Master 2 MIAGE

Mémoire de fin d'études

Interprétabilité et explicabilité des modèles boites noires

Auteur :
Damien **JAIME**

Tuteur :
Emanuel **HYON**

Promotion 2019-2020

Remerciements

LISTE DES ACRONYMES

IA *Intelligence artificielle*

NN *Réseau de neurones (Neuronal Network)*

DNN *Réseau de neurones profond (Deep Neuronal Network)*

RGPD *Règlement Général sur la Protection des Données*

NLP *Traitement automatique du langage naturel (Natural Language Processing)*

SVM *Machine à vecteurs de support (Support Vector Machine)*

TE *Ensemble d'arbres (Tree Ensemble)*

TABLE DES MATIÈRES

INTRODUCTION

Nous sommes au quotidien en contact avec une multitude de boites noires, c'est-à-dire un système cachant sa logique interne à l'utilisateur. L'utilisateur ne pouvant pas savoir ce qu'il se passe dans le système, de nombreuses questions se posent. Notamment sur la confiance à accorder en ces systèmes, mais aussi d'un point de vue éthique particulièrement sur les questions de discrimination et de protection de la vie privée de l'utilisateur.

Avec l'avènement de l'intelligence artificielle dans l'aide à la prise de décision, la problématique de compréhension de ces systèmes est devenu un des enjeux majeurs de notre génération. En effet, comment peut-on être assuré que notre modèle est réellement fiable ? Notre modèle a-t-il involontairement héritée de biais présent dans ses jeux de données d'entraînement ? Quel degré de compréhension est-il nécessaire de fournir et pour quelle complexité ?
Ces questions sont très importantes dans le domaine médical par exemple où un faux négatif peut être très problématique, car à la différence des faux positifs il n'y a pas de vérification postérieure effectué par l'humain. Par ailleurs, nombreux sont les cas où une intelligence artificielle a hérité de biais involontaire présent dans ses jeux de données d'entraînement engendrant différents types de discriminations. Comme par exemple l'intelligence artificielle de recrutement de l'entreprise Amazon qui rejetait automatiquement les CVs contenant le mot "femme"[Das18].

Depuis 2016, en Europe, le Règlement Général de l'Union européenne sur la Protection des Données (RGPD) impose un droit à l'explication des décisions algorithmiques prise au sujet d'un utilisateur [Bry16]. Or les raisons des décisions de ces algorithmes étant très abstraites même pour la personne l'aillant créée, un problème se pose et les utilisateurs ne peuvent pas réellement utiliser ce droit à l'explication.

En plus des questions d'éthique et de protections des utilisateurs, expli-

quer ces boites noires serai d'une très grande aide pour les entreprises afin de créer des produits plus sûr, plus efficacement et les aider à mieux gérer les problèmes de responsabilité.
Pour toutes ces raisons, fournir une explication à un système de décision type boite noire est un sujet de parfaite actualité suscitant de nombreuses attentes et tables rondes. Le "Gartner hype cycle for Emerging Technologies" de 2019 place ce sujet (Explainable AI) dans le pic des attentes et suppose une maturité dans les cinq à dix ans.

L'objectif de ce mémoire est donc de comprendre comment un système boite noire et plus spécifiquement comment une intelligence artificielle est amenée à prendre une décision. Pour cela, nous allons analyser les différentes méthodes existantes dans la littérature et les appliquer à plusieurs cas concrets. Afin de savoir si expliquer ces systèmes boite noire nous permettrai de résoudre les problèmes au-quels ils sont confrontés et pour lesquels ils sont tant critiqués. Les applications porteront donc sur les questions d'éthique, de fiabilité et de performance.

Le reste de ce mémoire sera organisé en plusieurs parties. Tout d'abord, nous commencerons par contextualiser afin de bien comprendre les différentes problématiques et dérives. Nous enchaînerons par la suite avec un état de l'art divisé en deux parties : la première sera axée sur des définitions et présentations d'éléments conceptuels portant sur l'interprétabilité et de l'explicabilité, tandis que la seconde traitera des différentes méthodes et outils existant afin d'expliquer différents types de boite noire. La suite consistera donc à appliquer ces méthodes afin de d'évaluer leurs capacités à rendre un système de décision compréhensible par l'homme et de voir si ces explications permettent de palier aux problématiques décrites précédemment.

Chapitre 1

CONTEXTE : UN BESOIN D'EXPLICABILITÉ

Expliquer les modèles de décisions boites noires est un besoin se faisant de plus en plus ressentir et suscitant de nombreux débats et tables rondes dans la communauté scientifique. Les entreprises recherchent avant toute chose la performance et ce au détriment de la transparence des algorithmes utilisés. En effet dans le domaine du machine learning il existe une multitude d'algorithmes pouvant être utilisés afin de fournir une prédiction, mais il existe une corrélation non négligeable entre les performances et la transparence de notre modèle comme le montre la figure 1.1. En général, plus notre modèle est performant, plus il est difficile à comprendre.

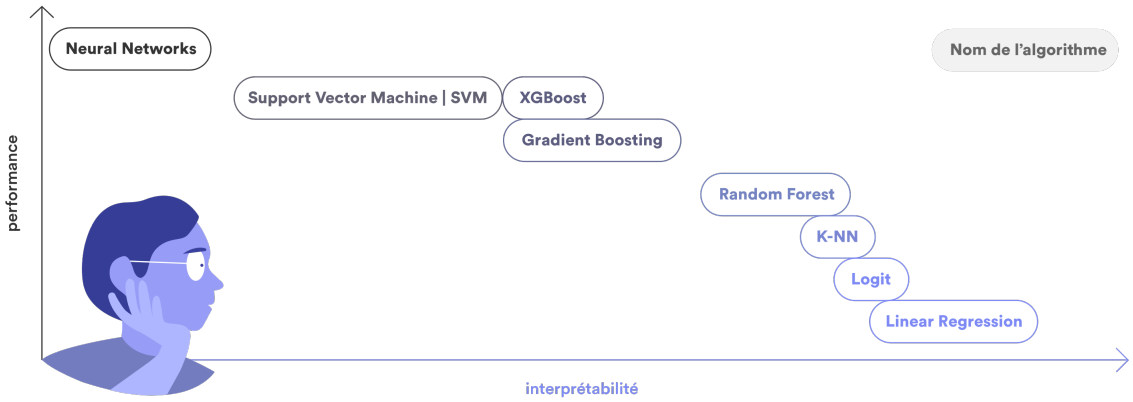


FIGURE 1.1 – Lien entre la performance et l'interprétabilité d'un algorithme de machine learning. *Source [Goo18]*

À l'heure actuelle, le réseau de neurones artificiel fait partie des algorithmes les plus performante et aussi est le plus couramment utilisée. La

non-explicabilité de ces modèles posent des problèmes de fiabilité, d'éthique et de responsabilité. Ce floue accentue aussi la méfiance et la peur des personnes envers l'intelligence artificielle, et entraîne un problème de compréhension entre les Data Scientist et les expert métier qui en plus de ne pas comprendre comment fonctionne le modèle ne comprennent pas non plus son résultat. Aussi, expliquer ces modèles permettrai d'améliorer notre conformité à la loi ainsi que d'augmenter les performances de nos modèles. Cette nouvelle problématique présente donc de nombreux enjeux de tailles.

1.1 L'éthique

Tout d'abord, de nombreux cas très exposés dans les médias nous montrent que cette incompréhension peut amener à des problèmes du point de vue éthique. La question de l'éthique est souvent ramenée à deux aspects principaux : la discrimination et la protection des données privées des usagers.

1.1.1 La discrimination

La question essentielle à se poser est de savoir comment une intelligence artificielle est amenée à prendre des décisions discriminante. Dans un article, Barocas et Selbst distinguent cinq façons par lesquelles une intelligence artificielle pourrait aboutir à une discrimination [Sol16]. Ces cas sont uniquement des cas involontaires et ne prennent pas en compte une discrimination délibérée qui pourrait bien évidemment être possible.

— **Définition des variables cibles et des étiquettes de classes** : le but d'une intelligence artificielle est de découvrir des corrélations dans des jeux de données. Ainsi, certain choix de variables cibles ou d'étiquettes de classes peuvent amener à faire des corrélations discriminatoire. Un exemple trivial serait de considérer un modèle permettant de savoir si un employé fait du travail de qualité, l'entreprise choisira de prendre en compte les retards des employés dans son modèle d'évaluation. Les personnes défavorisées habitant souvent loin de leur lieu de travail, ont plus souvent tendance à être en retard à cause des embouteillages et des problèmes de transport. Ces employés habitant loin, seront jugés comme faisant du travail de moins bonne qualité ce qui n'est pas forcément le cas. Si on ajoute à cela le fait que les personnes issus de l'immigration sont en moyenne plus pauvre et habitent en moyenne plus loin des centres villes, nous pouvons arriver involontairement à une IA discriminante par le simple fait d'utiliser l'étiquette "retard" dans l'évaluation des performances d'un employé.

- **Les données d'apprentissage** : les données d'apprentissage peuvent contenir des biais qui seront ensuite appris par notre modèle. L'exemple de l'IA de recrutement de l'entreprise Amazon qui rejetait les CVs contenant le mot "femme"[Das18] évoqué en introduction le montre bien. Cette IA avait enfaîte appris en analysant tous les profils recrutés par Amzon dans le passé, or les personnes recrutées étaient dans l'ensemble majoritairement des hommes et très majoritairement pour les postes technique. L'IA a donc déduis qu'il était préférable de recruter des hommes.
- **La collecte des données d'apprentissages** : Les lieux dans lesquels sont récoltées les données d'apprentissage sont aussi déterminant. Par exemple si l'on récolte des données concernant la criminalité dans un quartier avec des personnes issus majoritairement de l'immigration, l'IA aura plus tendance à considérer les immigrés comme de potentiels criminels par rapport aux personnes non-imigrées et inversement si on choisis un quartier avec des personnes majoritairement non-imigrées.
- **La sélection des caractéristiques** : Afin que l'IA puisse s'entraîner, il faut lui fournir des données qui sont en réalité une représentation simplifiée de notre monde. Ainsi, son créateur doit faire des choix pour sélectionner les caractéristiques qui constitueront cette représentation. Ce choix peut par concours de circonstance involontairement découler sur de la discrimination il faut donc les choisir avec parcimonie.
- **Données indirect** : Certaines données peuvent inclure des données indirect. Par exemple : *"un jeu de données qui ne contient pas de données explicites sur l'orientation sexuelle peut tout de même la dévoiler. Une étude de 2009 a montré que les liens d'"amis" sur Facebook révèlent par une méthode de prédiction précise de l'orientation sexuelle des utilisateurs de Facebook fondée sur l'analyse de leurs liens. Le pourcentage d'"amis" s'identifiant comme homosexuels serait fortement corrélé avec l'orientation sexuelle de l'utilisateur concerné"*[Bor18].

Mieux comprendre le fonctionnement de nos algorithmes permettrait donc de mieux se prémunir des discriminations involontaires. Aussi on pourrait prendre en compte l'explication en plus de la prédiction afin de fournir un résultat optimal. Par exemple, aux États-Unis, un algorithme appelé Compas est utilisé par les juges pour évaluer la probabilité pour un prévenu de se faire arrêter à nouveau dans les deux ans à venir. Cet algorithme est beaucoup remis en cause, car il est jugé "non-pertinant" et "discriminatoire" par certaines personnes alors que d'autres ont grandement confiance en lui, deux camps s'affrontent donc. Dans ce cas-là, fournir une explication conjointement à la prédiction permettrai de mettre tout le monde d'accord, ainsi nous pouvons prendre en compte la prévision et déterminer si dans le cas précis elle est

issue d'un jugement non-pertinant, discriminatoire ou autre.

