POLYPUBLIE
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Améliorer la confiance envers les outils d'IA pour sécuriser les communications avioniques avec l'IA eXplicable
Auteur: Author:	Charles de Malefette
Date:	2025
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	de Malefette, C. (2025). Améliorer la confiance envers les outils d'IA pour sécuriser les communications avioniques avec l'IA eXplicable [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/67093/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/67093/
Directeurs de recherche: Advisors:	Gabriela Nicolescu
Programme: Program:	Génie informatique

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Améliorer la confiance envers les outils d'IA pour sécuriser les communications avioniques avec l'IA eXplicable

CHARLES DE MALEFETTE
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Juillet 2025

© Charles de Malefette, 2025.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Améliorer la confiance envers les outils d'IA pour sécuriser les communications avioniques avec l'IA eXplicable

présenté par **Charles DE MALEFETTE**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Martine BELLAICHE, présidente
Gabriela NICOLESCU, membre et directrice de recherche
Ranwa AL-MALLAH, membre

REMERCIEMENTS

A ma directrice de recherche Mme. Gabriela Nicolescu, merci pour ton accompagnement et ta gentillesse lors de ces deux années de maîtrise. A l'attaché de recherche Jean-Yves Ouattara, merci pour ton soutien, tes nombreux conseils et encadrements et pour ton expertise qui m'ont été indispensable pour cette maîtrise. Au professeur Adel Abusitta, merci de m'avoir introduit au domaine d'XAI qui m'a particulièrement intéressé et a donné naissance à mon sujet de maîtrise. A tous mes camarades du HESL, merci de m'avoir accompagné lors de mon parcours à Polytechnique.

RÉSUMÉ

L'intégration de l'Intelligence Artificielle (IA) dans les systèmes avioniques, en particulier pour la détection d'intrusions (IDS), soulève un défi majeur : concilier performance algorithmique et transparence décisionnelle. Alors que le marché des systèmes avioniques intelligents devrait atteindre 10 milliards USD d'ici 2030, et que 78% des compagnies aériennes utilisent déjà des outils de cybersécurité intelligents, seulement 11% exploitent pleinement leurs données pour entraîner des modèles d'IA. Ce fossé s'explique par l'opacité des modèles complexes (LSTM, Transformers), qui, malgré leur précision, restent perçus comme des boîtes noires dans un domaine où 42% des erreurs humaines proviennent de décisions mal comprises.

Ce mémoire explore comment l'IA eXplicable (XAI) peut renforcer la confiance dans les IDS avioniques en fournissant des explications interprétables aux opérateurs humains. L'utilisation de l'XAI dans un tel contexte est encore inédite et cette recherche vise à approfondir les tenants et les aboutissants d'une telle application. Nous proposons d'abord un cadre conceptuel structurant plusieurs concepts clés pour guider l'implémentation de l'XAI en contexte avionique. Ensuite, nous présentons un formalisme mathématique unifié évaluant des techniques comme SHAP et LIME, théoriquement stables et précises et adaptées aux contraintes de temps réel. Enfin, nous développons une méthodologie d'intégration testée sur deux scénarios puis évaluons sa viabilité à l'aide de métriques adéquates. Nos applications fournissent d'excellents résultats tant sur la capacité à obtenir des explications approfondies sur le fonctionnement du modèle, que sur le respect de celles-ci des critères d'évaluation exigés comme la robustesse ou la stabilité. L'utilisation de l'XAI peut effectivement améliorer la confiance dans les IDS dans un contexte avionique. Cependant, notre travail révèle également que les explications générées ne sont interprétables et utilisables que par des experts en analyse de données, ou dans le domaine considéré (avionique, cybersécurité). Ces explications nécessitent ainsi une analyse a posteriori avant de pouvoir être effectivement transmises aux utilisateurs comme les pilotes. Cette recherche ouvre la voie à un approfondissement et à une standardisation de l'XAI pour l'avionique, pour pouvoir répondre aux besoins opérationnels tout en fournissant des résultats utiles et justes.

ABSTRACT

Integrating Artificial Intelligence (AI) into avionics systems, particularly for intrusion detection (IDS), raises a major challenge: reconciling algorithmic performance with decision-making transparency. While the market for intelligent avionics systems is expected to reach 10 billion USD by 2030, and 78% of airlines are already using intelligent cybersecurity tools, only 11% are fully exploiting their data to train AI models. This gap is explained by the opacity of complex models (LSTM, Transformers), which, despite their accuracy, remain perceived as black boxes in a field where 42% of human errors stem from poorly understood decisions.

This thesis explores how eXplainable AI (XAI) can enhance trust in avionics IDS by providing interpretable explanations to human operators. The use of XAI in such a context is still unprecedented, and this research aims to explore the ins and outs of such an application. We first propose a conceptual framework structuring several key concepts to guide the implementation of XAI in an avionics context. Next, we present a unified mathematical formalism evaluating techniques such as SHAP and LIME, which are theoretically stable and accurate and adapted to real-time constraints. Finally, we develop an integration methodology tested on two scenarios, then assess its viability using appropriate metrics. Our applications deliver excellent results, both in terms of their ability to provide in-depth explanations of how the model works, and in terms of their compliance with required evaluation criteria such as robustness and stability. The use of XAI can indeed improve confidence in IDS in an avionics context. However, our work also reveals that the explanations generated can only be interpreted and used by experts in data analysis, or in the domain under consideration (avionics, cybersecurity). These explanations thus require a posteriori analysis before they can be effectively transmitted to users such as pilots.

This research paves the way for further development and standardization of XAI for avionics, to meet operational needs while providing useful and accurate results.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

AI	Artificial Intelligence
XAI	eXplainable Artificial Intelligence
IDS	Intrusion Detection System
ADS-B	Automatic Dependent Surveillance Broadcast
MIL-STD	Military Standard
ARINC	Aeronautical Radio, Incorporated
SHAP	Shapley Additive exPlanation
G-DeepSHAP	Generalized Deep SHAP
LIME	Local Interpretable Model-agnostic Explanations
DeepLIFT	Deep Learn Important FeaTures
FPR	False Positive Rate
RNN	Recurrent Neural Network
ANN	Artificial Neural Network
LSTM	Long Short-Term Memory
SVM	Support Vector Machine

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

1.1 Contextualisation

L'Intelligence Artificielle (IA) a permis des avancées majeures dans les systèmes avioniques et la sécurité informatique, avec un marché des systèmes avioniques intelligents évalué en 2024 à 1,6 milliard USD, et projeté à près de 10 milliards USD d'ici 2030 [2]. Les Systèmes de Détection d'Intrusion (IDS), conçus pour identifier les cyberattaques ciblant les bus de communication avioniques, visent à protéger les bus de communication vulnérables comme l'ADS-B [3] ou le MIL-STD-1553 [4], utilisant notamment des outils d'IA. A ce titre, la sécurité informatique intelligente est jugée par 76% des compagnies aériennes et des aéroports comme étant la priorité numéro une avec 78% des compagnies en faisant déjà une utilisation régulière, et plus de 50% des investissements qui lui sont consacrés (selon SITA 2024 [5]). Cependant, la complexité des modèles d'IA et leur aspect probabiliste, limitent leur adoption dans des domaines où la transparence décisionnelle est essentielle [6]. Seulement 11% des aéroports utilisent stratégiquement les données avioniques récoltées pour entraîner des modèles d'IA, là où 86% sont encore en phase de récolte, sans utilisation concrète [5]. Ce fossé illustre le défi majeur de cette recherche : concilier performance algorithmique et transparence décisionnelle dans un domaine où les erreurs de décision représentent 42% des erreurs humaines [7].

L'IA eXplicable (XAI) émerge pour tenter de combler ce fossé en rendant les prédictions interprétables, favorisant une collaboration homme-machine plus transparente. Son objectif pour les modèles appliqués à l'avionique est d'améliorer la **confiance** des acteurs du secteur (pilotes, ingénieurs) envers les décisions automatisées [8]. L'XAI trouve de plus en plus sa place dans l'aviation mais reste aujourd'hui encore uniquement en phase de recherche avec aucun déploiement concret à ce jour [9].

1.2 Éléments de la problématique

L'intégration de l'IA dans les IDS avioniques soulève des défis majeurs liés à sa complexité, son opacité et son caractère probabiliste. Bien que les modèles modernes (LSTM, Transformers) offrent une haute précision, leur nature de « boîte noire » les rend difficilement auditables dans un domaine où la sécurité critique exige une transparence totale. Par exemple, un IDS basé sur un réseau de neurones profond peut détecter une anomalie dans un message MIL-STD-1553 avec plus de 99% de confiance [10], mais sans explication compréhensible, les ingénieurs ne peuvent ni valider ni agir en conséquence.

Les techniques d'XAI tentent de combler ce fossé, mais leur application aux protocoles avioniques reste inédite. Les contraintes temps-réel (délais <100 ms) limitent l'usage de méthodes coûteuses en calculs comme SHAP [11], tandis que la complexité des données (séries temporelles bruitées, dépendances contextuelles) exige des adaptations spécifiques. Par ailleurs, les attaques ciblées telles que les *adversarial injections* exploitent les failles des modèles de ML, rendant cruciale une explicabilité robuste pour éviter des explications trompeuses.

Enfin, l'absence de cadre méthodologique unifié en XAI pose un problème de standardisation. Comment traduire des sorties techniques (valeurs SHAP et LIME [12]) en alertes actionnables pour un pilote ? Comment gérer le fait que ces mêmes valeurs utilisent un formalisme théorique différent encore non abouti, et en quête de définition et de robustesse ? L'XAI peut-elle surmonter ses propres limites théoriques (pluralité des formalismes, robustesse inachevée) pour rendre l'IA véritablement fiable en avionique ?

Cette recherche vise donc à résoudre d'une part la tension entre performance et interprétabilité des modèles d'IA, en proposant un cadre conceptuel et théorique adapté aux spécificités avioniques. D'autre part, elle vise à trouver une méthodologie d'utilisation et d'évaluation pratique d'un tel cadre.

1.3 Objectifs de recherche

L'objectif de cette recherche est d'intégrer de l'XAI aux systèmes de sécurité informatique appliqués aux protocoles de communication avioniques. Le but étant de faire tomber le voile sur la complexité des nouveaux modèles d'apprentissage profond tels que ceux utilisés dans les IDS, afin de les rendre plus acceptables, manipulables et compréhensibles par leurs futurs utilisateurs particulièrement en aéronautique et avionique. Dans cette ligne de pensée, la question de recherche peut être formulée ainsi :

Comment l'XAI peut-elle adapter les IDS avioniques en systèmes dignes de confiance pour les opérateurs humains ?

Afin de donner réponse à cette question, je me suis fixé les objectifs suivants :

1. Définir un cadre conceptuel pour l'XAI, structurant les concepts clés autour du principe de confiance et de déploiement en contexte avionique.
2. Sélectionner des méthodes d'XAI adaptées aux contraintes avioniques, puis établir un formalisme mathématique unifié pour standardiser leur utilisation.
3. Établir une méthodologie d'intégration de l'XAI aux IDS avioniques, puis évaluer son applicabilité via deux cas d'usage concrets.

1.4 Plan du mémoire

Dans le chapitre 2, nous effectuerons une revue de littérature sur les différentes applications de l'IA en avionique et en sécurité des systèmes, ainsi que leur limitation. Nous analyserons ensuite les utilisations de l'XAI dans ces domaines. Le chapitre 3 détaillera les concepts relatifs à l'XAI : nous en donnerons d'abord une conceptualisation détaillée ainsi qu'une taxonomie, puis nous analyserons certaines des techniques existantes. Le chapitre 4 sera consacré à la présentation de la méthodologie d'utilisation de ces techniques, que j'ai développée pour répondre à la problématique. Les chapitres 5, 6 et 7 présenteront trois applications de nos travaux à des cas d'études spécifiques. Le chapitre 6 sera spécifiquement consacré à un article que j'ai soumis dans le journal *IEEE Transactions on Aerospace and Electronic System*. Finalement, nous discuterons nos résultats et leurs limitations dans le chapitre 8 avant de conclure dans le chapitre 9.

CHAPITRE 2 ÉTAT DE L'ART DE L'XAI POUR LA CYBERSÉCURITÉ AVIONIQUE

L'IA et particulièrement le ML deviennent progressivement essentiels dans les systèmes embarqués en raison de leur capacité à reconnaître des motifs caractéristiques et à extraire des informations précieuses de vastes ensembles de données en temps rapide. Cela fait de l'IA un outil puissant, en particulier pour les processus de prise de décision dans des industries telles que l'avionique. Dans ce contexte, la nécessité de prendre des décisions précises et mesurées est souvent compromise par des contraintes temporelles et un flux d'informations complexes à gérer en même temps. De nombreux accidents résultent d'une prise de décision mal gérée en situation critique [13]. Contrairement aux opérateurs humains, l'IA permet de relever ces défis sans être surchargée d'informations et sans ressentir de stress ou de fatigue. Ses applications en sécurité avionique sont larges allant de la sécurité des systèmes autonomes, aux modèles prédictifs de vols et d'attaque, en passant par la gestion d'interfaces hommes-machines. Garcia *et al.* [14] recensent justement en 2021 certaines approches possibles d'utilisation de l'IA dans la cybersécurité de l'aviation.

2.1 L'IA appliquée à la sécurisation des protocoles avioniques

L'IA offre des perspectives prometteuses pour renforcer la sécurité des communications avioniques, notamment pour le protocole ADS-B. Dès 2011, McCallie *et al.* [15] ont identifié les vulnérabilités de ce protocole, suivis par Manesh *et al.* [3] en 2017. Conçu pour diffuser ouvertement les données de trafic aérien, l'ADS-B ne chiffre pas ses messages, ce qui le rend vulnérable à diverses cyberattaques pouvant compromettre les opérations aériennes. Bien que ces travaux proposent des contre-mesures, celles-ci restent insuffisantes face à la sophistication des menaces actuelles, justifiant le recours à des IDS utilisant de l'IA pour sécuriser ces protocoles.

Parmi les attaques les plus critiques, le *spoofing* (usurpation d'identité) cible spécifiquement les stations au sol en falsifiant les adresses de communication, induisant des erreurs de navigation potentiellement dangereuses. Pour y remédier, Ying, Mazer *et al.* [16] ont développé en 2019 SODA, un réseau de neurones profonds détectant ces intrusions via une classification des messages. Une approche similaire a été adoptée par Karam *et al.* [17] en 2022 avec un détecteur d'anomalies basé sur un apprentissage supervisé, entraîné sur des messages injectés artificiellement pour contrer les attaques par modification de contenu et atteignant une précision de classification de 99%.

Cependant, comme le soulignent Fried et Last [18] en 2021, l'entraînement de modèles complexes est limité par le manque de données disponibles. Leur solution repose sur un *autoencoder* reconstruisant les trajectoires de vol : plus la reconstruction est fidèle, plus les données sont considérées comme intègres. Cette méthode élargit la détection à un spectre d'attaques plus variées (*spoofing*, injection, etc.).

D'autres protocoles avioniques font l'objet de recherches similaires comme le MIL-STD-1553, utilisé dans les bus avioniques militaires. Onodueze *et al.* [4] ont évalué en 2020 les limites des algorithmes classiques face aux attaques par émulation, en proposant des métriques d'évaluation standardisées. Marrocco *et al.* [10] ont ensuite développé TRIPT-IDS, un détecteur d'intrusions basé sur un Transformer, offrant des performances supérieures.

Pour finir, nous pouvons également mentionner le protocole ARINC 429 qui est le plus largement utilisé dans l'aviation civile et qui n'a pas de processus d'authentification dans son fonctionnement. Ce défaut expose ce protocole aux attaques *Man-in-the-Middle*. Wool *et al.* [19] contournent ce problème en exploitant les empreintes matérielles des composants avioniques, détectant ainsi toute altération physique des capteurs.

2.2 Défis et problèmes actuels de l'IA dans l'aviation

Pour développer un modèle d'IA au sein d'une infrastructure critique, la question de la sécurité est primordiale [20]. Rendre un avion plus connecté implique inévitablement le rendre vulnérable à un plus grand nombre d'attaques. Il est plus exposé. Alors, lorsque l'on veut déployer un modèle d'IA au sein d'un avion, celui-ci doit nécessairement vérifier des standards de sécurité stricts. En s'appuyant sur l'utilisation du ML pour les transports au sens large, Cheng *et al.* [21] en 2018 ont exploré les défis et perspectives qu'implique un tel déploiement, mettant l'accent sur trois exigences d'un modèle d'IA, nécessaires à la sécurité du système : sa justesse, sa validation et sa compréhension. Autrement dit, un modèle doit être précis, contenir des mesures de vérification des prédictions, mais également être compréhensible par ses utilisateurs. La question de la certification se pose également. Comment garantir que les résultats d'un modèle d'IA sont fiables ? Qui est responsable en cas de mauvaises prédictions ? De nombreux travaux sont menés pour tenter de fixer une norme de certification pour encadrer l'utilisation de l'IA dans ces domaines à risque. Tambon *et al.* [22] ont effectué en 2021 une revue de littérature systémique de tous les articles publiés entre 2015 et 2020, traitant du problème de la certification de l'IA en avionique. Ils ont pu extraire quatre piliers majeurs nécessaires au développement de l'IA dans un cadre certifié : la Vérification, le *Safe Reinforcement Learning*, la Certification Directe et enfin l'Explicabilité. Degas *et al.* [8] développent de surcroît en 2022 que certes, il est nécessaire d'avoir recours à l'IA dans les

outils d'Air Traffic Management (ATM), pour gérer un trafic toujours grandissant, mais ils concluent surtout que l'IA seule, à cause du manque de confiance des utilisateurs finaux en son fonctionnement, n'est pas une solution suffisante. Il faut également développer un cadre d'explicabilité autour de l'IA, pour que son acceptation par le milieu aéronautique soit favorisée et son utilisation plus sécurisée. L'XAI, bien qu'encore en pleine quête de définition, semble apparaître comme une solution à ces enjeux.

2.3 XAI dans le domaine avionique

Compte tenu de ces défis, l'XAI semble un outil intéressant à exploiter pour tenter de résoudre les problèmes de transparence et de confiance envers les systèmes d'IA pour la protection des systèmes avioniques. En 2021, Hernandez *et al.* [23] soulignent l'étonnant fossé qu'il existe entre les progrès de la recherche en IA actuels et sa mise en œuvre pratique dans les outils d'ATM, en effectuant une comparaison des nombres de publications de recherches publiés sur ces sujets. Cette différence s'explique majoritairement par des problèmes de réglementation et de manque de transparence des modèles que nous avons évoqués plus haut. Ils ont donc développé une méthodologie d'intégration de systèmes d'XAI, aux systèmes avioniques, pour répondre aux exigences réglementaires des outils de gestion du trafic aérien. Cette méthodologie est effectuée en s'appuyant également sur des retours d'utilisateurs détaillés.

Yadam *et al.* [24] réfléchissent en 2020 sur les diverses manières d'afficher visuellement les résultats d'un modèle d'IA afin de les rendre plus compréhensibles pour les utilisateurs finaux, à savoir ici pour les ingénieurs en avionique et aéronautique. Le but du modèle utilisé est de prédire le bruit régnant au voisinage de l'aile des avions, à l'aide d'un ensemble de données de la NASA. Il est primordial de prédire précisément ce bruit puisqu'il agit fortement sur les vibrations subies par l'aile. Ce bruit dépend de plusieurs facteurs tels que la fréquence, la vitesse de l'avion ou encore son angle d'attaque. L'utilisation de l'XAI ici vise alors à afficher sous forme de graphes, les importances relatives de chacun des facteurs, aux prédictions du modèle. Ces visualisations peuvent être sous forme de matrice de corrélation, de graphe d'importance de caractéristique, ou encore des distributions des contributions des caractéristiques en fonction de leurs valeurs. Un an plus tard, plusieurs travaux exploitent justement certaines de ces techniques de visualisations pour expliquer des modèles d'IA en aéronautique comme Dalmau *et al.* [25] pour prédire l'heure de décollage exacte d'un avion en fonction de différents paramètres relatifs à l'aéroport ou encore Nor *et al.* [26] pour détecter les anomalies dans les turbines de gaz des moteurs. Déjà en 2018, Zeldam *et al.* [27] avaient effectué un travail similaire pour expliquer les diagnostiques de dysfonctionnement dans les processus de maintenance des appareils avioniques.

De la même manière, Xie *et al.* [1] en 2021 cherchent à intégrer de l'XAI au sein des processus de décision d'urgence, relatifs aux outils d'ATM, que ce soit des outils techniques de manœuvre ou de communication. Le modèle étudié cherche à prédire le risque qu'un accident se déclare prochainement, en fonction des conditions météorologiques. Une interface homme-machine est conçue pour aider les pilotes ou le personnel ATM à traiter efficacement les informations dans des situations critiques, grâce à l'XAI. Cette interface n'est pas une proposition finale de produit, mais vise plutôt à lancer les discussions sur la meilleure stratégie à adopter pour afficher les résultats des modèles aux pilotes déjà surchargés d'informations. La Figure 2.1 illustre cette proposition d'interface avec trois types d'affichages différents. Une explication qui décrit les tendances globales du modèle sur l'ensemble des entrées sur lesquelles il est évalué. Une explication locale du modèle qui explique l'impact de chaque caractéristique sur une seule prédiction. Cet onglet ne s'ouvrirait que lorsque le modèle signalerait un haut risque d'accident. Enfin, une explication relative à chaque caractéristique, montrant l'importance d'une caractéristique à travers l'ensemble des données reçues.

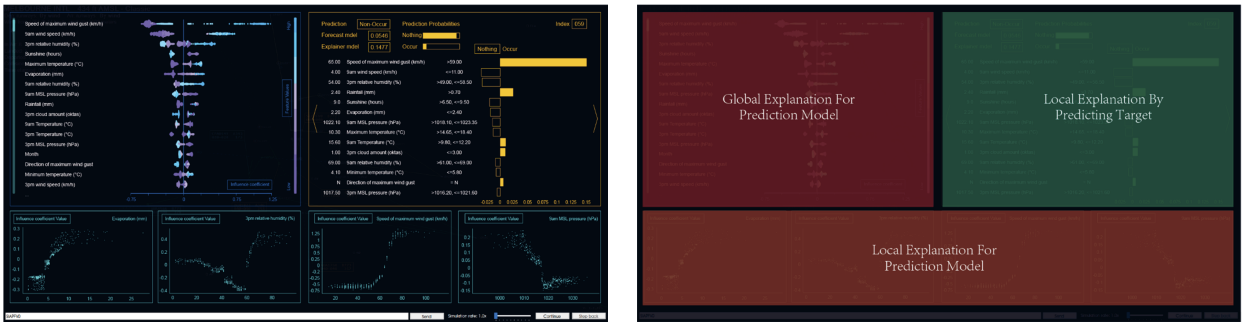


FIGURE 2.1 Interface homme-machine proposée par Xie *et al.* [1] pour expliquer en temps réel les modèles de prédiction d'accidents pour cause météorologique.

Toujours relatif au processus de prise de décision d'urgence, Sutthithatip *et al.* [28] la même année, explorent en profondeur dans quelle mesure l'XAI peut et doit trouver sa place dans les processus de décisions, étant donné une quantité d'informations à gérer déjà grande. L'XAI ne doit pas se présenter comme une information supplémentaire à gérer, mais bien comme un gestionnaire d'informations afin d'alléger le pilote et non de l'alourdir. Un an plus tard, les mêmes auteurs, [29] explicitent les différents degrés d'explicabilité que l'on peut recenser suivant les standards de l'*Intelligence Community Directive*. Ils définissent par ailleurs, trois acteurs principaux à considérer dans le processus d'intégration d'XAI :

— Les créateurs : Développent le modèle et assurent sa précision ainsi que son interprétabilité.

— Les garants : Vérifient que le modèle valide les standards imposés en termes de précision, utilisabilité et explicabilité.

— Les interpréteurs : Utilisent l'outil en bout de processus.

L'XAI doit être abordée de manière bien distincte en fonction des acteurs qui interagissent avec. Si pour les créateurs, il s'agit de lignes de code informatique et d'équations, pour les garants, c'est un ensemble de normes tandis que pour les interpréteurs, c'est un produit fini compréhensible par un non-expert de l'IA.

Il est intéressant de constater que l'XAI trouve également ses applications dans l'apprentissage par renforcement, comme le montre le travail de He *et al.* [30] qui ont utilisé de l'XAI sur un modèle d'apprentissage par renforcement profond pour les véhicules aériens sans pilote. L'objectif est de fournir des explications visuelles pour aider les utilisateurs à comprendre les décisions prises par le modèle dans des scénarios d'évitement d'obstacles, augmentant ainsi la transparence des systèmes autonomes. La Figure 2.2 montre l'interface qu'ils ont développée pour expliquer en temps réel les décisions de l'agent en sélectionnant à chaque pas de temps, trois actions que l'agent compte effectuer selon leur ordre de priorité, avec pour chacune une très brève explication de la raison.

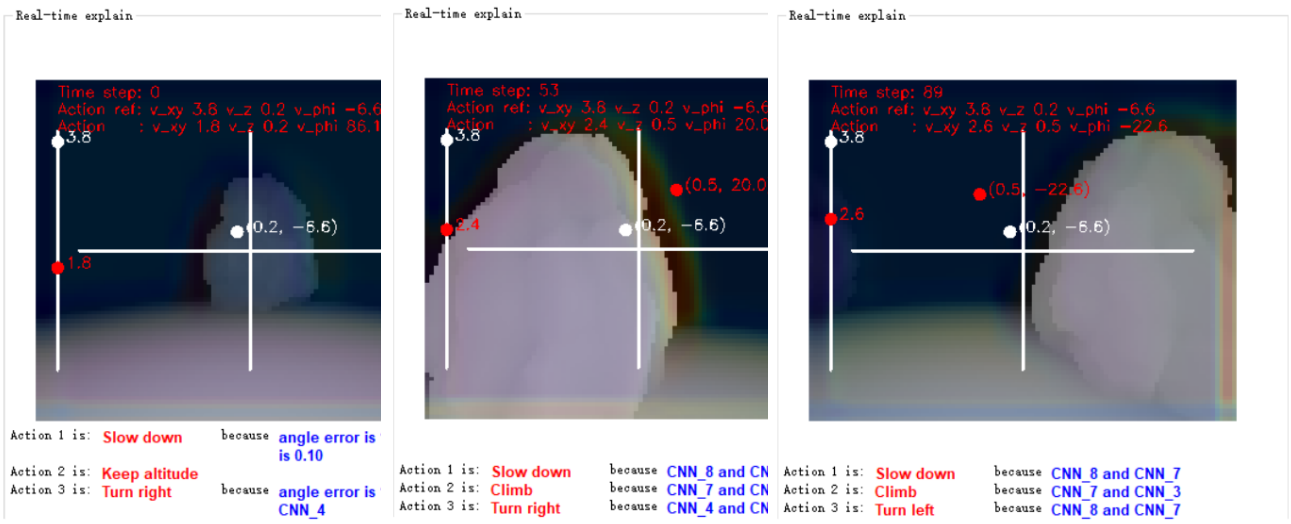


FIGURE 2.2 Explication d'une décision d'un agent de RL à trois temps différents

A travers tous ces travaux récents, nous avons constaté que l'XAI suscite depuis 2020 et de manière croissante, un intérêt dans la communauté en ingénierie avionique. Son application reste cependant dans la plupart des cas théoriques avec peu d'applications pratiques réelles. L'XAI cherche encore à se définir et à spécifier ses usages pour pouvoir finalement trouver des cas concrets en aéronautique et en avionique dans les années à venir.

Une autre discipline particulièrement intéressée par l'IA est celle de la sécurité des systèmes informatiques. Comme nous l'avons vu, l'IA est particulièrement efficace dans ces disciplines, mais lorsque les risques augmentent, une couche d'explicabilité semble également s'imposer.

2.4 XAI dans la sécurité des systèmes

En 2018, Vigano et Magazzeni [31] présentent *eXplainable Security* (XSec), une extension de l'XAI spécifique au domaine de la sécurité. En répondant aux questions des "six W" (qui donne, qui reçoit, quoi, où, quand, pourquoi et comment), ils établissent le spectre des caractéristiques de XSec, définissant ainsi les frontières d'application de l'XAI au domaine de la sécurité informatique. La Figure 2.3 représente ce spectre exploré par les "Six Ws".

En 2022, Srivastava *et al.* [32] étudient l'état de l'art et objectifs actuels relatifs à XSec. Ils recensent les secteurs d'application qui ont été concernés par XSec, spécifiquement les secteurs de l'industrie, et montrent que le spectre est très large allant de l'agriculture, au transport en passant par les systèmes bancaires et la santé. La Figure 2.4 montre l'ensemble de ces applications. On constate que des technologies telles que la 5G sont mentionnées ce qui nous permet également d'évoquer le travail de Guo [33] qui tente d'ajouter une couche d'explicabilité dans les systèmes d'installation des réseaux 6G. L'objectif serait de fournir une interface de compréhension du fonctionnement du réseau, à destination des utilisateurs, afin que ceux-ci soient mieux avertis des éventuels problèmes liés à cette technologie. Nos appareils téléphoniques sont déjà équipés de nombreux détecteurs d'anomalies qui ont pour but de bloquer les dangers menaçant l'intégrité du téléphone. Cependant, bien souvent, ces blocages ne sont pas expliqués poussant l'utilisateur à croire qu'il s'agit d'une erreur ou d'une fausse alerte.

L'application la plus fréquente et qui nous intéresse le plus dans nos travaux est l'utilisation de l'XAI pour expliquer les IDS. Mane et Rao [34] en 2021, Patil *et al.* [35] en 2022 et Nwakanma *et al.* [36] en 2023 sont trois exemples de travaux utilisant SHAP ou LIME pour expliquer des IDS en cybersécurité des réseaux.

Une autre application commune est l'explication les modèles d'analyse de menaces. Pour faire simple, l'objectif est de détecter quels éléments d'un modèle sont les plus susceptibles de détecter une alerte relative à un type d'attaque. En faisant cela pour toutes les attaques recensées, on peut tracer le profil d'importance de chaque caractéristique à chaque type d'attaque. C'est le travail d'Alenezi et Ludwig [37] qui établissent ces profils en utilisant SHAP, comme dans la Figure 2.5 qui se concentre sur l'importance de chaque caractéristique dans la détection d'URL malveillant.

Il est intéressant de noter que les modèles XAI eux-mêmes peuvent être la cible d'attaques. Ces scénarios sont étudiés par Kuppa et Le-Khac [38] [39] dans deux travaux visant à recenser les nouvelles attaques inhérentes à l'intégration de l'XAI. D'une part, l'explication elle-même peut-être corrompue, altérant la compréhension de l'utilisateur sur le problème actuel et

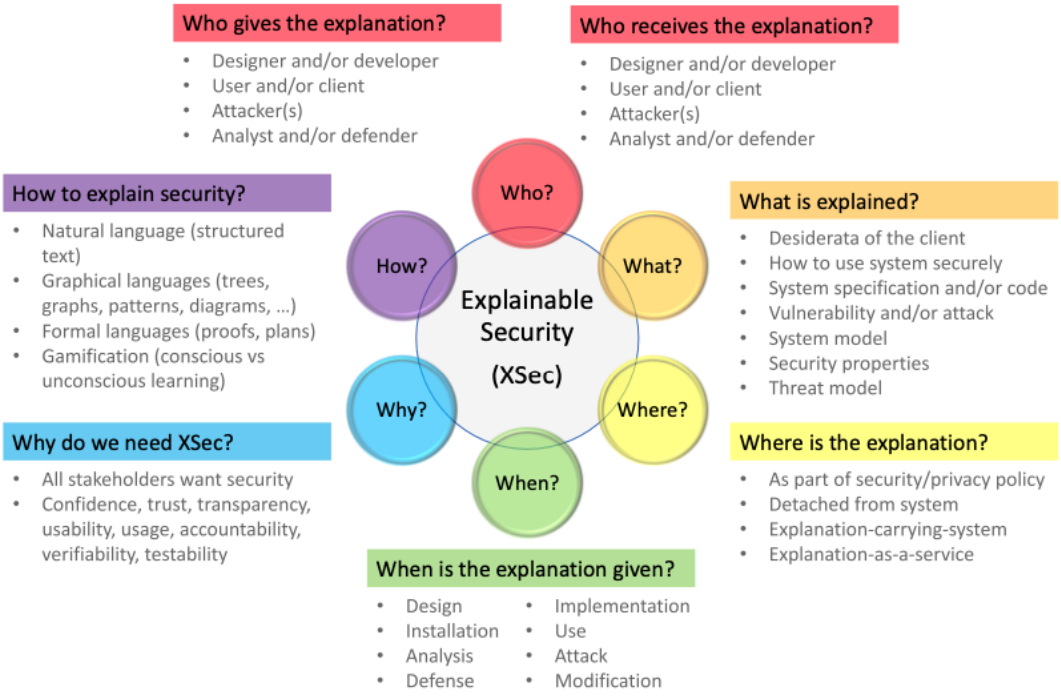


FIGURE 2.3 Les fondements de XSec

FIGURE 2.4 Domaines d'application de XSec

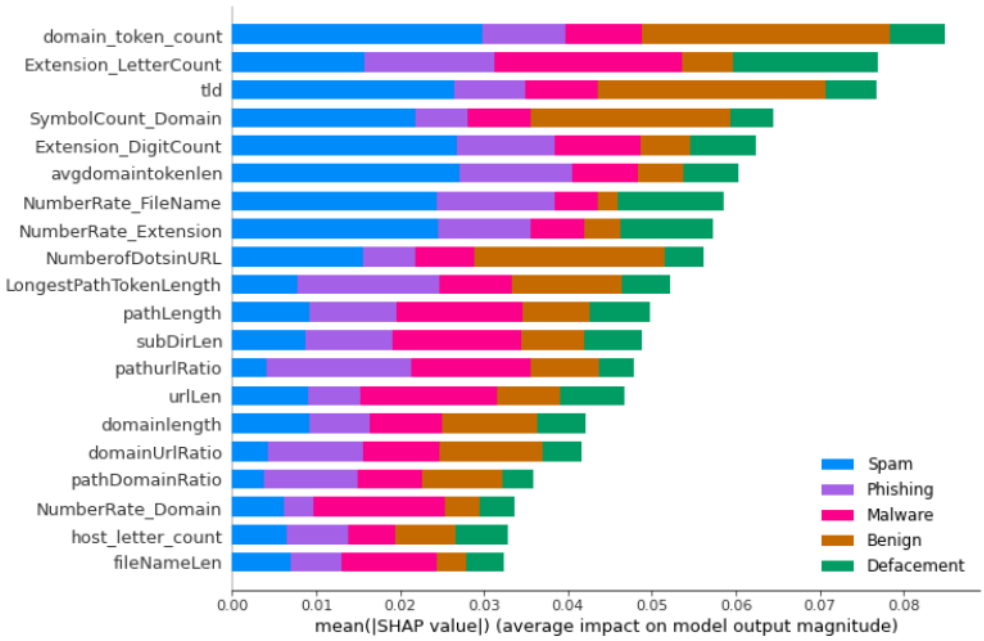


FIGURE 2.5 Profils SHAP pour la caractérisation d'URL malveillants

menant à des erreurs de décision. D'autre part, le modèle d'explication lui-même peut être attaqué, altérant par conséquent, le processus de génération des explications. Enfin, comme dans tout système, l'intégration d'une couche en plus augmente la possibilité d'intégration de porte dérobée par un éventuel attaquant. Ils développent alors des méthodes de protection contre ces différentes attaques. Dans la même direction, Slack *et al.* [40] analysent en 2020 les vulnérabilités que présentent les techniques d'XAI les plus utilisées que sont SHAP et LIME. Ils parviennent à introduire des biais racistes dans les jeux de données utilisés pour montrer que SHAP et LIME sont incapables d'en détecter l'intrusion. Les caractéristiques "caractère raciste" pour schématiser ne seront pas plus importantes suite à leur injection, relevant une faille dans la compréhension de la situation.

