributs pour l'instance 4. Il a été obtenu à partir du modèle explicatif local entraîné sur ses perturbations.



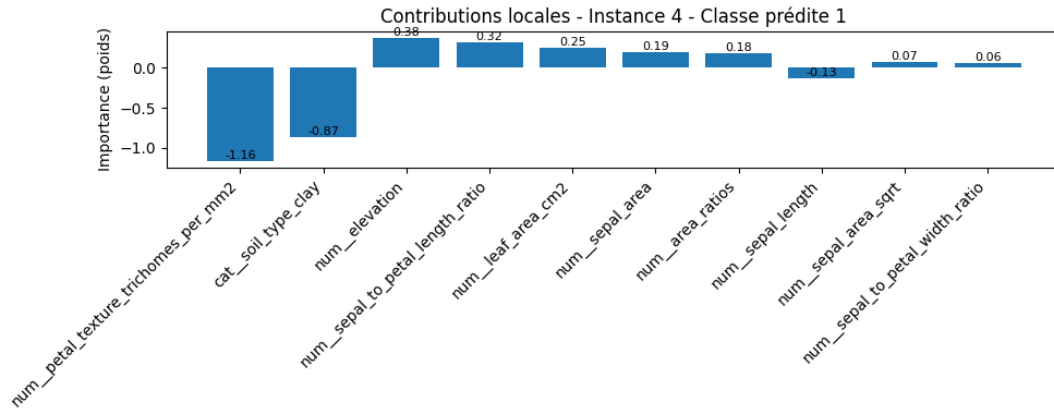


FIGURE 2.4 – Contributions locales des attributs – Instance 4

Ce diagramme permet de visualiser quels attributs ont le plus influencé la prédiction finale. Les attributs à poids négatif s’opposent à la classe prédite, tandis que ceux à poids positif y contribuent. Cela permet de mieux comprendre les décisions du modèle, en particulier dans les zones d’incertitude.

## 2.8 Influence de l’amplitude de perturbation

Nous avons comparé les contributions locales des attributs pour l’**instance 4** en utilisant deux amplitudes de bruit différentes lors de la génération des perturbations :

- Une première analyse avec un bruit de  $\pm 10\%$  autour de l’instance ;
- Une seconde analyse avec un bruit plus important de  $\pm 20\%$ .

Les graphiques ci-dessous montrent les contributions des attributs calculées par le modèle local dans chacun des cas :

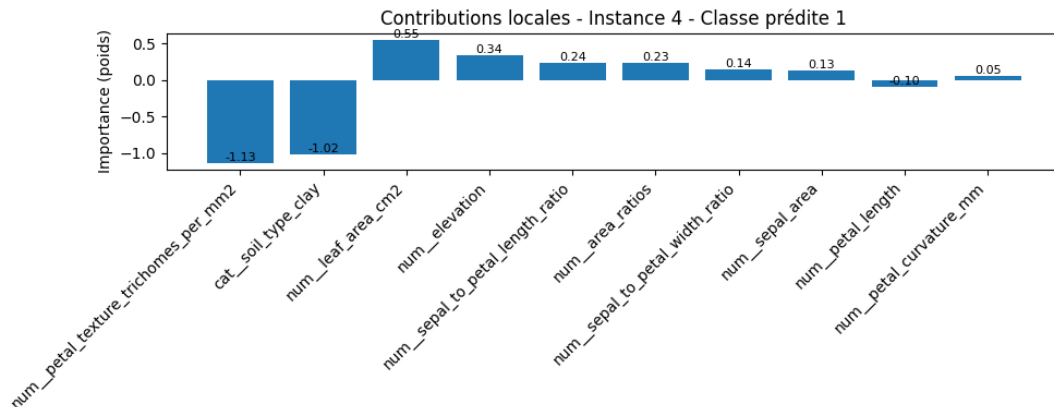


FIGURE 2.5 – Contributions locales avec perturbation  $\pm 10\%$  – Instance 4

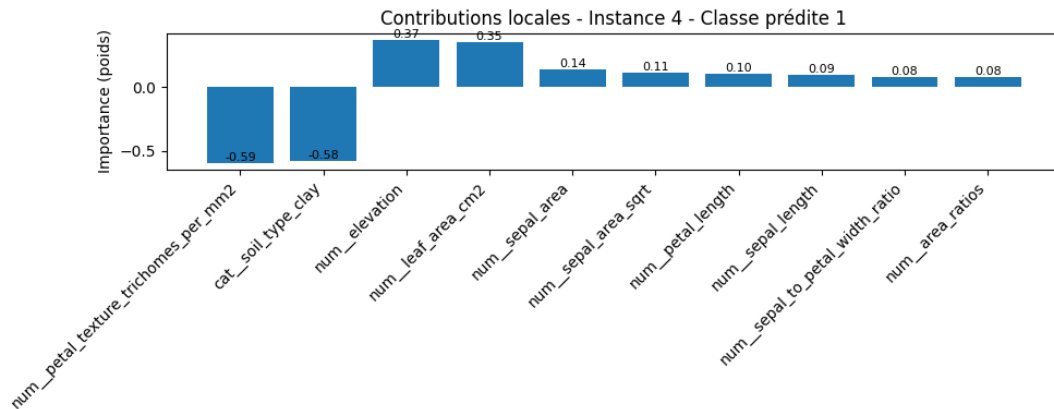


FIGURE 2.6 – Contributions locales avec perturbation  $\pm 20\%$  – Instance 4

### Analyse des différences :

- Certains attributs comme `petal_texture_trichomes_per_mm2` et `cat_soil_type_clay` conservent une contribution négative importante dans les deux cas, ce qui montre leur rôle stable.
- D'autres attributs voient leur contribution varier ou apparaître/disparaître selon l'amplitude choisie (ex. : `petal_curvature_mm` visible uniquement à  $\pm 20\%$ ).
- Globalement, l'ordre et l'importance relative des attributs changent légèrement : l'élargissement de la zone de perturbation fait intervenir des interactions différentes entre les variables.