1.1.2 Protection des données

L'apprentissage d'une intelligence artificielle requière un très grand nombre de données, parmi lesquelles peuvent se trouver des données personnelles. L'utilisation d'un modèle pourrait donc aussi nécessiter de fournir des données personnelles afin d'effectuer une prédiction. L'utilisation de ces données jugées critiques est problématique, car des restrictions et des règles de protection en découlent. Tel qu'évoqué dans l'introduction, le RGPD par l'addition de plusieurs de ses articles implique un "droit à l'explication". Ce droit a été démontré par Seth Flaxman et Bryce Goodman [Bry16], ce qui a donc pour effet d'accentuer le besoin d'applicabilité des modèles de décision boites noires. En effet, pour le moment ce droit est inaccessible par les usagers et les entreprises éprouvent des problèmes de responsabilités.

1.2 Fiabilité et confiance

Deuxièmement, expliquer les décisions prisent par un modèle boite noire permettrai d'augmenter la fiabilité que l'on lui accorde. Le modèle peut très bien fonctionner dans la plupart des cas, mais il est possible qu'il réagisse mal dans un certain cas bien spécifique. Avec l'absence de compréhension du fonctionnement interne de notre modèle, nous pouvons ne pas prévoir ce cas spécifique qui représente un risque. Le domaine médical par exemple est en attente d'une plus grande fiabilité de ses modèles en effet, une erreur peut avoir des conséquences graves et mettre en danger des personnes. La manière classique de mesurer la fiabilité d'un modèle est de calculer sa précision (accuracy), cela consiste à diviser le nombre de prédictions corrects par le nombre de prédiction total. Cette méthode indispensable ne permet pas à elle seule une fiabilité optimal en notre modèle, admettons que nous arrivons à une précision de 100% rien ne nous dis que dans un cas très particulier notre modèle ne nous donnera pas une prédiction fausse. La compréhension de la logique interne de notre modèle serai donc un facteur supplémentaire dans la fiabilité que l'on lui donne.

De plus, un soudage publié par OpinionWay[Opi17] montre que 30% des Français ont peur d'un jugement porté par l'intelligence artificielle dans le domaine financier et 21% dans le domaine médical. L'incompréhension de la logique interne des algorithmes décisionnels tend à accentuer la méfiance envers ces technologies. Fournir une explication permettrai d'augmenter l'acceptation de ces nouvelles technologies.

1.3 Performance

Enfin, expliquer la logique interne de la boite noire permettrai d'améliorer le développement de celle-ci, de la rendre plus compétitive et plus performante. Ces explications seront aussi utiles durant leur développement afin de comprendre pourquoi notre modèle ne fonctionne pas bien dans certains cas voir même pourquoi il ne converge pas du tout.

Chapitre 2

ÉTAT DE L'ART

2.1 Interprétabilité et explicabilité

Dans le domaine du machine learning, les mots interprétabilité et explicabilité sont très souvent utilisés conjointement et il peut être difficile au premier abords d'en saisir la différence. De plus, il existe de nombreuses définitions différentes pour ces termes et tous les experts ne s'accordent pas forcement à ce sujet. Après l'analyse de différentes propositions, nous nous accorderons, dans ce mémoire, à donner les définitions suivantes :

L'interprétabilité est le fait de pouvoir observer l'effet d'une cause dans un système c'est-à-dire de pouvoir prédire les changements dans la sortie lorsque l'on change une variable d'entrée.

L'explicabilité, quant à elle, est le fait d'expliquer dans des termes compréhensibles par l'homme la mécanique interne de notre système.

Afin de bien saisir la différence, le site *KDnuggets* dans un article[Gal18] nous invite à voir les choses comme si nous faisons une expérience scientifique à l'école : L'expérience peut être interprétable dans la mesure où vous pouvez voir ce que vous faites, mais elle n'est vraiment explicable qu'une fois que vous avez creusé la chimie derrière ce que vous pouvez voir se produire.

Créer un modèle interprétable implique de prendre en compte plusieurs facteurs :

Interprétabilité globale et local L'interprétabilité d'un modèle est dite *globale*, lorsque l'on comprend sa logique dans la totalité, nous sommes donc en mesure d'en expliquer toutes les solutions. À contrario, elle sera dite *local*, lorsqu'il est possible d'expliquer seulement une ou plusieurs solutions spécifiques. Un problème complexe pourra ainsi être découpé en un sous problème plus simple afin de pouvoir expliquer une partie des solutions comme le montre la figure 2.1 ci-dessous. Sur

cette figure, nous voyons à gauche la totalité du problème où les parties en bleu et en rouge correspondent aux domaines de prédiction de deux classes et où les croix et les ronds correspondent à des instances spécifiques de notre problème. On voit donc que le problème n'est pas linéaire et est assez complexe. Fournir une explication global à notre modèle de prédiction serait donc compliqué et pas forcement pertinent, or nous pouvons voir que la majorité des instances spécifiques se trouvent sur un problème simple et linéaire. L'explication local consiste donc à fournir une explication uniquement aux instances correspondantes à notre problème linéaire.

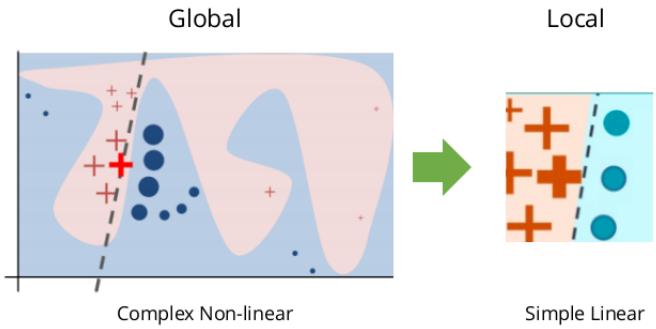


Figure 2.1 – Un problème global complexe pouvant être expliqué à échelle local. *Source : [MTR16]*

Limatation temporel Le temps que l'on peut allouer pour fournir une explication à notre modèle est aussi à prendre en compte. En effet, fournir une explication peut prendre du temps et cette explication peut être longue à appréhender pour un être humain. Dans certains contextes où la prise de décision devra être effectuée rapidement, il sera préférable d'avoir une explication simple, compréhensible et fournis rapidement par la machine. Dans d'autres cas, nous pourrons prendre le temps d'aborder une explication plus complexe et détaillée.

La cible de l'explication La nature de l'expertise de l'utilisateur est aussi un facteur à prendre en compte dans le choix de l'explication que nous voulons lui apporter. En effets, un expert dans le domaine aura tendance à préférer une explication exhaustive et précises qu'une explication simple et opaque et inversement pour une personne moins à l'aise.

Pré-requis d'un modèle interprétable :

En plus de prendre en compte les facteurs précédemment évoqués, un modèle interprétable doit être capable de satisfaire une liste de choses souhaitées. L'article "A survey of methods for explaining black box models"[RGu18] met en avant, après une analyse de différents états de l'art traitant de ce sujet, les desiderata d'un modèle interpretable :

Interpretabilité Dans quelle mesure le modèle ou la prédiction sont compréhensifs par l'homme. Ce sujet est encore en débat afin de savoir comment mesurer cette interprétabilité.

Précision Dans quelle mesure le modèle interprétable prédit avec précision les différentes instances. La précision d'un modèle peut être faite avec le *score de précision* (accuracy score), il s'agit simplement d'un rapport entre les observations correctement prédites et les observations totales. La précision est l'indicateur de base afin de calculer la précision d'un modèle, ce score fonctionne mieux si les faux positifs et les faux négatifs ont un coût similaire. Si le coût des faux positifs et des faux négatifs est très différent, il vaut mieux regarder le *F1-score*. Le F1-score est la moyenne pondérée de la précision et du rappel.

$$F1Score = 2 * \frac{Recall * Precision}{Recall + Precision}$$

Où la *précision* est le rapport des observations positives correctement prédites au total des observations positives prévues. Et le *recall* (rappel) est le rapport des observations positives correctement prédites à toutes les observations dans la classe réelle.

Éthique Si notre modèle traite des données personnelles, il devra garantir en plus une protection contre toutes formes de discrimination ainsi qu'une protection de la vie privée des personnes concernées.

Ces différents aspects jouent un rôle important quant à la confiance qu'un utilisateur va apporter à notre modèle interprétable. De plus d'autres notions viennent s'ajouter, notamment pour les modèles d'exploration de données et d'apprentissage automatique. Il est important de respecter des critères tels que la *robustesse*, la *causalité*, l'*évolutivité* et la *généralité*. Cela signifie qu'un modèle doit garantir un certain niveau de performance indépendamment des données d'entrée (robustesse), et que les changements d'entrée dû à une perturbation affectent le comportement du modèle (causalité). Enfin, étant donné que nous pouvons utiliser le même modèle avec une multitude données d'entrée et dans différents cas d'applications, il est préférable d'avoir

des modèles portables possédant un minimum de restriction (évolutivité, généralité).

2.2 Ouvrir les boites noires

Cette section se base principalement sur l'article "A survey of methods for explaining black box models"[RGu18] passe en revue cinquante-quatre méthodes aillant pour but d'ouvrir différentes boites noires, en nous fournissant une classification exposée dans un tableau disponible en annexe 1 et 2. Cette revue permet de distinguer différents types de problèmes, d'explicateurs, de boites noires, de données d'entrée ainsi que différentes restrictions sur le modèle. Mais elle est aussi complétée autre autre par le livre de Christoph Molnar "Interpretable Machine Learning" [Mol19] résumant l'essence de l'interprétabilité dans le machine learning et différentes autres ressources citées dans cette section. Nous allons commencer par expliquer ces différentes caractéristiques.

2.2.1 Types de problèmes

Nous pouvons distinguer deux grand types de problèmes. Le premier, la rétro-ingénierie, consiste à trouver une explication à un modèle déjà existant. Pour cela, nous allons utiliser des algorithmes externe à notre modèle afin de déduire une explication. Ce type de problème sera divisé en trois sous-problèmes : explication du modèle, explication du résultat et inspection de la boîte noire. Le second type de problème consiste à créer directement un modèle prédictif interprétable qui nous fournira une explication à ses résultats de lui même.

Explication du modèle

Ce problème consiste à fournir un modèle interprétable et transparent capable d'imiter le comportement d'une boite noire et de nous fournir un prédicat compréhensible par l'homme.
Étant donnée un prédicateur de boite noire b et un ensemble de données D, le problème d'explication du modèle consiste à trouver une fonction f telle que f(b,D)=c où c est un prédicateur compréhensible capable d'imiter le comportement de b et dérivable afin d'obtenir une explication.

Explication du résultat

Ce problème consiste à fournir un résultat interprétable, c'est-à-dire que le modèle devra fournir le résultat avec une explication sur les raisons qui l'ont poussé à donner cette prédiction. Il n'est pas nécessaire d'expliquer la logique interne du système mais seulement le processus de décision pour une instance donnée (interprétation local).