2.5 Évaluation des méthodes d'XAI

Lors du développement d'un modèle d'IA, celui-ci est souvent évalué suivant des métriques classiques permettant de le comparer avec d'autres modèles de l'état de l'art. Parmi ces métriques, on connaît la précision ou encore le taux de faux positifs/négatifs. Mais qu'en est-il pour les modèles d'XAI ? Il convient également de définir des métriques adaptées à l'XAI pour en évaluer les performances. Warnecke *et al.* [41] présentent en 2020 un ensemble de métriques pour évaluer une méthode d'XAI dans un contexte de sécurité informatique. Ces

métriques sont la précision descriptive, la dispersion, la robustesse, l'efficacité, la complétude et la stabilité. Ils appliquent ensuite ces critères pour comparer les méthodes d'XAI connues sur un cas classique de détection de logiciel malveillant. Ils comparent notamment les deux techniques SHAP et LIME au regard de ces métriques. Cugny *et al.* [42] développent en 2023, une méthodologie pour fournir la meilleure technique d'XAI avec les hyperparamètres optimaux pour répondre aux besoins spécifiques de l'utilisateur. Le problème est formulé sous forme de problème d'optimisation avec les contraintes utilisateur au cœur du processus. Pedro Lopes *et al.* [43] proposent en 2022, une taxonomie des techniques d'évaluation d'XAI pour séparer les techniques d'évaluation purement calculatoires [41], de celles dirigées vers l'expérience utilisateur. La Figure 2.6 présente cette taxonomie. [42]. Enfin, Osvaldo Arreche *et al.* [44] utilisent en 2024 les métriques de [41] pour les appliquer spécifiquement à trois IDS distincts. L'objectif étant de montrer les limites des techniques d'XAI actuels pour expliquer les IDS.

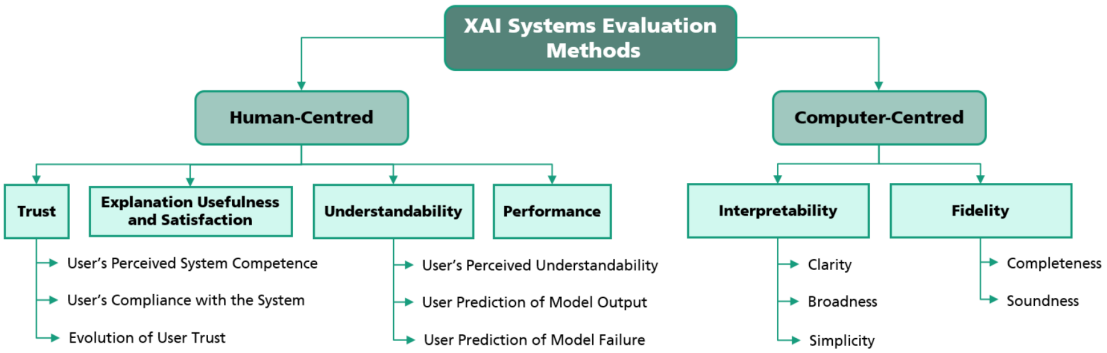


FIGURE 2.6 Taxonomie des méthodes d'évaluation d'XAI

CHAPITRE 3 L'IA EXPLICABLE : DES CONCEPTS, DES TECHNIQUES ET UNE HISTOIRE DE MATHÉMATIQUES

Ce chapitre, consacré à l'XAI se décompose en deux parties. La première vise à conceptualiser l'XAI en présentant les motivations et objectifs de cette discipline, les définitions ainsi que l'ensemble des champs d'application dans lequel on peut l'utiliser. L'objectif est ici de concevoir un cadre conceptuel permettant de mettre en lumière les concepts clés que nous retiendrons dans notre définition de l'XAI, au regard de l'application en avionique qui en sera faite. Nous présenterons également dans cette partie un répertoire de techniques existantes sous la forme d'une taxonomie pour achever la conceptualisation. La deuxième partie se concentre justement sur les techniques de ce répertoire, notamment une méthode appelée SHAP qui unifie plusieurs techniques d'XAI importantes dans la littérature. L'objectif de cette partie est de comprendre ce qui a motivé mon choix de sélectionner SHAP comme méthode phare pour ma recherche, répondant à la fois aux exigences avioniques de performance et de transparence et validant un grand nombre des piliers retenus dans notre conceptualisation.

3.1 Conceptualisation de l'XAI

Cette section vise à remplir mon premier objectif de recherche :

Définir un cadre conceptuel pour l'XAI, structurant les concepts clés autour du principe de confiance et de déploiement en contexte avionique.

Avant d'établir cette conceptualisation, présentons les objectifs et motivations qui ont donné naissance à cette discipline.

3.1.1 Objectifs et motivations

Les modèles d'apprentissage automatisés deviennent extrêmement complexes pour gagner en précision et en efficacité. Ces modèles deviennent également plus difficiles à comprendre et à interpréter. Même pour un expert en IA, il est presque impossible de prédire les résultats d'un modèle utilisant un réseau de neurones profond, et de saisir mathématiquement le cheminement qui a mené à ce résultat. L'XAI est un domaine de l'IA qui tente justement de la rendre compréhensible. Mais que signifie exactement compréhensible ? Et d'ailleurs, pourquoi chercher à comprendre un modèle d'IA ? Pourquoi ne pas simplement lui faire confiance ? D'autant qu'à mesure que les modèles progressent, leur précision augmente également, alors pourquoi une explication est-elle nécessaire ? Cette première partie a pour but de répondre

à ces questions. Déjà, il faut comprendre que dépendamment du domaine, une explication peut être plus ou moins utile. Dans des domaines sans risque, comme les algorithmes de recherche internet, l'utilisateur n'est pas vraiment intéressé par une quelconque interprétation ou explication du modèle qui a guidé l'algorithme à donner les suggestions. Seul le résultat l'intéresse et dans ce cas, l'XAI est inutile. Cependant, dans des domaines à risque tels que les domaines tels que la sécurité informatique, la médecine ou encore les transports, où les décisions ont des impacts sur des vies humaines ou impliquant des coûts élevés, l'XAI trouve particulièrement son utilité. Pour un modèle d'IA dont la tâche est de détecter les cancers, savoir que le patient est atteint de cancer est une information cruciale mais pas suffisante seule. Le médecin a besoin de savoir ce qui a poussé l'algorithme à conclure sur la présence d'un cancer. Déjà, en utilisant son expertise, il peut confirmer si les raisonnements du modèle sont cohérents, mais surtout, il peut avoir une meilleure compréhension du type de cancer dont il s'agit, sa dangerosité et comment mieux le traiter. Il y a une différence entre un modèle qui retourne "CANCER DE LA PEAU" et un modèle qui retourne : "CANCER DE LA PEAU, FIABILITÉ : 97%, RAISONS : LÉSIONS QUI NE GUÉRISSENT PAS, GRAINS DE BEAUTÉ ANORMAUX, PLAQUES BLANCHES". L'explication fournit ici semble permettre de confirmer que le diagnostic du modèle est correct. Cependant, le modèle pourrait aussi retourner : "CANCER DE LA PEAU, FIABILITÉ : 80%, RAISONS : PLAQUES ÉPAISSES ROUGES ET BRUNES". Dans ce cas-là, le médecin pourrait décider que les analyses ne sont pas suffisantes pour conclure à un cancer, mais qu'il pourrait aussi s'agir d'un psoriasis par exemple. L'XAI peut donc trouver son utilité dans la détection de faux positifs et faux négatifs. On appelle faux positif une instance du modèle qui est détectée positive, mais qui est en réalité négative, et faux négatif l'inverse. Dans l'exemple de la détection du cancer, un faux positif est lorsque le modèle affirme que le patient est atteint d'un cancer alors qu'il ne l'est pas. Ces faux résultats sont très problématiques dans les domaines à risque. Des explications accompagnant les prédictions permettraient de détecter ces erreurs, ou du moins d'impliquer un raisonnement humain dans le processus de décision. Une autre utilité de l'XAI est alors pour améliorer le modèle en lui-même. En comprenant le fonctionnement interne du modèle et en détectant les erreurs, il peut devenir plus aisé d'adapter le modèle en conséquence. Finalement, l'XAI se présente comme un ensemble de méthodes qui donnent aux humains les clés pour avoir une compréhension intellectuelle sur le fonctionnement d'un modèle d'IA. Le but principal étant de pouvoir raisonner sur les prédictions faites par le modèle. Nous allons voir par la suite certaines disciplines qu'explore l'XAI ainsi que les méthodes actuelles connues dans la littérature.

3.1.2 Conceptualisation

Maintenant que les objectifs sont clairement établis, présentons le cadre conceptuel que j'ai développé pour définir l'XAI en 11 concepts distincts. Ces concepts s'inspirent d'une part de nomenclatures existantes dans la littérature et d'autre part, ils visent à être pertinents dans le contexte de ma recherche, dans laquelle nous cherchons à utiliser l'XAI pour la cybersécurité avionique.

Sajid Ali propose dans *Explainable Artificial Intelligence (XAI) : What we know and what is left to attain Trustworthy Artificial Intelligence* [45], une nomenclature pour énoncer et définir des termes liés au concept d'XAI. La liste est non exhaustive, mais encadre cependant assez précisément les directions que peut prendre l'XAI, en donnant ainsi une bonne définition. J'ai suivi la même démarche pour créer mon cadre conceptuel en m'inspirant d'autres travaux du même type [46], [47], [48], [49].

La Figure 3.1 représente le cadre conceptuel finalement retenu. Il représente d'une part 8 concepts participant à augmenter la confiance des utilisateurs envers un modèle d'IA. L'octogone est choisi de telle sorte à ce que tous les concepts soient suffisamment indépendants les uns des autres, mais dont leur réunion est suffisant à établir une relation de confiance avec le modèle. D'autre part, certains concepts ne sont pas essentiels à l'amélioration de la confiance mais ils participent à l'applicabilité de tels modèles, au contexte avionique, prenant en considération certaines de ses spécificités. Les concepts de précision et de satisfaction participent aux deux pôles. Analysons désormais chacun de ces concepts.

1. **Explicabilité :** Selon le CNIL, c'est "la capacité de mettre en relation et de rendre compréhensibles les éléments pris en compte par le système d'IA pour la production d'un résultat". C'est la réponse à **pourquoi** le modèle a pris cette décision ? Pourquoi le malade est diagnostiqué positif ? Pourquoi le système a-t-il levé une alerte ? Pourquoi la maison est évaluée à ce prix ? Une explication résume la liste des raisons qui ont poussé le modèle à prendre une décision. Chistoph Molnar dans *A Guide for Making Black Box Models Explainable* [46] décrit l'explication comme une connexion entre les attributs d'un modèle et sa prédiction, dans un langage humainement compréhensible. Il recense également des critères d'évaluation d'une bonne explication, parmi lesquels on trouve :
   — *Sélectivité :* Capacité à sélectionner uniquement les éléments importants dans la prise de décision. Une explication ne doit pas lister exhaustivement toutes les raisons qui ont poussé le modèle à prédire tel résultat. Une à trois raisons sont souvent largement suffisantes pour donner à l'utilisateur une bonne compréhension sans lui donner trop d'informations.

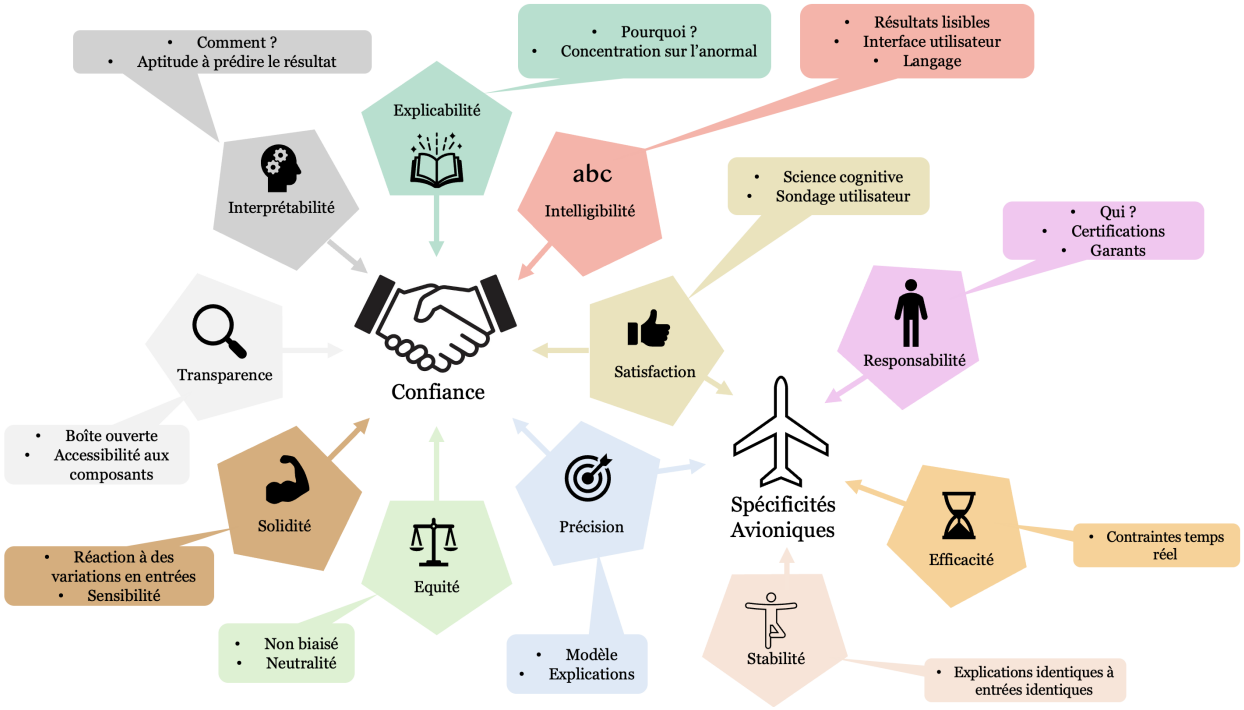


FIGURE 3.1 Cadre conceptuel pour définir l'XAI et sa capacité à améliorer la confiance en des modèles d'IA, tout en respectant des contraintes avioniques.

— *Adaptabilité* : Capacité à s'adapter au contexte et à l'utilisateur qui est visé par l'explication. On doit être capable d'expliquer le plus de situations possibles si l'on veut être utile à l'utilisateur. Si un générateur d'explication ne peut qu'expliquer les cas d'écoles simples, il ne sera pas vraiment utile en pratique.

— *Concentration sur l'anormal* : Mise en évidence des éléments qui sont inhabituels par rapport à une situation de référence. Cet élément est particulièrement important et est fondateur pour tout ce qui va suivre. En effet, l'esprit humain analyse très souvent une situation donnée par rapport à une situation de référence. Lorsque l'on demande pourquoi un patient est détecté malade, il convient naturellement de répondre ce qui a été détecté de différent par rapport aux individus sains. La réponse "il a deux bras et deux jambes" n'est pas fausse en soi, mais n'explique absolument pas pourquoi il a été détecté malade par rapport aux autres. Ainsi, toute question commençant par 'pourquoi', cache presque toujours une situation de référence implicite. Pourquoi la maison est-elle évaluée à tel prix, par rapport à la moyenne des prix des maisons ?

Une bonne explication doit prendre toutes ces considérations en compte. En pratique, l'explication est ce qui va être fourni à l'utilisateur souvent sous forme de phrases ou de

graphe si cela est plus approprié. Une question à poser dans le déploiement de l'XAI en avionique est la destination de l'explication. Le contenu de l'explication sera bien différent si le récepteur est le pilote ou les ingénieurs au sol.

2. **Interprétabilité :** L'interprétabilité permet de comprendre les mécanismes de fonctionnement du modèle. Ce n'est pas pourquoi le modèle a fait cette prédiction, mais **comment** il l'a fait. Cela implique une compréhension mathématique des interactions entre les différents composants du modèle. Un modèle interprétable est un modèle dont on pourrait faire des prédictions dessus. Selon Tim Miller dans *Explanation in Artificial Intelligence : Insights from the Social Sciences* [47], l'interprétabilité est une mesure d'à quel point un humain peut comprendre une décision. Une autre définition proche serait à quel point un humain peut prédire de manière cohérente, les résultats du modèle. C'est une propriété intrinsèque du modèle. On distingue les termes interprétabilité et explicabilité en se souvenant qu'un modèle est interprétable, et une prédiction est explicable. Lorsqu'un modèle n'est pas interprétable comme un réseau de neurones, on ne pourra pas le rendre interprétable directement, mais l'on pourra lui ajouter un modèle simplifié, qui lui le sera. Lorsque l'on développe un modèle interprétable, on accepte et il est même souvent conseillé, que les caractéristiques du modèle interprétable ne soient pas les mêmes que celles du modèle original. En effet, dans des modèles complexes, les caractéristiques d'un modèle sont souvent difficiles à appréhender ; dans le cas d'une image, les caractéristiques peuvent être sous forme de tenseur en trois dimensions contenant les canaux RGB de chaque pixel. On préférera alors mentionner dans notre interprétation si le pixel se distingue de ses voisins ou non. En somme, une caractéristique binaire simplifiée. Dans un modèle visant à classifier des textes, les caractéristiques sont des vecteurs projetés dans des espaces de plus hautes dimensions pour représenter les différents mots. En interprétation, on choisira plutôt de préciser si tel mot est présent ou non dans le texte. Ainsi, très souvent, lorsque l'entrée de notre modèle est sous la forme $x \in \mathbb{R}^d$, on utilisera une forme binaire simplifiée de ce vecteur sous la forme $x' \in \{0,1\}^d$. Ici d représente le nombre de caractéristiques du modèle.

   A partir de maintenant, nous définirons plus rapidement les autres concepts de la Figure 3.1 puisque ce mémoire explore principalement les concepts d'explicabilité et d'interprétabilité que nous venons de définir en détail.

3. **Transparence :** La transparence d'un modèle est sa capacité à rendre ses composants et décisions accessibles et lisibles par un humain. La transparence vient se mettre en opposition directe au concept de boîte-noire, très présent en IA. Un modèle de boîte-

noire est un modèle dont les décisions sont difficiles, voire impossibles, à accéder par un humain. Les réseaux de neurones profonds en sont un exemple classique. Métaphoriquement, rendre transparent un modèle, c'est pouvoir ouvrir la boîte. La transparence implique que les éléments internes du modèle (comme les poids dans un réseau de neurones ou les règles et nœuds d'un arbre de décision) soient accessibles et lisibles. Cela signifie que l'on peut examiner comment le modèle est structuré et comment il fonctionne.

4. **Robustesse (Solidité) :** La robustesse, aussi appelée la solidité d'un modèle se mesure à la variation de sa sortie lorsque l'on modifie légèrement l'entrée. Un modèle solide ne doit pas trop faire varier sa prédiction lorsque l'entrée varie légèrement. Par exemple, un modèle de reconnaissance d'images doit être capable de reconnaître un chat même si l'image est légèrement floue ou mal éclairée. C'est lors de la phase d'entraînement qu'il peut alors être pertinent de donner des entrées perturbées au modèle, et non pas seulement des données parfaitement saines. La robustesse est particulièrement importante pour résister aux attaques adverses, où des perturbations intentionnelles sont appliquées pour tromper le modèle. Par exemple, dans les véhicules autonomes, des perturbations mineures sur un panneau d'arrêt peuvent induire le modèle en erreur. Pour améliorer la robustesse, des techniques comme l'augmentation des données, l'apprentissage par injection de données malveillantes ou encore la régularisation sont couramment utilisées. Des travaux comme ceux de Goodfellow et al. [50] montrent comment un tel apprentissage par injection peut renforcer la résistance des modèles aux perturbations. Un modèle d'explicabilité doit lui aussi être solide dans le sens où il doit fournir des explications identiques à des situations qui le sont.

5. **Équité :** Un modèle doit être juste et non biaisé. Il en va de même pour ses explications. Les modèles utilisent de grands jeux de données issus de diverses sources qui peuvent avoir leurs propres biais, qu'ils soient raciaux, philosophiques ou sociétaux. Le modèle se doit de rester juste quant à tous ces sujets. Par exemple, un modèle de recrutement ne doit pas discriminer les candidats en fonction de leur genre ou de leur origine ethnique, même si les données historiques reflètent des biais de ce type. Il n'est pas si évident de s'assurer que ChatGPT reste parfaitement équitable alors qu'il s'entraîne probablement sur de nombreux jeux de données qui ne le sont pas. Les jeux de données doivent ainsi être filtrés au préalable pour respecter les principes d'inclusion, de neutralité et d'équité. Pour garantir l'équité, des techniques comme le rééchantillonnage des données, la pondération des classes, ou l'utilisation de méthodes d'apprentissage équitables peuvent être employées. Par exemple, dans le système judiciaire, des modèles comme COMPAS

ont été critiqués pour leur biais racial, ce qui a conduit à des recherches sur la détection et la correction de ces biais. Des travaux comme ceux de Friedler et al. [51] montrent comment les concepts de philosophie politique peuvent éclairer la conception de modèles équitables en IA.

6. **Intelligibilité :** L'intelligibilité en XAI se réfère à la capacité d'un modèle à fournir des explications compréhensibles et adaptées aux utilisateurs, qu'ils soient experts ou non. Une explication intelligible est claire, lisible et ajustée au niveau de connaissance de l'utilisateur. Par exemple, dans le domaine de la finance, un modèle qui refuse un prêt doit expliquer sa décision de manière accessible, comme : "le revenu mensuel est insuffisant par rapport aux charges fixes", plutôt que de fournir une réponse opaque à base de tableur ou de graphe. L'intelligibilité est essentielle pour inspirer confiance et favoriser l'adoption de l'IA, car les utilisateurs sont plus enclins à accepter les décisions d'un système qu'ils comprennent. L'intelligibilité est très proche de l'explicabilité, mais fait plutôt référence au caractère syntaxique ou graphique de la réponse, sa forme entre autres. L'explicabilité fait elle référence à son contenu dans le fond.

7. **Satisfaction des utilisateurs :** La satisfaction des utilisateurs est également un critère sur lequel l'XAI se doit d'être attentif. On peut même affirmer que c'est le plus important. Indépendamment de tous les autres concepts définis ci-dessus, si un pilote d'avion n'est pas satisfait d'une explication, alors elle est mauvaise puisqu'elle avait pour but principal de servir le pilote. La satisfaction peut se mesurer à l'aide de questionnaires, d'entretiens ou de tests utilisateurs, en évaluant des aspects comme la clarté, l'utilité et l'adaptation aux besoins. Janet H. Hsiao étudie dans *Roadmap of Designing Cognitive Metrics for Explainable Artificial Intelligence (XAI)* [49] les concepts inhérents à la satisfaction de l'utilisateur en adoptant une approche utilisant les sciences cognitives. Elle propose des métriques pour mesurer la charge cognitive, la confiance et l'utilité perçue des explications. Par exemple, une explication qui réduit l'effort mental requis pour comprendre une décision d'IA, sera perçue comme plus satisfaisante.

   La satisfaction des utilisateurs est étroitement liée aux autres concepts d'XAI. Par exemple, une explication peut être techniquement correcte (explicabilité) et fournie par un modèle robuste, mais si elle est trop complexe ou mal adaptée à l'utilisateur, elle ne sera pas satisfaisante. De même, un modèle transparent peut fournir des informations détaillées, mais si ces informations ne sont pas présentées de manière intelligible, la satisfaction en pâtira.

Par ailleurs, la satisfaction est un critère dynamique qui dépend du contexte et du profil de l'utilisateur. Par exemple, un ingénieur peut préférer des explications techniques détaillées, tandis qu'un pilote peut préférer des explications claires et concises. Adapter les explications à ces besoins est essentiel pour maximiser la satisfaction. En somme, la satisfaction est une métrique assurant un équilibre des concepts d'XAI, plutôt que la performance d'un seul.

8. **Précision :** Un modèle doit être extrêmement précis. D'autant plus précis que le domaine auquel on l'applique est à risque. La précision est le critère le plus facile à appréhender tant il est présent dans la littérature comme métrique d'évaluation principale. Il est pratiquement inutile d'expliquer ou d'interpréter un modèle très peu précis. Dans le contexte d'XAI, la précision s'étend également aux explications qui doivent également être précises. Une explication ne doit pas être fausse. Ce concept, est crucial en général et surtout pour correspondre aux contraintes avioniques qui exigent des précisions presque parfaites pour espérer que le modèle soit déployé. Si le modèle d'IA utilisé n'est pas parfaitement précis, il faut au moins que ses explications le soient.

9. **Responsabilité :** La responsabilité en XAI désigne la capacité à attribuer clairement les décisions d'un modèle d'IA à des entités ou processus spécifiques, qu'il s'agisse des concepteurs, des données ou de l'algorithme lui-même. En avionique, cela implique que les décisions doivent être traçables, justifiables et conformes aux normes éthiques et juridiques, pour gérer les problèmes liés à de mauvaises prises de décision. Par exemple, dans le domaine médical, si un modèle recommande un traitement erroné, l'XAI permet de retracer l'origine de l'erreur (biais dans les données, limitation de l'algorithme, etc.) et de prendre des mesures correctives. Ce concept ne participe pas directement à la confiance, un modèle peut être en termes de responsabilité clairement définie sans que la confiance en celui-ci ne soit particulièrement améliorée. En revanche, c'est un concept crucial en avionique.
10. **Efficacité :**

    L'efficacité est ici définie comme la complexité algorithmique temporelle, le temps effectif que prend un modèle d'une part à s'entraîner et d'autre part à prédire et à expliquer. Souvent, le temps d'entraînement n'est pas la contrainte principale puisque le modèle est entraîné au préalable de son utilisation. En revanche, dans des contextes temps-réel comme l'avionique, où les délais de décision sont critiques, le processus de prédiction et d'explication, qui vont être effectués en direct, ne doivent pas impacter significativement les délais requis de prise de décision.

11. **Stabilité :** Une dernière mesure importante dans une infrastructure critique est la stabilité des résultats générés. Une prédiction ou une explication doit être fiable dans le sens qu'elle ne doit pas fournir des résultats différents pour les mêmes données d'entrée. Cette métrique est importante à vérifier dans les méthodes probabilistes dont les résultats sont susceptibles de varier entre deux exécutions. Dans un contexte avionique, il est toujours préférable de réduire les degrés d'incertitude là où c'est possible.

Le tour d'horizon des concepts clés de l'XAI est désormais terminé. Bien que non-exhaustif, ce panorama avait pour objectif de mettre en lumière la diversité des approches possibles, qui s'articulent autour de notions mathématiques, philosophiques, mais aussi légales. Ces 11 concepts, trop nombreux pour être explorés simultanément, nous ont amenés à d'abord nous concentrer sur l'interprétabilité et l'explicabilité. Ces deux notions sont, en effet, fondamentales, car elles viennent enrichir les modèles existants en les rendant plus utilisables et compréhensibles, tout en répondant aux besoins pratiques des utilisateurs.

3.2 Taxonomie

Cette partie a pour but de présenter une taxonomie des techniques d'XAI proposée par Adel Abusitta dans *Survey on Explainable AI : Techniques, Challenges and Open Issues* [48]. L'objectif est de classifier les différentes méthodes d'explicabilité et d'interprétabilité et de spécifier sur quel type de modèles elles s'appliquent afin de répondre au deuxième objectif fixé :

Sélectionner des méthodes d'XAI adaptées aux contraintes avioniques, puis établir un formalisme mathématique unifié pour standardiser leur utilisation.

La taxonomie en elle-même n'est pas notre contribution, elle est un préambule au processus de sélection de la méthode à utiliser pour nos applications. La Figure 3.2 illustre la taxonomie que nous allons explorer dans ce qui suit. Elle se divise en deux types de modèles : les modèles directement interprétables et les modèles à interprétations a posteriori.

3.2.1 Modèles interprétables

Un modèle est interprétable si ses prédictions sont directement compréhensibles par l'utilisateur. Un tel modèle n'a pas besoin de nouveau modèle pour l'expliquer, il se suffit à lui-même en termes de compréhension. Les arbres de décision, de régression linéaire et logistique sont des exemples de modèles interprétables. Selon [48], un modèle doit satisfaire une des deux propriétés suivantes pour être considéré comme interprétable :

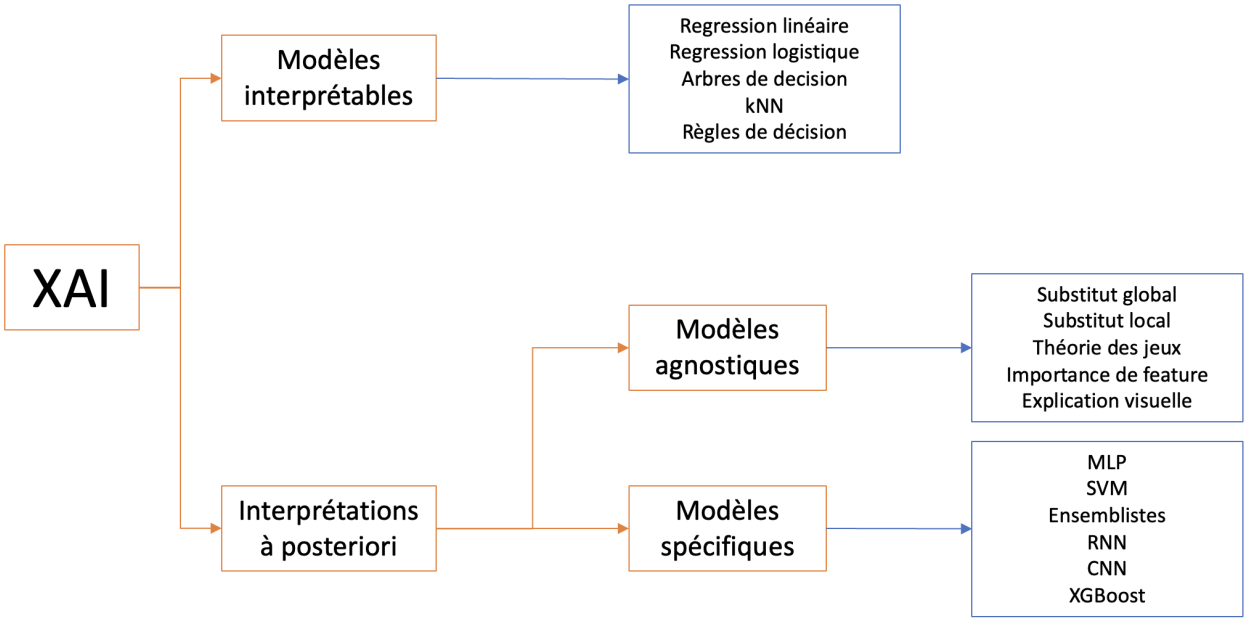


FIGURE 3.2 Taxonomie des techniques d'XAI

— *Simulabilité :* Un modèle est simulable si un humain peut décomposer le cheminement qui mène à la prédiction et comprendre chaque étape du modèle. Un arbre de décision est très simulable, car il est facile de décortiquer et d'analyser les nœuds un par un, pour suivre la progression de la prédiction.

— *Décomposable :* Un modèle est décomposable si chacun de ses sous-composants est interprétable en tant que tel. Si chaque règle d'un modèle de règle de décision est interprétable, alors le modèle est décomposable.

En XAI, ces modèles ne sont pas le cœur d'étude puisqu'ils ne nécessitent pas d'interprétations supplémentaires. En revanche, ils peuvent servir d'outils pour interpréter les modèles de la seconde catégorie.

3.2.2 Interprétations a posteriori

Les modèles évoqués précédemment ont une bonne interprétabilité, mais souvent des précisions moins performantes. C'est le compromis précision/interprétabilité que nous évoquerons un peu plus loin. Pour pallier ce problème, on peut utiliser des modèles très précis, mais peu interprétables comme les modèles d'apprentissage profond, et construire par-dessus une méthode d'explicabilité. Celle-ci vient donc s'ajouter après le modèle d'où le nom de méthodes "a posteriori". Nous allons analyser deux types de méthodes d'interprétabilité a posteriori : les techniques agnostiques et les techniques spécifiques.

1. **Techniques agnostiques :** Une technique agnostique ne dépend pas du modèle sur lequel elle s'applique. Elle est donc très adaptable. Elle nécessite uniquement les entrées et sorties du modèle et tente de détecter les relations et corrélations qui les relient. Ci-dessous, nous détaillons certaines de ces méthodes :

   Importance de la Permutation des caractéristiques (Permutation feature) : Dans cette technique, on regarde simplement l'impact sur la prédiction lorsque l'on change la valeur d'une caractéristique. Elle est importante si un changement de sa valeur implique un changement important dans la prédiction. Le module de calcul d'importance de caractéristique avec la librairie *scikit-learn* sur le modèle *XGBoost* utilise cette technique. Elle est simple et donne une bonne première idée des potentielles raisons derrière la prédiction. Cependant, elle ignore complètement les relations entre les caractéristiques elles-mêmes. Par ailleurs, il peut être très coûteux de faire varier chaque caractéristique, d'autant qu'il n'est pas clair si les permutations doivent être faites au sein du jeu de données de test ou d'entraînement, ni même les nouvelles valeurs qu'elles doivent prendre.