**Conclusion :** Les contributions locales dépendent de l'amplitude du bruit appliqué aux perturbations. Un bruit plus faible ( $\pm 10\%$ ) donne une explication plus précise et centrée sur la zone immédiate autour de l'instance, tandis qu'un bruit plus fort ( $\pm 20\%$ ) permet de capturer des effets plus globaux mais peut lisser ou diluer certaines influences locales. Il est donc important de choisir l'amplitude en fonction du niveau de granularité souhaité pour l'analyse.

# Chapitre 3

## Conclusion Réflexive

### 3.1 Estimation directe des contributions

Dans un réseau de neurones, il est difficile de comprendre directement le rôle de chaque attribut dans une prédiction. Les couches multiples, les fonctions non linéaires comme `tanh`, et les interactions complexes entre les variables rendent l'analyse peu claire. C'est pourquoi on a besoin d'un modèle local plus simple pour estimer la contribution de chaque attribut de façon compréhensible.

### 3.2 Apport des approximations locales

L'approche locale permet de se concentrer sur une petite région de l'espace des données, autour d'une instance donnée. Dans ce voisinage restreint, le comportement du réseau peut être approché par un modèle simple comme une régression linéaire. Cette simplification permet d'associer à chaque attribut un coefficient expliquant son rôle dans la décision. C'est une manière concrète d'observer comment le réseau « raisonne » localement, et de détecter les zones où ses prédictions deviennent instables ou ambiguës.

### 3.3 Forces et faiblesses de l'approche locale

#### **Forces :**

- Applicable à tout modèle boîte noire, quelle que soit sa complexité.
- Offre une interprétation intuitive des décisions locales à travers des modèles simples.
- Permet d'identifier des attributs discriminants ou ambigus dans des zones spécifiques.

#### **Faiblesses :**

- Ne fournit qu'une explication locale : elle ne généralise pas à l'ensemble du modèle.
- Sensible à la manière dont les perturbations sont générées (bruit, échelle, distribution).
- Peut donner des explications peu fiables si le voisinage généré est trop hétérogène.

# Chapitre 4

## Conclusion Générale

Ce travail s’inscrit dans le cadre de l’analyse locale des décisions d’un réseau de neurones, avec pour objectif de mieux comprendre les prédictions d’un modèle complexe à travers des méthodes interprétables.

**Contexte et problématique.** Les réseaux de neurones sont des modèles puissants mais difficiles à interpréter. Dans un contexte où l’explicabilité est devenue cruciale (ex. : médecine, finance, éthique), il est essentiel de pouvoir justifier les décisions automatiques, notamment au niveau individuel.

**Limites des approches existantes.** Les poids internes du modèle ne permettent pas à eux seuls de comprendre les décisions. Les méthodes globales sont souvent inadaptées pour saisir les subtilités locales, notamment dans des zones où le modèle hésite.

**Approche proposée.** Nous avons mis en œuvre une méthode inspirée de LIME, consistant à :

- Générer des perturbations locales autour d’instances données ;
- Appliquer le modèle global (MLP) sur ces perturbations ;
- Entraîner un modèle explicatif local (régression logistique) pour estimer la contribution des attributs.

**Résultats et apports.** Cette approche nous a permis de :

- Visualiser clairement les attributs influents pour chaque décision ;
- Détecter les zones d’instabilité (proches des frontières de décision) ;
- Analyser la sensibilité des explications au paramètre de bruit utilisé pour la génération des perturbations.

**Limites.** L’approche reste locale : elle ne reflète pas le comportement global du réseau. Les résultats dépendent du nombre de perturbations, de l’amplitude du bruit, et du modèle explicatif choisi. De plus, certaines instances très stables apportent peu d’informations nouvelles.

**Pistes d’amélioration.** Parmi les prolongements possibles :

- Étendre l’analyse à d’autres types de modèles (arbres) ;
- Explorer d’autres méthodes d’explication ;
- Améliorer la robustesse des perturbations (e.g. perturbations structurelles ou générées par modèle de densité).

Ce projet montre qu’il est possible d’apporter de la transparence aux modèles neuro-naux en combinant apprentissage automatique et analyse locale. Il constitue une première étape vers des systèmes plus explicables et responsables.

# Chapitre 5

## Annexes

### 5.1 Structure du projet

L'organisation du projet est la suivante :

- `main.py` : point d'entrée principal du projet (entraînement, test, explication locale).
- `src/` :
  - `train.py` : fonctions d'entraînement du MLP.
  - `evaluate.py` : évaluation du modèle (précision, matrice de confusion, etc.).
  - `explain_local.py` : génération de perturbations et modèles explicatifs locaux.
  - `pre_process.py` : traitement des données et encodage des attributs.
- `results/models/` : sauvegarde du modèle entraîné.
- `results/visualisations/` : courbes de perte/précision, visualisations des contributions locales, etc.
- `data/` : fichier CSV utilisé dans les expériences.

### 5.2 Instructions de reproduction

1. Installer les dépendances
2. Lancer `python3 main.py`
3. Visualiser les sorties dans `results/`

### 5.3 Références

- Documentation officielle de scikit-learn : <https://scikit-learn.org/stable/documentation.html>
- Documentation officielle de PyTorch : <https://pytorch.org/docs/stable/index.html>
- NumPy : <https://numpy.org/doc/>
- Matplotlib : <https://matplotlib.org/stable/contents.html>