Inspection de la boîte noire

Ce problème consiste à fournir une représentation visuelle ou textuelle afin de comprendre le fonctionnement interne de notre boite noire.
Étant donnée un prédicateur de boite noire b et un ensemble de données D, le problème d'explication du modèle consiste à trouver une fonction f telle que f(b,D)=V où V est une représentation du fonctionnement de la boite noire.

Conception transparente

Ce problème consiste à créer un modèle prédictif transparent qui soit directement interprétable globalement ou localement. Il fournira donc de lui même une explication à ses prédictions sans que l'ont ai besoin d'utiliser des algorithmes externe à notre modèle.

2.2.2 Types d'explicateurs

Il existe différentes méthodes afin de fournir une explication à un modèle type boîte noire, nous allons dans cette sous-section présenter les principales.

- **Arbre de décision (Decision Tree)** : L'arbre de décision exploite un arbre qui a pour noeuds des conditions, les arcs correspondent à la valeur d'une variable d'entrée et les feuilles correspondent aux différents labels possible. La figure 2.2 montre un exemple trivial d'arbre de décision. Ainsi, la décision sera simplement expliquée en exprimant les chemins de l'arbre empruntés.
- **Règles de décision (Decion Rules)** : Utilisé pour expliquer le modèle, le résultat ainsi que pour la conception transparente. Il est aussi possible de transformer un arbre en un ensemble de règles. Les règles de décisions sont simplement des conditions IF-THEN :
  if condition1, condition2, condition3 then outcome
- **Importance des variables (Features Importance)** : Solution souvent utilisée, elle consiste à trouver les entrées de la boite noire pour lesquelles les poids sont les plus importants. Par exemple, pour une

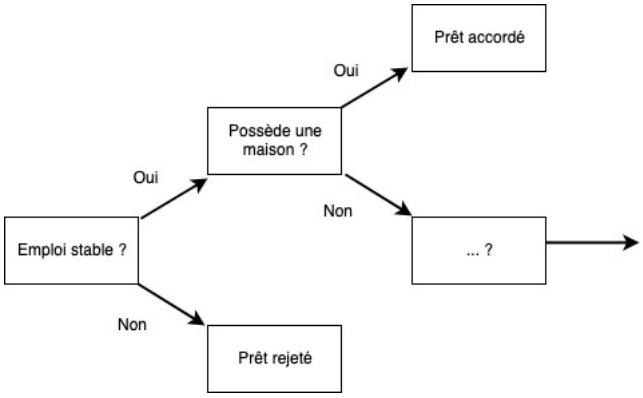


Figure 2.2 – Exemple d'un arbre de décision

classification d'image, de trouver les pixels les plus importants dans l'entrée pour en arriver à une prédiction. Comme le montre la figure 2.3 tirée de l'article *"Explainable Artificial Intelligence : Understanding, Visualizing and Interpreting Deep Learning Models"* [WSa17].

Figure 2.3 – Importance des fonctionnalités : explication de la prédiction "coq"

— **Salient Mask** Généralement utilisé pour expliquer localement les réseaux de neurones profond (DNN), le salient mask permet de mettre en évidence visuellement les parties déterminantes de l'entrée analysée. L'article *"Real Time Image Saliency for Black Box Classifiers"*[Pio17] décrit son fonctionnement et la figure 2.4 montrant un exemple de salient mask est tirée de cet article.
— **Analyse de sensibilité (Sensitivity Analysis)** : Généralement utilisée pour l'inspection de boite noire. L'analyse de sensibilité consiste

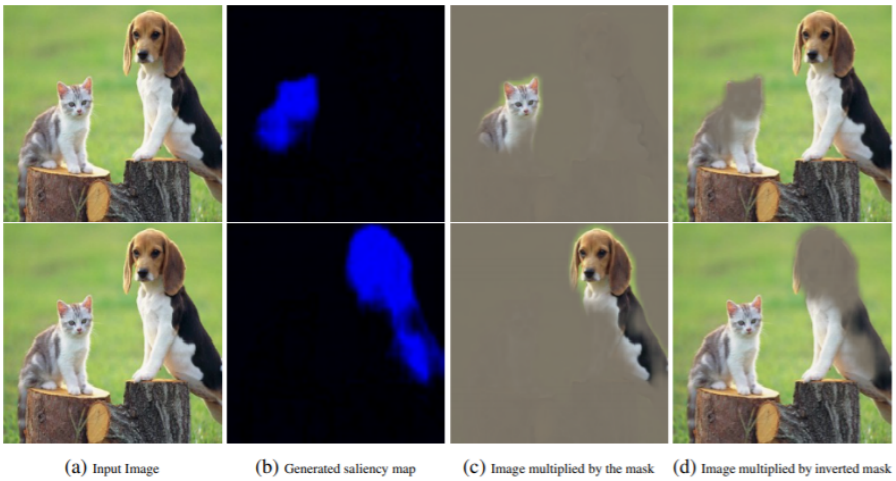


Figure 2.4 – Masque saillant : explication de la prédiction chien et chat

à évaluer l'incertitude statistique du résultat d'une boîte noire avec les différentes sources d'incertitude dans ses entrés. En d'autres termes, cela consiste à modifier des variables d'entrées afin de voir si cela a un impact sur le résultat en sorti et donc de savoir comment elles affectent notre prédiction.

— **Diagramme de dépendance partielle (Partial Dependence Plot)** : Ces graphiques permettent de comprendre l'effet d'une ou deux variables d'entrée sur la sortie du modèle. Le but étant de montrer si la relation entre une caractéristique d'entrée et la sortie est linéaire, monotone ou plus complexe. Par exemple, appliqué à un modèle de régression linéaire, les tracés de dépendance partielle montrent toujours une relation linéaire. Nous somme limité par une ou deux variables à la fois, car une variable donne une représentation en 2 dimensions du problème et deux variables fournissent donc une représentation 3 en dimensions. Par exemple, la figure 2.5 tirée du livre [Mol19] montre trois diagrammes de dépendances différents sur trois valeurs d'entrées (la température, l'humidité et la vitesse du vent) et leurs impactes linéaire sur la prédiction du nombre de vélos.

— **Sélection de prototype (Prototype Selection)** : Cet explicateur consiste à retourner, avec le résultat, un exemple très similaire à l'enregistrement classifié, afin de préciser avec quel critère la prédiction a été renvoyée.

— **Activation des neurones (Neurons Activation)** : L'analyse des réseaux de neurones permet aussi de comprendre son comportement. Cela consiste à analyser les neurones activés pour chaque entrés pas-

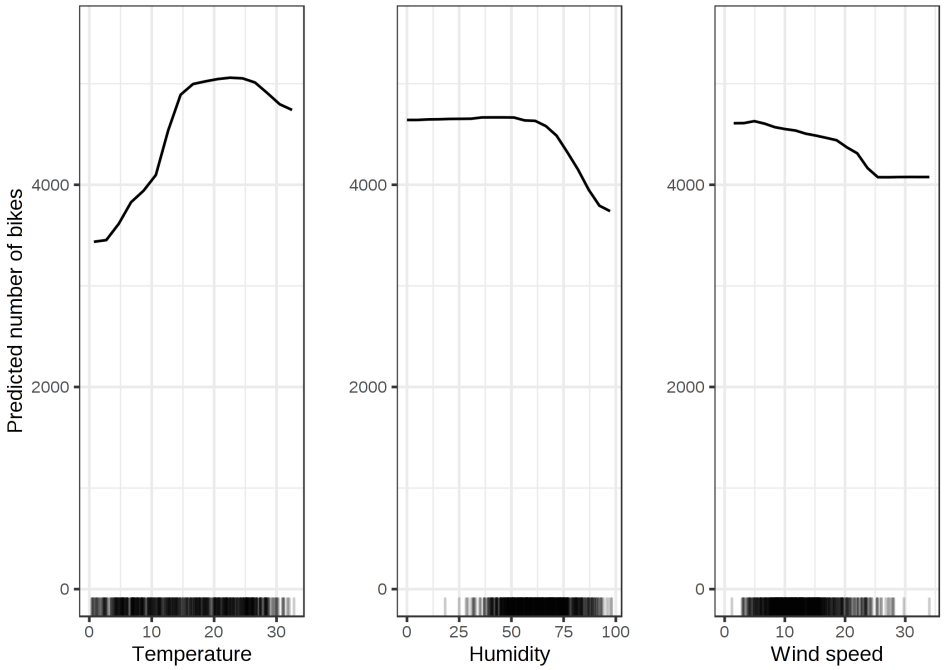


Figure 2.5 – Exemple de diagramme de dépendance partielle.

sées en argument à notre modèle.

L'explicateur varies en fonction de notre type de problème, de notre type de boite noire, de notre type de données d'entrée (images, texte...) et de nos différentes restrictions sur le modèle (accès au code, aux données...).

2.2.3 Spécificité de la boite noire

Le type d'explicateur à utiliser dépend aussi de notre boite noire et des données d'entrée utilisées. Nous distinguons principalement trois types de données d'entrées : les données tabulaires, textuelles et les images.

Données d'entrées

— **Tabulaire :** Dans une donnée tabulaire, chaque enregistrement partagent le même ensemble de caractéristiques qui sont de type textuel, numérique ou booléen. La donnée la plus couramment utilisée est le fichier csv.

— **Image :** Le modèle prévisionnel le plus couramment utilisé est la prédiction d'image, cela comprend aussi l'exploitation de vidéos. Pour pouvoir être traité, l'image est transformée en une liste nombres de

la taille de l'image (hauteur x largeur) où chaque nombre de la liste correspond à la couleur d'un pixel. Ces images peuvent être traitées telles quelles par la boite noire ou être pré-traitées afin qu'elles aient toutes la même taille ou leur appliquer un filtre par exemple.

— **Textuel :** Souvent utilisé pour le Traitement automatique du langage naturel (NLP), cette donnée est logiquement dépendante de la langue utilisée à l'entrainement de notre boite noire (un modèle entraîné en français ne pourra pas prendre un texte anglais en entrée). Un exemple serai un modèle de classification de sujet ou bien de détection de spam dans une boite mail.

Type de boites noires

Dans ce mémoire, nous nous concentrons sur l'apprentissage supervisé résolvant des problèmes de classification, les familles de boites noires énuméré dans cette sous-section répondent donc à ces critères et sont elles-mêmes découpées en sous-familles.

— **Réseau de neurones (NN) :** Un neurone artificiel (appelé perception) prend un certain nombre d'entrées et associe un poids à chacune afin de leur donner une importance (pouvant être positive ou négative). Le neurones nous donnera en sortie l'addition de chaque entrée multipliée par leurs poids respectifs, cette sortie correspond à la prédiction de notre neurone. L'apprentissage consiste donc à donner des exemples à notre neurone afin d'ajuster les poids associés aux entrées. Un réseau de neurones consiste à ajouter une couche intermédiaire de neurones (appelé layer) entre les données d'entrées et le neurone fournissant la prédiction. Cette couche a pour effet d'augmenter la complexité de notre modèle et donc de pouvoir détecter plus de paternes dans notre entrée.

— **Réseau de neurones profond (DNN) :** Un réseau de neurones profond est un réseau de neurones contenant une multitude de couches dans le but de résoudre des problèmes non-linéaire complexe. Il existe de nombreuse architectures différentes de réseaux de neurones profond, certains réseaux peuvent contenir des milliers voir même des millions de neurones. Nous en verrons un exemple dans la partie application.