   Modèles substituts : Dans cette technique, on utilise un modèle interprétable, comme un arbre de décision, pour approximer le modèle boîte noire à expliquer. Pour cela, on doit alors entraîner le modèle interprétable à prédire des résultats proches de ceux du modèle original. On mesure la qualité de ce modèle en calculant simplement la différence entre leurs prédictions sur un ensemble d'entrée d'évaluation. On parle de substitut global lorsque le modèle substitut essaie d'approximer le modèle original complet, et de substitut local lorsqu'il essaie d'approximer le modèle pour une seule prédiction. Le défaut principal de la technique de substitut global est la potentielle perte en précision. En effet, le modèle substitut sera difficilement aussi précis que le modèle de base, sinon, cela veut dire qu'il n'était pas nécessaire dès le départ d'avoir un modèle si complexe. En revanche, le substitut local peut s'avérer très utile pour expliquer une prédiction spécifique comme nous le verrons plus tard avec la technique LIME.

   Techniques de théorie des jeux : Dans ces techniques, on associe les caractéristiques du modèle aux joueurs d'un jeu, et l'on utilise les différents concepts de la théorie des jeux pour avoir une meilleure interprétation du modèle. Nous ne détaillerons pas plus cette section puisqu'elle fait l'objet d'une partie entière 3.3.

2. **Techniques spécifiques :** Les techniques spécifiques sont spécialement construites

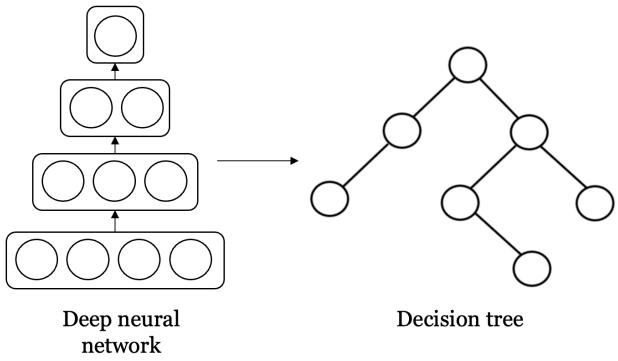


FIGURE 3.3 Modèle substitut

pour interpréter un type de modèle en particulier. On trouve dans la littérature plusieurs travaux afin d'expliquer un modèle ensembliste, un *Support Vector Machine* (SVM) ou encore un modèle *XGBoost*. Ces techniques sont souvent plus efficaces que les techniques agnostiques puisqu'elles ont dans leur construction, totalement pris en considération les spécificités du modèle qu'elles tentent d'expliquer. Elles sont cependant moins adaptables aux changements réguliers de modèle les rendant difficilement pérennes dans le temps. DeepLIFT est une technique créee spécifiquement pour les réseaux profonds, et sera l'objet d'étude dans la suite.

3.2.3 Compromis entre précision et interprétabilité

Lorsqu'un utilisateur décide d'utiliser un modèle d'apprentissage automatisé, il doit décider du modèle qu'il va utiliser. Notamment, il va décider d'une part le niveau de précision du modèle en fonction de ses besoins. D'autre part, il doit également en considérer le niveau d'interprétabilité. Il se trouve qu'empiriquement, comme le schématise la Figure 3.4, plus un modèle est précis, moins il est interprétable. C'est le fameux compromis entre précision et interprétabilité, très connu en XAI.

Nous en avons désormais terminé avec les concepts généraux et les définitions que l'on peut donner sur l'XAI. Mon travail ici a été de filtrer les informations et de conserver les concepts qui me semblaient pertinents dans la tentative de définition de l'XAI. Encore une fois, il s'agit d'un domaine récent qui n'a pas de bornes précisément définies. Nous avons également vu l'ensemble des techniques d'XAI qu'il pouvait exister et à cet égard, la prochaine partie a pour but d'en explorer trois d'entre elles : les valeurs de Shapley, LIME et DeepLIFT. Ces trois techniques se regroupent ces dernières années sous une même bannière qui tente de les réunir,

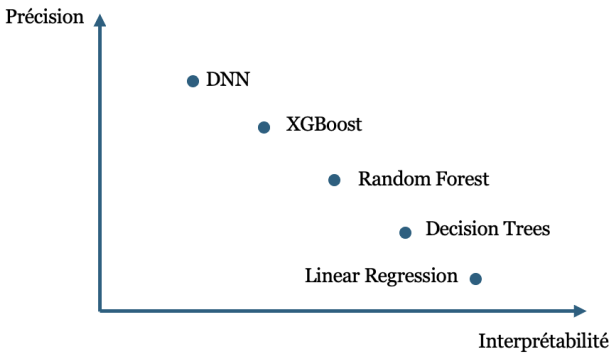


FIGURE 3.4 Compromis entre précision et interprétabilité des modèles d'IA

du nom de SHAP. C'est cette technique que nous allons explorer en profondeur. Cette partie sera donc très mathématique et une grosse part de ma recherche a consisté à formaliser de manière la plus cohérente possible, tout ce qui va suivre. Afin d'être le plus rigoureux possible dans ma démarche, j'avancerai dans les concepts en utilisant des théorèmes et des propriétés s'appuyant sur des preuves, parfois extraites de la littérature, parfois de ma propre démarche.

Avant d'entrer dans cette partie, reprenons l'énoncé de l'objectif de recherche :

Sélectionner des méthodes d'XAI adaptées aux contraintes avioniques, puis établir un formalisme mathématique unifié pour standardiser leur utilisation.

Précisons alors pourquoi SHAP a été retenue dans le cadre de cette recherche. SHAP est une technique agnostique rendant ses **explications adaptables** à différents types de modèles d'IA. Elle utilise un modèle **interprétable** et **transparent** pour substituer le modèle original et utilise un cas de référence pour analyser les situations anormales. Par ailleurs c'est une technique connue pour être **robuste**, **stable** et **précise** grâce à ses fondements théoriques. Les graphes fournis en sortie par SHAP visent du mieux possible à être **intelligible** pour **satisfaire** les utilisateurs. De nombreuses extensions à SHAP ont été également trouvées afin d'améliorer son **efficacité**. En résumé, SHAP est une technique répondant presque parfaitement aux exigences fixées par notre cadre conceptuel.

3.3 SHAP, DE LA THÉORIE DES JEUX A L'EXPLICATION DE RÉSEAUX PROFONDS

SHAP est une technique d'XAI développé par Scott Lundberg et Su-In Lee dans *A Unified Approach to Interpreting Model Predictions* [11]. Cette technique attribue une importance à chaque caractéristique d'un modèle d'apprentissage automatisé, en se basant sur leur valeur de Shapley qui est issue de la théorie de jeux. Ces contributions des caractéristiques sont alors utilisées pour expliquer les prédictions du modèle. Le long processus qui a permis d'aboutir à la technique G-DeepSHAP a commencé en 1952 lorsque Lloyd Shapley formule pour la première fois la notion de valeur de Shapley, aujourd'hui notion pilier de la théorie des jeux. Scott Lundberg et Su-In Lee en 2017, ont trouvé une application à ces valeurs de Shapley dans le contexte de l'apprentissage automatisé en créant SHAP. C'est en 2019 que les deux chercheurs, rejoints par Hugh Chen, formulent une version améliorée de SHAP, DeepSHAP, spécifiquement applicables aux réseaux d'apprentissage profond. Enfin en 2022, ces mêmes chercheurs publient une version encore plus aboutie, G-DeepSHAP, adaptée aux séries de réseaux de neurones. Ce chapitre a pour but de retracer toute cette évolution, de 1952 à aujourd'hui.

3.3.1 L'origine de la valeur de Shapley, 1952

Au début des années 1950, Lloyd Shapley, un mathématicien et économiste américain, se demande comment, dans un jeu collaboratif, donner équitablement une attribution à chaque joueur du jeu. Attribution censée refléter la valeur ajoutée du joueur à la coalition jouant au jeu, sous forme de valeur scalaire. Un exemple concret serait de se demander comme un directeur d'équipe peut récompenser ses employés équitablement, suite à un gros profit généré par la boîte, et connaissant la valeur ajoutée de chaque groupe d'employé au profit global. Shapley cherche à formuler les règles que devraient respecter cette répartition pour qu'elle soit considérée équitable par tous les joueurs, et ce, quel que soit le jeu. Dans l'article *A value for a n-person game* [52] publié dans le *Cambridge University Presse Books* le 18 Mars 1952, il formule mathématiquement ces questionnements. Il montre alors que, selon quelques axiomes, il n'existe qu'une seule solution qui réponde à son problème, qu'il appellera la valeur de Shapley. Détaillons ce formalisme, base mathématique de tout ce qui suivra jusqu'à la création de G-DeepSHAP.

Définitions d'un jeu

On appelle *jeu collaboratif*, un jeu où les joueurs coopèrent pour maximiser leur gain collectif. Dans tous ce qui suit, on ne considérera que des jeux coopératifs. Dans l'exemple des employés de bureau, on suppose donc qu'aucuns groupe d'employés ne peut se tirer vers le bas. Dans le pire des scénarios, ils ne s'apportent rien en coopérant.

Definition 3.3.1. *Soit U l'ensemble des joueurs du jeu. On dit que le jeu est adéquatement représenté par une fonction $v : P(U) \to \mathbb{R}$, où $P(U)$ représente l'ensemble des sous-ensembles de U. v doit alors vérifier :*

1. $v(\emptyset) = 0$
2. $v(S) \geq v(S \cap T) + v(S \backslash T), \forall S, T \subset U$
3. $N \subset U$ *est support de* $v \Leftrightarrow v(S) = v(S \cap N), \forall S \subset U$

On dit que v évalue la force d'une coalition dans le jeu.

La première condition s'assure de n'attribuer aucune force à la coalition vide. Par ailleurs, la deuxième condition impose au jeu d'être collaboratif en s'assurant que l'effort combiné d'un groupe doit être au moins aussi fort que la somme de sous-groupes le composant. Enfin, on définit le support d'un jeu comme étant un ensemble contenant au moins tous les joueurs ayant une contribution significative (non nulle) dans le jeu. Autrement dit, tout joueur qui n'est pas dans ce support a une contribution nulle au jeu, mais y être n'implique pas nécessairement d'en avoir une non nulle. On notera souvent N un tel support. Réciproquement, tout ensemble noté N sera considéré support. De plus, tout ensemble N' contenant N est également un support du jeu, auquel on a simplement ajouté des joueurs inutiles. En un sens, un support est l'ensemble des joueurs suffisant au jeu.

Par la suite, on confondra un jeu avec la fonction le décrivant. On parlera donc du jeu v.

Manipulations et permutations d'un ensemble

La notion de permutation d'un ensemble représente l'ensemble des mélanges possibles que l'on peut obtenir avec cet ensemble. Ainsi si $U = \{Paul, Pierre, Jacques\}$, on note $\Pi(U)$ l'ensemble des permutations de U. Ici, $\Pi(U) = \{$(Paul, Pierre, Jacques), (Paul, Jacques, Pierre), (Pierre, Paul, Jacques),(Pierre, Jacques, Paul), (Jacques, Paul, Pierre), (Jacques, Pierre, Paul)$\}$. A noter que pour un ensemble de taille n, il y a $n!$ permutations possibles.

Pour manipuler nos ensembles de joueurs, lorsque l'on prend une coalition S dans U, on considère la position des joueurs dans l'ensemble U pour créer S. Mettons que l'on veuille étudier le binôme $\{Pierre, Jacques\}$, on prendra $S = (2, 3)$ dans $U = \{Paul, Pierre, Jacques\}$. Cependant, si on travaille avec une permutation $\pi = \{3, 1, 2\}$, transformant U en $\pi(U) = \{Jacques, Paul, Pierre\}$, ce même binôme sera maintenant représenté par une coalition équivalente notée $\pi S = \pi(S) = (3, 1)$. Cette indexation des joueurs est utile à des fins mathématiques, mais il est évidemment nécessaire qu'au final, le binôme, qu'importe l'ordre utilisé, ait la même force. Or, à priori, $v(S) \neq v(\pi S)$. C'est pourquoi l'on doit définir une nouvelle fonction associée à cette nouvelle réindexation des joueurs.

Definition 3.3.2 (Jeu abstrait)**.** *Soit une permutation π dans $\Pi(U)$. On appelle jeu abstrait du jeu v, le jeu πv vérifiant :* $\pi v(\pi S) = v(S), \quad \forall S \subset U$

Ce nouveau jeu πv permet d'examiner le jeu sous une autre organisation des joueurs, tout en conservant les mêmes répartitions des gains.

Axiomes

Dans un jeu v, on cherche à définir un vecteur $\varphi(v)$ de même taille que U qui contient les attributions finales de chaque joueur au jeu. $\varphi_i(v)$ décrivant ainsi l'attribution du joueur i et l'ensemble des $\varphi_i(v)$ est ce qu'on appelle les **valeurs de Shapley** du jeu v. On souhaite que ces valeurs respectent quatre axiomes : la symétrie, l'efficacité, l'agrégation et le joueur nul. On appelle *contribution* d'un joueur au sein d'une coalition, la différence de résultat au jeu de cette coalition, avec et sans le joueur. A différencier de *l'attribution*, représentée par une valeur de Shapley, qui est la valeur finale recherchée, décrivant la valeur ajoutée du joueur au jeu entier et non pas par rapport à une coalition spécifique. Les quatre axiomes à respecter par les $\varphi_i(v)$ sont donc :

— **Symétrie :** L'attribution d'un joueur au jeu est indépendante de la permutation qui a été choisie, c'est une propriété intrinsèque du jeu :

$$\forall \pi \in \Pi(U), \varphi_{\pi i}(\pi v) = \varphi_i(v), \forall i \in U \tag{3.1}$$

Si nos employés sont initialement triés par âge, mais que l'on veut par soucis pratique, les réorganiser par taille, cela ne doit pas changer la répartition des gains de chacun au final.

— **Efficacité :** Pour une coalition, support du jeu, la somme des attributions de chaque joueur de cette coalition doit être égale à la force de la coalition complète :

$$\forall N \subset U, \sum_{i \in N} \varphi_i(v) = v(N) \tag{3.2}$$

Si l'entreprise fait un profit de 1000$, la somme des attributions des joueurs doit en effet valoir 1000$.

— **Linéarité ou loi de l'agrégation :** Pour deux jeux indépendants, les attributions de chaque jeu s'additionnent joueur par joueur :

$$\varphi(v + w) = \varphi(v) + \varphi(w), \forall v, w \tag{3.3}$$

Si l'entreprise fait deux profits différents, le gain d'un employé dans le profit total est simplement la somme de ses gains aux deux profits.

— **Joueur nul :** Si un joueur n'apporte aucune contribution à aucune coalition, alors son attribution au jeu doit être nul :

$$v(S \cup \{i\}) = v(S), \forall S \subset U \implies \varphi_i(v) = 0 \tag{3.4}$$

L'employé inutile ne reçoit rien.

Valeurs de Shapley

L'Annexe A détaille la démonstration via l'utilisation de trois lemmes préliminaires du théorème suivant :

Theorem 3.3.3. *Un unique φ existe satisfaisant tous les axiomes :*

$$\varphi_i(v) = \sum_{S \subseteq U, S \ni i} \underbrace{\gamma_U(S)}_{poids} \underbrace{[v(S) - v(S \setminus \{i\})]}_{contribution}, \quad \forall i \in U \tag{3.5}$$

Où $\gamma_U(S) = \frac{(|S|-1)!(|U|-|S|)!}{|U|!} = \frac{1}{|U|}\binom{|U|-1}{|S|-1}^{-1}$.

Cette formule énonce que l'attribution d'un joueur au jeu est une moyenne pondérée des contributions de ce joueur à l'ensemble des coalitions auxquelles il peut participer. L'attribution d'un employé se calcule donc en mesurant les contributions de cet employé dans tout sous-groupe de l'entreprise. Puisque pour toute coalition S ne contenant pas le joueur i, le terme $v(S) - v(S \setminus \{i\})$ dans la somme est nul, on peut simplement sommer sur $U \setminus \{i\}$:

$$\varphi_i(v) = \sum_{S \subseteq U \setminus \{i\}} \gamma'_U(S) \left[v(S \cup \{i\}) - v(S) \right], \quad \forall i \in U \tag{3.6}$$

Où $\gamma'_U(S) = \frac{(|S|)!(|U|-|S|-1)!}{|U|!} = \frac{1}{|U|} \binom{|U|-1}{|S|}^{-1}$.
Le changement du coefficient de poids γ s'explique simplement par le fait que dans la première formulation, on cherchait $|S|-1$ éléments à ajouter à $\{i\}$, à choisir dans $U \setminus \{i\}$ pour former S, tandis que dans la deuxième, on choisit $|S|$ éléments à choisir dans $U \setminus \{i\}$.
Cette formule admet une autre écriture si l'on ne considère non plus que l'on somme sur tous les sous-ensemble S, mais sur toutes les permutations π de U et où l'on considère l'ensemble P^i_π comment étant tous les éléments situés avant l'élément i dans la permutation π. Au final, dans chaque permutation, la place de l'élément i détermine le cardinal de l'ensemble et la permutation en détermine les éléments. Il y a donc une équivalence entre les deux écritures.

Definition 3.3.4 (Formulation en permutation)**.** *Les valeurs de Shapley admettent une formulation équivalente, raisonnant sur l'ensemble des permutations comme :*

$$\varphi_i(v) = \frac{1}{|U|!} \sum_R \left(v(P^i_R) - v(P^i_R \setminus \{i\}) \right) \tag{3.7}$$

Exemple d'une entreprise

Voyons un exemple issu de Wikipédia https://en.wikipedia.org/wiki/Shapley_value. Soit une entreprise dont le directeur est le joueur o et les employés sont les joueurs $w_1, ..., w_m$ tel que $U = \{o, w_1, ..., w_m\}$. On considère trois types de joueurs :

- Le directeur qui ne contribue pas directement, mais qui a besoin d'être présent dans une contribution pour que ses employés travaillent.
- Les employés $w_1, ..., w_n$ où $n < m$ qui ne travaillent pas si le directeur n'est pas là et qui fournisse une valeur de p si le patron est là.
- Les employés $w_{n+1}, ..., w_m$ qui ne travaillent jamais.

On cherche à répartir les attributions de chaque participant au jeu correspondant au gain produit par l'entreprise. On constate que l'ensemble $N = \{o, w_1, ..., w_n\}$ est support du jeu puisqu'il contient tous les employés qui peuvent aider à la production de l'entreprise. On travaille donc sur N en affirmant d'office que $\forall i \in n-1, ..., m, \varphi_{w_i} = 0$ d'après l'axiome du

joueur nul. Formellement, on a donc :

$$\forall S \subseteq N, v(S) = \begin{cases} (|S|-1)p, \text{si } o \in S \\ 0, \text{sinon} \end{cases} \tag{3.8}$$

On calcule alors les valeurs de Shapley :

$$\forall i \in 1, ..., n, \varphi_i(v) = \sum_{S \subseteq U \setminus \{i\}} \gamma'_U(S) \left[v(S \cup \{i\}) - v(S) \right] \tag{3.9}$$

$$= \sum_{S \subseteq U \setminus \{i\}, S \ni o} \gamma'_U(S) \left[v(S \cup \{i\}) - v(S) \right] \tag{3.10}$$

$$= p \sum_{S \subseteq U \setminus \{i\}, S \ni o} \gamma'_U(S) \tag{3.11}$$

$$= \frac{p}{2} \tag{3.12}$$

En effet :

$$1 = \sum_{S \subseteq U \setminus \{i\}} \gamma'_U(S), \tag{3.13}$$

$$= \sum_{S \subseteq U \setminus \{i\}, S \ni o} \gamma'_U(S) + \sum_{S \subseteq U \setminus \{i\}, o \notin S} \gamma'_U(S) \tag{3.14}$$

$$= 2 \sum_{S \subseteq U \setminus \{i\}, o \notin S} \gamma'_U(S) \tag{3.15}$$

Pour le directeur :

$$\varphi_o(v) = \sum_{S \subseteq U \setminus \{o\}} \gamma'_U(S) \left[v(S \cup \{o\}) - \underbrace{v(S)}_{0} \right] \tag{3.16}$$

$$= \sum_{S \subseteq U \setminus \{o\}} \gamma'_U(S) |S| p \tag{3.17}$$

$$= \frac{np}{2} \tag{3.18}$$

On vérifie bien que $\sum_{i \in N} \varphi_i(v) = np = v(N)$ qui vérifie l'axiome d'efficacité. On voit à travers cet exemple la pertinence de travailler sur des coalitions de joueurs et pas simplement sur les performances des joueurs seuls. Effectivement, $v(\{o\}) = 0$, pourtant on voit bien que le directeur a une importance cruciale au jeu. Lui donner une attribution nulle serait donc une très mauvaise répartition des gains du jeu.

3.3.2 Axiome de monotonicité, 1982

En mars 1982, Hobart Peyton Young formule dans *Monotonic solutions of cooperative games* [53] un nouvel axiome permettant de s'affranchir des axiomes de linéarité et de nullité : l'axiome de monotonicité.

Monotonicité L'axiome de monotonicité stipule que si la force d'une coalition augmente, toute autre coalition ayant des forces inchangées, l'attribution de chaque joueur de cette coalition ne peut qu'augmenter :

$$\forall S' \subseteq U, (v(S') \geq w(S'), v(S) = w(S), \forall S \neq S' \implies \varphi_i(v) \geq \varphi_i(w), \forall i \in S') \tag{3.19}$$

Monotonicité forte La monotonicité forte stipule que si la contribution d'un joueur augmente dans toutes les coalitions, alors l'attribution du joueur dans le jeu ne peut qu'augmenter :

$$\forall i \in U, (v^i(S) \geq w^i(S), \forall S \subset U \implies \varphi_i(v) \geq \varphi_i(w)) \tag{3.20}$$

où $v^i(S) = \begin{cases} v(S \cup \{i\}) - v(S) & \text{si } i \notin S, \\ v(S) - v(S \backslash \{i\}) & \text{si } i \in S. \end{cases}$ est la contribution de i dans S.

On peut d'ailleurs introduire cette nouvelle notation dans l'écriture des valeurs de Shapley :

$$\varphi_i(v) = \sum_{S \subseteq U, S \ni i} \gamma_U(S) v^i(S), \quad \forall i \in U \tag{3.21}$$

Young donne également une définition de ce que représente une allocation :

Definition 3.3.5. *Une allocation est une fonction ψ qui s'applique à un jeu v et à un ensemble de joueurs N tel que :*

$$\sum_{i \in N} \psi_i(v) = v(N) \tag{3.22}$$

Cette définition est simplement l'équivalent de l'axiome d'efficacité. Ainsi, lorsque l'on parle d'allocation, l'axiome d'efficacité est implicitement pris en compte.

Theorem 3.3.6. *La valeur de Shapley est l'unique allocation symétrique qui est fortement monotone.*

Ce théorème est une réduction du théorème 3.3.3 de quatre à trois axiomes. La preuve du

théorème est détaillée en annexe A. Plus tard, Scott Lundberg et Su-In Lee prouveront même que l'axiome de monotonicité implique l'axiome de symétrie. La preuve est en annexe A. Le théorème final stipule donc que la valeur de Shapley est l'unique allocation qui est strictement monotone.

3.3.3 Shapley Additive exPlanations, 2017

Les valeurs de Shapley trouvent en 2017 leur application concrète dans une nouvelle théorie du nom de SHAP, pour *Shapley Additive exPlanations*, spécialisée dans l'apprentissage automatisé, spécifiquement dans l'explication de modèle. Lundberg et Lee, deux chercheurs de l'université de Washington présentèrent les prémices de SHAP dans *An unexpected unity among methods for interpreting model predictions* [54] en 2016 pour publier la version finale de SHAP dans *A Unified Approach to Interpreting Model Predictions* [11] en novembre 2017.

Méthode additive d'attribution de caractéristique

Soit un modèle d'apprentissage automatisé f et des caractéristiques $x_1, ..., x_n$. Une attribution de caractéristique assigne une valeur scalaire à chacune d'elles, reflétant l'importance de cette dernière sur la prédiction du modèle. Pour se faire une première idée, considérons un modèle linéaire tel que $f(x) = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n$. Un tel modèle est dit interprétable, car chaque caractéristique x_i est directement reliée à la prédiction par un unique paramètre β_i. Dans ce cas, l'attribution de caractéristique consiste simplement à attribuer à chaque x_i, son paramètre β_i. Pour des modèles d'apprentissage plus complexes, la fonction de prédiction n'est pas directement interprétable et il convient de définir un nouvel modèle en parallèle, plus simple, que l'on appelle le modèle explicatif et qui cherche à approximer le modèle original en le rendant interprétable. Pour la suite, appelons f le modèle original et g le modèle explicatif. SHAP est une méthode d'attribution de caractéristique locale, c'est-à-dire que le modèle g va être construit afin d'expliquer une prédiction spécifique $f(x)$. Ainsi pour tout $x' \neq x$ il faudra construire un nouveau modèle explicatif pour expliquer la nouvelle prédiction. On pourrait noter g_x une telle fonction pour souligner sa dépendance à x, mais nous garderons g par simplicité.

Dans cette partie nous nous intéressons à une catégorie de méthodes d'explication donnant une forme particulière à g, il s'agit des méthodes additives d'attribution de caractéristique. Un modèle d'apprentissage automatisé prend en entrée un vecteur $x \in \mathbb{R}^N$, où N est le nombre de caractéristiques du modèle. Pour la mise en parallèle avec les notions de théorie des jeux développées dans les parties précédentes, le jeu est ici représenté par un modèle f et les joueurs sont les caractéristiques du modèle. Désormais, nous dirons le modèle f pour désigner le jeu v.

Choisir une coalition de joueurs revient alors à choisir l'ensemble des caractéristiques qui seront données au modèle. Il conviendra de définir ce qu'il advient des caractéristiques absentes. Pour le moment, considérons un vecteur $x' \in \{0,1\}^N$ dont l'élément x'_i vaut 1 si la caractéristique est présente dans la coalition et 0 si elle est absente. On appelle x' la variable binaire de x. $\{0,1\}^N$ est alors l'espace binaire de notre modèle indiquant les caractéristiques présentes. g est construite pour aller de l'espace binaire dans $\mathbb{R}$ soit $g : \{0,1\}^N \to \mathbb{R}$. Définissons de plus $h_x : \{0,1\}^N \to \mathbb{R}^N$ comme étant une fonction de correspondance spécifique à x, telle que $h_x(x') = x$, donc permettant le passage de l'espace binaire à l'espace des caractéristiques $\mathbb{R}^N$. A noter que bien que x' contienne moins d'information que x, puisque h_x est construite spécifiquement sur x, elle peut le reconstruire à partir de x'. Soit maintenant $z' \in \{0,1\}^N$ un élément de l'espace binaire, les méthodes explicatives locales s'assurent que $g(z') = f(h_x(z'))$ lorsque $z' \approx x'$. Autrement dit, tout élément z' dans l'espace binaire, proche de x', la version binaire de x, est évalué par g de telle sorte à obtenir une prédiction proche de celle faite par f, sur un voisin $z = h_x(z')$ de x.

Definition 3.3.7. *Une méthode additive d'attribution de caractéristique a un modèle explicatif linéaire en ses variables binaires :*

$$g(z') = \phi_0 + \sum_{i \in N} \phi_i z'_i \tag{3.23}$$

Où, $\phi_i \in \mathbb{R}$ et ϕ_0 est la prédiction du modèle quand aucune caractéristique n'est présente (coalition vide).

Les méthodes présentant cette propriété donnent ainsi une attribution à chaque caractéristique, dont la somme pondérée par les variables binaires de présence/absence, donne la valeur de la prédiction.

Lien avec les valeurs de Shapley

La notation précédente ϕ est différente de la notation utilisée depuis le début pour désigner les valeurs de Shapley φ, puisqu'a priori, les deux n'ont pas de lien. Nous allons voir maintenant qu'en réalité, elles sont égales. On cherche donc une méthode explicative g pour expliquer $f(x)$. On souhaite que cette méthode soit une méthode additive d'attribution de caractéristique. Par ailleurs, on souhaite qu'elle respecte trois propriétés :

— **Justesse locale :** Le modèle explicatif doit parfaitement correspondre au modèle

original quand il est évalué en x' :

$$f(x) = g(x') = \phi_0 + \sum_{i \in N} \phi_i x'_i \tag{3.24}$$

— **Absence :** Une caractéristique absente du modèle n'a aucun impact :

$$x'_i = 0 \implies \phi_i = 0 \tag{3.25}$$

Cette propriété permet de simplifier l'expression de g en :

$$g(z') = \phi_0 + \sum_{i \in M} \phi_i z'_i \tag{3.26}$$

où M désigne le nombre de caractéristiques présentes. Les caractéristiques absentes ayant une attribution nulle peuvent être retirées de la somme.

— **Cohérence :** Notons $f_x(z') = f(h_x(z'))$ comme étant la prédiction du modèle évalué selon la variable binaire z' à travers la fonction de correspondance h_x. Soit $z' \backslash i$ la notation pour désigner que la caractéristique i est absente soit $z'_i = 0$. Alors pour tout modèles f et f' :

$$f'_x(z') - f'_x(z' \backslash i) \geq f_x(z') - f_x(z' \backslash i), \forall z' \in \{0,1\}^N \implies \phi_i(f', x) \geq \phi_i(f, x) \tag{3.27}$$

Autrement dit, si un modèle change de sorte que la contribution d'une caractéristique augmente ou reste la même, indépendamment des autres caractéristiques, l'attribution de cette caractéristique ne doit pas diminuer.

Theorem 3.3.8. *Un seul modèle g satisfait les 3 propriétés ci-dessus, celui dont les coefficients sont :*

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!} [f_x(z') - f_x(z' \backslash i)] \tag{3.28}$$

où $|z'|$ est le nombre de coefficients non nuls dans z', et $z' \subseteq x'$ représente tous les sous-ensembles dont les coefficients non nuls sont des sous-ensembles des coefficients non nuls de x'. $|z'|$ varie donc entre 0 et M.

Pour schématiser, l'entrée x contient un certain nombre de voyants allumés et d'autres éteints. On cherche l'attribution de chaque voyant allumé à la prédiction du modèle. Les voyants éteints ayant une attribution nulle. Pour cela, on effectue toutes les combinaisons de

voyants on/off parmi ceux qui sont allumés dans x, et on calcule la contribution de chaque combinaison avec et sans le voyant i. i est donc nécessairement parmi ceux qui sont allumés, sinon $\phi_i(f, x) = 0$. Les ϕ_i sont donc l'exact équivalent des valeurs de Shapley dans le cas de modèle d'apprentissage. Les axiomes de justesse local et de cohérence sont exactement ceux d'efficacité et de monotonicité dont on a prouvé qu'ils étaient suffisants à déterminer une unique solution pour ϕ. Un corollaire de ce théorème est que toute méthode qui n'est pas basée sur les valeurs de Shapley, viole la justesse locale ou la cohérence.

Valeurs SHAP

Pour compléter l'ensemble, il convient maintenant de définir ce qu'implique réellement la présence ou l'absence d'une caractéristique. Plus concrètement, on se demande que vaut la prédiction $f_x(z') = f(h_x(z'))$. Cette valeur correspond en pratique à ce que donnerait le modèle comme prédiction, si on ne lui donnait que les caractéristiques présentes dans z'. SHAP calcule l'espérance de la prédiction sur toutes les valeurs absentes, connaissant les valeurs présentes. Autrement dit, pour une entrée binaire z', tel que $h_x(z') = z_S$. z_S est un vecteur de $\mathbb{R}^N$ dont les caractéristiques non incluses dans S ont des valeurs manquantes et celles inclus dans S ont leur valeur de x dans $\mathbb{R}$. Ainsi :

$$f_x(z') = E[f(z)|z_S] \tag{3.29}$$

Détaillons le calcul des premiers termes pour illustrer. Dans le cas où aucune caractéristique n'est présente, soit $z' = 0^N$ et $S = \emptyset$, la prédiction vaut : $f_x(z') = E[f(z)]$ qui est simplement l'espérance du modèle sur toutes les caractéristiques et $g(z') = \phi_0$. Ainsi par justesse locale :

$$\phi_0 = E[f(z)] \tag{3.30}$$

Lorsqu'une seule caractéristique i est présente, z' a des 0 partout et un 1 à la position i, et $S = \{i\}$. Alors :

$$\phi_1 = f_x(z') - f_x(\emptyset) = E[f(z)|z_i = x_i] - \phi_0 \tag{3.31}$$

On peut ainsi construire tous les termes ϕ_i jusqu'à converger, pour le modèle explicatif, vers la prédiction original : $g(x') = \phi_0 + \sum_{i \in N} \phi_i = f(x)$. La Figure 3.5 illustre ce calcul. Les espérances mentionnées ici peuvent être faites sur l'ensemble de la base de données ou alors sur un échantillon que l'on considère représentatif. Les parties suivantes approfondiront ces notions de base de données représentative du problème.

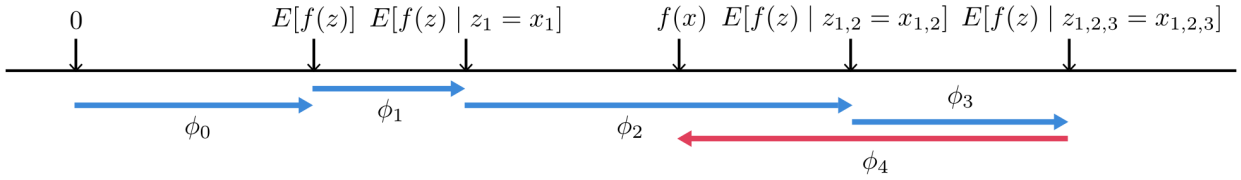


FIGURE 3.5 Valeurs SHAP : Explications graphiques des calculs récursifs des ϕ_i, permettant de converger de $E[f(z)]$ vers $f(x)$

Nous en avons terminé avec le formalisme portant sur SHAP. Un problème pratique persiste cependant ; la formulation des valeurs SHAP comme présentées dans le théorème 3.3.8 est très coûteuse à calculer en pratique. Il faut en effet sommer sur tous les sous-ensembles possibles de caractéristiques qui, si effectué de front, a une complexité en factorielle du nombre de caractéristiques, ce qui est bien trop coûteux. Faisons alors un détour en étudiant deux autres techniques d'explicabilité très répandues dans la littérature : DeepLIFT et LIME, puis voyons comment ces techniques, combinées à SHAP, permettent de résoudre notre problème.