— **Ensemble d'arbres (TE) :** Utilisé notamment en fouille de données et en apprentissage automatique, les méthodes d'ensemble combinent des algorithmes d'apprentissage afin d'améliorer le pouvoir prédictif des algorithmes qu'ils combinent. Ils combinent les prédictions de différents arbres de décision, chacun étant formé sur un sous-ensemble

indépendant des données d'entrée. Nous en verrons un exemple dans la partie application.

— **Machine à vecteurs de support (SVM) :** Couramment utilisé, les SVM reposent sur la distance entre la frontière de séparation et les échantillons les plus proches cette distance est appelée la marge. La frontière de séparation entre nos classes de prédiction est choisie comme celle qui maximise la marge, le problème est donc de trouver la frontière séparatrice optimale. Cette méthode permet donc uniquement de résoudre des problèmes linéaires, pour pouvoir traiter un problème non-linéaire le SVM va transformer l'espace de représentation des données d'entrées en un espace de plus grande dimension où il existe une probabilité de trouver une séparation linéaire.

Accessibilité de la boite noire

Le dernier facteur à prendre en compte pour choisir notre explicateur de boite noire est l'accessibilité de la boite noire. Il faudra donc se poser les questions suivantes :

Puis-je sonder le modèle autant de fois que je le souhaite ? Ai-je accès au code source du modèle ? Ai-je accès aux jeux de données qui ont servi à l'entraînement du modèle ? Le modèle va-t-il perturber mes données d'entrée

Par exemple si nous ne pouvons pas sonder la boite noire autant de fois que l'on désire, nous ne pourrons pas utiliser d'explicateur de type rétro-ingénierie. À l'inverse si nous n'avons pas accès au code du modèle, nous ne pourrons pas créer directement un modèle prédictif interprétable.

2.3 Approches utilisées

Face à ce besoin d'explicabilité grandissant, les entreprises peuvent préférer se tourner vers des modèles moins performant mais plus explicable afin de contourner la problématique de boite noire. Ainsi, nombreuse sont les entreprises qui se détournent de l'apprentissage profond au profil de l'apprentissage par arbre de décision ou les modèles linéaires par exemple qui fournissent un résultat plus compréhensibles et interprétable. Mais, la recherche dans ces domaines n'a pas évoluée depuis un certain temps et l'éventail des algorithmes disponibles est assez limité.

Sur-couche explicative

Le but étant d'essayer de fournir à l'utilisateur des éléments approximatifs permettant de comprendre son modèle boite noire, comme par exemple

identifier les variables d'entrée les plus importantes dans la prise de décision du modèle. Comme le montre la figure 2.6, la sur-couche explicative vient se greffer après la prédiction du modèle. Différents outils proposent une

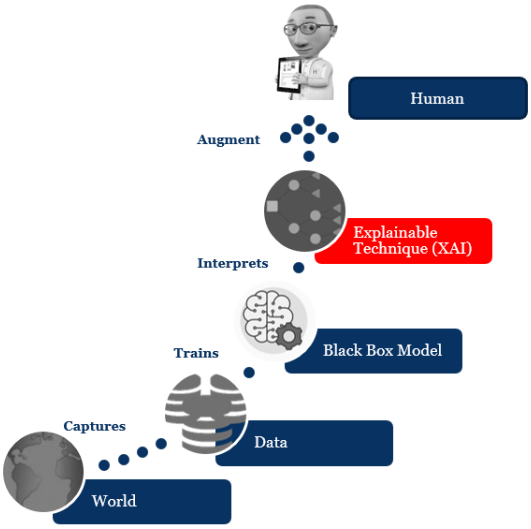


Figure 2.6 – Position de la sur-couche explicative dans le processus de prédiction. *Source : [Kau18]*

approche permettant de livrer des éléments de compréhension approximatif pour un modèle boite noire simple. Nous allons présenter les deux outils les plus populaires, LIME et SHAP.

2.3.1 LIME

LIME signifie Local Interpretable Model-Agnostic Explanations (explications locales interprétables par modèle-agnostique). L'objectif est donc de fournir une interprétation local à un modèle de classification ou de régression (model agnostic) en se basant sur l'importance des variables comme vu précédemment. L'idée est de sonder la boite noire autant de fois que nécessaire en perturbant la donnée d'entrée ce qui aura pour effet de produire un résultat différent à chaque fois. Pour cela, nous devons récupérer l'image d'entrée pour laquelle nous voulons expliquer la prédiction et définir une multitude de zones perturbations qui seront affichées ou non comme le montre la figure 2.7. À partir de ces zones perturbations, nous allons générer une multitude d'image en cachant ou non chaque zone et les donner en entrée à notre modèle (entre cinquante et cent-cinquante images dans la plupart des cas). Il

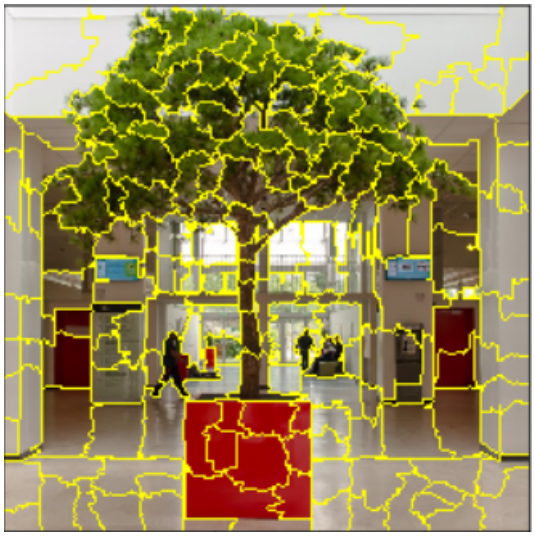
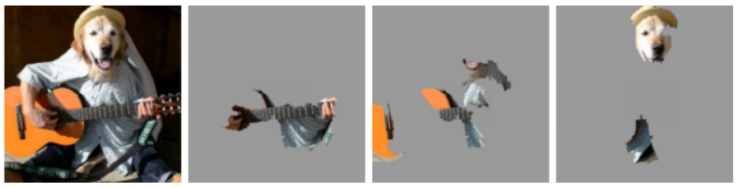


Figure 2.7 – Exemple de zones de perturbations sur une image

faudra donc stocker les prédictions effectuées par notre modèle pour chaque perturbation. Une fois que toutes les perturbations sont effectuées il est alors possible de récupérer les coefficients qui composent la droite de la régression linéaire et ainsi de déduire l'importance des variables d'entrées. Notre exemple porte sur une image, mais LIME accepte aussi des données d'entrée de type tabulaire ou textuel et le principe reste le même.

Par exemple dans la figure 2.8 tirée de l'article original de la présentation de LIME[MTR16] : L'image originelle (a) est donnée à notre boite noire qui prédit "Guitare électrique", "Guitare acoustique" et "Labrador". Nous constatons que le modèle s'est trompé sur la reconnaissance de la guitare électrique et nous utilisons LIME afin de comprendre ce qu'il s'est passé. LIME va prendre notre image (a) et la dériver de plusieurs façons en cachant certaines parties et envoyer ces nouvelles images à notre modèle. Le but étant de trouver les parties qui une fois cachées font que le modèle ne prédit plus la même chose. Puis une fois les trois classes trouvées, nous obtenons une explication où l'on voit les parties de l'image aillant joué un rôle dans la prédiction de chaque labels de notre modèle.

Une librairie Python est disponible afin d'utiliser l'algorithme LIME simplement sur nos modèles. Cette librairie a été créée par Marco Tulio Ribeiro qui est un des auteurs du papier original de LIME [MTR16], le code source de cette librairie est disponible sur le GitHub de son auteur [Rib16]. Cette librairie, disponible via le gestionnaire de paquets "pip", rend l'interprétation local d'un modèle très accessible, elle met à notre disposition entre autre trois méthodes majeures : LimeTabularExplainer, LimeImageExplainer et Lime-



(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 4: Explaining an image classification prediction made by Google's Inception network, highlighting positive pixels. The top 3 classes predicted are "Electric Guitar" ($p = 0.32$), "Acoustic guitar" ($p = 0.24$) and "Labrador" ($p = 0.21$)

Figure 2.8 – Source : LIME Paper [MTR16]

TextExplainer permettant de fournir une explication local à tout type de données d'entrées. Nous aurons, dans la partie application, l'occasion d'utiliser LIME afin de voir s'il pourrait servir à rendre nos modèles de classification plus fiable.

2.3.2 SHAP

Un an après la sortie de LIME, un nouvel outil fait son apparition afin de compenser entre autre le fait de ne pas pouvoir fournir une interprétation globale, cet outil s'appelle SHAP. SHAP signifie SHapley Additive exPlanations, c'est une méthode permettant de fournir une interprétation globale ou local à notre modèle. Cette méthode est basée sur la valeur de Shapley issus de la théorie des jeux, il est donc nécessaire dans un premier temps d'expliquer brièvement cette valeur de Shapley.

La valeur de Shapley introduit par Shapley en 1953, permet de repartir les gains équitablement dans un jeu coopératif. Le but étant que tous les joueurs coopérant ensemble reçoivent un certain gain en fonction de leur contribution.

Proposée par Lundberg et al. dans l'article [Sco19] faisant suite à l'article [Sco17], SHAP reprends l'idée de la valeur de Shapley afin d'expliquer toutes sortes de modèle de machine learning. Le but étant d'associer une valeur égale à la contribution dans la prédiction de chaque caractéristique d'entrées. SHAP commence donc par déduire la contribution des caractéristiques pour chaque instance et moyenne les résultats obtenus afin de donner la contribution globale de chacune de ces caractéristiques

Prenons par exemple un modèle permettant de prédire le prix d'un logement. Une multitude de caractéristiques sont données en entrés à notre modèle afin d'estimer le prix du logement, le but de SHAP est donc de définir pour chacune de ces caractéristiques leur impact monétaire sur le prix final du logement. Il interrogera donc la boite noire avec une multitude de

cas différents afin d'isoler le prix de chaque caractéristique. Prenons le cas de la figure 2.9 où l'on veut expliquer la prévision de 310 000 euros pour un logement de 50m2, au premier étage, près d'un parc et où les animaux sont interdits.

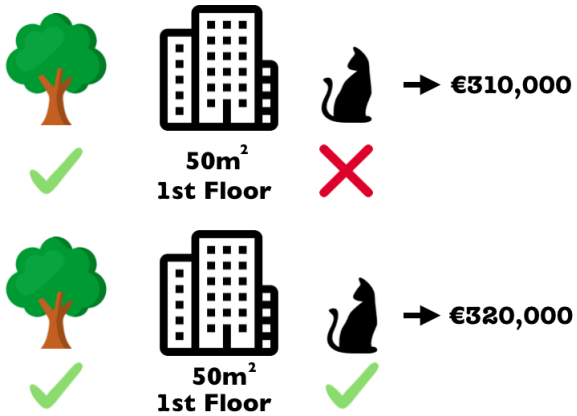


Figure 2.9 – Exemple de l'explication du coût d'un logement avec SHAP. *Source : [Mol19]*

SHAP va donc recréer la même l'instance et la donner en entrée à la boite noire en autorisant les animaux afin d'en déduire le coût (dans notre exemple 10 000 euros). Cette opération sera effectuée sur toutes les caractéristiques du problème afin de trouver la contribution de chacune pour chacune des instances de notre jeu de donnée. L'interprétation globale sera ensuite donnée en faisant la moyenne de ces contributions. Cela peut sembler trivial dans cet exemple mais il peut exister des dépendances entre les caractéristiques qui modifient leurs coût en fonction de la présence ou non d'une ou plusieurs autre(s) caractéristique(s).

Une librairie Python est disponible afin d'utiliser SHAP simplement sur nos modèles. Cette librairie a été créée par Scott Lundberg qui est un des auteurs des papiers originaux de SHAP cités plus haut, le code source de cette librairie est disponible sur le GitHub de son auteur [Lun18]. Cette librairie, disponible via le gestionnaire de paquets "pip", rend l'interprétation global et local et d'un modèle très accessible. Elle met à notre disposition différent types d'explicateurs permettant de fournir une explication global et local à différents types de boites noires et différents types de données d'entrées. Elle met aussi à notre disposition plusieurs graphiques permettant d'illustrer ses interprétations, ce qui simplifie la compréhension et la rend plus agréable. Nous aurons, dans la partie application, l'occasion d'utiliser la librairie python de SHAP et de voir des exemples de graphiques généré afin

de voir si SHAP peut servir à détecter des problèmes d'ordre discriminatoire.

2.3.3 Limites de ces implémentations

Aujourd'hui, de nombreuses implémentations de ces méthodes sont utilisées, mais elles essuient aussi quelques critiques et ne conviennent pas dans tous les cas de figures. Premièrement, ces algorithmes représentent un coût non négligeable. Ces méthodes basées sur des simulations et interrogeant notre modèle plusieurs fois pour une seule prédiction, peuvent poser des problèmes de performances lorsque l'on manipule un très grand nombre de données. Aussi la pertinence des explications fournies sont sujettes à débat, en effet ce sont des approximations et rien ne nous garantis que ces explications correspondent réellement au fonctionnement de notre modèle. Ces explications pourraient même donner l'effet inverse à celui escompté, en effet on peut arriver dans des cas où l'on fait confiance à notre modèle grâce à ces explications alors qu'elles ne sont pas fondées et ainsi prendre une décision critique appuyée sur une explication erroné.

Il est donc nécessaire d'utiliser ces algorithmes avec parcimonie et en aillant bien conscience qu'ils ne fournissent que des approximations de notre problème. Mais ce genre d'approches sont très bénéfiques pour la recherche, elles nous permettent d'apporter de nouvelles solutions ainsi que de mettre en lumière de nouvelles problématiques.

Chapitre 3

APPLICATION

Nous avons vu dans le chapitre 1 de ce mémoire que l'utilisation d'algorithmes de type boites noires dans l'aide à la prise de décision soulevait des questions, notamment sur l'éthique, la fiabilité et la performance. Dans cette partie, nous allons mettre en application nos recherches présentées dans l'état de l'art afin de déterminer si l'interprétabilité d'un modèle boîte noire nous permettrai de résoudre ces problèmes.

3.1 Mise en évidence de biais

L'objectif de cette partie est de démontrer que rendre un modèle boite noire interprétable permettrai de mettre en évidence des biais discriminatoire. Pour cela, nous allons créer un modèle prédictif à partir d'un jeu de données comportant possiblement des biais, puis d'utiliser l'outil SHAP afin d'essayer de mettre en évidence les éventuels biais présents dans notre jeu de données. Le code de cette section est disponible sur mon GitHub [Jai20c] et suit, entre autre, le tutoriel de "Towards Data Science"[Dat19]

3.1.1 Le jeu de données

Afin de savoir si utiliser une méthode de compréhension de boites noires aurai permis de mettre en évidence les biais présents dans l'intelligence artificielle dite sexiste de l'entreprise Amazon (évoquée dans l'introduction de ce mémoire), nous appliquerons ces méthodes sur un exemple similaire. Pour rappel, cette IA rejetait les CVs de femmes car elle avait été entraînée en se basant sur les personnes employées par l'entreprise dans le passé et que ces personnes étaient majoritairement des hommes, l'IA a donc conclue qu'il était préférable d'embaucher un homme plutôt qu'une femme.

Pour cette expérimentation, nous allons utiliser un jeu de données qui nous donne des informations sur des étudiants ainsi que leurs notes à différents examens. Les données d'entrée de notre modèle seront donc de type tabulaire. Ce jeu de données disponible ici [Kim12] ressemble à cela (Figure 3.1) :

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
2	female	group B	bachelor's degree	standard	none	72	72	74
3	female	group C	some college	standard	completed	69	90	88
4	female	group B	master's degree	standard	none	90	95	93
5	male	group A	associate's degree	free/reduced	none	47	57	44
6	male	group C	some college	standard	none	76	78	75
7	female	group B	associate's degree	standard	none	71	83	78

Figure 3.1 – Jeu de donnée des performances des étudiants

À noter, que ce jeu de données est fictif et généré aléatoirement. Notre but est de créer un modèle qui prédira la moyenne d'un étudiant en fonction de son sexe, de son ethnie, du niveau d'éducation de ses parents, de son régime alimentaire et de sa préparation aux tests. Pour ce faire, le jeu de données a été modifié en retirant les trois colonnes de scores afin d'en créer qu'une seule correspondant à la moyenne des trois. Nous obtenons donc le résultat présenté dans la figure 3.2 : Une fois notre modèle entraîné à prédire la moyenne

	gender	race/ethnicity	parental level of edu	lunch	test preparation course	average
2	female	group B	bachelor's degree	standard	none	72.66666667
3	female	group C	some college	standard	completed	82.33333333
4	female	group B	master's degree	standard	none	92.66666667
5	male	group A	associate's degree	free/reduced	none	49.33333333
6	male	group C	some college	standard	none	76.33333333
7	female	group B	associate's degree	standard	none	77.33333333

Figure 3.2 – Jeu de donnée final

potentielle d'un étudiant à partir de ces caractéristiques, on pourrai imaginer l'utiliser pour le recrutement universitaire. Mais (en dépit du fait que ces caractéristiques ne sont pas très pertinentes) notre modèle de recrutement est-il réellement fiable ? Pourrait-il être discriminant pour certaines ethnies ou pour un certain sexe ? Nous allons essayer de le découvrir en comprenant comment les caractéristiques d'entrée influent sur notre prédiction.

3.1.2 Création du modèle prédictif

La première étape sera donc de créer un modèle prédictif et de l'entraîner sur notre jeu de données. Pour ce faire, nous allons utilisé l'algorithme des

forêts d'arbres décisionnels (random forest) qui fait partie de la famille des ensembles d'arbres vu dans l'état de l'art partie 2.2.3. Cet algorithme reposant sur l'apprentissage par arbre de décision consiste à effectuer un apprentissage en parallèle sur de multiples arbres de décision construits aléatoirement et entraînés sur des sous-ensembles de données différentes selon le principe du bagging. Le baggin consiste à réduire la variance d'un arbre de décision afin de palier à son instabilité. Les prédictions de tous ces arbres sont ensuite moyennées afin de nous donner un résultat.

Afin de créer notre modèle, nous allons utiliser une bibliothèque Python destinée à l'apprentissage automatique appelée Scikit-learn (sklearn)[Cou10]. Cette bibliothèque a été initiée en 2007 par David Cournapeau en tant que projet Google Summer of Code qui est un programme organisé par l'entreprise Google où des étudiants travaillent pendant l'été sur un projet pour lequel l'étudiant a postulé. Open source et comptant de nombreux collaborateurs, cette bibliothèque mettant à notre disposition une multitude d'outils pour le machine learning est devenue la référence dans son domaine notamment grâce à sa complétude et sa simplicité d'utilisation.

Avec Scikit-learn, la création de forêts d'arbres décisionnels se fait à l'aide d'une classe appelée *RandomForestRegressor*, mais avant de l'utiliser nos données doivent être adaptées afin d'y être compatible. Dans un premier temps, il nécessaire d'enlever la colonne "average" car ce ne sera pas une caractéristique d'entrée mais la valeur à prédire (target). De plus, la fonction de forêts d'arbres décisionnels ne peux pas prendre en données d'entrée un tableau contenant des valeurs qualitatives (non-numérique). Afin d'être adapté à notre modèle, notre jeu de données a donc dû être encodé en créant une colonne par variables dans le but de n'avoir que des entrées numériques, par exemple la colonne "gender" devient deux colonnes "female" et "male" prenant 0 ou 1 comme valeur. Cette transformation des données est appelée "encodage one-hot" ou "encodage 1 parmi n". Pour ce faire, Scikit-learn met à notre disposition une classe : *"from sklearn.preprocessing import OneHotEncoder"* nous permettant de procéder simplement à cet encodage. Ainsi, pour notre jeu de données, nous passons de cinq colonnes d'entrée à dix-sept colonnes qui sont :

['x0_female' 'x0_male' 'x1_group A' 'x1_group B' 'x1_group C' 'x1_group D' 'x1_group E' "x2_associate's degree" "x2_bachelor's degree" 'x2_high school' "x2_master's degree" 'x2_some college' 'x2_some high school' 'x3_free/reduced' 'x3_standard' 'x4_completed' 'x4_none']

Où "x0" correspond à la transformation de la première colonne, "x1" à la transformation de la deuxième colonne et ainsi de suite.

Maintenant que nos données sont prêtes, nous allons créer notre modèle

à l'aide de la classe *RandomForestRegressor*. Pour ce faire, il nous suffit de l'initialiser en choisissant, pour notre cas, : la profondeur maximale des arbres (six dans notre cas), le nombre d'arbres dans la forêt (dix dans notre cas). Puis de l'entraîner à l'aide de la fonction *"fit"* prenant en argument les caractéristiques d'entrées et la colonne que l'on souhaite prédire (average dans notre cas).

3.1.3 Explication de notre prédiction

Notre modèle étant prêt, nous allons pouvoir nous demander s'il n'a pas été biaisé involontairement par notre jeu de données. Pour ce faire, nous allons utiliser l'outil d'interprétation Python SHAP vu dans notre état de l'art. La première chose à faire est de déterminer la valeur de Shapley de notre modèle. Pour ce faire, la librairie SHAP met à notre disposition une fonction prenant en paramètre notre modèle ainsi que nos données d'entraînement. Comme décrit dans l'état de l'art partie 2.3.2, cette fonction va sonder notre modèle de multiple fois en faisant varier les données d'entrées afin de déterminer l'impact de ces données sur notre prédiction.

Une fois notre valeur de Shapley obtenue, nous pouvons afficher différent graphique afin de mieux comprendre les raisons de nos prédictions. La figure 3.3 est un graphique généré par la fonction *"summary_plot avec pour type "bar""* de SHAP fournissant une explication globale de notre modèle de type importance des variables (features importance). L'importance des variables est calculée en moyennant la valeur absolue des valeurs de Shapley de chaque variable pour chaque exemple de notre jeu de données.

On voit donc sur notre graphique que la non-préparation aux tests et qu'un régime alimentaire équilibré sont les deux entrées aillent le plus d'impact sur notre prédiction, mais nous ne sommes pour le moment pas en mesure de savoir si cet impact est positif ou négatif.