3.3.4 LIME, 2016

En 2016, Marco Tulio Ribeiro, Sameer Singh et Carlos Guestrin publient dans *"Why Should I Trust You ?" Explaining the Predictions of Any Classifier* [12], une nouvelle technique d'explicabilité appliquée aux modèles de classification. La technique s'appelle LIME pour *Local Interpretable Model-agnostic Explanations*. Il s'agit d'une technique d'explication locale, comme SHAP, c'est-à-dire qu'elle vise à expliquer une seule prédiction et non l'ensemble du jeu de données. De plus, c'est une technique agnostique, c'est-à-dire qu'elle est indépendante du modèle sur lequel elle s'applique, la rendant très polyvalente. Son objectif est d'ajouter un modèle substitut interprétable par-dessus le modèle original, en s'assurant que localement autour de la prédiction, le modèle substitut est fidèle au modèle original. L'idée est très proche de celle de SHAP, son approche est cependant bien différente.

Formalisme

Comme souvent, lorsque l'on crée un modèle interprétable, si $x \in \mathbb{R}^N$ est l'entrée du modèle original, on souhaite que notre modèle interprétable prenne plutôt en entrée une version simplifiée $x' \in \{0,1\}^N$. Soit $g \in G$ un modèle interprétable où G représente la classe des modèles interprétables (linéaires, arbre de décision...). Si f est notre modèle original, on a donc : $f : \mathbb{R}^N \to \mathbb{R}$ et $g : \{0,1\}^N \to \mathbb{R}$. Les modèles dans G sont par construction tous

interprétables, mais ont des niveaux de complexité qui peuvent varier. On note alors $\Omega(g)$ la complexité du modèle g. Pour un modèle en arbre de décision, la complexité peut être la profondeur de l'arbre mais pour un modèle linéaire, la complexité peut simplement être le nombre de coefficients non nuls. Plus la complexité est élevée, moins le modèle est interprétable. Soit $z \in \mathbb{R}^N$, on définit la fonction $\pi_x(z)$ la fonction qui mesure la proximité entre x et z. π_x définit ainsi un voisinage de x. Soit enfin $\mathcal{L}(f, g, \pi_x)$ une fonction de coût, qui évalue à quel point le modèle g approxime correctement f, au voisinage de x. Ainsi, puisque $\mathcal{L}$ évalue la fidélité de g, et Ω évalue son interprétabilité, LIME produit un modèle explicatif final, relatif à l'entrée x, notée $\xi(x)$, tel que :

$$\xi(x) = argmin_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{3.32}$$

Cette formulation est très générique et il conviendra de spécifier quel type de famille de modèles interprétables G on souhaite utiliser, comment définir le voisinage π_x ainsi que la fonction d'approximation $\mathcal{L}$. Si x' est la représentation binaire de x, on crée un ensemble $\mathcal{Z}$ constitué d'éléments $z' \in \{0, 1\}^N$ représentant des perturbations autour de x'. Soit z leur représentation dans l'espace des caractéristiques tel que $h_x(z') = z$ (pour reprendre les notations de SHAP). On souhaite donc que $f(z) = f_x(z')$ soit d'autant plus proche de $g(z')$ que (z', z) est proche de (x', x) . La proximité entre $f(z)$ et $g(z')$ est mesurée par $\mathcal{L}$ et celle entre z et x par π_x. En x, on souhaite toujours vérifier la justesse locale telle que $f(x) = g(x')$.

Classe de modèles linéaires

Dans ce qui suit, on considère que G représente la classe des modèles linéaires (donc bien interprétables). Ainsi g est de la forme $g(z') = \sum_i w'_i z'_i$. Soit D une mesure de distance (comme la norme L^2 par exemple pour des distances entre pixels). On choisit alors comme voisinage pour x une fonction noyau exponentiel :

$$\pi_x(z) = exp(-\frac{D(x, z)^2}{\sigma^2}) \tag{3.33}$$

σ est l'épaisseur de peau et définit l'épaisseur du voisinage. On utilise pour $\mathcal{L}$ la fonction pondérée locale de perte carrée :

$$\mathcal{L}(f, g, \pi_x) = \sum_{z' \in \mathcal{Z}} \pi_x(z)(f(z) - g(z'))^2 \tag{3.34}$$

De plus, pour la complexité du modèle, on laisse le choix à l'utilisateur de fixer le nombre d'éléments noté K qu'il y aura dans le voisinage de x. Dans le cas d'un texte, cela permet de

limiter l'explication d'un mot à ses K voisins les plus proches, idem pour le cas d'une image. Plus exactement, K définit le nombre de voisins qui auront un coefficient non nul dans le modèle linéaire g. Ainsi, le voisinage peut être de taille $N > K$ tant qu'il y a au moins $N - K$ coefficients nuls. On définit alors la complexité du modèle comme étant infini si il y a plus que K coefficient non nuls :

$$\Omega(g) = \infty 1[|w_g| > K] \tag{3.35}$$

$|w_g|$ représente le nombre d'éléments non nuls de w_g. On peut alors entraîner un modèle linéaire g avec des caractéristiques z'_i dans $\mathcal{Z}$ et une cible $f(z)$. Finalement, sous le formalisme présenté ici, on peut trouver la forme de g avec la méthode des moindres carrés.

3.3.5 KernelSHAP, 2017/2021

Dans cette partie, nous allons tenter de faire converger les théories de SHAP et *Linear LIME*. *Linear LIME* est la version de LIME dans laquelle on donne une forme linéaire à g. Dans *A Unified Approach to Interpreting Model Predictions* [11], les auteurs donnent une première version de KernelSHAP et dans *Improving KernelSHAP : Practical Shapley Value Estimation via Linear Regression* [55] Ian Covert et Su-In-Lee approfondissent le sujet. L'idée de regrouper ces deux théories vient du constat suivant : dans LIME, la forme donnée au modèle $g(z') = \sum_i w_i z'_i$ montre qu'il s'agit bien d'une méthode additive d'attribution de caractéristique. Pourtant, les w_i de *Linear LIME* que nous avons trouvé suite aux choix pour $\mathcal{L}, \Omega$ et π_x, sont bien différents des valeurs de Shapley φ_i, issus de SHAP. Le corollaire du théorème 3.28 stipule que les valeurs de LIME violent une des trois propriétés parmi la justesse locale, l'absence et la cohérence. Considérons d'abord une caractéristique absente dans l'espace binaire, soit $z'_i = 0$. Alors l'algorithme LIME ne va même pas la choisir parmi ses K représentants et ne lui attribuera pas de poids w_i. Autrement dit, elle lui attribue d'office un poids nul. La propriété d'absence est donc vérifiée, mais une des deux autres propriétés est forcément violée. Puisque le calcul de g dans LIME est bien plus rapide que celui de g dans SHAP, mais que l'on souhaite que g respecte les propriétés imposées par SHAP, on se demande comment choisir $\mathcal{L}, \Omega$ et π_x pour que la forme finale de g dans LIME, respecte les propriétés de SHAP. Autrement dit pour que $g_{LIME} = g_{SHAP}$.

Theorem 3.3.9. *Pour obtenir un modèle g, linéaire en ses caractéristiques, et qui est*

solution de l'équation 3.32, l'unique possibilité pour $\mathcal{L}, \Omega$ et $\pi_{x'}$ est :

$$\Omega(g) = 0 \tag{3.36}$$

$$\pi_{x'}(z') = \binom{M}{|z'|}^{-1} \frac{M-1}{|z'|(M-|z'|)} \tag{3.37}$$

$$\mathcal{L}(f, g, \pi_{x'}) = \sum_{z' \in \mathcal{Z}} [f(z) - g(z')]^2 \pi_{x'}(z'), \tag{3.38}$$

$$\tag{3.39}$$

où $|z'|$ est le nombre d'éléments non nuls de z'.

KernelSHAP permet donc de calculer les valeurs SHAP, sans passer par la formule coûteuse 3.28, en utilisant la méthode des moindres carrés de LIME avec des paramètres bien choisis. De nombreuses utilisations de SHAP dans la littérature utilisent KernelSHAP. Voyons cependant dans la prochaine section une autre technique d'XAI permettant elle aussi un calcul optimisé des valeurs SHAP.

3.3.6 DeepLIFT, 2017

DeepLIFT (*Deep learning Important Features*, développée dans *Learning Important Features Through Propagating Activation Differences* [56] [57], est une méthode d'explication de réseaux de neurones profonds dont le principe est de propager la contribution de chaque neurone du réseau, à chaque caractéristique qui lui est reliée. En comparant l'activation d'un neurone à sa valeur de "référence", DeepLIFT assigne un score de contribution à la prédiction à chaque neurone. Une seule passe en arrière suffit ainsi à donner ce score et par voie de conséquence, donne une attribution à chaque caractéristique. Cette partie explore en détail le fonctionnement de DeepLIFT.

Contributions

Soit t un neurone du réseau et t^0 son activation de référence. On note $\Delta t = t - t^0$ la différence à la référence du neurone t. On définit également la contribution d'un neurone x à t comme étant la variable $C_{\Delta x \Delta y}$.

Proposition 3.3.10 (Sommation à delta)**.** *Soient $x_1, x_2 ... x_n$ des neurones nécessaires et suffisants au calcul de l'activation de t. Alors : $\sum_i C_{\Delta x_i \Delta t} = \Delta t$*

$C_{\Delta x \Delta t}$ représente ainsi à quel point l'activation de x_i influence celle de t.

Proposition 3.3.11 (Contribution positive/négative)**.** *Soit un neurone y, et Δy sa différence à la référence. Soit Δy^+ et Δy^- ses composants positifs et négatifs tels que $\Delta y = \Delta y^+ + \Delta y^-$. Alors, pour un neurone t quelconque, toute contribution de y à t peut se décomposer en composante positive et négative.*

$$C_{\Delta y \Delta t} = C_{\Delta y^+ \Delta t} + C_{\Delta y^- \Delta t} \tag{3.40}$$

Multiplicateurs

Definition 3.3.12. *Pour un neurone x, contribuant à un neurone t, on définit le multiplicateur $m_{\Delta x \Delta y}$ comme :*

$$m_{\Delta x \Delta t} = \frac{C_{\Delta x \Delta t}}{\Delta x} \tag{3.41}$$

Le multiplicateur est en quelque sorte l'équivalent de la dérivée partielle $\frac{\partial t}{\partial x}$, mais avec des différences finis et non infinitésimales. Lorsque l'on utilise la sommation à delta aux multiplicateurs, on prouve la règle suivante :

Proposition 3.3.13 (Règle de chaînage)**.** *Soient $x_1 ... x_n$ la couche de neurones d'entrée, $y_1 ... y_n$ la couche de neurones cachée et t le neurone de sortie. Alors, les multiplicateurs respectent la règle de chaînage suivante :*

$$\sum_j m_{\Delta x_i \Delta y_j} m_{\Delta y_j \Delta t} = m_{\Delta x_i \Delta t}, \quad \forall i \tag{3.42}$$

Avec $C_{\Delta x_i \Delta y_j}$ et $C_{\Delta y_j \Delta t}$ vérifiant tous deux la sommation à delta, on montre en annexe A que définir le multiplicateur $m_{\Delta x_i \Delta t}$ suivant la règle de chaînage permet également à $C_{\Delta x_i \Delta t}$ de vérifier la sommation à delta. Cette propriété poursuit l'analogie avec les dérivées partielles et leur règle de chaînage. Cette règle permet ainsi une propagation des contributions à travers le réseau.

Définir la référence

Pour connaître les activations de référence des neurones, il suffit de choisir une entrée de référence x^0, et de la propager en avant dans tout le réseau. L'entrée x^0 dépend beaucoup du problème considéré. Pour un modèle prenant en entrée des images, soit un ensemble de pixels, on peut définir x^0 comme l'image noire (tous les pixels à 0) ou l'image blanche (tous à 1). Pour des données de vol, on peut considérer les valeurs typiques de vol classiques. Il faut donc parfois avoir accès aux bases de données pour trouver de bonnes valeurs de référence.

Cette référence permet de définir par rapport à quoi on cherche à expliquer la prédiction. Pour le diagnostic de maladie, la référence sera les caractéristiques d'un individu sain, les neurones dont la différence à l'activation est grande sont donc les neurones contribuant le plus au diagnostic anormal.

Attribution des contributions

Jusqu'ici, nous n'avons jamais vraiment expliqué comment calculer les valeurs des contributions $C_{\Delta y \Delta t}$. L'objectif de cette partie est de donner des méthodes d'attribution en fonction des types de réseaux de neurones étudiés.

Règle linéaire Soit y un neurone, linéaire en ses entrées x_i modulo un biais b tel que : $y = b + \sum_i w_i x_i$, respectivement $\Delta y = b + \sum_i w_i \Delta x_i$. On peut alors directement affirmer que $C_{\Delta x_i \Delta y} = w_i \Delta x_i$ et $m_{\Delta x_i \Delta y} = w_i$. On peut cependant aller plus loin en utilisant la décomposition en éléments négatif/positif :

$$\Delta y^+ = \sum_i 1(w_i \Delta x_i > 0) w_i \Delta x_i \tag{3.43}$$

$$= \sum_i 1(w_i \Delta x_i > 0) w_i (\Delta x_i^+ + \Delta x_i^-) \tag{3.44}$$

Et

$$\Delta y^- = \sum_i 1(w_i \Delta x_i < 0) w_i \Delta x_i \tag{3.45}$$

$$= \sum_i 1(w_i \Delta x_i < 0) w_i (\Delta x_i^+ + \Delta x_i^-) \tag{3.46}$$

On obtient alors les contributions suivantes :

$$C_{\Delta x_i^+ \Delta y^+} = 1(w_i \Delta x_i > 0) w_i \Delta x_i^+ \tag{3.47}$$

$$C_{\Delta x_i^- \Delta y^+} = 1(w_i \Delta x_i > 0) w_i \Delta x_i^- \tag{3.48}$$

$$C_{\Delta x_i^+ \Delta y^-} = 1(w_i \Delta x_i < 0) w_i \Delta x_i^+ \tag{3.49}$$

$$C_{\Delta x_i^- \Delta y^-} = 1(w_i \Delta x_i < 0) w_i \Delta x_i^- \tag{3.50}$$

$$\tag{3.51}$$

De la même manière on peut déduire les multiplicateurs comme étant : $m_{\Delta x_i^+ \Delta y^+} = m_{\Delta x_i^- \Delta y^+} = 1(w_i \Delta x_i > 0) w_i$ et $m_{\Delta x_i^+ \Delta y^-} = m_{\Delta x_i^- \Delta y^-} = 1(w_i \Delta x_i < 0) w_i$. Par convention, on choisit

$m_{\Delta x_i^- \Delta y^+} = m_{\Delta x_i^- \Delta y^-} = 0.5 w_i$ si $\Delta x_i^- = 0$ et $m_{\Delta x_i^+ \Delta y^+} = m_{\Delta x_i^+ \Delta y^-} = 0.5 w_i$ si $\Delta x_i^+ = 0$.

Voyons maintenant comment les multiplicateurs se propagent à travers une couche dense. Soit ΔX et ΔY les matrices de taille respectivement $(caractéristiques, échantillon)$ et $(hidden, échantillon)$ et W la matrice de poids de la première couche de taille $(hidden, caractéristiques)$. Ainsi $\Delta Y = W \Delta X$. Soit t le neurone de sortie et $M_{\Delta x \Delta t}, M_{\Delta y \Delta t}$ les matrices contenant les multiplicateurs. Alors :

$$
\begin{aligned}
M_{\Delta x \Delta t} &= M_{\Delta x \Delta y^+} M_{\Delta y^+ \Delta t} + M_{\Delta x \Delta y^-} M_{\Delta y^- \Delta t} + \quad \text{par règle de chaînage sur } y && (3.52)\\
&= (M_{\Delta x^+ \Delta y^+} + M_{\Delta x^- \Delta y^+} + M_{\Delta x = 0 \Delta y^+}) M_{\Delta y^+ \Delta t} && (3.53)\\
&+ (M_{\Delta x^+ \Delta y^-} + M_{\Delta x^- \Delta y^-} + M_{\Delta x = 0 \Delta y^-}) M_{\Delta y^- \Delta t} \quad \text{par règle de chaînage sur } x && (3.54)\\
&= ((W^T \odot 1(W^T > 0) M_{\Delta y^+ \Delta t}) \odot 1(\Delta X > 0) + (W^T \odot 1(W^T < 0) M_{\Delta y^+ \Delta t}) \odot 1(\Delta X < 0)) && (3.55)\\
&+ ((W^T \odot 1(W^T < 0) M_{\Delta y^+ \Delta t}) \odot 1(\Delta X > 0) + (W^T \odot 1(W^T > 0) M_{\Delta y^+ \Delta t}) \odot 1(\Delta X < 0)) && (3.56)\\
&+ (W^T (M_{\Delta y^+ \Delta t}) + M_{\Delta y^+ \Delta t}) \odot 1(\Delta X = 0) && (3.57)
\end{aligned}
$$

$\odot$ correspond à la multiplication de matrice élément par élément.

3.3.7 DeepSHAP, 2019

Dans *Explaining Models by Propagating Shapley Values of Local Components* [58], Hugh Chen, Scott Lundberg et Su-In Lee regroupent les travaux faits sur SHAP et DeepLIFT pour créer une nouvelle théorie unifiée, DeepSHAP, qui exploite les aspects des deux théories. DeepSHAP a le même champ d'application que DeepLIFT dans le sens où elle s'applique aux réseaux de neurones profonds. Cependant, DeepSHAP utilise les valeurs de Shapley pour définir les contributions, et non pas les méthodes présentées juste au-dessus. Ainsi, si deux neurones x, y sont directement reliés par une fonction f dans le réseau tel que $y = f(x)$, alors : $\varphi_i(f, x) = C_{\Delta x_i \Delta y}$. Pour rappel $\Delta x = x - x_0$ définit l'écart à la référence où x_0 est la référence. Ainsi, dans le cas linéaire où $\Delta y = b + \sum_i w_i \Delta x_i$, plutôt que de définir $C_{\Delta x_i \Delta y} = w_i \Delta x_i$ on définit $C_{\Delta x_i \Delta y} = \varphi_i(f, x)$. L'avantage d'utiliser les valeurs de Shapley est que l'on va pouvoir l'appliquer à davantages de types de fonctions différentes. On peut alors, en utilisant les règles de chaînage issues de DeepLIFT, propager les valeurs de Shapley à travers tout le réseau.

Il convient en premier lieu de définir une référence. Soit D une distribution de vecteurs de référence. Le choix de D est libre et dépend du problème, il doit être choisi de façon à adéquatement représenter la situation de référence du problème. Un choix simple pour D

est basiquement de prendre le jeu de donnée X^{train}, mais dans d'autres situations, il peut être préférable et moins coûteux de faire un autre choix. On peut par exemple effectuer un algorithme de *k-clustering* et choisir le cluster μ_i tel que :

$$\mu_i = argmin_j ||x - \mu_j||^2 \tag{3.58}$$

De cette façon, on s'assure de ne comparer notre entrée qu'aux autres entrées du même cluster. Comparons ce qui est comparable en somme.

Considérons alors que nous avons construit D et choisissons un vecteur de référence $x^b \in D$ (b pour *baseline*). On note $\varphi(f, x)$ les valeurs de Shapley globales, c'est-à-dire celles calculées par rapport à D et $\varphi(f, x, x^b)$ les valeurs de Shapley locales, c'est-à-dire celles calculées par rapport à une seule référence. Étudions l'exemple de la Figure 6.1 qui présente un réseau profond simple avec deux caractéristiques, deux neurones cachés, une fonction d'activation g non linéaire et un neurone de sortie.

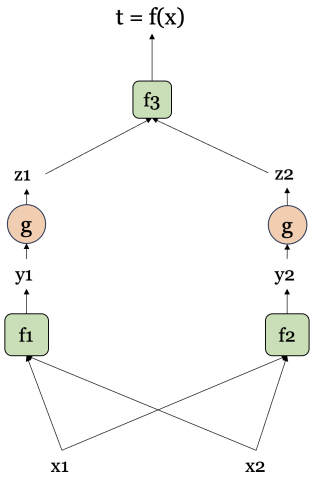


FIGURE 3.6 Un réseau de neurone simple avec 1 neurone caché et une fonction d'activation.

Soit $x = (x_1, x_2)$ l'entrée du modèle, $y = (y_1, y_2)$ le vecteur après la première couche et $z = (z_1, z_2)$ le vecteur après la fonction d'activation. Soit $t = f(x)$ la prédiction du modèle. On calcule d'abord les vecteurs de référence intermédiaires y^b et z^b en propageant x^b dans le réseau. Ensuite, on commence par la fin du réseau avec $C_{\Delta z_i \Delta t} = \varphi_i(f_3, z, z^b)$ pour $i \in \{1, 2\}$, puis on calcule le multiplicateur comme :

$$m_{\Delta z_i \Delta t} = \frac{\varphi_i(f_3, z, z^b)}{z_i - z_i^b}, i \in \{1, 2\} \tag{3.59}$$

De même on a $C_{\Delta x_j \Delta y_i} = \varphi_j(f_i, x, x^b)$ et :

$$m_{\Delta x_j \Delta y_i} = \frac{\varphi_j(f_i, x, x^b)}{x_j - x_j^b}, j \in \{1, 2\} \tag{3.60}$$

Ensuite, on traite linéairement la fonction d'activation non linéaire comme $\varphi_i(g, y, y^b) = g(y_i) - g(y_i^b)$. Cela permet de respecter les caractères additifs et linéaires des valeurs de Shapley. Bien que cette approximation puisse sembler simplificatrice, les impacts des non-linéarités sont souvent marginaux dans le cadre d'une explication globale ou pour des petits changements autour de la référence. Ainsi, ce traitement est justifié par un compromis acceptable entre précision et

efficacité. Cela donne alors :

$$m_{\Delta y_i \Delta z_i} = \frac{g(y_i) - g(y_i^b)}{y_i - y_i^b}, i \in \{1, 2\} \tag{3.61}$$

En utilisant la règle de chaînage avec (6.5), (6.6) et (6.7), on peut finalement calculer la contribution de chaque caractéristique $x_j, j \in \{1, 2\}$ à la sortie du modèle $f(x)$ comme :

$$\begin{aligned}
\varphi_j(f, x, x^b) &= (x_j - x_j^b) m_{\Delta x_j \Delta t} && (3.62)\\
&= (x_j - x_j^b) \sum_{i \in 1,2} m_{\Delta x_j \Delta y_i} m_{\Delta y_i \Delta z_i} m_{\Delta z_i \Delta t} && (3.63)\\
&= \sum_{i \in 1,2} \varphi_j(f_i, x, x^b) \frac{g(y_i) - g(y_i^b)}{y_i - y_i^b} \frac{\varphi_i(f_3, z, z^b)}{z_i - z_i^b} && (3.64)
\end{aligned}$$

Nous avons ainsi trouvé les attributions de chacune des caractéristiques, à la prédiction finale, par rapport à une référence x^b. Le théorème suivant permet de trouver les valeurs de Shapley globales à partir des valeurs locales.

Theorem 3.3.14. *La moyenne des valeurs de Shapley sur l'ensemble des références vaut exactement la valeur de Shapley globale sur la distribution soit :*

$$\varphi(f, x) = \frac{1}{|D|} \sum_{x^b \in D} \varphi(f, x, x^b) \tag{3.65}$$

La preuve du théorème est en annexe A. Finalement, il est possible d'accéder aux valeurs de Shapley globales de chaque caractéristique du modèle. DeepSHAP apparaît donc comme une méthode très rapide de calcul des valeurs SHAP à travers un réseau de neurones, en ne faisant plus tourner le réseau sur un nombre factoriel de combinaisons, mais plutôt en propageant directement les attributions à travers le réseau. DeepSHAP est encore plus rapide que KernelSHAP, mais il est spécifique aux réseaux de neurones là où KernelSHAP est plus adaptable.

3.3.8 G-DeepSHAP, 2022

Hugh Chen, Scott Lundbergh et Su-In Lee poursuivent à nouveau leurs travaux dans *Explaining a series of models by propagating Shapley values* [59] où ils vont étendre la propagation des valeurs de Shapley, non plus seulement à un réseau de neurones, mais à une série de réseaux. La généralisation de DeepSHAP porte alors le nom de G-DeepSHAP pour *Generalized DeepSHAP*. Dans les modèles modernes, il est souvent efficace de superposer en série différents réseaux,

chacun ayant une mission spécifique dans le modèle. Par exemple, un modèle peut être constitué d'un réseau servant à la vectorisation, un autre pour la transformation matricielle des données et un dernier pour la détection d'anomalies. Nous verrons plus loin des études de cas montrant un tel enchaînement de réseaux. Pour le moment, considérons notre modèle comme étant une fonction $f(x) = f_k(x) = (h_k \circ ... \circ h_1)(x)$, où $h_i : \mathbb{R}^{m_i} \to \mathbb{R}^{o_i}$, tel que $m_i = o_{i-1}, \forall i \in 2...k$ et $o_k = 1$. k est le nombre de sous-modèles constituant le modèle global. On a donc un ensemble de modèles intermédiaires $f_i(x) = (h_i \circ ... \circ h_1)(x)$. Concrètement, m_1 est le nombre de caractéristiques du modèle, les m_i sont les dimensions d'entrée de chaque modèle intermédiaire, et les o_i les dimensions de sortie pour chaque couche i.

Proposition 3.3.15. *G-DeepSHAP calcule les attributions le long du modèle selon :*

$$\psi^k = \varphi(h_k, f_{k-1}(x), f_{k-1}(x^b)) \tag{3.66}$$

$$\psi^i = \varphi(h_i, f_{i-1}(x), f_{i-1}(x^b))(\psi^{i+1} \oslash (f_i(x) - f_i(x^b)), i \in 1, ..., k-1. \tag{3.67}$$

La division d'Hadamard $\oslash$ effectue la division entre a et b, élément par élément, où a_i/b_i vaut 0 si b_i vaut 0. On l'appelle aussi division sûre. L'attribution finale à calculer est donc :

$$\varphi(f_k, x, x^b) = \psi^1 \tag{3.68}$$

C'est la contribution de chaque caractéristique au calcul de la prédiction finale, soit le calcul de f_k. C'est ce qu'on a toujours cherché à calculer, la contribution des caractéristiques à la boîte noire représentée par f_k. La Figure 3.7 illustre comment calculer les valeurs SHAP dans une série de réseaux. Les ψ^i donnent également les attributions des entrées $f_{i-1}(x)$ au modèle

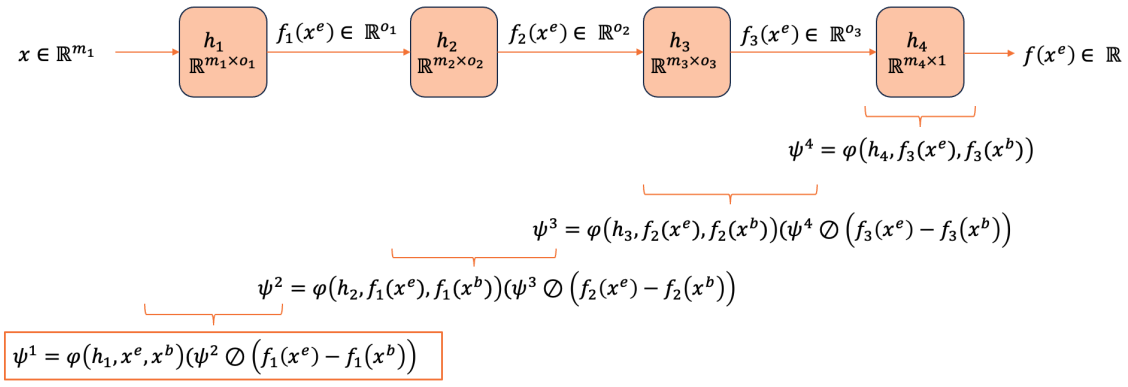


FIGURE 3.7 Calcul des valeurs SHAP dans une série de modèles

$(h_k \circ ...h_i)$. On peut montrer que les ψ_i sont bien des attributions.

Theorem 3.3.16. *Chaque attribution $\psi^i \in \mathbb{R}^m, \forall i \in 1...k$ satisfait l'axiome d'efficacité i.e :*

$$\hat{1}_{1\times m_i}\psi^i = f_k(x) - f_k(x^b), \forall i \in 1, ..., k \tag{3.69}$$

La preuve est en annexe A

3.3.9 Récapitulatif des techniques étudiées

Récapitulons le tour d'horizon des techniques d'XAI que nous avons étudiées dans cette section, à l'aide de la Figure 3.8.

Nous avons d'abord défini une catégorie de méthodes d'XAI : les méthodes additives d'attribution de caractéristiques. Ces méthodes modélisent l'explicabilité comme une combinaison linéaire de variables binaires, où chaque variable représente la présence ou l'absence d'une caractéristique. Trois techniques relèvent de cette catégorie : les valeurs de Shapley, DeepLIFT et LIME, chacune avec une approche distincte. Shapley utilise la théorie des jeux pour trouver les coefficients φ, décrivant les attributions associées à chaque joueur. DeepLIFT évalue les contributions $C_{\Delta y \Delta z}$ des neurones d'un réseau en propageant leurs influences via des multiplicateurs, puis agrège ces contributions pour expliquer la sortie. LIME approxime localement le modèle original par un modèle linéaire interprétable, en optimisant les paramètres π_x (voisinage), $\mathcal{L}$ (fonction de perte) et Ω (régularisation), via une régression des moindres carrés. Ces trois méthodes donnent par défaut trois résultats différents et donc trois modèles d'explicabilité distincts en fin de processus.

SHAP impose trois axiomes pour garantir des explications rigoureuses :

1. Justesse locale : Égalité entre f (modèle original) et g (modèle explicable) au point évalué.
2. Absence : Exclusion des caractéristiques sans influence.
3. Cohérence : Stabilité des attributions lors de modifications du modèle.

Seules les valeurs de Shapley satisfont toujours ces axiomes, ce qui en fait la base théorique de SHAP (*Shapley Additive exPlanations*). Cependant, le calcul exact des valeurs de Shapley est prohibitif pour des modèles complexes. DeepLIFT et LIME offrent des alternatives efficaces (mais non axiomatiques). En adaptant leurs paramètres — par exemple, les multiplicateurs de DeepLIFT ou le noyau de LIME — pour respecter les axiomes de SHAP, on obtient deux approximations pratiques :

— DeepSHAP (extension de DeepLIFT aux réseaux de neurones).

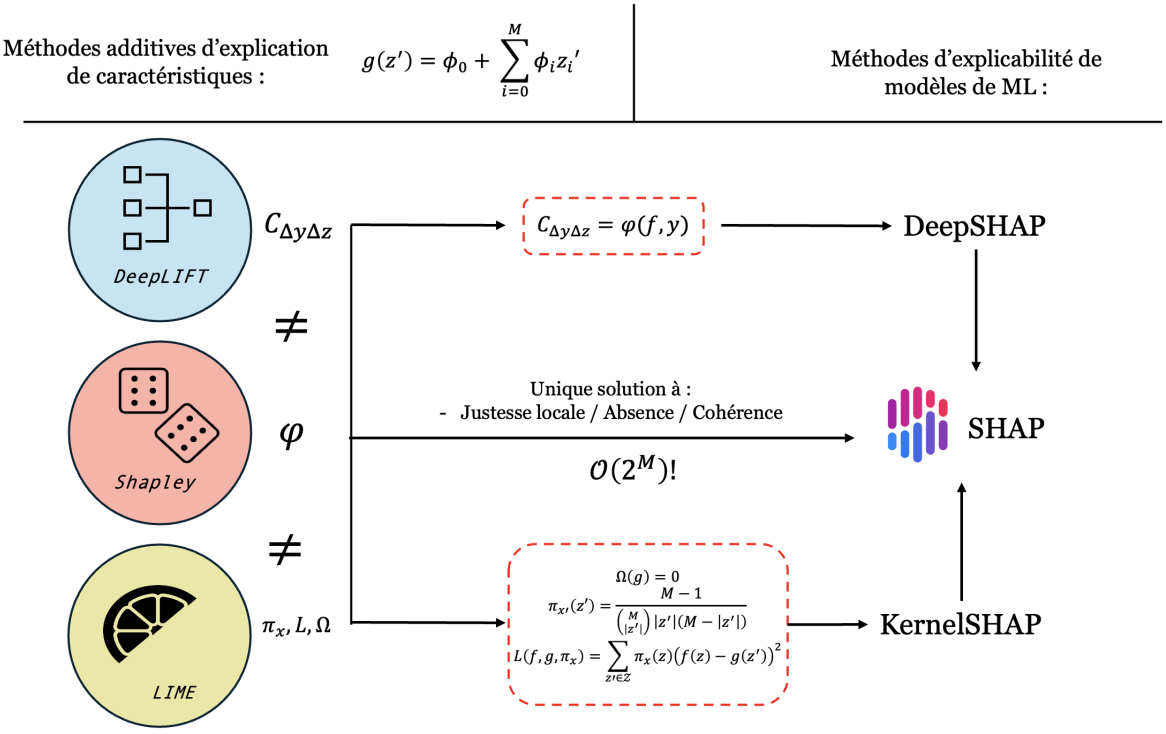


FIGURE 3.8 Récapitulatif de l'étude des techniques SHAP, Shapley, DeepLIFT et LIME

— KernelSHAP (adaptation de LIME via un noyau spécifique).

Pour revenir à notre cadre conceptuel, SHAP aborde dans sa démarche les concepts de transparence, d'interprétabilité et d'explicabilité. De plus, les méthodes dérivées sont motivées par le concept d'efficacité. La théorie présentée dans ce chapitre permet d'affirmer que SHAP assure des résultats précis, solides et stables. Lors de la discussion des résultats obtenus, nous aborderons également les concepts d'intelligibilité et de satisfaction. Ainsi, le choix de SHAP permet de prendre en considération 8 des 11 concepts retenus pour le cadre conceptuel, en faisant la meilleure méthode d'XAI pour ma recherche. L'équité et la responsabilité ne seront donc jamais abordées, nous y reviendrons dans la partie consacrée aux travaux futurs.

CHAPITRE 4 MÉTHODOLOGIE

L'objectif de ce chapitre est d'appliquer les méthodes d'XAI, et notamment SHAP, aux modèles d'IA utilisés en cybersécurité avionique, afin d'en expliquer les décisions critiques. Après avoir justifié l'importance de l'XAI pour la sécurité aérienne (cf. introduction) et approfondi la technique SHAP, nous détaillons ici son intégration pour la détection d'intrusions. Cette partie présente le cadre méthodologique que nous avons développé. Celui-ci définit une démarche standardisée pour intégrer l'XAI dans les systèmes de détection d'intrusions, et servira de base à nos applications futures. Il indique la démarche à suivre pour comprendre dans quel cadre l'intégration de l'XAI dans le domaine étudié peut se faire. Notre démarche s'articule autour d'un cadre méthodologique structurant, illustré par la Figure 4.1, qui se décompose en cinq phases distinctes :

1. Génération des données
2. Prétraitement des données pour l'analyse
3. Développement de l'IDS
4. Génération des explications via SHAP
5. Évaluation des explications produites

Le chapitre suivant proposera des scénarios d'application concrets validant cette approche à travers plusieurs cas d'étude représentatifs.