Afin d'entrer plus dans le détail, SHAP nous offre d'autres graphiques comme par exemple la figure 3.4 toujours généré avec la fonction *"summary_plot"*. Sur ce graphique, chaque point représente une valeur de Shapley pour la variable pour chacune des instances. Les points rouges représentent un impact positif sur notre prédiction et les points bleu représentent un impact négatif. Nos données d'entrée étant constituées de valeurs binaires, ce graphique se lis de cette manière : si la valeur est positive et que la couleur est bleu alors la variable a un impact négatif sur notre valeur de sortie et inversement pour la couleur rouge. Par exemple la non-préparation au test et un régime alimentaire équilibré ont une influence négative sur notre moyenne. On peut aussi voir qu'être une femme a un impact positif alors qu'être un homme a un impact négatif. L'ethnie de type D sera aussi favorisée. Ces ré-

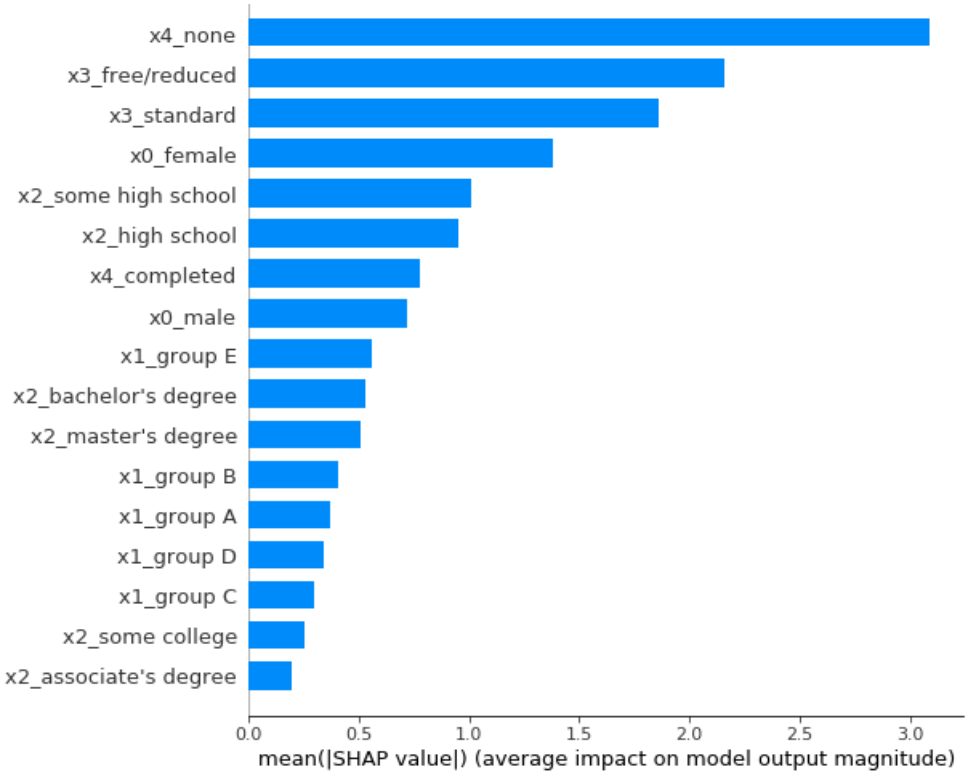


Figure 3.3 – Importance des variables d'entrées dans notre prédiction

sultats laissent penser qu'utiliser ce modèle pour le recrutement universitaire aura un effet discriminatoire pour les hommes qui seront défavorisés, ce qui ressemble à l'IA d'Amazon évoquée précédemment. Il nous est aussi possible de fournir une interprétation local c'est-à-dire de voir pourquoi notre boite noire a donnée un tel résultat pour une instance précise. La figure 3.5, généré avec la fonction *"dependence_plot"* de SHAP, nous montre un exemple d'interprétation local. On vois que pour cette instance notre modèle a estimé que l'étudiant aurai 60.62 de moyenne. Comme pour l'exemple précédent, les variables aillant un impact positif sont affichées en rouge et celles ayant un impact négatif sont affichées en bleu. On voit donc que pour cette instance l'entraînement à l'épreuve, le régime alimentaire et le sexe influence négativement notre moyenne alors que le niveau d'éducation des parents a une influence positive.

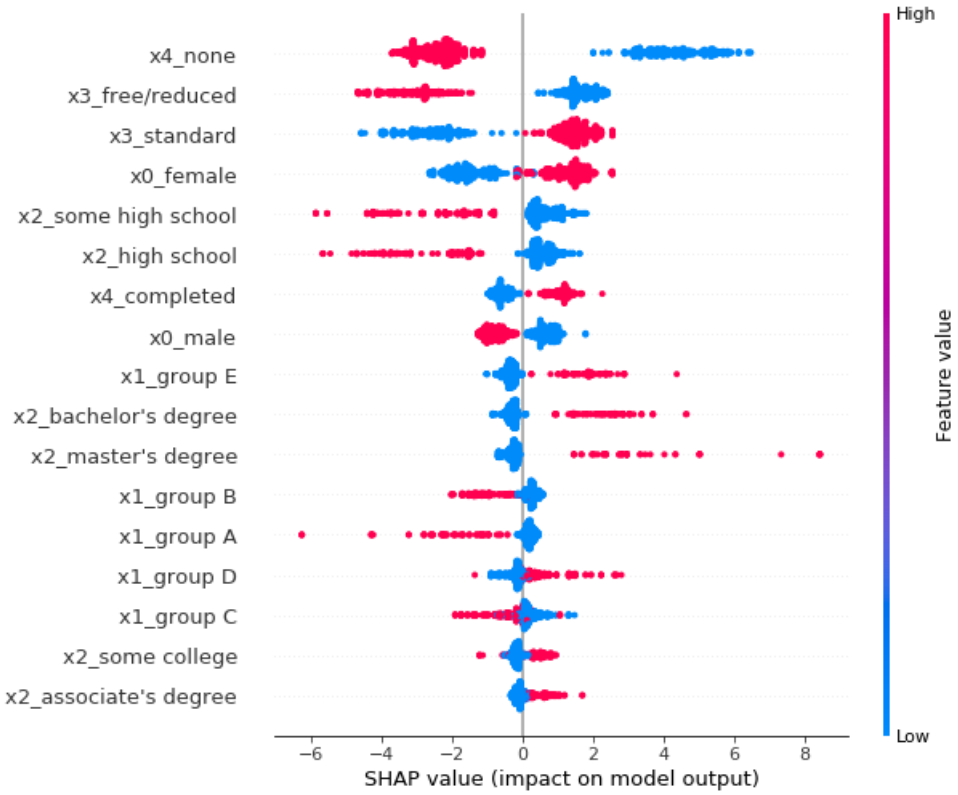


Figure 3.4 – Détail de l'explication de l'importance des variables

3.1.4 Bilan

Après l'inspection de notre modèle, nous avons pu voir qu'il jugeait les femmes plus performantes que les hommes et que son utilisation dans le recrutement universitaire aurai eut un impact discriminant sur les étudiants postulant. L'utilisation d'une méthode d'interprétabilité sur notre boite noire nous a donc permis de mettre en évidence un biais de discrimination présent dans notre jeu de donnée. Bien sûr, notre exemple étant trivial on aurai pu directement se rendre compte que les caractéristiques d'entrées n'étaient pas réellement pertinentes pour prédire la moyenne d'un étudiant.

Nous pouvons donc dire que qu'il est important de vérifier la pertinence des entrées de notre modèle, aussi utiliser un algorithme permettant que comprendre la logique interne de notre boite noire est important afin d'éviter tous problèmes.

Figure 3.5 – Explication de la prédiction pour une instance précise

3.2 Améliorer la fiabilité

L'intelligence artificielle pose aussi des problèmes de fiabilités, notamment dans le domaine médical. Afin de savoir si rendre un modèle interprétable permettrai d'améliorer sa fiabilité et de vérifier s'il se base sur les bons éléments pour donner ses prédictions, nous allons appliquer l'algorithme LIME vu dans l'état de l'art à un modèle issu du monde médical.

3.2.1 Le jeu de données

Pour cette expérimentation, nous allons utiliser un jeu de données contenant des images de radiographies. Le but sera de créer un modèle permettant de détecter si le patient de la radiographie est atteint de pneumonie. Ce jeu de données disponible ici [Moo18] est visible sur la Figure 3.6 : Le jeu de

Figure 3.6 – Exemple d'images présentes dans le jeu de donnée de pneumonie

données contient donc des radiographies de cages thoraciques saines ou malade. Dans la partie suivante, nous allons donc créer un modèle prenant en entrée une radiographie et nous donnant une prédiction sur l'état de santé du patient.

3.2.2 Création du modèle prédictif

Afin de prédire si la radiographie présente une pneumonie ou pas, nous allons créer un réseau de réseau de neurones à convolution (Convolutional Neural Networks). Cet algorithme, faisant partie de la famille des réseaux de neurones profond vu dans l'état de l'art parti 2.2.3, va ajouter des filtres et des couches à notre image d'entrée afin de pouvoir détecter plus de patterns sur notre image que le ferai un réseau de neurones classique. La figure 3.7 nous montre le fonctionnement d'une convolution, elle consiste en l'enchaînement de deux étapes : le "filtrage" et le "pooling" pour finir sur un réseau de neurones profond classique.

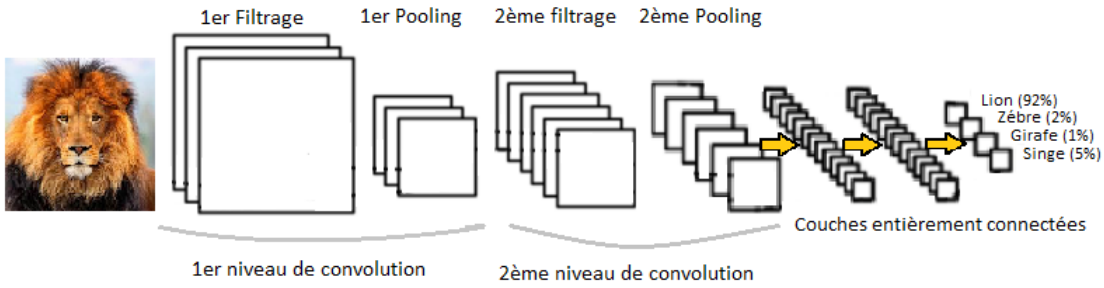


Figure 3.7 – Schéma expliquant la convolution. *Source : [Sim18]*

Le filtrage consiste à appliquer un filtre sur l'image afin de créer plusieurs images différentes à partir de l'image d'entrée, il permet de faire ressortir par exemple les contours, la couleur, la luminosité, etc... Ensuite, le pooling va prendre les images filtrées et vas les réduire en extrayant les valeurs importantes des pixels. Pour ce faire il va prendre, par exemple, un carré de trois pixel sur trois et le transformé en un seul pixel de la valeur du pixel le plus haut ou de la moyenne des pixels du carré. Afin d'y voir plus clair, la figure 3.8 nous schématise le fonctionnement du pooling. Ainsi, sur la figure, l'image qui faisait à l'origine neuf par neuf pixels en fait plus que sept par sept ce qui aura pour effet de diminuer les valeurs d'entrées et donc notre temps de calcul final.

Le code du modèle de cette application est disponible sur mon GitHub [Jai20a] et a été créé en Python à l'aide de TensorFlow 2.0.

TensorFlow[Ten15] est un outil d'apprentissage automatique initié par Google en 2011 devenu en 2015 open source et comptant de nombreux collaborateurs. TensorFlow met à notre disposition une API stable pour les langages Python et C++, et est devenu l'un des outils les plus utilisés en matière d'intelligence artificielle notamment pour la création d'architecture de réseau de neurones artificiels.