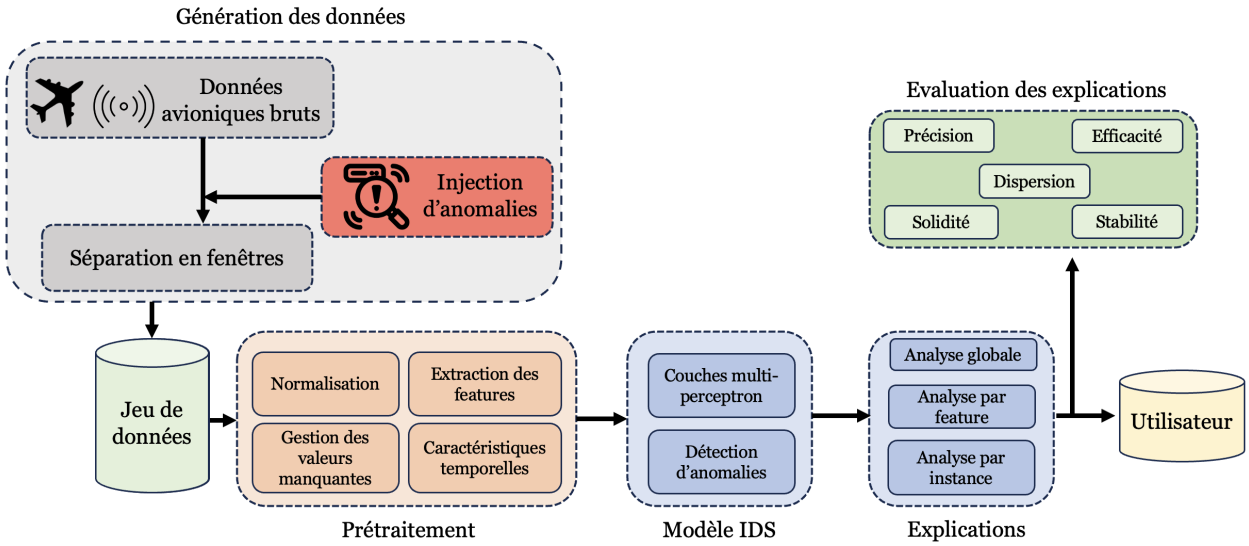


FIGURE 4.1 Framework de l'intégration de l'XAI pour la détection d'intrusions en cybersécurité avionique.

4.1 Génération des données

La première étape consiste à sélectionner un protocole de communication avionique (ARINC, ADS-B, MIL-STD-1553, etc.) qui servira de base à notre étude. Ce choix initial est délibérément laissé ouvert pour garantir l'adaptabilité de notre approche à différents contextes opérationnels. On étendra même dans le chapitre 5 au cas des communications réseaux. Une fois le protocole identifié, nous procédons à la collecte des données nécessaires à l'entraînement de notre IDS. Bien que centrée sur les communications avioniques, la méthodologie reste applicable à d'autres domaines de sécurité, sous réserve de disposer d'un volume de données suffisant pour assurer un apprentissage efficace du modèle. Pour permettre la détection d'anomalies, nous introduisons artificiellement dans le jeu de données des altérations reproduisant différents scénarios d'attaque. Cette approche présente deux avantages :

— Elle conserve une flexibilité quant à la nature des menaces considérées

— Elle maintient un équilibre entre données normales et anomalies, essentiel pour l'apprentissage des modèles

Les données avioniques présentant une forte dépendance temporelle, nous appliquons un fenêtrage glissant aussi appelé découpage en séries temporelles, pour trois raisons techniques. D'abord cela permet de préserver les relations temporelles entre les données. Les communications aéronautiques forment des séquences cohérentes où chaque élément dépend des précédents (ex : la variation de position est plus significative que la position absolue). Ensuite, cela permet d'être plus robuste au bruit en révélant les tendances globales plutôt que les valeurs ponctuelles. Enfin, ce découpage permet une compatibilité de nos données avec les architectures modernes ; cette structure est optimale pour les modèles exploitant les dépendances temporelles (RNN, LSTM, Transformers). À l'issue de cette phase, nous disposons d'un jeu de données prêt à être traité en préparation de leur utilisation pour l'entraînement du modèle.

4.2 Prétraitement des données

La phase de prétraitement convertit les données brutes en un format adapté aux modèles d'IA. Elle comprend quatre étapes indépendantes :

— *Normalisation* : Pour homogénéiser les échelles de valeurs entre différentes caractéristiques (par exemple, une variable variant entre 0-1 et une autre entre 0-1000), nous appliquons une normalisation centrée réduite :

$$x' = \frac{x - \mu}{\sigma} \tag{4.1}$$

où μ est la moyenne et σ l'écart-type du jeu de données.

— *Gestion des valeurs manquantes* : Les jeux de données ne garantissent pas de pouvoir toujours fournir une valeur pour chaque caractéristique, à chaque donnée. C'est pourquoi il faut fixer au préalable une méthode de gestion de ces valeurs manquantes. Plusieurs stratégies sont envisageables :
  — Remplacement par la valeur moyenne
  — Suppression des instances incomplètes
  — Interpolation temporelle (méthode retenue) : Pour une fenêtre donnée, les valeurs manquantes sont estimées à partir des valeurs adjacentes (par exemple, une vitesse manquante est interpolée à partir des vitesses mesurées dans la même fenêtre de 10 secondes)

— *Encodage des caractéristiques temporelles* : Cette étape capture les informations temporelles brutes (secondes, heures) via des encodages cycliques (sinus/cosinus). La stationnarité des données est également analysée ; une fenêtre est considérée non stationnaire si ses points temporels sont irréguliers (exemple : t=1, t=2, t=8, t=45) et nécessite un traitement spécifique.

— *Extraction des caractéristiques importantes* : Chaque fenêtre est convertie en une instance unique contenant des métriques synthétiques dont les statistiques de base : moyenne, médiane, écart-type, min-max. Elle contient également le *Mean Absolute Change* qui capture la variabilité temporelle de la fenêtre par :

$$MeanAbsoluteChange = \frac{1}{n-1} \sum_{i=1}^{n-1} |x_{i+1} - x_i| \tag{4.2}$$

Ces quatre transformations visent à améliorer la qualité de l'apprentissage du modèle, à réduire la sensibilité aux artefacts des données brutes et à optimiser l'efficacité algorithmique.

4.3 Modèle d'apprentissage

Cette phase centrale consiste à concevoir l'architecture de l'IDS. Plusieurs approches de classification supervisée peuvent être envisagées pour identifier les anomalies, avec des architectures communes comprenant :

— Une couche d'analyse (typiquement un réseau de neurones de type *Multi-Layer Perceptron*) chargée d'extraire les motifs caractéristiques des données prétraitées.

— Une couche de décision spécialisée dans la détection des déviations par rapport au comportement normal.

Le choix final du modèle (Random Forest, SVM, réseaux de neurones profonds, etc.) et l'optimisation des hyperparamètres (taux d'apprentissage, taille des couches, fonctions d'activation) seront déterminés de manière empirique pour chaque scénario opérationnel.

4.4 Explications

Cette phase cruciale intègre la couche d'explicabilité entre la détection d'anomalies et l'utilisateur final. Nous utilisons la théorie SHAP (cf. Chapitre 3.3) via la librairie Python officielle (`https://github.com/shap/shap`), avec deux niveaux d'implémentation. Un premier niveau pour les cas standards où l'on pourra utiliser directement l'API, ainsi qu'un second niveau pour les cas plus complexes, dans lequel on mènera une adaptation raffinée de l'analyse, basée sur les fondements mathématiques de SHAP.

Le processus d'explication s'appuie sur trois types de visualisations SHAP, illustrées ici avec un modèle de prédiction immobilière en Californie [60] :

Analyse globale : La phase d'explication débute par une évaluation globale du modèle sur l'ensemble du jeu de données. Cette approche permet d'identifier les caractéristiques ayant l'impact moyen le plus significatif sur les prédictions, représenté par leurs valeurs SHAP. Deux types de visualisations sont ici possibles :

- *Visualisation en barres (Fig. 4.2)* : Présente la moyenne des valeurs SHAP absolues sur l'ensemble du jeu de données. Cette visualisation met en évidence ici que la localisation géographique est le facteur prédominant et que le revenu médian est le second contributeur majeur. On peut également voir les autres caractéristiques avec des contributions marginales mais non négligeables.
- *Diagramme en essaim (Fig. 4.3)* :
  - Ordonnée : caractéristiques classées par importance
  - Abscisse : valeurs SHAP individuelles
  - Code couleur : valeur des caractéristiques (rouge = élevée, bleu = faible)

  Ce diagramme permet de visualiser les valeurs SHAP sur l'ensemble du jeu de donnée. Pour chaque caractéristique, le diagramme affiche toutes les valeurs SHAP relatives à cette caractéristique. On détecte ainsi une corrélation négative entre coordonnées géographiques et prix prédit. Le motif géographique qui se dégage est clair : les valeurs immobilières augmentent vers les zones métropolitaines côtières (e.g., San Francisco, Los Angeles) situés au Sud-Ouest (latitude et longitude basse). Par ailleurs, le revenu médian

présente une corrélation positive avec les prix des maisons. Corrélation relativement prévisible, plus le revenu d'un habitant est élevé, plus le prix de sa maison l'est aussi.

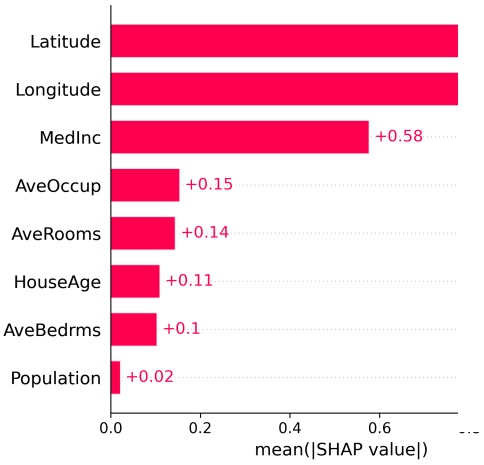


FIGURE 4.2 Tracé en bar

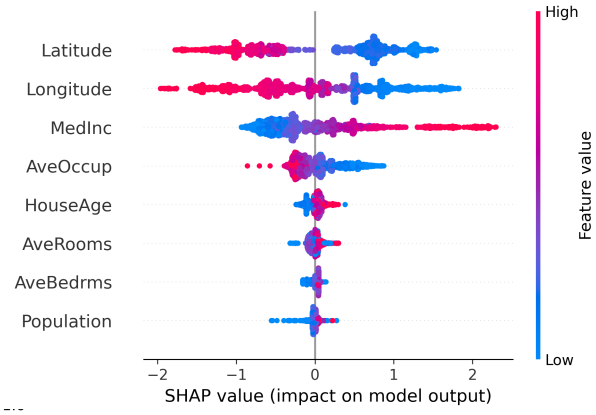


FIGURE 4.3 Tracé en cascade

Analyse par caractéristique : La deuxième étape consiste en une analyse spécifique par caractéristique, visant à établir la relation entre la valeur d'une variable et son influence sur les prédictions du modèle. Cette analyse s'appuie sur des diagrammes de dispersion (Fig. 4.4) qui mettent en relation :

— En abscisse : la valeur brute de la caractéristique

— En ordonnée : la valeur SHAP correspondante (impact sur la prédiction)

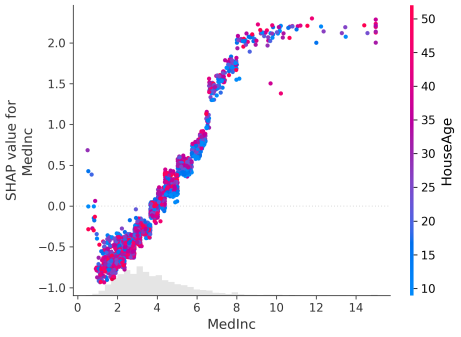
— En couleur : une seconde caractéristique pour une analyse croisée

Analysons un à un les diagrammes. Le premier diagramme de dispersion est relatif au revenu médian (*MedInc*). L'analyse révèle plusieurs régimes distincts :

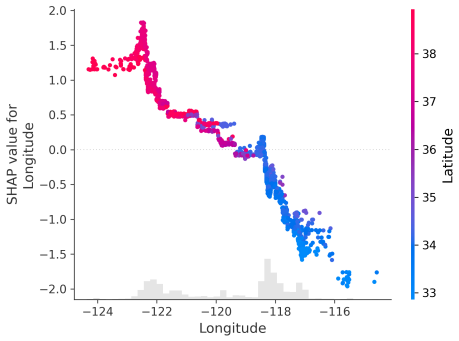
— Régime linéaire ($0 < MedInc < 4$) → Valeur SHAP négative traduisant un impact négatif des revenus sur la prédiction des prix des habitants. Impact d'autant plus négatif que le revenu est bas.

— Zone neutre ($MedInc \approx 4$) → Valeur SHAP proche de 0. La caractéristique n'influence pas la prédiction et le modèle considère ce revenu comme *baseline*.

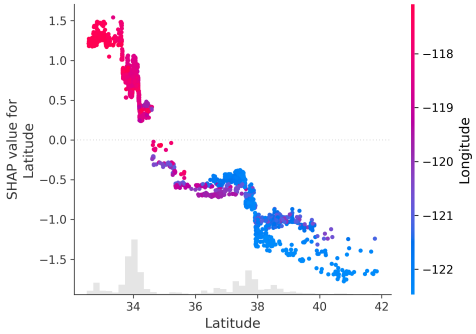
— Régime linéaire ($MedInc < 4 \rightarrow 10$) → Relation positive monotone avec les valeurs SHAP montrant un impact croissant du revenu sur la prédiction du prix avec un gradient de +0.25 SHAP/unité de revenu. Plus le salaire est élevé, plus il impactera le prix de la maison.



(a) MedInc



(b) Longitude



(c) Latitude

FIGURE 4.4 Tracés dispersés pour les caractéristiques de localisation et revenu médian

— Effet de saturation ($MedInc > 10$) → Plateau des valeurs SHAP. On peut supposer qu'un seuil psychologique/économique est atteint dû à des effets de plafond lié aux mécanismes de marché.

Pour un système de détection d'intrusions avioniques, cette approche permettrait par exemple d'identifier les **seuils critiques des paramètres réseaux**, les **relations non-linéaires** entre variables ainsi que les **points de bascule** dans le comportement du modèle.

Analyse locale : L'analyse locale permet d'expliquer des prédictions individuelles, particulièrement utiles pour investiguer les anomalies détectées par le modèle. Cette approche ciblée génère des rapports d'explication concis pour chaque alerte, comme illustré par le diagramme en cascade (Fig. 4.5).

Dans la visualisation en cascade, on trouve en ordonnée les influences de chaque caractéristique à la prédiction en abscisse. Toutes conclusions faites sur ce type de tracé est relative à

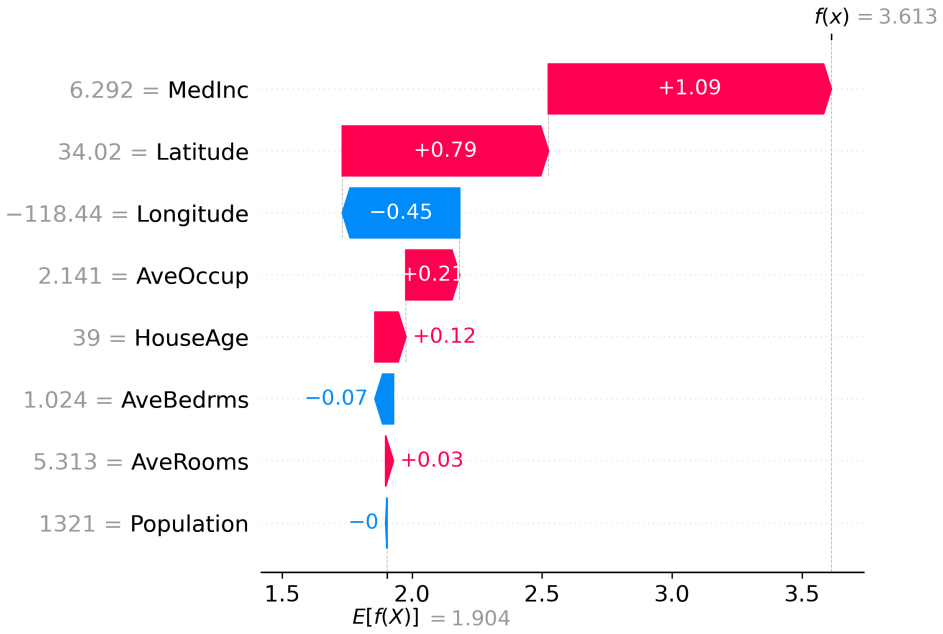


FIGURE 4.5 Tracé en cascade

une maison en particulier puisqu'il s'agit d'une analyse locale. Au départ, lorsqu'aucune caractéristique n'est spécifiée, le modèle explicatif vaut $g(z') = \varphi_0 = E[f(X)] = 1.904$. C'est simplement l'espérance du modèle relative à la distribution choisie. On ajoute ensuite caractéristique par caractéristique, les attributions φ_i jusqu'à arriver à la prédiction finale $f(x) = 3.613$. Le caractère additif des valeurs SHAP ($E[f(X)] + \sum \varphi_i = f(x)$), qui permet de converger de l'espérance à la prédiction en additionnant une par une les attributions, est ici parfaitement visible. Dans cet exemple, on peut légitimement supposer que le quartier est plutôt récent, mal placé en longitude, bien en latitude, mais surtout qu'il s'agit d'un quartier aisé.

Pour un système de détection d'intrusions, cette analyse permettrait de prioriser les alertes en identifiant les paramètres réseaux ayant le plus contribué à une alerte, afin de distinguer les vraies menaces (contributions cohérentes) des **faux positifs** ainsi que de détecter des motifs d'attaque émergents (ex : temps anormal des requêtes) permettant des comparaisons avec des cas nominaux.

Cette méthodologie en trois étapes (globale/caractéristique/locale) assure une explicabilité complète, depuis les tendances système jusqu'aux événements ponctuels. La phase suivante détaillera comment évaluer quantitativement la qualité de ces explications.

4.5 Méthode d'évaluation

Cette section vise à évaluer objectivement la qualité des explications obtenues précédemment. Bien que les résultats semblent satisfaisants sur le plan qualitatif, il est nécessaire d'en quantifier la pertinence. Nous nous appuyons pour cela sur les travaux de Molnar [46], adaptés selon les métriques proposées dans [41], que nous avons modifiées pour répondre à nos besoins spécifiques. Cinq critères d'évaluation seront présentés : précision, dispersion, robustesse, stabilité et efficacité. Pour chaque métrique, nous en justifierons d'abord la pertinence avant de détailler son mode de calcul et son application concrète à nos résultats. Ces critères ont été choisis en écho à la conceptualisation réalisée en partie 3.

1. **Précision :** La précision mesure la capacité du modèle explicatif à identifier correctement les caractéristiques réellement déterminantes pour la prédiction. Pour la vérifier, nous évaluons la dégradation des performances du modèle lorsqu'on neutralise ses caractéristiques jugées importantes (en les remplaçant par leurs valeurs de référence). Formellement, la précision d'ordre k (où k représente le nombre de caractéristiques considérées comme significatives) se calcule ainsi :

$$ACY_k(x, f) = ||f(x|x_1 = x_1^b, ..., x_k^b) - y| - |f(x) - y)|| \tag{4.3}$$

   avec y la classification réelle et $f(x)$ la prédiction du modèle. Dans l'idéal la suppression des k caractéristiques principales devrait inverser la prédiction $f(x|x_1 = x_1^b, ..., x_k^b) \approx 1 - y$. Dans le pire cas, la prédiction est inchangée, $f(x) = f(x|x_1 = x_1^b, ..., x_k^b)$ et $ACY_k(x, f) = 0$. Ainsi, plus la précision est proche de 1, meilleur est le modèle explicable. Le choix de k est important : inspirés par la règle des trois en communication opérationnelle, nous retenons généralement trois caractéristiques comme suffisantes pour expliquer une décision à un utilisateur final comme un pilote.

2. **Densité :** Cette métrique vérifie que seule une minorité de caractéristiques a un impact significatif sur la prédiction. Nous analysons la distribution des valeurs SHAP via leur histogramme sur l'intervalle [-1,1]. La masse autour de zéro (MAZ) se calcule par :

$$MAZ(r) = \int_{-r}^{r} h(x)dx \tag{4.4}$$

   Plus la fonction $MAZ(r)$ croît rapidement vers 1 lorsque r augmente, plus la rareté des caractéristiques influentes est marquée, signe d'une bonne explicabilité. Pour $r = 1$, la MAZ convertit l'ensemble des valeurs et $MAZ(1) = 1$. Inversement, si $r = 0$, la

fenêtre est vide et $MAZ(0) = 0$. Nous traçons ensuite l'histogramme des valeurs de SHAP et l'évolution du MAZ en fonction du rayon. La vitesse à laquelle le MAZ évolue vers 1 indique la qualité de la rareté du modèle explicatif. Cette métrique mesure la **sélectivité** du modèle explicatif.

3. **Efficacité :** Dans des contextes temps réel comme l'avionique, où les délais de décision sont critiques, le processus d'explication ne doit pas grever les performances du système. Nous mesurons donc strictement le temps de calcul requis pour générer les explications, sans seuil absolu mais en référence aux contraintes opérationnelles spécifiques.

4. **Stabilité :** Une autre mesure importante dans une infrastructure critique est la stabilité des résultats générés. Une explication doit être fiable dans le sens qu'elle ne doit pas fournir des résultats différents pour les mêmes données d'entrée. En d'autres termes, pour deux exécutions i et j du modèle explicatif, les ensembles de caractéristiques importantes des deux exécutions T_i et T_j doivent avoir une intersection proche de 1, c'est-à-dire $IS(i, j) > 1 - \epsilon$, où IS est la taille de l'intersection et ϵ un seuil proche de 0. Cette métrique est importante à vérifier dans les méthodes probabilistes dont les résultats sont susceptibles de varier entre deux exécutions.

5. **Solidité :** La solidité évalue la capacité des explications à résister aux perturbations intentionnelles ou accidentelles des données d'entrée. Contrairement aux tests de stabilité qui mesurent la cohérence interne, cette métrique vérifie comment les explications se comportent face à des altérations externes. Pour la quantifier, nous proposons une approche en deux étapes :
   - Perturbation contrôlée : Nous appliquons des transformations progressives aux données d'entrée, allant de modifications mineures (bruit gaussien faible) à des altérations significatives (décalages systématiques).
   - Mesure de déviation : Pour chaque niveau de perturbation, nous calculons la distance entre les explications originales et perturbées.

   L'analyse de solidité est particulièrement cruciale pour les systèmes de sécurité où des acteurs malveillants pourraient tenter de manipuler les explications pour masquer des intrusions.

La présentation de la méthodologie est désormais achevée, nous pouvons ainsi entrer dans les applications. Le chapitre 5 se concentre sur l'étude des communications réseaux sur le jeu de données NSL-KDD. Le chapitre 6 étudie le cas du protocole ADS-B, travail qui a donné

lieu à l'écriture d'un article de recherche. Enfin, le chapitre 7 aborde le cas du protocole MIL-STD-1553.

CHAPITRE 5 APPLICATION DES ANALYSES SHAP À LA BASE DE DONNÉES NSL-KDD

Cette première application consiste à utiliser SHAP pour expliquer un IDS sur des communications réseau. Nous allons reprendre étape par étape notre méthodologie présentée plus haut et l'appliquer à ce cas d'étude. Cette première étude vise à remplir plusieurs objectifs :

- Prendre en main la méthodologie développée et vérifier sur un cas simple, sa faisabilité ainsi que sa pertinence.
- Apprendre à développer un IDS en utilisant un cas d'étude particulièrement adapté aux développements de ces systèmes, tout en se familiarisant avec les communications réseaux.
- Utiliser un jeu de données de référence en cybersécurité informatique, largement validé par la communauté scientifique afin de nous concentrer sur l'évaluation rigoureuse de notre approche, tout en garantissant la pertinence et la fiabilité des données d'entrée.

Cette première étude ne prend pas en considération le caractère avionique inhérent à ma problématique, cependant, il sert de bonne introduction à tous les enjeux que nous allons rencontrer.

5.1 Présentation de la base de données

La base de données NSL-KDD [61] est une version améliorée du célèbre KDD Cup 1999 [62], largement utilisé pour la recherche en détection d'intrusions réseau. Elle contient 41 caractéristiques décrivant des connexions réseau, incluant des métriques temporelles (comme *duration* indiquant la durée de la connexion), des statistiques de trafic (*src_bytes, dst_bytes* donnant le volume d'octets envoyés et reçus), et des indicateurs comportementaux (*wrong_fragment* pour le nombre de paquets corrompus ou *logged_in* pour indiquer si l'utilisateur est authentifié). Chaque connexion appartient à une des cinq catégories : une bénigne (0) et quatre de type attaques (DoS : 1, Sonde : 2, R2L : 3, U2R : 4). Ce jeu de données présente un déséquilibre de classes, notamment avec des attaques U2R et R2L sous-représentées, reflétant la complexité des scénarios réels. Voici une brève explication des attaques :

- **DoS** (*Denial of Service*) [Classe 1] : Surcharge une ressource (serveur, bande passante) pour la rendre indisponible. On peut citer l'attaque *neptune* qui inonde le serveur de requête de synchronisation, aussi appelé *SYN flooding*, ou encore l'attaque *mailbomb* consistant en un envoi massif d'email.

— **Probe** (Surveillance/Balayage) [Classe 2] : Collecte des informations sur le réseau (ports, services vulnérables) pour préparer une attaque future. L'attaque *nmap* qui scan les ports ou *satan* qui fait un audit de vulnérabilités, en sont des exemples.

— **R2L** (*Remote to Local*) [Classe 3] : Obtient un accès non autorisé à distance en exploitant des failles de sécurité. L'attaque *guess_passwd* qui cherche un mot de passe par force brute et *ftp_write* qui envoie des fichiers malveillants via FTP, sont des attaques R2L.

— **U2R** (*User to Root*) [Classe 4] : Élève les privilèges d'un utilisateur local vers un accès administrateur (root). L'attaque *buffer_overflow* qui exploite les débordements dans l'espace mémoire est une U2R.

5.2 Prétraitement des données

Certaines caractéristiques sont catégorielles comme *protocol_type* qui peut prendre les valeurs *tcp, udp, icmp* ou encore *service* (http, ftp_data, private). Ces données ne peuvent être fournies comme telle au modèle, elles sont alors encodées en deux étapes : Une étape d'encodage de la catégorie qui convertit les noms des catégories en valeurs numériques discrètes. (tcp $\rightarrow$ 0, udp $\rightarrow$ 1, icmp $\rightarrow$ 2). La seconde étape est l'encodage *One-Hot* et traduit les valeurs des catégories en vecteur de même taille que le nombre de catégories, avec un 1 à la position de sa catégorie et 0 ailleurs. Cet encodage sert à s'affranchir de toute notion d'ordre entre les catégories. Enfin, toutes les données numériques sont normalisées avec *StandardScaler* pour garantir une échelle homogène entre les valeurs. Les labels d'attaques sont unifiés en cinq classes selon la taxonomie standard, remplaçant les 39 sous-catégories d'attaques originales en seulement 5, précédemment décrites.

5.3 Architecture et entraînement du modèle

Un réseau de neurones profond est construit avec Keras, composé de :

— Deux couches cachées (128 et 64 neurones) avec activation ReLU, suivies d'une *Batch-Normalization* et d'un *Dropout* (taux=0.3) pour limiter le surapprentissage.

— Une couche de sortie à 5 neurones avec activation *softmax* pour la classification multi-classes.

Le modèle est optimisé via Adam (taux d'apprentissage=0.001) et entraîné sur 3 époques avec une taille de lot de 64. La métrique *sparse_categorical_crossentropy* est utilisée pour tenir compte des labels entiers non encodés en one-hot. Une validation croisée sur 20% des données d'entraînement permet de surveiller les performances, atteignant une précision supérieure à 90% sur l'ensemble de test.

5.4 Explications avec SHAP

Comme présenté dans la méthodologie, nous allons d'abord effectuer une analyse globale des valeurs SHAP pour ensuite faire une analyse par caractéristique et finir avec une analyse locale.

5.4.1 Analyse globale

La Figure 5.1 montre le graphe en essaim relatif aux valeurs SHAP pour les instances relevées comme des dénis de service (DoS). Les caractéristiques *dst_host_serror_rate, dst_host_srv_serror_rate, srv_serror_rate*, et *flag_S0* contribuent significativement vers une prédiction DoS, ce qui est cohérent avec leur lien direct avec des schémas d'attaque par saturation :

— *flag_S0* indique des connexions non établies (SYN non-ACK), signature classique des inondations TCP.

— *srv_serror_rate* et ses variantes *dst_host* capturent les taux d'erreurs SYN anormaux, typiques des attaques distribuées.

À l'inverse, les caractéristiques *same_srv_rate, dst_host_diff_srv_rate, et dst_host_srv_count* réduisent le score DoS, car elles caractérisent un trafic structuré et diversifié :

— *same_srv_rate* élevé suggère des échanges normaux avec un service unique (ex : session HTTP stable).

— *dst_host_diff_srv_rate* et *dst_host_srv_count* reflètent une répartition équilibrée des services, opposée au trafic ciblé des DoS.

5.4.2 Analyse par caractéristique

La Figure 5.2 montre l'analyse de la caractéristique *dst_host_same_src_port_rate*, qui mesure la proportion de connexions vers un même port de destination, avec un codage couleur relatif à la caractéristique *service_ftp_data*, dans le cas d'une attaque R2L.

Le comportement de la caractéristique *dst_host_same_src_port_rate* suit une distribution en "L inversé", la plupart des valeurs ont un impact quasi nul (SHAP ≈ 0), sauf pour une valeur extrême (≈ 2.5), où les valeurs SHAP deviennent brusquement élevées. Cela suggère que cette caractéristique n'est décisive que dans un cas très spécifique (valeur anormalement haute), probablement lié à une activité suspecte. Analysons désormais le code couleur, bleu quand le service FTP n'est pas utilisé, rouge sinon. Quand le FTP n'est pas actif, *dst_host_same_src_port_rate* a peu d'impact, même à des valeurs élevées tandis que quand il

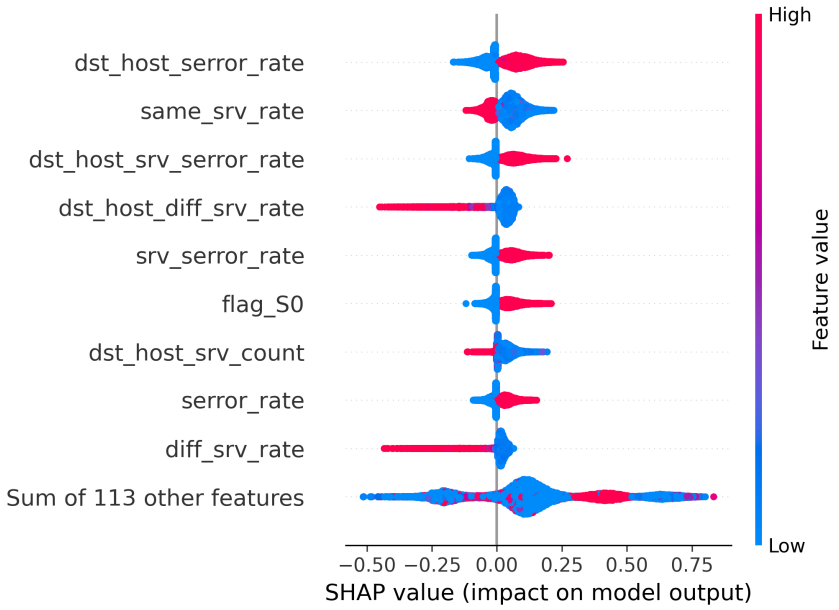


FIGURE 5.1 Analyse en essaim des valeurs SHAP, pour l'attaque DoS

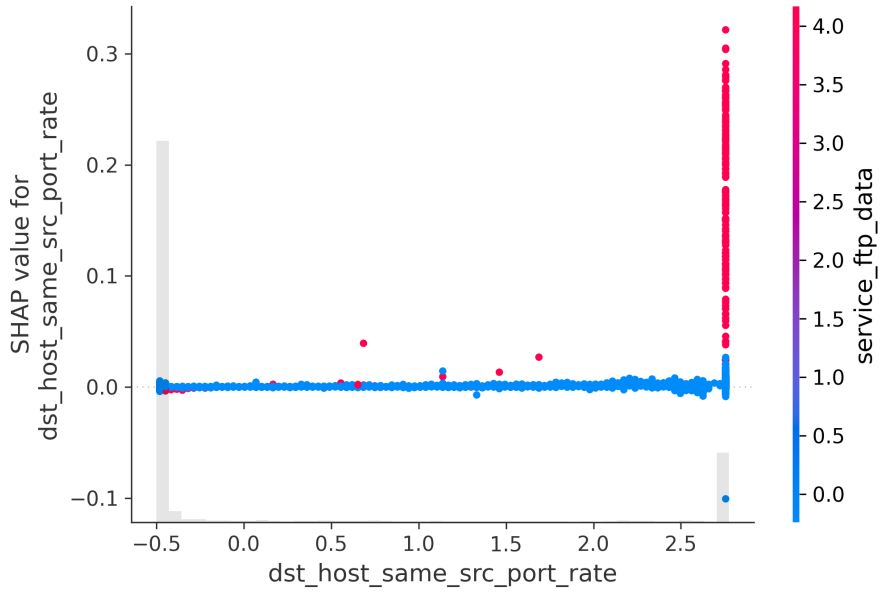


FIGURE 5.2 Tracé dispersé de la caractéristique *dst_host_same_src_port_rate* , codage couleur par *service_ftp_data*.

est actif, une valeur extrême de *dst_host_same_src_port_rate* déclenche une forte prédiction R2L. Autrement dit, une valeur anormale de *dst_host_same_src_port_rate* (ex : un hôte se connectant massivement au même port de destination) combinée à l'utilisation de FTP est un signal fort d'attaque R2L. L'attaque par force brute FTP consiste en effet en un attaquant tentant de deviner un mot de passe en ouvrant de multiples connexions vers ce port. Le modèle a ainsi identifié une règle implicite : "Un pic de connexions vers un même port + FTP = risque R2L", ce qui est cohérent en cybersécurité. Cependant, sa dépendance à une seule valeur critique (2.5) rend la détection vulnérable aux variations. Un attaquant pourrait éviter ce seuil.