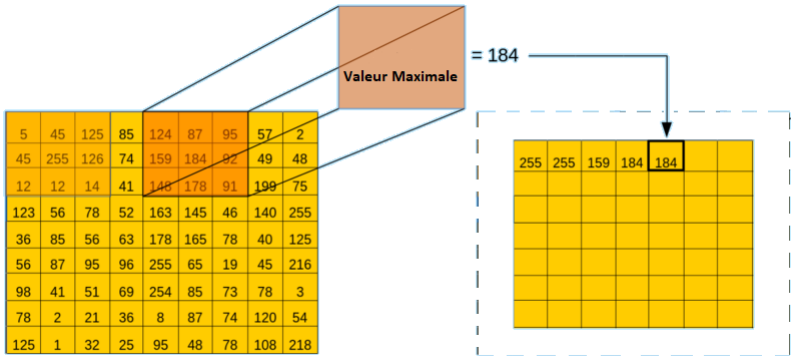


Figure 3.8 – Explication d'un pooling de taille trois par trois. *Source : [Sim18]*

Tensorflow va donc nous permettre de créer notre réseau à convolution pour la détection de pneumonie dont l'architecture est présenté sur la figure 3.9.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 203, 203, 32)	320
max_pooling2d (MaxPooling2D)	(None, 101, 101, 32)	0
conv2d_1 (Conv2D)	(None, 99, 99, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 49, 49, 64)	0
conv2d_2 (Conv2D)	(None, 47, 47, 64)	36928
flatten (Flatten)	(None, 141376)	0
dense (Dense)	(None, 64)	9048128
dense_1 (Dense)	(None, 2)	130

Total params: 9,104,002
Trainable params: 9,104,002
Non-trainable params: 0

Figure 3.9 – Architecture de notre modèle de prédiction de pneumonie.

Notre modèle prend en argument une image de dimension (205, 205, 1) les deux premières valeurs correspondent à la taille de l'image en pixel et la dernière valeur correspond aux canaux de couleur (1 pour notre cas car l'image est en noir et blanc). L'image vas donc passer dans le premier filtrage (conv2d) qui va nous donner une sortie de dimension (203, 203, 32), nous constatons donc que trente-deux filtres ont été créés sur l'image. Puis ces

trente-deux filtres passent par une phase de pooling pour au final avoir une dimension de (101, 101, 32), nous constatons donc que les images ont bien été réduites. On répète l'opération une deuxième fois puis on passe dans la couche "flatten" qui a pour effet de tout représenter sur une liste simple (141 376 correspondant donc à 47 x 47 x 64) afin d'être analysée par une couche de neurones. Cette liste est donc donnée à une couche de soixante-quatre neurones (dense) qui vont l'analyser et passer le résultat à une dernière couche contenant deux neurones (dense_1) où le résultat d'un neurone correspond à la probabilité que l'image d'entrée montre une pneumonie et l'autre neurone à la probabilité qu'elle n'en montre pas. Ainsi, la convolution nous a fait passer d'une image d'entrée de taille 42 025 (205 x 205, la taille de l'image) à une entrée de taille 141 376. Cela ajoute de la précision à notre modèle, mais a un temps de traitement et de calcul très conséquent.

3.2.3 Explication de notre prédiction

Notre modèle prêt, nous allons utiliser l'algorithme de LIME présenté dans notre état de l'art afin de vérifier la fiabilité de notre modèle. Dans cette partie, nous n'utiliserons pas la librairie LIME afin de pouvoir entrer plus dans le détail du fonctionnement de cette algorithme et de mieux le comprendre. Le code correspondant à l'implémentions de LIME est disponible sur mon GitHub [Jai20b] et suit, entre autre, le tutoriel de "Towards Data Science" [Art19]. L'explication sera locale et effectuée sur une seule image. Pour rappel, l'idée de LIME est de sonder la boite noire autant de fois que nécessaire avec différentes versions de notre image d'entrée qui aura subi de multiples perturbations.

La première étape consiste donc à créer des perturbations pour notre image. Pour ce faire, nous allons utiliser la librairie Skimage afin de générer 75 perturbations de notre image. La figure 3.10 montre le squelette de perturbation de notre image, chaque zone sera cachée ou affichée comme le montre la figure 3.11.

La seconde étape consiste à utiliser notre modèle créé précédemment afin d'effectuer une prédiction pour l'ensemble des perturbations générées. La prédiction ne doit pas nous donner le résultat, mais sa probabilité. Pour notre exemple la prédiction ne renverra donc pas "pneumonie", mais un tableau de deux valeurs avec la probabilité d'avoir que les poumons soient sains ou non "[0.23, 0.77]".

Une fois nos prédictions faites, nous allons évaluer la distance entre chaque prédiction et notre image de base. La distance correspond aux nombre de zones du squelette de perturbation de la figure 3.10 cachées. L'image de base aillant toutes les zones de perturbation visible.

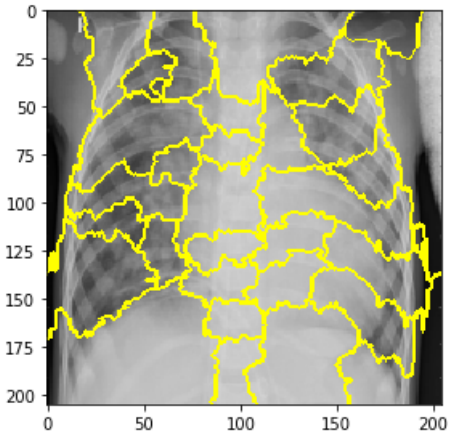


Figure 3.10 – Zones de perturbation

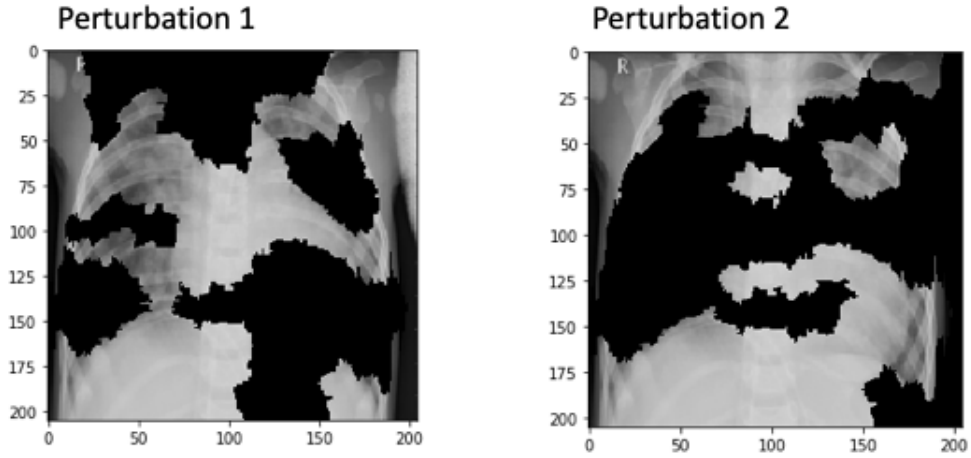


Figure 3.11 – Exemple de perturbations générées

Pour finir, nous créons un modèle linéaire que nous entraînons avec les informations obtenues dans les étapes précédentes afin d'obtenir un coefficient pour chacune des zones de perturbations représentant l'importance de la zone pour notre prédiction. Il ne nous reste donc plus qu'à trier ces coefficient afin d'afficher les quatre plus élevés. Pour notre exemple nous obtenons le résultat présenté en figure 3.12 sur l'image de gauche.

Afin d'y voir plus clair, nous pouvons superposer l'image obtenue avec l'image de base afin de voir la zone du poumon touchée par la pneumonie selon notre réseau de neurones comme le montre la figure 3.12 sur l'image de droite.

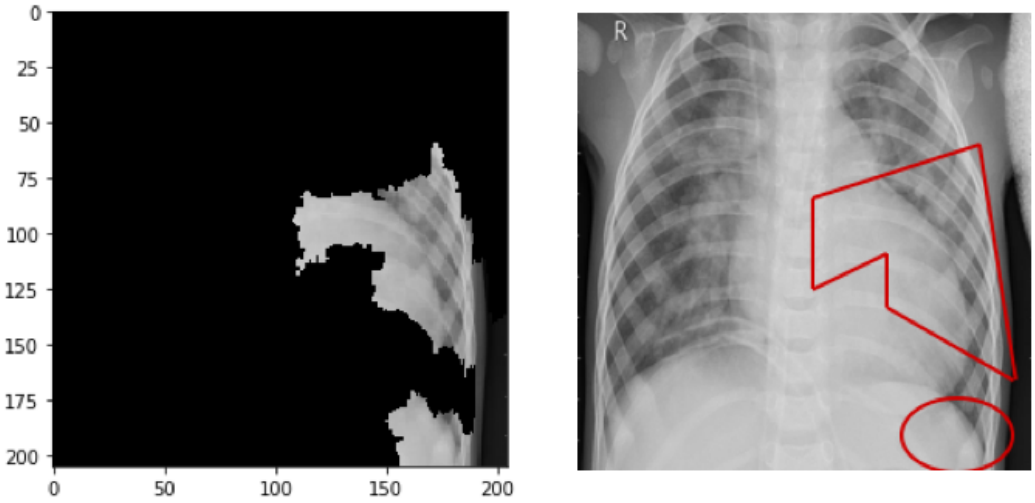


Figure 3.12 – À gauche : Zones de perturbations avec les coefficients les plus élevés À droite : Explication de la prédiction de pneumonie par LIME

3.2.4 Bilan

L'utilisation de l'outil d'interprétabilité nous a donc bien permit de déterminer quelles zones de notre image sont déterminantes pour notre prédiction. Dans notre exemple l'explication apporte un plus à la simple prédiction "pneumonie", car elle montre la zone la plus touchée par la maladie.

Accompagner chaque prédiction avec son explication serait un grand plus et améliorer grandement la fiabilité de nos modèles. Aussi, cela permettrai d'augmenter la confiance de la population à l'égard de ces technologies. Bien-sûr le but n'est pas de faire vérifier chaque prédiction par un médecin, cela serai contre-productif et l'IA permettant de faire gagner du temps aux médecins serait inutile. L'idée serait d'utiliser ces prédictions pendant les phases de développement et de test de notre IA afin de vérifier qu'elle évolue dans le bon sens et qu'elle détecte bien les pneumonies et pas autre chose.