5.4.3 Analyse locale

La Figure 5.3 analyse une instance détectée comme une attaque *Probe*. Cette figure révèle que trois caractéristiques ont particulièrement contribué à la prédiction.

D'abord la caractéristique *flag_RSTR* est celle avec la plus haute valeur SHAP. Le flag RSTR indique une connexion brutalement réinitialisée (ex : réponse TCP RST). Les scans réseau génèrent souvent des réponses RST lorsque des ports fermés sont détectés. La présence de cette caractéristique en haut de la liste est donc cohérente et nous donne un des mécanismes derrière l'attaque *Probe*.

La deuxième caractéristique, *dst_host_same_src_port_rate* (nombre de connexions sur un seul port) joue également un rôle important. Des attaques comme un balayage horizontal peuvent effectivement utiliser des connexions répétées vers un port spécifique. La dernière caractéristique est plus curieuse, il s'agit de *dst_host_diff_srv_rate* qui est pourtant souvent associée à un trafic normal car diversifié. Une explication possible est que certains outils de scan (ex : satan) testent plusieurs services rapidement, créant un trafic "diversifié mais anormal". Le modèle est donc capable de repérer ce genre de comportements.

5.5 Bilan de l'étude

Cette étude a démontré l'utilité de SHAP pour expliquer un IDS basé sur un réseau de neurones profond, en révélant des mécanismes décisionnels alignés avec la théorie des attaques réseau. Grâce aux analyses globales, par caractéristique et locale, nous avons identifié certaines signatures clé des attaques ; par exemple, les flags TCP (S0) et les taux d'erreurs SYN (*serror_rate*) comme marqueurs fiables des attaques DoS. Nous avons également repéré des interactions caractéristiques comme la combinaison FTP + pics de connexions vers un même port, pour les attaques R2L, validant des règles connues dans le domaine. Nous avons

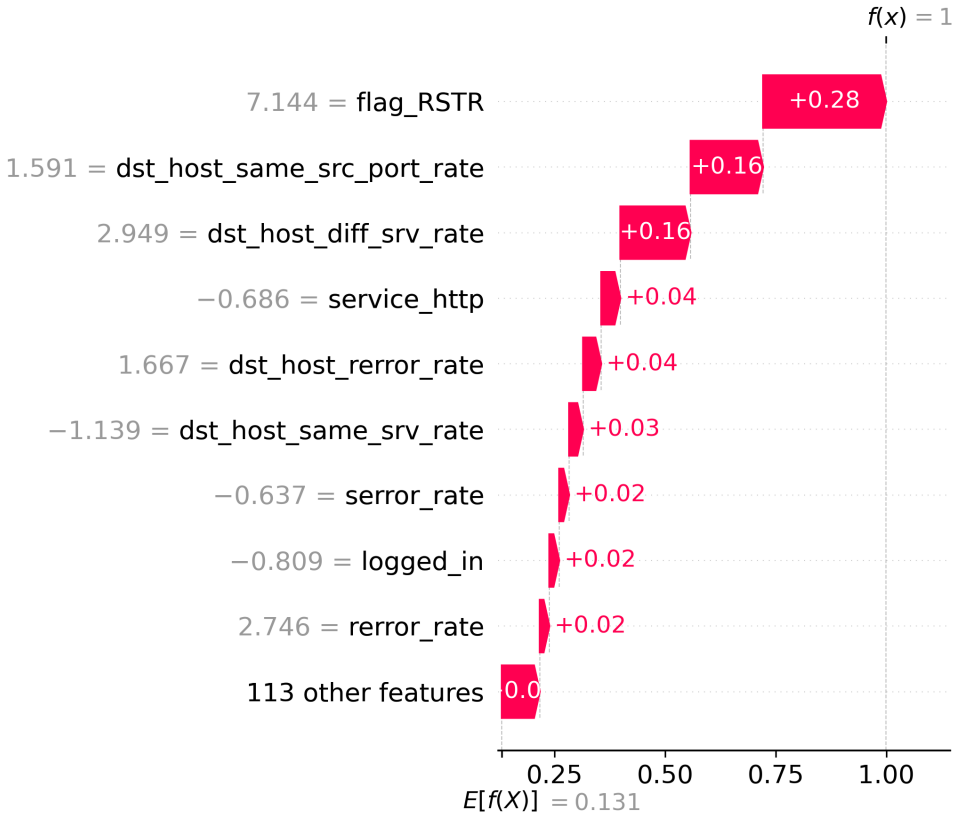


FIGURE 5.3 Tracé en cascade pour une instance de l'attaque *Probe*

également pu voir certaines limites du modèle, dont sa dépendance excessive à des seuils spécifiques ou des biais liés aux classes rares (U2R/R2L).

SHAP a permis de transformer des prédictions "boîte noire" en explications, facilitant l'audit des décisions et l'amélioration ciblée de l'IDS (ex : ajout de règles logiques pour les faux positifs). Cette approche est particulièrement pertinente en cybersécurité, où la compréhension des alertes est aussi cruciale que leur détection. Nous discuterons plus en détail ces résultats dans la partie 8.

CHAPITRE 6 ARTICLE 1 - AN XAI-BASED FRAMEWORK FOR TRUSTWORTHY COMMUNICATIONS IN AVIONICS

Charles de Malefette, Jean-Yves Ouattara, Felipe Magalhaes
Ahmad Shanejat Bushehri, Adel Abusitta, Gabriela Nicolescu
Polytechnique Montréal

La deuxième application de notre méthodologie a été pour un IDS appliqué au protocole de communication ADS-B. Ce travail fournissant à la fois des résultats prometteurs et encadrant exhaustivement mes objectifs de recherche, à ainsi donné naissance à un article de recherche [63], soumis dans le journal *IEEE Transactions on Aerospace and Electronics System* le 30 Mai 2025. Ce chapitre présente cet article dont les coauteurs sont écrits au-dessus.

Abstract

Artificial Intelligence (AI) techniques have made significant progress in avionic systems, improving domains such as aircraft design, operation, and maintenance, as well as flight prediction. A particularly important application is the development of efficient Intrusion Detection Systems (IDS) to detect abnormal intrusions in avionic communication systems, including ARINC-664, ADS-B, and MIL-STD-1553. These systems are essential for detecting potential cyberattacks or communications faults. However, the adoption of AI-powered techniques in this field is limited due to their opaque "black-box" nature. Further, they lack determinism, which hinders their understandability and acceptance by commercial aviation regulators. While advanced models may achieve high accuracy, they often lack the ability to clearly and comprehensibly explain their decisions to users. The emerging field of eXplainable AI (XAI) aims to address these challenges by making machine learning models more transparent, interpretable, and comprehensible for end-users. XAI is particularly consistent in critical infrastructure, where trust and clarity are paramount. In this paper, we present a novel approach incorporating XAI to explain the outputs of a deep-learning-based IDS designed for avionic communication systems. We explore an XAI feature attribution technique called Deep SHapley Additive exPlanations (DeepSHAP), which meets time-constraint requirements and theoretical stability, both of which are necessary in critical infrastructure, such as avionics. This approach aims to enhance user trust and improve the practical applicability of machine learning models in securing avionic communication systems. Applied to ADS-B communications, our approach

successfully explains anomalies detected by an IDS in this protocol by analyzing feature importance.

Reproducibility

The scripts to reproduce the results are available in open-source at https://github.com/CharlesdeMalefette/XAI4ADS-B. The dataset used for this research is available at [64].

6.1 Introduction

In avionics, performance-based operations are increasingly relying on complex, high-performance autonomous systems, an industry that has traditionally favored deterministic tools with well-defined risks. Furthermore, aircraft are becoming more interconnected, increasing their vulnerability to cybersecurity threats. One application of Machine Learning (ML) in avionics is the development of Intrusion Detection Systems (IDS) and anomaly detectors, designed to detect and prevent external threats to aircraft communication systems or any other abnormal situations. However, today's ML techniques introduce a level of uncertainty and are often not fully understandable or controllable by users. In critical environments like aviation, the results of ML models must not only be accurate but also interpretable and understandable, particularly for professionals who may lack expertise in Artificial Intelligence (AI). Thus, the opaque, "black-box" nature of current ML models requires an additional layer of explainability to foster trust and ensure usability.

EXplainable Artificial Intelligence (XAI), a new discipline in the AI context, aims to address this need by providing explanations for how ML models arrive at their predictions, thereby increasing confidence in their use. The level of explanation is based on three key concepts : transparency, understandability, and interpretability. Transparency refers to making the components of a system accessible to the user, in contrast to a black box. Understandability ensures that results are presented in an explicitly comprehensible format (e.g., natural language descriptions). Interpretability enables users to better grasp the model's inner workings, for example, by highlighting the importance of certain features and their relationship to the outcome.

Developing an XAI model raises important questions : What should the final form of the explanation be ? For instance, graphs may be preferred over tables and figures, while visual indicators such as HUD alerts or graphical user interfaces may be more suitable in other cases. Who will benefit from the explanation ? The design of the model should differ depending on whether it targets data scientists, flight engineers, pilots, or end-users at airports. Like

any computer science model, maintenance, updates, and extensions must be managed by whom ? Finally, the explanations must comply with certification requirements. In a cockpit, for instance, there are strict regulations on colors, shapes, text, and auditory signals that can be presented to pilots.

In this paper, we explore Deep SHapley Additive exPlanations (DeepSHAP), a recent XAI technique based on feature attribution using Shapley values from game theory. We aim to address the aforementioned questions by proposing a novel approach incorporating DeepSHAP to handle intrusions in avionic communication systems. We chose DeepSHAP for its solid mathematical foundation, which provides theoretical guarantees, as well as for its applicability to complex neural networks. In this approach, we aim to demonstrate how XAI can help end-users better understand and respond to anomalies detected by complex ML models. We present a use case applying DeepSHAP to an IDS for ADS-B communications.

The contributions of this paper are as follows :

— A novel framework integrating XAI into an avionic communication anomaly detection system using DeepSHAP ;

— The application of evaluation metrics on the proposed XAI framework, to quantify its usability with avionics constraints, and ;

— A use case of the framework within a scenario to explain anomalies in ADS-B communication, involving multiple stakeholders like pilots or flight engineers.

The rest of this paper is structured as follows : Section II reviews the literature on the use of XAI in avionics and cybersecurity. The fundamentals of DeepSHAP and the rationale for selecting this technique are presented in Section III. Section IV details the proposed approach for integrating XAI into avionic communication systems. Finally, Section V demonstrates an application of DeepSHAP in the context of ADS-B communications.

6.2 Background

The application of AI, particularly XAI in avionics and other critical security systems presents challenges and opportunities. This section reviews key research areas and open issues regarding the integration of AI in avionics and security, with a focus on the importance of transparency and explainability in critical systems.

6.2.1 Challenges and Open Issues of AI in Avionics

ML is gradually becoming essential in embedded systems due to its ability to recognize objects and extract valuable insights from large datasets in real-time. This makes ML a powerful tool,

particularly for decision-making processes in industries such as avionics [28]. In these settings, the need for accurate and timely decisions is often hindered by time constraints and complex information. In some cases, this has led to accidents due to poor decision-making [13]. Unlike human operators, ML can handle these challenges without being overloaded by information and without experiencing stress or fatigue.

As outlined by [22], the use of AI in avionics systems is growing, but it also raises new certification challenges and issues that must be addressed. [21] explored the broader implications of deploying neural networks in critical infrastructures, highlighting the potential opportunities they present. One of the most significant challenges in AI deployment, as discussed by Molnar [46], is its "black-box" nature, which limits transparency and interpretability. This challenge underscores the need for XAI, a discipline that aims to demystify ML decision-making.

6.2.2 XAI in Avionic Domain

Given these challenges, XAI has emerged as a critical tool to address transparency issues in avionics systems. [23] identified a gap between the advancements in AI research and its practical implementation. They emphasized the importance of compliance with regulations and the need for sufficient guarantees in Air Traffic Management (ATM) tools. Thus, they developed an XAI framework to meet the regulatory requirements of AI-based ATM tools.

[24] explore various ways of visually displaying ML results to make them more understandable for end-users, thus bridging the gap between flight engineers and ML scientists. They focused on explaining the prediction of airfoil noise around aircraft using a NASA dataset. Similarly, [1] developed a risk and accident prediction model for aircraft based on meteorological data. Using explainable methods such as SHapley Additive exPlanation (SHAP) [11] and Local Interpretable Model-agnostic Explanation (LIME) [12], they aimed to enhance trust between air traffic controllers and prediction models. Their human-machine interface is designed to assist pilots or ATC personnel in handling critical information.

Finally, [30] applied XAI to a Deep-Reinforcement-Learning (DRL) model for Unmanned Aerial Vehicles (UAVs). They provided visual explanations to help users understand the decisions made by the DRL model in obstacle-avoidance scenarios, thereby increasing the transparency in autonomous systems.

6.2.3 XAI in Security Domain

Explainable Security (Xsec), an extension of XAI in the security domain, was proposed by [31]. By answering the « Six Ws » questions (Who gives, Who receives What, Where, When, Why, and How), they demonstrated that Xsec involves multiple stakeholders and has a multi-faceted nature, as it requires reasoning about threat, security properties, and privacy measures in deploying countermeasures. [32] provided a comprehensive review of XAI in security systems, covering applications from banking to smart agriculture.

Interestingly, XAI models themselves can be targets of attacks. This is analyzed by [38] where they divided the explainability into three sub-domains : explanations of predictions, explanations covering security and privacy properties, and explanations concerning threat models. Each domain could be vulnerable to attacks, raising critical questions [39] about how attackers might exploit explainable models to perform attacks such as membership inference, model extraction, or adversarial examples.

Several studies apply XAI techniques like LIME or SHAP to explain security ML models. [35] developed a framework using LIME to explain an IDS, thereby increasing trust in the original model. [33] pointed out the need for XAI in critical infrastructure, particularly with the advent of new technologies such as 6G. They explored how to incorporate XAI in future 6G critical systems, such as precision manufacturing, healthcare, and human-machine-brain interfaces. [37] applied different extensions of SHAP to explain a security threat detection model, demonstrating the flexibility of SHAP in explaining different types of complex models. [34] also proposed an XAI framework to explain IDS models specifying that different user types require different explanations. For instance, end-users might need local explanations, while data analysts may prefer global ones.

Building upon these previous works, this paper aims to apply DeepSHAP, an improvement of SHAP, to explain ML-based anomaly detection in avionic communication systems. The next section delves deeper into the functionality of DeepSHAP.

6.3 Explainability Using DeepSHAP

This section discusses the development and application of DeepSHAP, beginning with its theoretical foundation in Shapley values from game theory and moving through its extension to machine learning (SHAP) and deep learning (DeepSHAP). We conclude by explaining why DeepSHAP is used in our work.

A common way to explain a complex model is by analyzing feature importance. Techniques such as Permutation Importance, Feature Importance, Joint Feature Importance, or LIME [12]

employ various mathematical analyses on features to indentify patterns and correlations to explain model output. Shapley Additive exPlanation (SHAP), developed by [11], assigns importance to each feature according to its Shapley value from game theory, aiming to represent its contribution to the output prediction.

6.3.1 Shapley values

In a game v, we define the contribution of a player i within a coalition S as the difference in the outcome when i is included versus excluded from S. Shapley values, theorized by [52], associate a real value $\varphi_i(v)$ for each player i, in a game v with player set U, satisfying three axioms : symmetry, efficiency, and linearity.

— **Symmetry** requires that players i, j, with the same contribution in all subsets, receive the same attribution : $\varphi_i(v) = \varphi_j(v)$.

— **Efficiency** states that the sum of contributions should equal the total game outcome : $f(x) = \sum_{i \in U} \varphi_i(v)$.

— **Linearity** ensures that the contribution of a player across two games is the sum of its contributions in each game.

Shapley proved that the only solution satisfying these three axioms is given by :

$$\varphi_i(v) = \sum_{S \subseteq U} \gamma_U(S) \left[v(S) - v(S \setminus \{i\}) \right], \quad \forall i \in U \tag{6.1}$$

Where $\gamma_U(S) = \frac{(|S|-1)!(|U|-|S|)!}{|U|!}$ is a weight factor for the subset S regarding all the subsets with the same size in U.

This formula provides a fair distribution of gains among players based on their marginal contributions. [53] later expanded Shapley values' properties, demonstrating the monotonicity of these solutions, which ensures stability by preventing allocation decreases when contributions increase. [11] will later show that this axiom also allows for the removal of the symmetry axiom, reducing the required axioms to just two. This concept has practical applications in fields such as mathematics, economics, and AI, as seen in SHAP.

6.3.2 SHAP

[11] extended the Shapley values to ML, enabling the interpretation of complex models. SHAP treats model features as game players and assigns importance to each feature using the Shapley values to explain prediction outcomes. Formally, SHAP explicitly defines an explainable model g that explains a specific prediction made by the model f being analyzed.

In SHAP, we define g as a linear function of binary variables z', where each coefficient is a Shapley value, and $z'_i = 0$ indicates the absence of feature i. With x belonging to $\mathbb{R}^U$, and z' to $\{0,1\}^U$, the explainable model is :

$$g(z') = \varphi_0(f) + \sum_{i \in U} \varphi_i(f) z'_i \tag{6.2}$$

Where $\varphi_0(f)$ is the prediction when no features are given.

In other words, the g explainable model takes as input all the features of the model in the form of present/absent variables, and resorts to the Shapley-weighted average of the present variables. In a game, considering a player's absence is easy to model - do not count them in the team. However, in ML, all features must have a value. So we need to define what we mean by an absent variable. To achieve this, we introduce a reference instance, denoted x^b, which describes a reference situation of the case under study. In our applications, this could be a message typical of a communication channel, a perfect flight path, or even a set of black pixels, especially when working with images. The reference situation depends entirely on the situation under study. Suppose there is no single good instance that describes a reference situation. In that case, we can choose a set D of references, containing a set of references, calculate the SHAP values for each element of D, and get the average. Consider we have $x^b \in D$, then :

$$z_i = \begin{cases} x^b_i, if z'_i = 0, \\ x_i, if z'_i = 1 \end{cases} \tag{6.3}$$

When all variables are present, $z' = x'$ is the real situation. The other z' variables describe only theoretical cases, in order to calculate SHAP values. Since these are Shapley values, they verify the properties defined earlier. In particular, efficiency ensures that $g(x') = f(x)$, meaning that the explainable model should match the original one if evaluated when all the variables are present.

SHAP derivatives, such as LinearSHAP, KernelSHAP, and TreeSHAP, were developed for various contexts and applications, with DeepSHAP being best suited for deep neural networks.

6.3.3 DeepSHAP

[56] and [57] developed DeepLIFT, a method to backpropagate each neuron's contribution across a neural network to every input feature. It does so by comparing the activation of each neuron to its reference activation. It assigns a score to the difference. Formally, they

define $C_{\Delta x \Delta y}$ as the contribution of neuron x to neuron y, where $\Delta x = x - x_0$, with x_0 as the reference activation of the neuron and x its current value. The contribution $C_{\Delta x \Delta y}$ must follow the summing-to-delta property :

$$\sum_i C_{\Delta x_i \Delta y} = \Delta y \tag{6.4}$$

when summing over all entries of neuron y.

They also define a relevance measure per input, called the multiplier, as $m_{\Delta x \Delta y} = \frac{C_{\Delta x \Delta y}}{\Delta x}$. Conceptually, while $C_{\Delta x \Delta y}$ represents how much y varies with x, $m_{\Delta x \Delta y}$ approximates the partial derivative $\frac{\partial y}{\partial x}$. The summing-to-delta property applied to multipliers leads to the chain rule : $\sum_j m_{\Delta x_i \Delta y_j} m_{\Delta y_j \Delta t} = m_{\Delta x_i \Delta t}$. In a network where $x_{i \dots m}$ represents the input layer, $y_{1 \dots n}$ the hidden layer, and t the target layer, the chain rule allows for the computation of the contribution of x directly to t, bypassing the neuron y implicitly.

Building upon DeepLIFT, DeepSHAP [58] adapted SHAP to neural networks, propagating the Shapley values across the network using the multipliers. Shapley values $\varphi_i(f, x)$ become contributions $C_{\Delta x_i \Delta y}$ where $f(x) = y$, with the reference x_0 as the baseline x^b.

Fig. 6.1 presents a simple neural network with two features as input (represented by $x = (x_1, x_2)$), one hidden layer with two neurons ($y = (y_1, y_2)$, a non-linear activation function g, and an output layer with one neuron t. Thus, $t = f(x)$ is the prediction of the model. Consider we have chosen a reference x^b. We apply the chain rule to this network :

First, $C_{\Delta z_i \Delta t} = \varphi_i(f_3, z)$ for $i \in 1, 2$. Then, we define the multiplier :

$$m_{\Delta z_i \Delta t} = \frac{\varphi_i(f_3, z)}{z_i - z_i^b}, i \in \{1, 2\} \tag{6.5}$$

Similarly, we have $C_{\Delta x_j \Delta y_i} = \varphi_j(f_i, x)$ and :

$$m_{\Delta x_j \Delta y_i} = \frac{\varphi_j(f_i, x)}{x_j - E[x_j]}, j \in \{1, 2\} \tag{6.6}$$

These two multipliers give the contribution of their inputs to their outputs. Next, we handle the activation function by defining $\varphi_i(g, y) = g(y_i) - g(y_i^b)$ leading to

$$m_{\Delta y_i \Delta z_i} = \frac{g(y_i) - g(E[y_i])}{(y_i - E[y_i])}, i \in \{1, 2\} \tag{6.7}$$

Using the chain rule with (6.5), (6.6) and (6.7), we can finally compute the contribution of

each x_j to the output $f(x)$ as :

$$\varphi_j(f,x) = (x_j - x_j^b) \sum_{i \in 1,2} m_{\Delta x_j \Delta y_i} m_{\Delta y_i \Delta z_i} m_{\Delta z_i \Delta t}, , i \in \{1,2\} \tag{6.8}$$

This contribution calculation precisely represents the relative importance of each feature in the model prediction with respect to one single reference x^b. The exact notation should be $\varphi_j(f,x) = \varphi_j(f,x,x^b)$. We can then compute the SHAP values with respect to a reference distribution D as :

$$\varphi(f,x) = \frac{1}{|D|} \sum_{x^b \in D} \varphi(f,x,x^b) \tag{6.9}$$

This final method thus extends the use of SHAP values to a deep neural network with non-linearities.

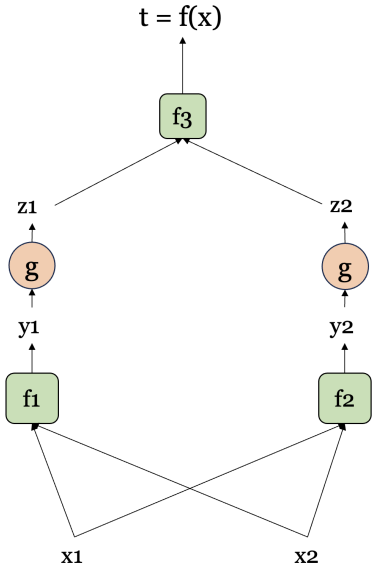


FIGURE 6.1 A simple neural network

6.3.4 Why DeepSHAP ?

DeepSHAP is well-suited for this work because its theoretical rigor ensures consistency and transparency, which are key factors in XAI for critical systems, such as avionics. The unicity of the solution provided by SHAP removes any notions of probability or uncertainty from the explanation results. It provides both local explanations—useful for understanding specific alerts or events, within an IDS context—and also provides global explanations for flight engineers to get the overall behavior of the model. This broader perspective helps identify tendencies and enhances the model's interpretability. Unlike LIME, which provides local

explanations by perturbing the inputs locally, SHAP focuses on the exact input values. Besides, the avionic domain requires the most accurate model of ML due to its critical environment. A deep neural network provides one of the most powerful and accurate models. The IDS developed in [10] is a powerful one that has shown great results in detecting anomalies in the avionic communication protocol MIL-STD-1553, using a complex transformer-based network. DeepSHAP is well-suited for complex models, as it handles non-linear networks and deeply analyzes feature connections.

6.4 Methodology

In this section, we explain our approach, first presenting the framework showing the path to integrate XAI into the IDS process, and then detailing an application scenario for this framework.

6.4.1 Framework

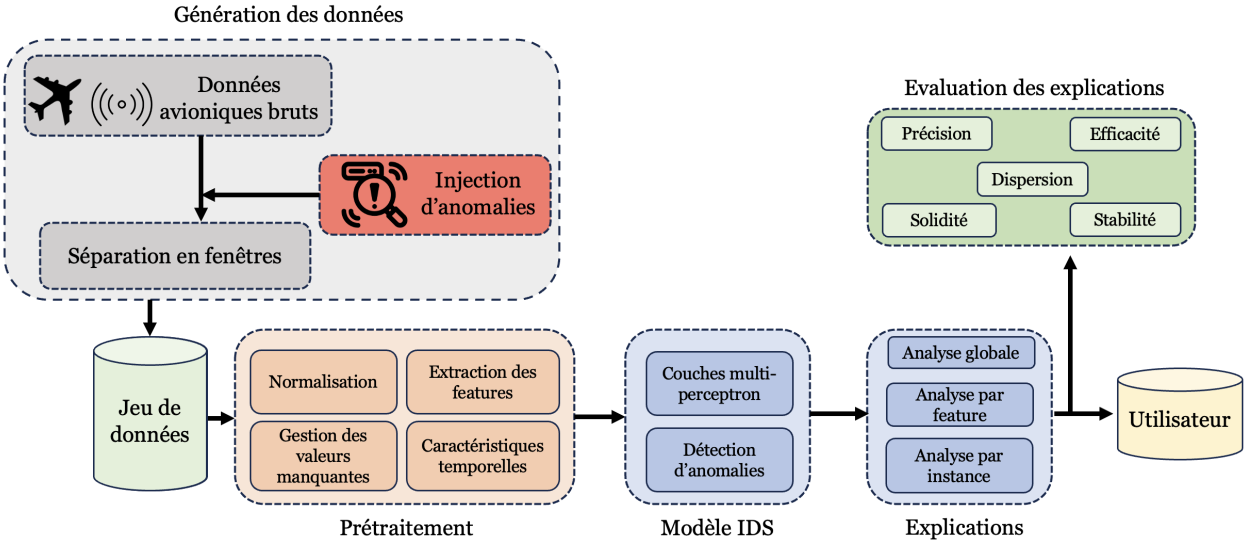


FIGURE 6.2 Framework

We introduce our approach to integrating XAI for explaining anomaly detectors in avionic communication systems in this section. Fig. 6.2 shows the framework of this approach. First, the gray module collects data from the communication system to construct a dataset used to train the IDS. We corrupt a part of the dataset using alteration techniques presented later in this paper. The rest of the dataset remains benign, allowing the model to learn to classify anomalies. A single data point alone does not carry enough context, so we consider a

data window instead. Then, the orange module aims to preprocess the data. In this module, we extract relevant features from this window (e.g., standard deviation, mean square error), which serve as the input to the model. We also handle missing values and scale the data. Next, we train the model in the blue module using a multi-layer perceptron deep neural network to classify whether a data window contains an anomaly. When an anomaly is detected, the explanation layer, shown in purple, provides a deeper analysis of the output. This analysis can take three forms : global, feature-based, or sample-based.

The global analysis summarizes the overall importance of each feature by ranking them based on their average impact (SHAP values) on the model's decisions, revealing which features contribute most. The feature analysis explores how individual features influence predictions by examining the relationship between their values and the corresponding SHAP contributions, which helps identify general trends. The sample analysis breaks down a single prediction, showing how each feature's value in that specific instance affects the model's output, allowing for localized interpretation.

In the last part of the framework, the green module, we explore methods for evaluating the techniques we use, firstly to take a critical look at our results, and above all, to check that they respect the constraints inherent in the field in which we are trying to apply them, in this case, avionics safety.

6.4.2 Scenario

We present the application of our approach for a specific scenario : air-ground communication. This approach aims to provide real-time support to both pilots and engineers in the event of a cyberattack on avionics. Specifically, consider an avionic communication system receiving or transmitting status messages between the aircraft and ground control, for instance. In this scenario, an external hacker managed to corrupt the message, threatening its integrity. When detecting the anomaly, the IDS on the ground station triggers an alert to indicate a potential intrusion.

At this moment, the pilot becomes aware of the attack. However, he has neither the time to deeply investigate such cybersecurity issues nor the expertise to do so. The priority for the pilot is to maintain the safety of the aircraft and passengers, which requires full attention. In contrast, the engineer, based on the ground, who has high expertise and time to investigate, is simultaneously alerted by the IDS about the breach. His role is to assist the pilot by analyzing the intrusion and providing instructions.

Our DeepSHAP mechanism is now integrated into the process, interpreting the IDS's output

to provide engineers with a precise understanding of attacks by visualizing the contribution of each feature. This helps identify compromised aspects of communication, enabling confident decisions.

We provide global and feature analyses to engineers, leveraging their expertise and time to interpret these complex insights. For pilots, we offer sample analysis, delivering an immediate, actionable understanding of their specific situation.

Possible engineer actions range from instructing pilots to switch communication channels to deferring to specialized response teams. The specific protocols for pilot action and communication fall outside the scope of this paper, as they depend on aviation command structures.

This scenario is designed to preserve the adequate mental state of each stakeholder concerning their roles. On the one hand, the pilot must remain calm and responsive to avoid stress and make informed decisions, thereby ensuring the aircraft's safety. On the other hand, the engineer must focus on having a thorough comprehension and analysis of the situation to enhance the system's resilience and identify the most effective countermeasures. By managing the workload distribution among the involved actors, this framework ensures that cybersecurity threats are addressed in a manner that minimizes stress and maximizes efficiency.

6.5 Results and discussions

In this section, we apply the scenario described above to the ADS-B system to detect and explain airborne attacks.

6.5.1 ADS-B

Automatic Dependent Surveillance-Broadcast (ADS-B) is a communication system for real-time air traffic management, recently mandated for nearly all aircraft to access controlled airspace. ADS-B has two modes of operation : *ADS-B In* enables an aircraft to receive information about its position or other kinematic data, such as groundspeed, directly from a transmitting satellite. *ADS-B In* can also receive meteorological and air traffic data from ground control towers. *ADS-B Out* enables the aircraft to broadcast the same information to other aircraft and nearby air traffic control towers. The primary goal of ADS-B is to improve aircraft access to traffic data, enhance situational awareness, and reduce accidents caused by insufficient knowledge of surrounding airspace.

ADS-B messages are broadcast by aircraft as 1090 MHz ES (Extended Squitter) signals (for Mode S transponders) or 978 MHz UAT (Universal Access Transceiver) signals (mainly in

general aviation). The message is comprised of a synchronization pattern to help the receiver determine the start of the message and an ICAO address to identify the aircraft. Moreover, the message type is specified. What interests us the most is the payload of the message, which contains the actual ADS-B data.

A significant drawback of this protocol is its lack of authentication and data encryption, which leaves communications vulnerable to external tampering and communication spoofing. Attackers can impersonate legitimate aircraft, and receivers cannot verify whether the data has been manipulated. Below, we briefly introduce a specific attack on the ADS-B protocol and a basic anomaly detector for its detection. We then demonstrate how XAI can provide valuable insights into the detection process.

6.5.2 Airborne Attacks

In this application, we follow the work of [18], which focuses on airborne attacks that primarily target message integrity and authentication, such as message modification, deletion, or insertion. An airborne attacker operates directly from an aerial position, either onboard the aircraft or remotely. The attacker is an external entity whose motivation is unspecified here. The attack scenario involves injecting anomalies into a targeted flight trajectory by exploiting one of the ADS-B protocol vulnerabilities studied in [3]. To simulate the attack, they injected a single data window (comprising 600 ADS-B messages) with anomalous behavior to alter the original flight trajectory as transmitted in the communication channel. We focus on two types of modifications :

1. Gaussian Noise Addition : Adds Gaussian noise to the original sample based on the feature's mean value.
2. Trajectory Modification : Alters the target flight trajectory by applying the cumulative sum of message differences from another real flight trajectory. The two types of flight used to alter other flights' trajectories are :
   — Maneuver (Flight AIB232E), which performs a complex maneuver resembling a large Christmas tree over Northern Europe, and ;
   — Landing (Flight TAP070), which represents a landing approach at Lisbon Airport.

Our work builds upon the foundational dataset and attack scenarios introduced in [18], which consists of 40 real flights (including seven altered via simulated attacks). While this dataset was designed for initial proof-of-concept validation, focusing on basic spoofing and injection attacks, we intentionally adopt it for two key reasons.

First, the simplicity of the attacks allows us to isolate the explainability of the IDS's decisions,

free from confounding factors like complex adversarial noise. This aligns with our core objective : to evaluate whether XAI methods, especially DeepSHAP, can provide human-interpretable insights into why an IDS flags anomalies, even in elementary cases. Understanding simple scenarios is a prerequisite for tackling complex ones.

Second, by reusing a published dataset and attack framework, we ensure our XAI analysis is grounded in prior work, enabling direct comparison with existing detection models. Future work can scale to richer datasets, but our contribution lies in methodology (XAI integration), not in curating new attack vectors.

We acknowledge that real-world attacks may involve more sophisticated tactics. However, our goal is not to propose a new IDS or attack model but to demonstrate that XAI can retrofit transparency into existing systems. The insights from this preliminary study, such as which features most influence the IDS, are generalizable to more complex scenarios.

6.5.3 Model Architecture

The model used is a classifier using the PyTorch library and employing a multi-layer perceptron with two hidden layers. The contribution here is the explanation layer added at the end to interpret the model's predictions, not the model itself. The model predicts whether a sequence of ADS-B messages contains an anomaly or is benign. If the classifier predicts an anomaly, the explanation layer, based on DeepSHAP, is activated. The features used in the model are ground speed, vertical rate, latitude, longitude, and altitude. They act as the "players" of the interpretative process. From these raw features, after normalizing them using standard scalarization, we extract the most relevant characteristics for each window : standard deviation, minimum and maximum values, median, and mean change. The final feature set comprises five sub-features for each feature, allowing us to reduce each window of 60 ADS-B messages to a single instance of data.

6.5.4 Results

We begin by simulating airborne attacks on three reference flights : BAW175, BAW9L, and CFG114. Our classifier model, trained to detect anomalies, successfully flags each of these attacks. We then perform a detailed analysis of these flights using the explanation layer. Fig. 6.3 presents the results of a global analysis for these three flights, each subjected to one type of attack at a time. Each column of the figure gives information for one specific attack. When analyzing the SHAP profiles, we see that similar attacks generate quite similar profiles. For instance, the first column is for the landing attack and for the three flights, the "Vertical Rate"

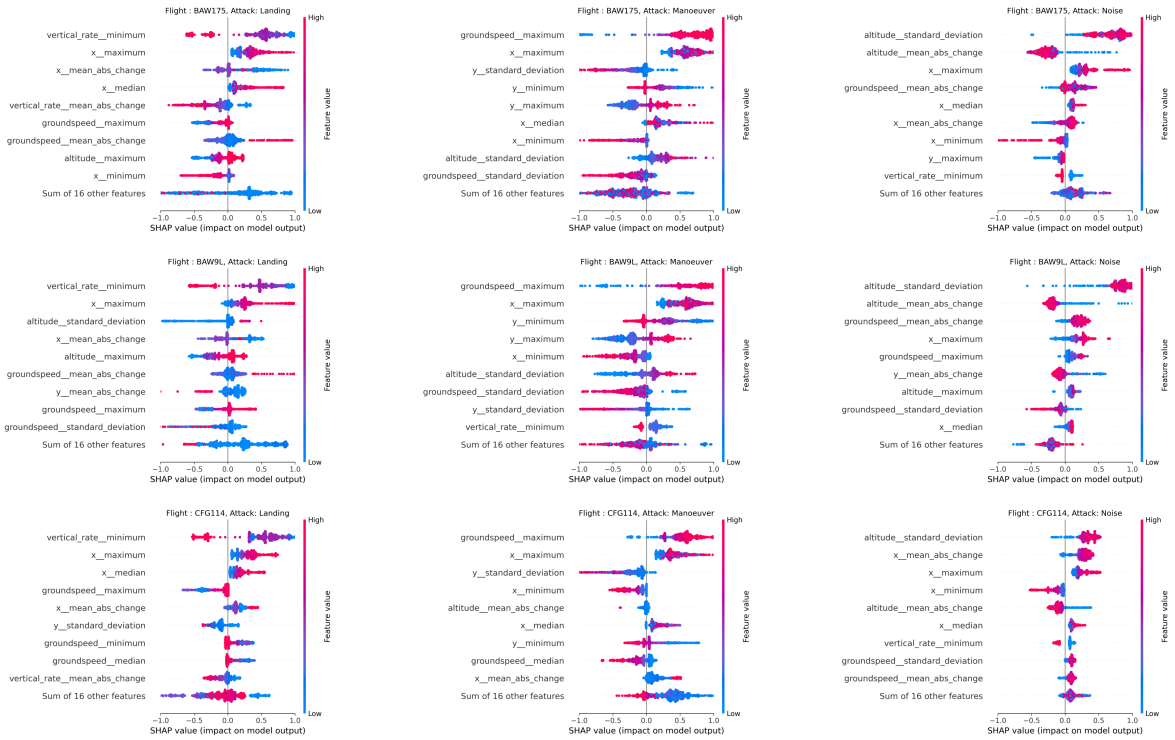


FIGURE 6.3 Global analysis on three flights for three different attacks

feature plays a significant role in the model's decision process, which aligns with expectations since this attack injects a landing flight (TAP070) into the target flights, where the vertical rate is a critical parameter. Not only the first feature, but also the subsequent ones, present the same profile. For the maneuver alteration attack, the most influential features shift to "longitude" (x), "latitude" (y), and "groundspeed," while "Vertical Rate" is less relevant. What does this reveal about how the model operates ? First of all, it indicates that SHAP profile analysis is a reliable method for identifying the attack under consideration. Indeed, we observe that the profiles exhibit a distinct pattern for each type of attack. By analyzing the SHAP profile of a flight detected under attack, we may be able to determine the type of attack in question. Secondly, when we analyze the three profiles of the same flight, we see almost no similarity. As SHAP values are calculated in relation to a reference, this reference has here been chosen as being the same flight without an attack. Therefore, since SHAP aims to identify the features responsible for a deviation from a reference, any similarity relative to this reference between two analyses is eliminated, displaying only the anomalies. This aspect is particularly relevant in a context where the aim is to minimize the amount of information to be transmitted. In this way, we do not transmit all the analyses of the flight in question, just what led the model to conclude that it was an attack.

Now, consider that we detect an alert on a new flight, VIR63, without knowing the type of attack it represents. For the sake of simplicity, we assume that the attack is among the three considered. We will revisit this aspect later. We then analyze the SHAP profile of the flight (Fig. 6.4) and we observe a distinctive attack pattern : "ground speed maximum" emerges as the most important feature, followed by "longitude" and "latitude." This analysis indicates that the flight, in relation to a normal flight reference, exhibited abnormal variations in position (x,y) and speed, reaching alarmingly extreme values. Based on the above profiles (6.3), we can conclude that this attack has a high probability of being a **maneuver attack**. Here again, we ask ourselves what more we have learned. In addition to detecting the type of attack, we have gained a better understanding of the strange trajectory taken by the aircraft, and thus why the model reacted to this situation. It is important to remember that, in reality, the plane did not follow such a trajectory ; it was only the data it transmitted about its position that was altered. Our understanding has therefore increased. Before proceeding, we should point out that, in a real case, there is not an accessible number of attacks listed, so we can not really establish a profile for each attack and choose the type of attack according to the profile. Again, this is a simplified scenario to open the door to the use of XAI in such a context.

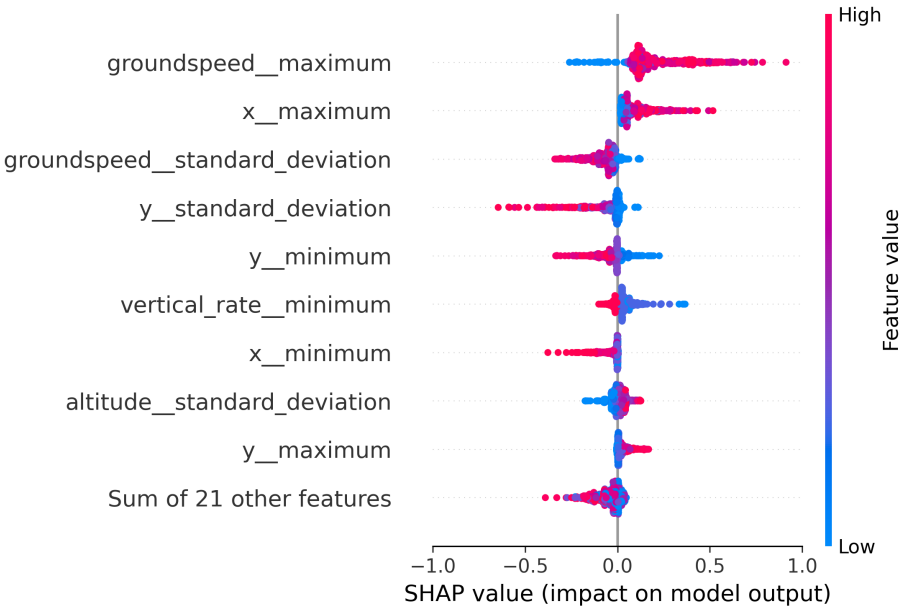


FIGURE 6.4 Profile analysis of the flight VIR63

We now consider another situation in which the flight is in progress, and the model, assumed to be onboard or on the ground but at least capable of detecting anomalies in real time, raises an alert. We are not talking here about waiting for the flight to end and analyzing the entire

flight a posteriori, as in the previous examples ; we are talking here about directly analyzing the instance that has been raised as abnormal. An attack throughout the entire flight would likely trigger numerous anomalies (each time abnormal data is detected). The more anomalies detected, the higher the likelihood that the detected behavior is genuinely anomalous, rather than a false positive. However, in this case, we are analyzing only the first anomaly raised, without any prior context on the detector's behavior throughout the flight. Fig. 6.5 illustrates an instance analysis for flight VIR63. We observe that the anomaly is primarily influenced by the "altitude" and "vertical rate" features. The personnel in charge of directing the aircraft, whether pilots or ground crew, are typically aware of the aircraft's current trajectory. In this specific case, they are expected to follow a flight path at a constant altitude and near-constant cruise speed. Thus, a model emphasizing abnormal values in altitude, speed, and angle of attack helps identify that the aircraft is under attack and that this attack has corrupted the data transmitted by the aircraft. One could also assume that the aircraft is indeed following an abnormal trajectory and that the attack involves the manipulation of internal cockpit indicators, but this scenario is outside the scope. An alert is triggered by a single data burst. Thus, verifying whether the aircraft is genuinely under attack requires analyzing subsequent data bursts to assess whether the initial hypotheses are invalidated or corroborated.

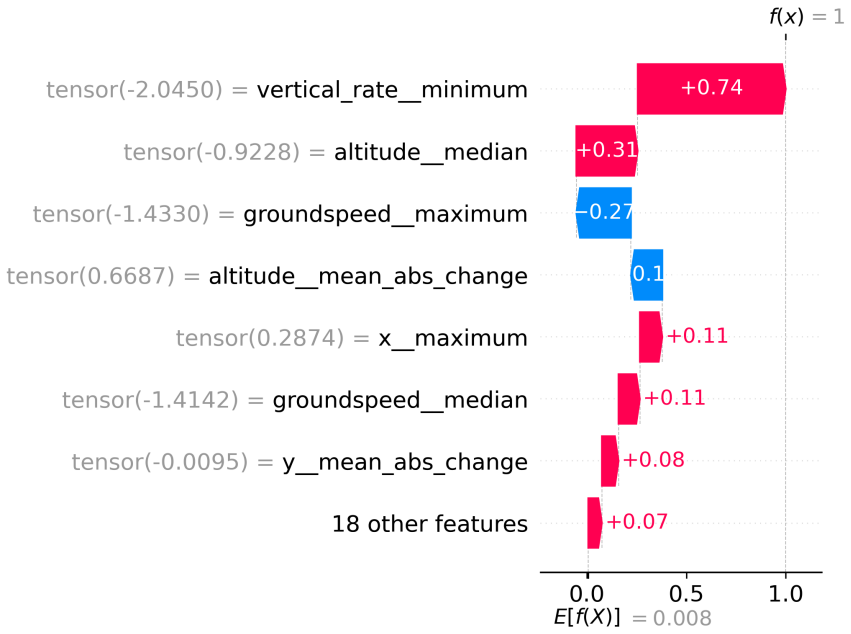


FIGURE 6.5 Sample analysis on the flight VIR63

In this last example, we examine how analyzing SHAP values helps us understand the impact of the "altitude standard deviation" feature on the model's decision process. Fig. 6.6 shows the

evolution of the SHAP value for this feature as its value increases. For low values, the SHAP value is also low, indicating minimal influence on the decision process. This corresponds to stable altitude periods in the flight. For intermediate values, however, the model's suspicion increases significantly. Specifically, when the "altitude standard deviation" reaches a value of 1, the feature alone is enough to trigger an alert (SHAP value = 1.0). For high values, the impact diminishes as these values align with typical altitude fluctuations during takeoff and landing. This pattern suggests that intermediate values are unusual as they do not correspond to standard flight phases (takeoff, cruising, or landing) but rather indicate a level of variation that doesn't fit these expected patterns. These intermediate values could thus represent random noise affecting the model's stability. This evidence suggests that the detected anomaly is likely due to a **noise attack**. This type of analysis is primarily reserved for ground analysts, as it requires examining each feature individually, and the resulting behavioral patterns and conclusions are more complex to derive. It enables analysts to gain a detailed understanding of the relationship between the model and a specific feature.

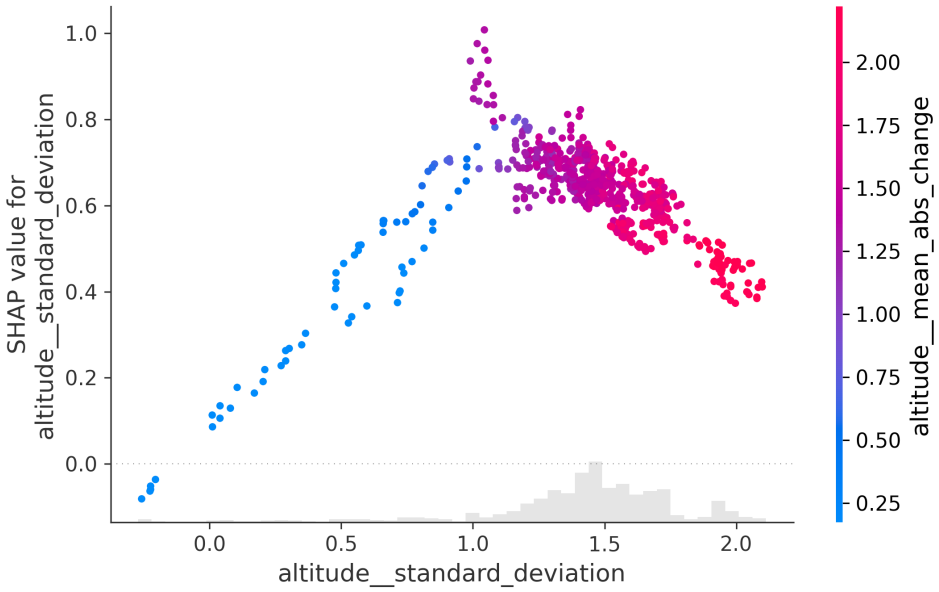


FIGURE 6.6 Shap analysis of the feature 'altitude standard deviation' through the flight VIR63

In this section, we have seen how the analysis of SHAP profiles can provide a more detailed understanding of the anomalies detected, and can even give a good idea of the attack in question from a list of pre-determined attacks. However, our case study allows us to cover only those cases where we have prior knowledge of the potential attacks. This layer of

explicability needs to be further developed before it can be used to explain and identify new attacks. Furthermore, although SHAP values are calculated deterministically and uniquely, the conclusions drawn from their profiles remain speculative. The importance of specialized aeronautical engineers becomes crucial for analyzing these profiles because the more one understands the behavior of features during flight, the more they will be able to draw relevant conclusions from SHAP profiles. As our knowledge of aeronautical behavior in this paper is bounded, our analyses and conclusions are also fairly limited. The aim here is, above all, to demonstrate the applicability of the framework presented in a specific case.

6.6 Evaluation

In this section, we attempt to evaluate the results obtained earlier using rational metrics. In fact, although we were able to obtain satisfactory results enabling a better understanding of the model, we do not really have any information on the quality of these explanations. In his book, Molnar [46] presents several components to consider when evaluating the quality of an explanation ; however, we will use the metrics derived from [41], which have been adapted and modified to meet our specific needs. We will present 5 metrics : accuracy, sparsity, robustness, stability, and efficiency. For each of these metrics, we will first define its relevance to our work and then explain how to calculate and apply it to our results.

6.6.1 Accuracy

Accuracy assesses the explainable model's ability to detect the features that are most important for prediction. To assess this, we examine how the model behaves in the absence of these important features. If the explainable model has done its job well, the model's prediction without the important features should be poor. Removing important features simply means replacing their value with their baseline value. We then introduce a parameter k to designate the k most important features of the model. This is known as k-order accuracy. Formally :

$$ACY_k(x, y, f) = ((f(x|x_1 = x_1^b, ..., x < -k = x_k^b) - y)^2 - (f(x) - y)^2)^2 \tag{6.10}$$

In this equation, y is the actual classification of the instance and $f(x)$ is the model prediction. $(f(x) - y)^2$ describes the local accuracy of our model for the instance x, and $ACY_k(x, y, f)$ describes how much we have lost in accuracy by removing the k most important features. Assuming that $f(x) \approx y$, the ideal case is that $f(x|x_1 = x_1^b, ..., x_k = x_k^b) \approx 1 - y$ leading to $ACY_k(x, y, f) = 1$. Removing the most important features changes the classification. The worst case for an explainable model is that the prediction is the same ie $f(x) = f(x|x_1 = x_1^b, ..., x_k = x_k^b)$ meaning that $ACY_k(x, y, f) = 0$. Thus, the closer the accuracy is to 1, the

better the explainable model. We need to correctly choose k to be consistent with the scenario. To find k, we need to ask, "How many features should I get to explain the situation to a pilot ?". In real life, we often use the rule of three to describe a situation, meaning that 3 arguments are most of the time sufficient to have a good understanding of a situation. The Fig. 6.7 shows the evolution of the accuracy of our explainable model with the parameter k. For $k = 3$, we have $ACY(x, y, f(x)) \approx 0.9$, which is a pretty good result. When removing the three features detected as most important, the model's accuracy drops by 90%.

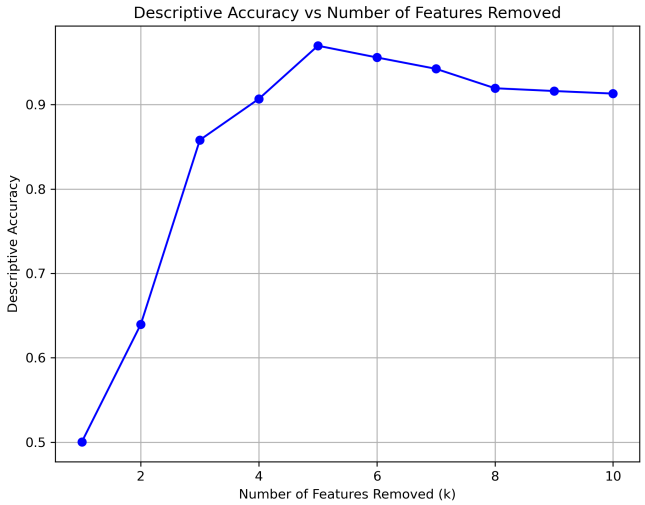


FIGURE 6.7 Accuracy of the explainable model

6.6.2 Sparsity

A complementary metric to accuracy is sparsity, which ensures that the majority of features have a minimal or negligible impact on prediction. Only a handful of features should be significant in a good explanation, as we said. To calculate this, we plot the histogram of SHAP values over the interval $[-1, 1]$ and verify that it has a high mass around 0. Let h be the histogram of SHAP values and r a radius to define a neighborhood of 0. We define the mass around 0 as :

$$MAZ(r) = \int_{-r}^{r} h(x)dx \tag{6.11}$$

The MAZ is a window of radius r that measures the density of SHAP values contained in the window. For $r = 1$, the MAZ covers all the values and $MAZ(1) = 1$. Inversely, if $r = 0$, the window is empty and $MAZ(0) = 0$. We then plot the histogram of SHAP values and the evolution of MAZ as a function of the radius. The speed at which the MAZ evolves towards 1

indicates the quality of the sparsity of the explainable model. Fig. 6.8 shows the histogram of the SHAP values while Fig. 6.9 presents the evolution of the MAZ with the radius r. We see that most of the values are low, as the histogram shows a quick high mass around zero. With $r = 0.2$, we have $MAZ(r) = 0.8$, meaning that 80% of the SHAP values are below 0.2.

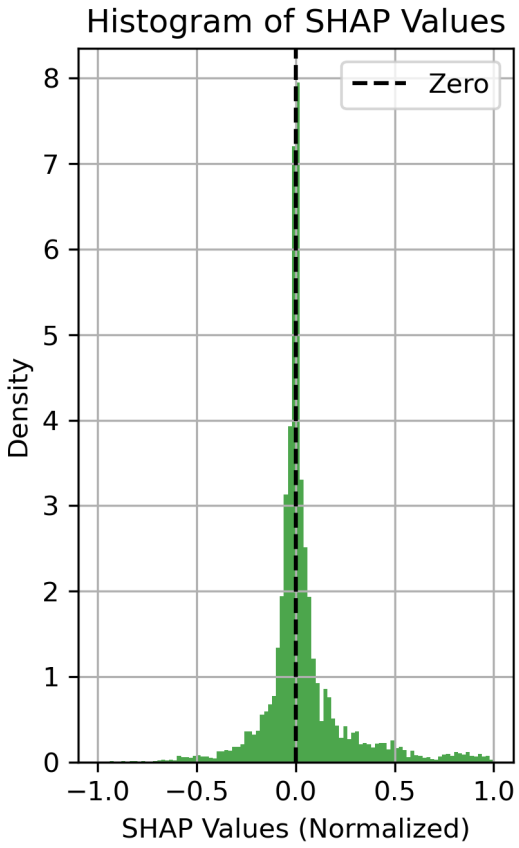


FIGURE 6.8 SHAP values histogram

6.6.3 Efficiency

The explanation process is an additional process to the initial intrusion detection process. An IDS seeks to be as fast as possible to meet the time constraints imposed by high-risk fields such as avionics, where the decision-making process is crucial and very short. In this context, the explanation process must be efficient in the sense that it must not slow down the already constrained process. To measure the efficiency of an explainable model, we simply need to measure the time taken by the model to provide one or more explanations. There are no absolute good values for the efficiency, but they must be put into perspective with the

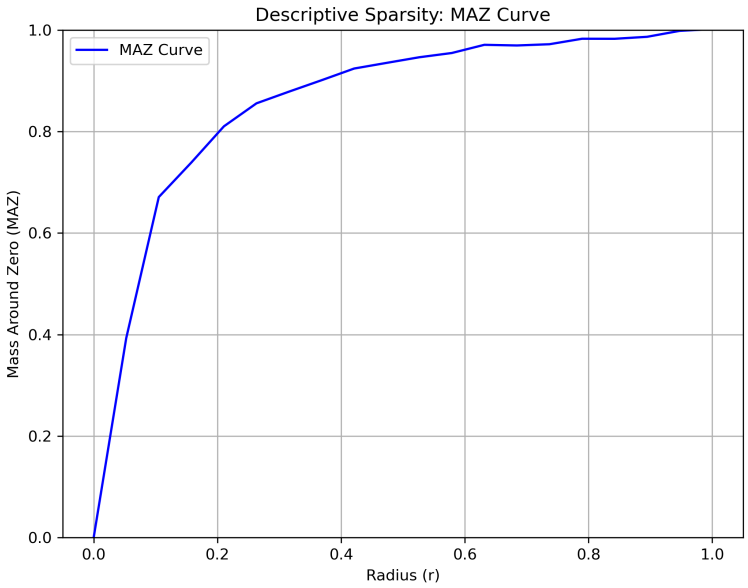


FIGURE 6.9 Mass Around Zero (MAZ) of the explainable model

context in which they are applied. As the Fig. 6.10 shows, the explanation process is logically linear with the instances it explains and takes around 2.5 seconds to explain an entire flight (about 2500 instances). The explanation rate is consequently 1ms per instance. In the ADS-B protocol, the interval between two messages is between 0.5 and 1 second. Our explanation frequency is therefore more than satisfactory, which was indeed the desired goal when we chose DeepSHAP, specially designed to quickly explain deep networks.

6.6.4 Stability

Another important metric in a critical infrastructure is the stability of the results generated. An explanation must be reliable in the sense that it must not provide different results for the same inputs. In other words, for two runs i and j of the explainable model, the most important feature sets of the two runs T_i and T_j must have an intersection close to 1, i.e. $IS(i,j) > 1 - \epsilon$, where IS is the intersection size and ϵ a threshold close to 0. This metric is important to check for probabilistic methods whose results are likely to vary between two runs. However, DeepSHAP is purely deterministic, as we saw in the theoretical section. DeepSHAP ensures that the intersection size is strictly equal to 1 under all circumstances. Like efficiency, this is one of the reasons that motivated us to use DeepSHAP.

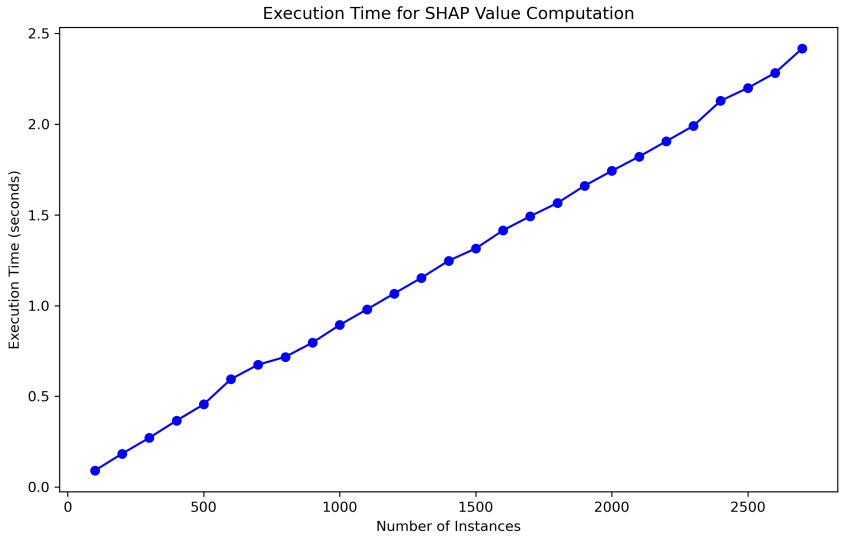


FIGURE 6.10 Efficiency

6.6.5 Robustness

Finally, an explanation must be robust in the sense that it must be resilient to adverse perturbations designed to alter the results of the explainable model. To check this, we add Gaussian noise to the instances to be explained and see how this affects the explanations. We compute the robustness as the square root distances between the SHAP values computed before and after adding the noise. Fig. 6.11 shows that our method is highly resilient to random modifications by Gaussian noise, as it maintains a robustness score above 0.9 even with very high levels of noise. In terms of magnitude, our instances are normalized, so they have values close to unity. However, our method for measuring this metric is too naive. Indeed, injection attacks do not randomly attack data, but rather target data precisely so that the intended model, in this case, the explainable model, is incorrect in its prediction, while keeping the attack minimal. Random noise simulates such an attack very poorly. However, we will content ourselves with this metric for the time being and keep the in-depth version for future work.

Finally, these five evaluation metrics enable us to conclude that our evaluation model is highly precise in identifying important features while assigning little weight to trivial ones. In addition, it respects the execution times imposed by the avionic context, and its stability and robustness make it a valid tool for use in a critical infrastructure. The limitations of this section are the poverty of the robustness calculation method and, above all, the absence of a

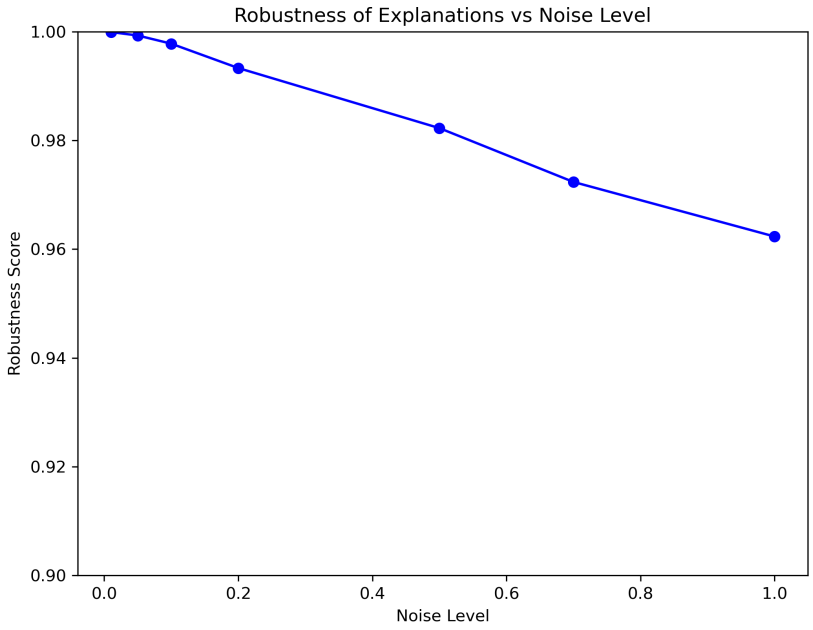


FIGURE 6.11 Robustness

benchmark with which to compare our measurements. Since XAI and these metrics are still in their infancy, we present our results here without benchmarks. Nevertheless, we can say that, according to our criteria, they are encouraging.

6.7 Conclusion

This study presents a novel approach that integrates XAI into an anomaly detector specifically designed for avionics communication networks. Our approach uses DeepSHAP to provide clear explanations at the feature level of detected anomalies, enhancing confidence and ease of use within the aviation industry. By providing a transparent view of how each feature contributes to model decisions, our framework ensures that even non-AI expert stakeholders can interpret and respond effectively to IDS alerts.

Our results demonstrate that DeepSHAP-based explanations facilitate a deeper understanding of anomaly sources, providing engineers with the necessary information to make informed safety decisions without imposing additional cognitive load on pilots. Furthermore, the method's rigorous theoretical underpinnings align well with aviation's high standards for reliability and deterministic results, bridging the gap between cutting-edge ML techniques and the rigorous requirements of avionics systems.

In this work, we have chosen a simple model to build our IDS, on the ADS-B protocol, as its features are easier to grasp. In the future, it will be interesting to extend this work to more advanced projects, such as the IDS developed in [**?**] to detect intrusions in the MIL-STD-1553 protocol. Other future works could extend this study by testing it using a flight simulator to conduct experiments on pilots to test the acceptance of XAI among their tools, as well as their response to attacks when understanding their origin. Further, we will be able to better train defense models by better knowing their main impacts and root causes, increasing the efficacy of such defenses.

The aim of this future work is to further explore the major limitations identified in this article. Indeed, we have identified that the interpretation of SHAP graphs requires expertise in data analysis and a certain amount of processing time, making them difficult to use directly in the cockpit. In addition, an in-depth analysis is required if the attack in question is not a listed attack. Work on the simulator is therefore aimed at separating SHAP results that can be directly interpreted and acted upon from those that require a posteriori analysis.

CHAPITRE 7 TENTATIVE D'EXPLICATION D'UN *TRANSFORMER* AVEC SHAP POUR LES COMMUNICATIONS SUR LE PROTOCOLE MIL-STD-1553

Ce chapitre est un peu à part des deux autres applications que nous venons de présenter. Il vise plutôt à utiliser le formalisme mathématique que nous avons développé en chapitre 3 pour tenter d'étendre le champ d'application de SHAP à des modèles plus complexe. Dans la forme, ce chapitre ne suivra donc pas la méthodologie développée dans le chapitre 4 et ses résultats ne seront pas comparés à ceux des autres applications. Il s'agit d'une exploration purement théorique.

7.0.1 Contexte

Dans *TRIPT-IDS : Triplet Loss Pre-trained Transformer for Avionic Intrusion Detection System* [10], Jean-Simon Marrocco, un ancien élève de Polytechnique Montréal en maîtrise, a développé un IDS utilisant un *Transformer* pour identifier les messages corrompus dans le protocole MIL-STD-1553. TRIPT-IDS utilise non seulement un Transformer, mais il va surtout l'utiliser en série avec d'autres couches de neurones, comme les couches de vectorisation et celle de détection d'anomalies en fin de réseau. Ce modèle est donc une bonne référence de ce que l'on peut appeler un réseau de neurones profonds complexes, en opposition aux cas précédemment étudiés. Le principe même de *Transformer*, que l'on commencera par présenter, est d'une complexité algorithmique telle que la librairie SHAP en python ne peut pas être utilisée directement dessus. L'architecture du modèle complet, étudiée en second temps ne facilite pas non plus le travail d'explicabilité. Jusqu'ici, l'étude théorique de SHAP a été certes utile pour mieux comprendre chacune des utilisations qui en a été faite, cependant, cela n'était pas strictement nécessaire pour utiliser la librairie SHAP qui est pensée pour être facile d'utilisation même pour ceux avec peu de connaissances sur le sujet. Notre application au protocole ADS-B aurait pu être effectuée à peu près de la même manière avec des connaissances moindres sur SHAP. Ce qui n'est plus le cas à partir d'ici où l'on va devoir ré-invoquer notre formalisme étudié en partie 3.3. La troisième partie de cette section aura donc pour but de propager à la main les valeurs SHAP à travers TRIPT-IDS, à l'aide d'une approche mathématique.

7.0.2 *Transformer* et mécanisme d'attention

Un *Transformer* est un modèle d'apprentissage profond qui utilise principalement un mécanisme appelé *self-attention*, pour traiter des séries de données, comme du texte ou des séquences temporelles. Contrairement aux modèles précédents comme les RNNs et les LSTMs, les *Transformers* ne traitent pas les séquences de manière séquentielle, mais de manière parallèle améliorant ainsi l'efficacité des calculs.

Le mécanisme clé dans un *Transformer* est l'attention, qui permet au modèle de pondérer l'importance de chaque élément d'une séquence par rapport aux autres. Plus précisément, on utilise le *Scaled Dot-Product Attention*, où chaque mot d'une séquence est représenté par un vecteur, et l'attention est calculée en fonction de trois vecteurs : *Query* (Q), *Key* (K) et *Value* (V). La similarité entre la *Query* et la *Key* détermine l'attention, et les valeurs associées sont ensuite combinées pour produire la sortie. Le *Transformer* est composé de plusieurs couches d'encodeurs et de décodeurs. Chaque couche d'encodeur comprend deux sous-couches : une couche d'attention multi-tête (qui permet de capturer différentes relations entre les éléments) et une couche de réseau de neurones *feed-forward*. Chaque sous-couche est suivie de mécanismes de normalisation et de résidus. Les *Transformers* modernes peuvent avoir des architectures très profondes, avec des centaines de couches, et un grand nombre de paramètres (parfois plusieurs milliards). Cette profondeur et cette largeur ajoutent une couche supplémentaire de complexité à l'interprétation, car chaque couche peut *Transformer* l'information de manière non triviale, ce qui rend difficile de retracer comment chaque entrée initiale influence la sortie finale.

Les équations pour décrire comment sont calculées les valeurs sont détaillés dans *Attention is all you need* [65] et sont les suivantes :

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{7.1}$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{7.2}$$

$$\text{où head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{7.3}$$

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{7.4}$$

Ces équations ne sont pas importantes en soit dans notre contexte, mais il est important de comprendre qu'il va falloir être capable d'y faire traverser les valeurs SHAP, si nous voulons l'utiliser dessus. Ce sont ces mêmes équations, entre autres, qui font que la librairie actuelle SHAP ne fonctionne pas sur un *Transformer*. Intéressons-nous maintenant plus largement à l'architecture complète du modèle.

7.0.3 Préparation des données du protocole MIL-STD-1553

Le modèle prend en entrée des messages issus du protocole MIL-STD-1553. Présentons alors rapidement comment fonctionne ce protocole. D'abord, trois composants constituent le réseau :

— *Bus Controller (BC)* : le BC est responsable de la gestion du trafic des données sur l'ensemble du bus. C'est le maître des communications et supervise l'ensemble des terminaux à distances.