3.3 Performance

Chapitre 4

ÉVALUATION DE L'APPORT

Chapitre 5

ÉVOLUTIONS POSSIBLES

CONCLUSION

Annexes

.1 Revue des méthodes explicatives

Name	Ref.	Authors	Year	Problem	Explanator	Black Box	Data Type	General	Random	Examples	Code	Dataset
Trepan	[20]	Craven et al.	1996	Model Expl.	DT	NN	TAB	✓				✓
-	[50]	Krishnan et al.	1999	Model Expl.	DT	NN	TAB	✓		✓		✓
DecText	[9]	Boz	2002	Model Expl.	DT	NN	TAB	✓	✓			✓
GPDT	[39]	Johansson et al.	2009	Model Expl.	DT	NN	TAB	✓	✓	✓		✓
Tree Metrics	[14]	Chipman et al.	1998	Model Expl.	DT	TE	TAB					✓
CCM	[23]	Domingos et al.	1998	Model Expl.	DT	TE	TAB	✓	✓			✓
-	[29]	Gibbons et al.	2013	Model Expl.	DT	TE	TAB	✓	✓			
STA	[114]	Zhou et al.	2016	Model Expl.	DT	TE	TAB		✓			
CDT	[87]	Schetinin et al.	2007	Model Expl.	DT	TE	TAB			✓		
-	[32]	Hara et al.	2016	Model Expl.	DT	TE	TAB		✓	✓		✓
TSP	[94]	Tan et al.	2016	Model Expl.	DT	TE	TAB					✓
Conj Rules	[19]	Craven et al.	1994	Model Expl.	DR	NN	TAB		✓			
G-REX	[37]	Johansson et al.	2003	Model Expl.	DR	NN	TAB	✓	✓	✓		
REFNE	[115]	Zhou et al.	2003	Model Expl.	DR	NN	TAB	✓	✓	✓		✓
RxREN	[6]	Augasta et al.	2012	Model Expl.	DR	NN	TAB		✓	✓		✓
SVM+P	[70]	Nunez et al.	2002	Model Expl.	DR	SVM	TAB			✓		✓
-	[28]	Fung et al.	2005	Model Expl.	DR	SVM	TAB			✓		✓
inTrees	[22]	Deng	2014	Model Expl.	DR	TE	TAB			✓		✓
-	[61]	Lou et al.	2013	Model Expl.	FI	AGN	TAB	✓		✓	✓	✓
GoldenEye	[33]	Henelius et al.	2014	Model Expl.	FI	AGN	TAB	✓	✓	✓	✓	✓
PALM	[51]	Krishnan et al.	2017	Model Expl.	DT	AGN	ANY	✓		✓		✓
-	[97]	Tolomei et al.	2017	Model Expl.	FI	TE	TAB			✓		
-	[108]	Xu et al.	2015	Outcome Expl.	SM	DNN	IMG			✓	✓	✓
-	[25]	Fong et al.	2017	Outcome Expl.	SM	DNN	IMG			✓		✓
CAM	[113]	Zhou et al.	2016	Outcome Expl.	SM	DNN	IMG			✓	✓	✓
Grad-CAM	[89]	Selvaraju et al.	2016	Outcome Expl.	SM	DNN	IMG			✓	✓	✓
-	[56]	Lei et al.	2016	Outcome Expl.	SM	DNN	TXT			✓		✓
LIME	[83]	Ribeiro et al.	2016	Outcome Expl.	FI	AGN	ANY	✓	✓	✓	✓	✓
MES	[98]	Turner et al.	2016	Outcome Expl.	DR	AGN	ANY	✓		✓		✓
NID	[71]	Olden et al.	2002	Inspection	SA	NN	TAB			✓		
GDP	[7]	Baehrens	2010	Inspection	SA	AGN	TAB	✓		✓		✓
IG	[92]	Sundararajan	2017	Inspection	SA	DNN	ANY			✓		✓
VEC	[16]	Cortez et al.	2011	Inspection	SA	AGN	TAB	✓		✓		✓
VIN	[35]	Hooker	2004	Inspection	PDP	AGN	TAB	✓		✓		✓
ICE	[30]	Goldstein et al.	2015	Inspection	PDP	AGN	TAB	✓		✓	✓	✓
Prospector	[48]	Krause et al.	2016	Inspection	PDP	AGN	TAB	✓		✓		✓
Auditing	[2]	Adler et al.	2016	Inspection	PDP	AGN	TAB	✓		✓	✓	✓
OPIA	[1]	Adebayo et al.	2016	Inspection	PDP	AGN	TAB	✓		✓		
-	[110]	Yosinski et al.	2015	Inspection	NA	DNN	IMG			✓		✓
TreeView	[96]	Thiagarajan et al.	2016	Inspection	DT	DNN	TAB			✓		✓
IP	[90]	Shwartz et al.	2017	Inspection	NA	DNN	TAB			✓		
-	[81]	Radford	2017	Inspection	NA	DNN	TXT			✓		
CPAR	[109]	Yin et al.	2003	Transp. Design	DR	-	TAB					✓
FRL	[102]	Wang et al.	2015	Transp. Design	DR	-	TAB			✓	✓	✓
BRL	[57]	Letham et al.	2015	Transp. Design	DR	-	TAB			✓		
TLBR	[91]	Su et al.	2015	Transp. Design	DR	-	TAB			✓		✓
IDS	[53]	Lakkaraju et al.	2016	Transp. Design	DR	-	TAB			✓		
Rule Set	[104]	Wang et al.	2016	Transp. Design	DR	-	TAB			✓	✓	✓
1Rule	[64]	Malioutov et al.	2017	Transp. Design	DR	-	TAB			✓		✓
PS	[8]	Bien et al.	2011	Transp. Design	PS	-	ANY			✓		✓
BCM	[44]	Kim et al.	2014	Transp. Design	PS	-	ANY			✓		✓
-	[63]	Mahendran et al.	2015	Transp. Design	PS	-	IMG			✓	✓	✓
-	[47]	Kononenko et al.	2010	Transp. Design	FI	-	TAB			✓		✓
OT-SpAMs	[103]	Wang et al.	2015	Transp. Design	DT	-	TAB			✓	✓	✓

FIGURE 1 – Tableau résumant l'ensemble des méthodes expliquant les boites noires présent dans la littérature. Description en annexe 2. Tiré de [RGu18]

.2 Description des méthodes explicatives

Feature	Description
Problem	Model Explanation, Outcome Explanation, Black Box Inspection, Transparent Design
Explanator	DT - Decision Tree, DR - Decision Rules, FI - Features Importance, SM - Saliency Masks, SA - Sensitivity Analysis, PDP - Partial Dependence Plot, NA - Neurons Activation, PS - Prototype Selection
Black Box	NN - Neural Network, TE - Tree Ensemble, SVM - Support Vector Machines, DNN - Deep Neural Network, AGN - AGNostic black box
Data Type	TAB - TABular, IMG - IMaGe, TXT - TeXT, ANY - ANY type of data
General	Indicates if an explanatory approach can be generalized for every black box, i.e., it does not consider peculiarities of the black box to produce the explanation
Random	Indicates if any kind of random perturbation or permutation of the original dataset is required for the explanation
Examples	Indicates if example of explanations are shown in the paper
Code	Indicates if the source code is available
Dataset	Indicates if the datasets used in the experiments are available

FIGURE 2 – Description du tableau présent en annexe 1. Tiré de [RGu18]

BIBLIOGRAPHIE

[Bor18] Frederik Zuiderveen Borgesius. « Discrimination,intelligence artificielle et décisions algorithmiques ». In : (2018).

[Bry16] Seth Flaxman Bryce Goodman. « European Union regulations on algorithmic decision-making and a "right to explanation" ». In : (2016).

[Mol19] Christoph Molnar. *Interpretable Machine Learning*. 2019.

[MTR16] C.Guestrin MT.Ribeiro S.Singh. « "Why Should I Trust You ?" Explaining the Predictions of Any Classifier ». In : (2016).

[Pio17] Yarin Gal Piotr Dabkowski. « Real Time Image Saliency for Black Box Classifiers ». In : (2017).

[RGu18] S.Ruggieri R.Guidotti A.Monreale. « A Survey Of Methods For Explaining Black Box Models ». In : (2018).

[Sco17] Su-In Lee Scott M. Lundberg. « Consistent feature attribution for tree ensembles ». In : (2017).

[Sco19] Su-In Lee Scott M. Lundberg Gabriel G. Erion. « Consistent Individualized Feature Attribution for Tree Ensembles ». In : (2019).

[Sol16] Andrew D. Selbst Solon Barocas. « Big Data's Disparate Impact ». In : (2016).

[WSa17] KR.Müller W.Samek T.Wiegand. « Explainable Artificial Intelligence : Understanding, Visualizing and Interpreting Deep Learning Models ». In : (2017).

WEBOGRAPHIE

[Art19] Cristian Arteaga. *Interpretable Machine Learning for Image Classification with LIME*. 2019. URL : https://towardsdatascience.com/interpretable-machine-learning-for-image-classification-with-lime-ea947e82ca13 (visité le 16/05/2020).

[Cou10] David Cournapeau. *Scikit-learn : Machine Learning in Python*. 2010. URL : https://github.com/scikit-learn/scikit-learn (visité le 18/05/2020).

[Das18] Jeffrey Dastin. *Amazon scraps secret AI recruiting tool that showed bias against women*. 2018. URL : https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G (visité le 16/05/2020).

[Dat19] "Dr. Dataman". *Explain Your Model with the SHAP Values*. 2019. URL : https://towardsdatascience.com/explain-your-model-with-the-shap-values-bc36aac4de3d (visité le 16/05/2020).

[Gal18] Richard Gall. *Machine Learning Explainability vs Interpretability : Two concepts that could help restore trust in AI*. 2018. URL : https://www.kdnuggets.com/2018/12/machine-learning-explainability-interpretability-ai.html (visité le 27/01/2020).

[Goo18] Data for Good. *Serment d'Hippocrate pour Data Scientist*. 2018. URL : https://hippocrate.tech/ (visité le 23/03/2020).

[Jai20a] Damien Jaime. *Classification of pneumonia case in chest x-ray using Tensorflow*. 2020. URL : https://github.com/DaJaime/CNN_pneumonia_detection (visité le 16/05/2020).

[Jai20b] Damien Jaime. *LIME explain - classification of pneumonia case in chest x-ray*. 2020. URL : https://github.com/DaJaime/LIME_pneumonia_explain (visité le 16/05/2020).

[Jai20c] Damien JAIME. *Use of SHAP to highlight discriminatory bias.* 2020. URL : https://github.com/DaJaime/SHAP_highlight_bias (visité le 16/05/2020).

[Kau18] Saurabh KAUSHIK. *Holy Grail of AI for Enterprise - Explainable AI.* 2018. URL : https://www.kdnuggets.com/2018/10/enterprise-explainable-ai.html (visité le 27/01/2020).

[Kim12] Royce KIMMONS. *Exam scores for students at a public school.* 2012. URL : http://roycekimmons.com/tools/generated_data/exams (visité le 20/04/2020).

[Lun18] Scott LUNDBERG. *SHAP : A game theoretic approach to explain the output of any machine learning model.* 2018. URL : https://github.com/slundberg/shap (visité le 16/05/2020).

[Moo18] Paul MOONEY. *Chest X-Ray Images (Pneumonia).* 2018. URL : https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia (visité le 20/04/2020).

[Opi17] OPINIONWAY. *OpinionWay pour VMware - Les Français et l'usage de l'Intelligence Artificielle.* 2017. URL : https://www.opinion-way.com/fr/ (visité le 27/01/2020).

[Rib16] Marco Tulio RIBEIRO. *Lime : Explaining the predictions of any machine learning classifier.* 2016. URL : https://github.com/marcotcr/lime (visité le 16/05/2020).

[Sim18] Florent SIMON. *Deep Learning, le réseau à convolution.* 2018. URL : https://www.supinfo.com/articles/single/8037-deep-learning-reseau-convolution (visité le 21/05/2020).

[Ten15] "TENSORFLOW". *An Open Source Machine Learning Framework for Everyone.* 2015. URL : https://github.com/tensorflow/tensorflow (visité le 18/05/2020).

TABLE DES FIGURES

Université Paris-Nanterre
200 Avenue de la République
92000 Nanterre