— *Remote Terminals (RT)* : Les RTs sont des terminaux esclaves connectés au BC, répondant à ses directives. Un RT peut envoyer ou recevoir des données en fonction des directives.

— *Bus Monitor (BM)* : Le BM est un dispositif qui surveille et enregistre les activités du bus. Il ne participe pas directement aux échanges, mais est utilisé pour la détection d'erreurs et l'analyse des communications.

Les messages émis par ces différents composants sont constitués de ce qu'on appelle des "mots", qui servent d'unité de base pour constituer des communications. Il y a trois types de mots :

— *Mot de commande* : Émis par le BC, il indique au RT les actions à effectuer. Il est composé de 16 bits, dont 5 pour identifier le terminal visé (allant de 1 à 30), 1 bit indiquant le sens de la communication (1 pour BC→RT et 0 pour RT→BC), 5 bits pour spécifier le type données à transmettre, et 5 bits pour préciser le nombre de données nécessaire pour décrire l'action.

— *Mot de statut* : Émis par le RT en réponse au BC pour décrire son état. Il comprend également 16 bits dont 5 pour décrire l'adresse du RT qui écrit et 11 bits pour donner des informations supplémentaires si nécessaire (état actif/inactif, état de la réception, erreurs détectées, etc...).

— *Mot de données* : Émis soit par le BC soit par le RT selon le sens de la communication, il contient le cœur du message à transmettre. Il est composé de 16 bits de données bruts. Le nombre de mots de données transmis est spécifié par le mot de commande sur 5 bits. Il peut donc y avoir jusqu'à 32 mots de données dans un message.

Pour chaque type de mots, il y a en plus 3 bits de synchronisation au début et 1 bit de parité à la fin. Les bits de synchronisation servent à identifier le type de mots (1 0 0 pour un mot de commande, 0 1 0 pour un mot de statut et 0 0 1 pour un mot de données). Le bit de parité permet d'assurer que le nombre de bits à 1 dans le mot est bien impair. L'émetteur compte donc à la fin de son message le nombre de 1 et ajuste le bit de parité en conséquence. Cela ajoute une couche de sécurité pour prévenir d'interférences ou d'autres problèmes d'intégrité. La Figure 7.1 récapitule la constitution de chacun des types de mots.

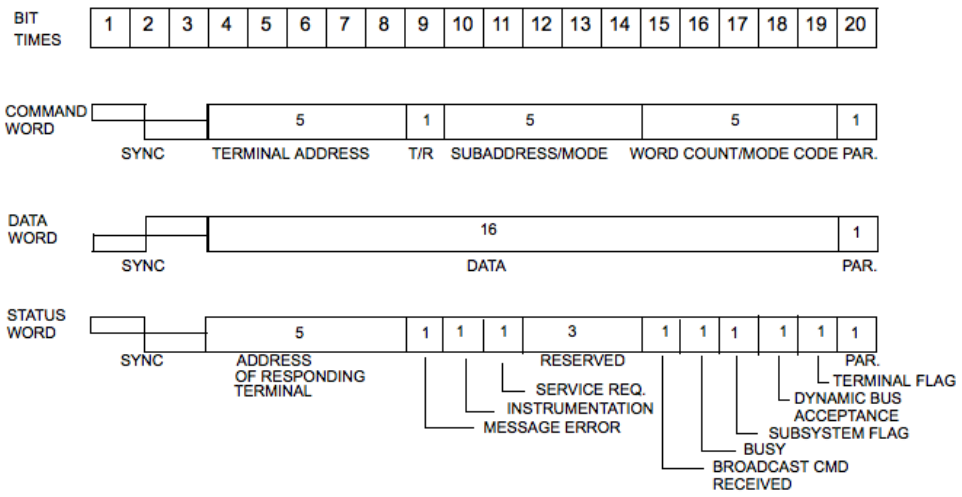


FIGURE 7.1 Structure des mots dans le protocole MIL-STD-1553

On choisit des fenêtres de 5 mots pour constituer une instance du modèle. Celui-ci ne prend cependant pas directement en entrée le message, mais en prend une version vectorisée. Cette vectorisation est faite au préalable et a pour but d'identifier clairement les différents mots composant le message et également leur type. La Figure 7.2 illustre les différentes façons d'appréhender le message d'origine. En haut, on trouve le message d'origine composé de 5 mots : 2 mots de commandes, 2 mots de données et 1 mot de statut. La première ligne du tableau concatène simplement ces 5 mots dans un seul vecteur que l'on note x^o. La seconde ligne est une écriture semblable à l'écriture originale, mais à laquelle on a ajouté un tag, propre à chaque type de mot. On note x^t cette écriture. Enfin, la troisième ligne du tableau montre la version vectorielle notée x^v. Le vecteur est de la même taille que le nombre de mots (5), auquel on ajoute 1 ligne pour les éléments de classe, sur lesquels nous reviendrons plus loin. Pour chaque ligne i de x^v, si le mot i est un mot de données, on remplit les éléments x^v_{i1} et x^v_{i2}. Si c'est un mot de commande, on remplit les éléments de x^v_{i2} à x^v_{i5}. Si c'est un mot de statut, on remplit les éléments de x^v_{i6} à x^v_{i17}. Les éléments de classe servent à la classification finale et sont les éléments qui porteront leur "attention" sur tous les autres composants tout au long du modèle. Pour plus de détail, voir le papier *BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding* [66]. Pour le moment, ne les considérons pas. Nous avons donc $x^v \in \mathbb{R}^{5\times 17}$.

7.0.4 Architecture de TRIPT-IDS

La Figure 7.4 présente l'architecture du modèle TRIPT-IDS avec en entrée, un message de 5 mots et en sortie, 0 si le message est sain et 1 s'il est corrompu. Cette architecture est

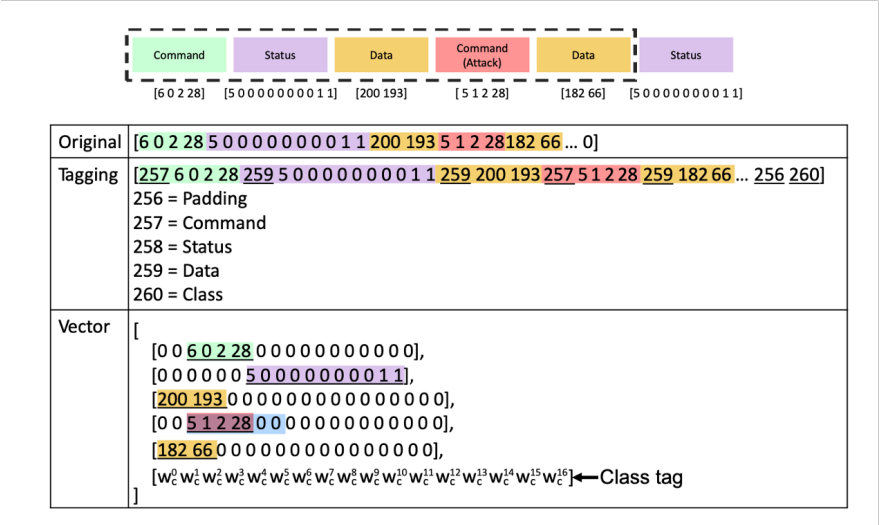


FIGURE 7.2 Vectorisation du message d'entrée

constituée d'une part du module TRIPT, et d'autre part de la tête d'anomalie réalisant la tâche finale de classification. La Figure 7.3 présente le fonctionnement interne du module TRIPT. Voici une explication détaillée couche par couche.

— *Couche de vectorisation* : convertit des données discrètes (comme des mots ou des indices) en vecteurs continus denses pour détecter des relations sémantiques ou contextuelles entre les composants.

— *Encodage de position* : ajoute des informations sur la position dans une séquence pour permettre au modèle d'intégrer l'ordre des éléments dans sa compréhension des relations.

— *Encodage Transformer* : traite les données séquentielles en capturant, à l'aide de mécanismes d'attention, les relations entre les éléments.

— *Couche dense* : transforme les caractéristiques apprises en une représentation adaptée à la tâche finale, ici une tâche de classification.

7.0.5 Propagation des valeurs SHAP

Dans cette partie nous allons utiliser G-DeepSHAP au modèle TRIPT-IDS, en propageant manuellement les valeurs SHAP à travers le réseau, en le considérant comme une série de modèle. Décomposons le réseau suivant le modèle de la Figure 3.7.

La Figure 7.5 représente mathématiquement la structure du modèle couche par couche, avec les dimensions de notre vecteur à chaque étape, ainsi que les valeurs SHAP associées.

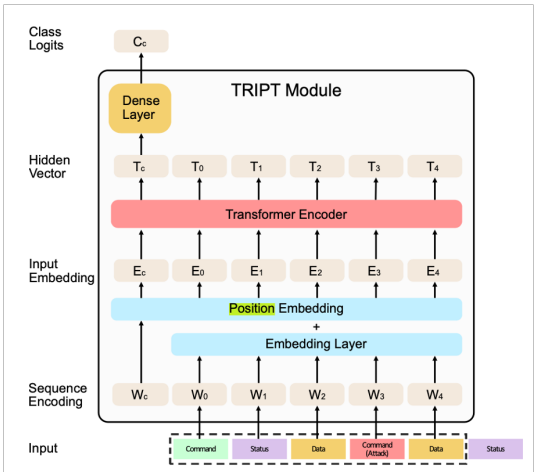


FIGURE 7.3 Architecture du module TRIPT

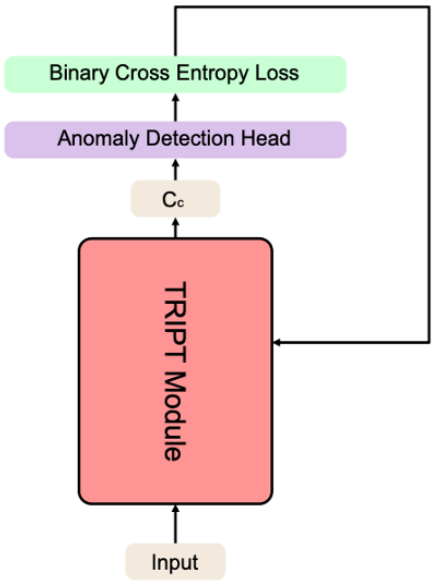


FIGURE 7.4 Architecture de TRIPT-IDS

Analysons cette figure en commençant par l'entrée $x^v \in \mathbb{R}^{5\times17}$. Puisque nous ne travaillerons qu'avec la représentation vectorielle du message, on confondra par simplicité x^v avec x. La couche de vectorisation, représentée par la fonction h_1, est implémentée par un neurone linéaire transformant la dimension $d_{sequence} = 17$, propre au message d'entrée, en dimension de vectorisation $d_{model} = 16$, propre au modèle. La matrice de h_1 est donc dans $\mathbb{R}^{17\times16}$ et la sortie du premier bloc vaut $f_1(x) = h_1(x) \in \mathbb{R}^{5\times16}$.

On ajoute alors à $f_1(x)$ une sixième dimension, la dimension de classe. Cette étape s'appelle la transformation en jeton de classe ou *class tokenization* qui transforme simplement via une fonction t_1, $f_1(x)$ en $f_1'(x) = t_2(f_1(x)) \in \mathbb{R}^{6\times16}$. Cette dernière ligne ajoutée est initialisée aléatoirement. Elle trouvera son intérêt dans le bloc *Transformer*.

Ensuite, l'encodeur de position transforme le vecteur suivant la fonction $h_2 = PE + Id$, où PE encode des informations sur la position de chaque élément. Elle est définie dans la littérature,

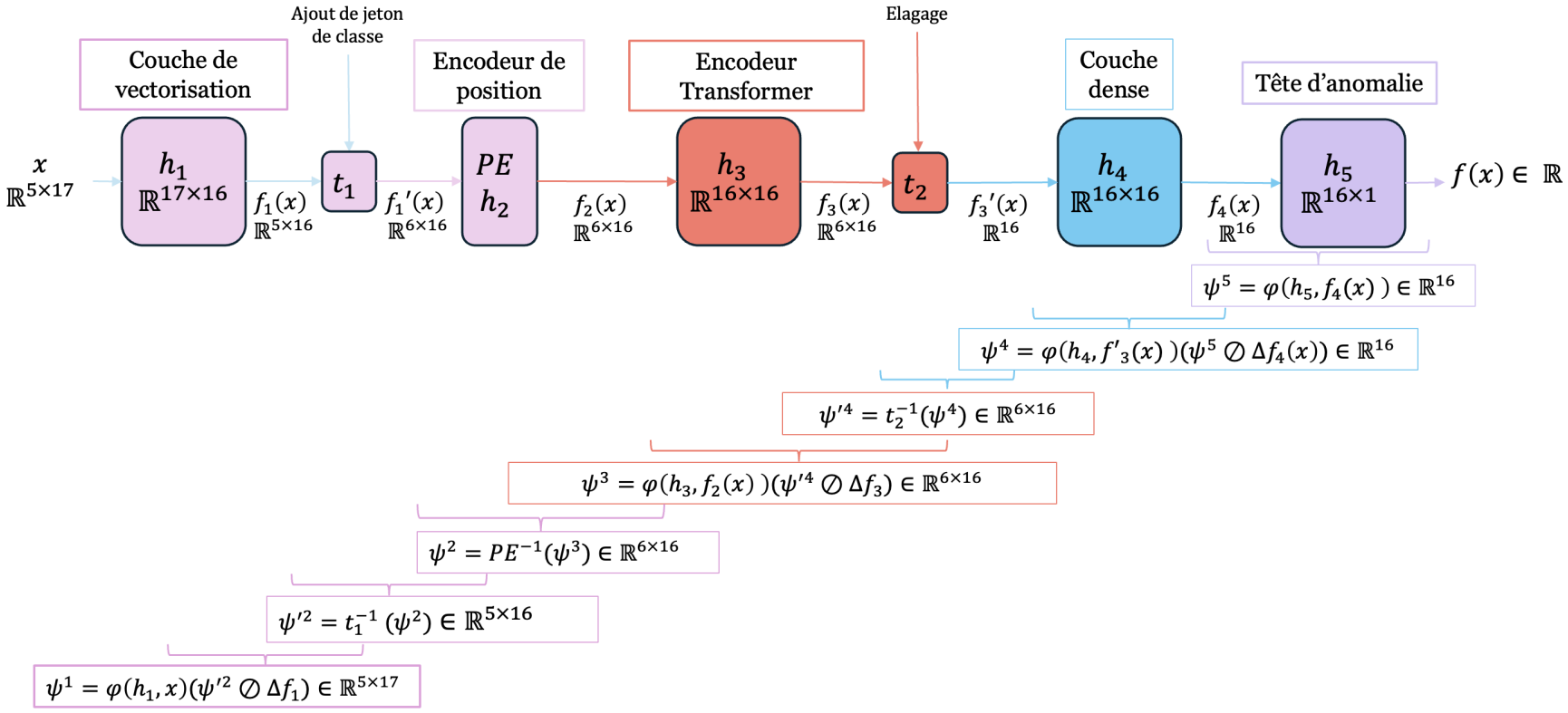


FIGURE 7.5 Décomposition du modèle TRIPT-IDS et propagation des valeurs SHAP

telle que :

$$PE_{(\text{pos, 2i})} = \sin\left(\frac{\text{pos}}{1000^{\frac{2i}{d_{\text{model}}}}}\right) \tag{7.5}$$

$$PE_{(\text{pos, 2(i+1)})} = \cos\left(\frac{\text{pos}}{1000^{\frac{2i+1}{d_{\text{model}}}}}\right) \tag{7.6}$$

h_2 ne change donc pas la dimension du vecteur d'entrée soit $f_2(x) = h_2 \circ f_1'(x) \in \mathbb{R}^{6\times16}$.

L'encodeur *Transformer*, cœur du modèle et basé sur le mécanisme d'attention, transforme le vecteur via une série de mécanismes complexes que nous avons présenté dans les équations 7.1. Celui-ci est représenté par la fonction h_3, qui conserve également les dimensions. Ainsi $f_3(x) = h_3 \circ f_2(x) \in \mathbb{R}^{6\times16}$.

Suite au *Transformer*, le vecteur de classe, dernière dimension de notre vecteur propagé, a pu "enregistrer" via le mécanisme d'attention tous les éléments nécessaires à la future classification. Les 5 premières lignes du vecteur ne sont alors plus utiles et on effectue un élagage via la fonction t_2, pour ne garder que la ligne de classe. Cet élagage est visible également sur la Figure 7.3. Notre instance vaut donc à cette étape : $f_3'(x) = t_2 \circ f_3(x) \in \mathbb{R}^{16}$.

La couche dense, représentée par la fonction h_4, est composée d'un neurone linéaire qui conserve la dimension avec une fonction d'activation en *LeakyRelu*. En sortie de cette couche, on a $f_4(x) = h_4 \circ f_3'(x) \in \mathbb{R}^{16}$.

Enfin, la tête d'anomalie extrait de la sortie du module TRIPT, la classification du message d'origine via un neurone linéaire et une fonction d'activation en *sigmoïd*. h_5 permet en définitive d'obtenir la prédiction finale $f(x) = f_5(x) = h_5 \circ f_4(x) \in \mathbb{R}$. $f(x)$ est plus exactement dans $[0, 1]$ mais peu importe.

Remontons alors en arrière du réseau et propageons les valeurs SHAP suivant la formule de récurrence de G-DeepSHAP 3.3.15. Dans les notations, oublions par simplicité d'ajouter les références dans la notation des valeurs SHAP soit $\varphi(f, x, x^b) = \varphi(f, x)$, et notons $f_i(x) - f_i(x^b) = \Delta f_i(x)$ conformément à la notation issue de DeepLIFT. On notera en rouge les variables encore à calculer ou à définir, et en vert celles directement accessibles et connues.

D'abord $\psi^5 = \varphi(h_5, f_4(x)) \in \mathbb{R}^{16}$. Ce vecteur décrit les contributions de chaque élément de $f_4(x)$ à la prédiction finale $f(x) = h_5(f_4(x))$.

Ensuite, on calcule $\varphi(h_4, f_3'(x)) \in \mathbb{R}^{16\times16}$ qui décrit les contributions de $f_3'(x)$ à $f_4(x)$. On en déduit $\psi^4 = \varphi(h_4, f_3'(x))(\psi^5 \oslash \Delta f_4(x)) \in \mathbb{R}^{16}$ qui donne la contribution de $f_3(x)$ à la prédiction finale $f(x)$.

Comment ensuite remonter la fonction d'élagage ? On pourrait simplement continuer en calculant $\varphi(t_2, f_3(x)) \in \mathbb{R}^{6\times16\times16}$, mais en réalité cette fonction va pouvoir se traiter bien plus simplement. Déjà, il faut comprendre que la fonction t_2 supprime les éléments $f_3(x)_i, i \in \{1, ..., 5\}$ pour ne conserver que l'élément $f_3(x)_6$. Cela implique que la contribution des 5 premiers éléments de $f_3(x)$ à $f_3'(x)$ est nulle et celle du dernier élément vaut exactement $f_3'(x)$ par axiome d'efficacité. On peut donc créer $\psi'^4 \in \mathbb{R}^{6\times16}$ tel que $\psi_6'^4 = \psi^4$ et $\psi_i'^3 = 0^{16}$ lorsque $i < 6$. On peut noter t_2^{-1} une telle transformation, soit $\psi'^4 = t_2^{-1}(\psi^4)$.

Le *Transformer* se traite avec la méthode classique soit $\psi^3 = \varphi(h_3, f_2(x))(\psi'^4 \oslash \Delta f_3(x)) \in \mathbb{R}^{6\times16}$.

Pour l'encodeur de position, puisque $h_2 = Id + PE$, les valeurs SHAP de $f_1'(x)$ hérite directement des valeurs SHAP de $f_2(x)$ car PE est indépendant de l'entrée. On considère donc que $\psi_2 = \psi_3$. Les encodages de position fixes ne contribuent pas aux variations par rapport à une référence fixe.

On traite la fonction de transformation en jeton de classe t_1 d'une manière analogue à t_2 en construisant une fonction t_1^{-1} telle que $\psi'^2 = t_1^{-1}(\psi^2) \in \mathbb{R}^{5\times16}$ avec $\psi_i'^2 = \psi_i^2, i \in \{1, ..., 5\}$.

Enfin, on traite la couche de vectorisation par la méthode de récurrence classique, soit : $\psi^1 = \varphi(h_1, x)(\psi'^2 \oslash \Delta f_1(x)) \in \mathbb{R}^{5\times17}$. ψ^1 décrit exactement les contributions de chaque caractéristique de x à la prédiction finale $f(x)$.

Dans tout ce qui précède, recensons les valeurs connues et celles qu'il nous faudra calculer. Nous connaissons, grâce au modèle, tous les $f_i(x)$ et donc tous les $\Delta f_i(x)$. Par ailleurs, nous connaissons les fonctions $t_1^{-1}, t_2^{-1}, PE^{-1}$ que nous avons construites. De plus, nous avons accès à tous les poids des modèles intermédiaires qui ont été entraîné. Finalement, il nous reste à

calculer les valeurs SHAP suivantes : $\begin{cases} \varphi(h_5, f_4(x)) \\ \varphi(h_4, f'_3(x)) \\ \varphi(h_3, f_2(x)) \\ \varphi(h_1, x) \end{cases}$

Dans ce qui suit, la notation $\odot$ correspond à la multiplication matricielle terme à terme.

Calcul de $\varphi(h_5, f_4(x))$: On cherche ici à étudier la tête d'anomalie responsable de la classification en fin de modèle. Cette tête est implémentée par un neurone linéaire avec une fonction d'activation en *sigmoïd* schématisé dans la Figure 7.6. Les équations décrivant le neurone sont :

$$z = \sum_{i=1}^{16} w_i f_4(x_i) + w_0 \tag{7.7}$$

$$z = w(f_4(x_i)), \text{on note } w \text{ la transformation} \tag{7.8}$$

$$f(x) = sig(z) \tag{7.9}$$

$$sig(z) = \frac{1}{1 + e^{-z}} \tag{7.10}$$

Où w est le vecteur de poids du modèle. En utilisant les règles issues de DeepSHAP, on calcule les valeurs SHAP comme :

$$\varphi(sig, z) = \Delta sig(z) = sig(z) - sig(z^b) \in \mathbb{R}, \text{traitement linéaire de la fonction d'activation.} \tag{7.11}$$

$$\varphi(w, f_4(x)) = w \odot \Delta f_4(x) \in \mathbb{R}^{16}, \text{valeurs SHAP d'un neurone linéaire} \tag{7.12}$$

$$\varphi(h_4, f_4(x)) = m_{\Delta f_4 \Delta f_5} \odot \Delta f_4(x), \text{par définition des multiplicateurs} \tag{7.13}$$

$$= m_{\Delta f_4 \Delta z} \odot m_{\Delta z \Delta f_5} \odot \Delta f_4(x), \text{règle de chaînage} \tag{7.14}$$

$$= (\varphi(w, f_4) \oslash \Delta f_4(x)) \odot \frac{\Delta sig}{\Delta z} \Delta f_4(x) \tag{7.15}$$

$$= w \oslash \Delta f_4(x) \frac{\Delta sig}{\Delta z} \in \mathbb{R}^{16} \tag{7.16}$$

Les divisions présentes ici sont des divisions d'Hadamard, s'assurant de ne jamais diviser par 0 en cas d'égalité entre la référence et l'instance étudiée.

Calcul de $\varphi(h_4, f'_3(x))$: La couche dense est également décrite par un neurone linéaire mais dont la sortie est en dimension 16, donc la matrice de poids sera en deux dimensions

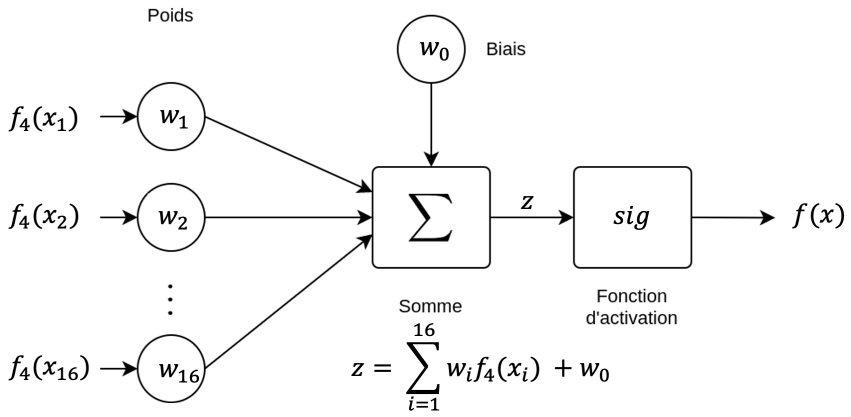


FIGURE 7.6 Architecture de la tête d'anomalie

(16x16). La Figure 7.7 schématise cette couche. Les équations sont les suivantes :

$$z = W^T f'_3(x) + b \tag{7.17}$$

$$z = W(f'_3(x)), \text{on note } W \text{ la transformation} \tag{7.18}$$

$$f_4(x) = LeakyReLU(z) \tag{7.19}$$

$$LeakyReLU(z) = max(0, z) + \alpha * min(0, z) \tag{7.20}$$

Comme précédemment :

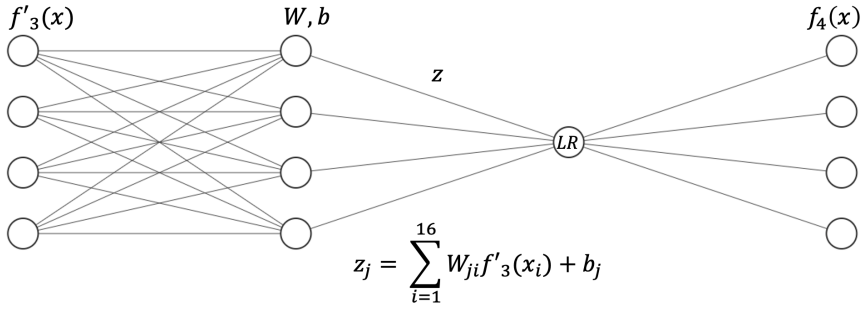


FIGURE 7.7 Architecture de la couche dense

$$\varphi(LR, z) = \Delta LR(z) = LR(z) - LR(z^b) = \Delta f_4(x) \in \mathbb{R}^{16}, \text{traitement linéaire} \tag{7.21}$$

$$\varphi(W, f_3')_{ij} = W_{ji}\Delta f_3'(x)_i \in \mathbb{R} \tag{7.22}$$

$$\varphi(W, f_3') = W^T \odot (\Delta f_3'(x) \cdot 1^T) \in \mathbb{R}^{16\times 16}, \text{valeurs SHAP d'un neurone linéaire} \tag{7.23}$$

$$\varphi(h_4, f_3'(x)) = m_{\Delta f_3' \Delta f_4} \odot (\Delta f_3'(x) \cdot 1^T) \tag{7.24}$$

$$= m_{\Delta f_3' \Delta z} m_{\Delta z \Delta f_4} \odot (\Delta f_3'(x) \cdot 1^T), \text{règle de chaînage} \tag{7.25}$$

$$= (\varphi(W, f_3') \oslash (\Delta f_3'(x) \cdot 1^T))(\varphi(LR, z) \oslash \Delta z) \odot (\Delta f_3'(x) \cdot 1^T) \tag{7.26}$$

$$= W^T \odot (\Delta f_3'(x) \cdot 1^T)(\Delta f_4(x) \oslash \Delta z) \in \mathbb{R}^{16\times 16} \tag{7.27}$$

Calcul de $\varphi(h_3, f_2(x))$: L'encodeur *Transformer* est plus complexe. Commençons alors simplement par utiliser la librairie Python des auteurs :

```
1 Transformer_explainer = shap.DeepExplainer(self.Transformer_encoder, f3_b)
2 Transformer_shap_values = Transformer_explainer.shap_values(f3)
```

La commande ci-dessus dépasse de loin la limite de mémoire que le serveur est capable d'allouer. Quand bien même on utiliserait la méthode KernelSHAP (ou même GradientSHAP, alternative aux méthodes existantes) qui utilise d'autres méthodes d'optimisation, on dépasse le seuil de mémoire disponible. Bien que ces deux méthodes soient optimisées, voyons quel calcul elles essaient de calculer. Le calcul de $\varphi(h_3, f_2(x))$ cherche les contributions de chaque caractéristique de $f_2(x)$ à $f_3(x)$ soit $\varphi(h_3, f_2(x)) \in \mathbb{R}^{6\times 16\times 6\times 16}$. C'est immense comme vecteur à calculer, bien au-delà de ce qu'il est envisageable pour nos méthodes. Cependant nous pouvons utiliser une caractéristique des *Transformers* pour gagner un temps considérable, il s'agira d'une approximation, mais elle aura le mérite de réussir à franchir cette étape. Pour cela, utilisons le fait que les poids issus des couches d'attention contiennent par essence les contributions relatives de chaque jeton d'entrée. Autrement dit, chaque jeton $i \in \{1, 6\}$, inséré dans le *Transformer* passe plusieurs têtes d'attention qui lui donnent chacune un score d'attention $\alpha_{i,classe}$, reflétant sa contribution à l'élément de classe de sortie $\psi_{classe}'^4$. Pour simplifier, nous agrégeons les poids d'attention $\alpha_{i,classe}$ en calculant leur moyenne sur toutes les têtes et couches du *Transformer*, puis en normalisant ces valeurs pour qu'elles somment à 1. Ainsi, on transfère directement les contributions permettant de ne pas rentrer dans l'analyse caractéristique par caractéristique, mais de rester au niveau des jetons, soit de rester en dimensions $\mathbb{R}^{6\times 6}$. Puisque l'on effectue un élagage sur les 5 autres dimensions qui ne sont pas des dimensions de classe, on peut s'affranchir des contributions de f_2 à tous ces éléments, réduisant encore la dimension de $\varphi(h_3, f_2(x))$ à $\mathbb{R}^6$. Ainsi, avec $\alpha_{i,j} \in \mathbb{R}$ et $\psi_{classe}'^4 \in \mathbb{R}^{16}$, on

a :

$$\forall i \in \{1,6\}, \varphi(h_3, f_2(x))_i = \sum_{couchesl} \sum_{têtesk} \alpha^{l,k}_{i,classe} \tag{7.28}$$

$$\psi^3 = ((\sum_{couchesl} \sum_{têtesk} \alpha^{l,k}_{classe}) \cdot \psi'^4_{classe}) \oslash \Delta f_3(x), \in \mathbb{R}^{6\times 16} \tag{7.29}$$

Le jeton de classe résume l'information du message via l'attention. Sa valeur SHAP ψ'^4_{classe} est ainsi redistribuée aux jetons d'entrée proportionnellement aux $\alpha_{i,classe}$. Dans cette formulation, nous ne considérons donc pas les couches *feed_forward* qui font partie intégrante du mécanisme de *Transformer*. Un travail futur sera donc de raffiner cette définition pour les considérer.

Calcul de $\varphi(h_1, x)$: Cette couche est similaire à la couche dense considérée plus haut. L'entrée de la couche est $x \in \mathbb{R}^{5\times 17}$, la matrice de poids est $W_e \in \mathbb{R}^{17\times 16}$ avec un biais $b_e \in \mathbb{R}^{16}$. La sortie vaut ainsi $f_1(x) \in \mathbb{R}^{5\times 16}$ telle que $f_1(x) = xW_e + b_e$. Il s'agit d'un cas simple déjà étudié menant à :

$$\varphi(h_1, x) = (1^T \cdot W_e) \odot (\Delta x \cdot 1^T) \tag{7.30}$$

7.0.6 Bilan

Cette étude nous a permis de mettre réellement en pratique nos connaissances mathématiques de SHAP, pour débloquer une situation dans laquelle l'utilisation classique de la librairie ne fonctionne pas. Il s'agit ici d'une tentative, à laquelle nous avons ajouté un certain nombre d'approximations et dont le résultat final pratique ne garantit rien pour le moment. Une des approximations a été de considérer que les poids d'attention étaient directement convertibles en valeurs SHAP, ignorant ainsi les couches *feed_forward* [67]. Cette motivation s'inspire notamment d'une approximation similaire faite par les auteurs de SHAP, dans le traitements des fonctions d'activation non-linéaires. Contrairement à LIME, qui repose sur des perturbations locales, notre méthode exploite la structure intrinsèque du *Transformer*, évitant ainsi des calculs coûteux. Bien que les poids d'attention fournissent une heuristique utile, des travaux récents montrent qu'ils peuvent être découplés de l'importance réelle des caractéristiques [68]. Notre approximation doit donc être validée empiriquement via des tests.

Cette étude a néanmoins mis en lumière que l'établissement d'un cadre mathématique permet d'affronter des gros modèles de manière armée. Un travail futur sera de raffiner cette étude en l'implémentant au code original pour obtenir des premiers jets d'explication, jusqu'à converger vers des explications convenables de la situation.

Si les deux premières applications démontraient la faisabilité de l'XAI en détection d'intrusions,

celle-ci étend la méthodologie à des modèles complexes, ouvrant la voie à une explicabilité adaptée même pour les *Transformer*.

CHAPITRE 8 RÉSULTATS, LIMITATIONS, DISCUSSIONS ET PERSPECTIVES D'AMÉLIORATIONS

8.1 Résultats des études et discussions

Dans cette partie, revenons sur les résultats des applications ADS-B et NSL-KDD. Nous n'allons pas analyser à nouveau tous les résultats issus de nos deux études. L'organisation de ce mémoire est telle que chaque résultat ainsi que son interprétation a été présenté dans le contexte de son application respective. Nous présentons ici un récapitulatif sous forme de tableau 8.1 de l'ensemble des résultats obtenus. Le tableau se décompose en deux catégories : d'une part les métriques d'évaluation de l'IDS, d'autre part celles des explications.

Les métriques retenues pour évaluer la performance des IDS, n'ont jusqu'alors pas été présentées, et sont les suivantes :

— Exactitude (*Accuracy*) : Taux de prédictions correctes (toutes classes confondues).
— Précision : Proportion des vraies attaques parmi les instances détectées comme malignes.
— Rappel (*Recall*) : Taux de détection des attaques réelles.
— Temps d'inférence : Temps de traitement par échantillon (en ms).
— FPR (*False Positive Rate*) : Taux de fausses alarmes (signaux normaux classés comme attaques), impactant la charge cognitive des opérateurs.

Les métriques pour évaluer les explications sont celles présentées en 4.5 et dont les résultats ont déjà été présentés pour l'application ADS-B **??**. Le tableau ajoute à ces résultats ceux de l'application NSL-KDD.

Les systèmes de détection d'intrusion (IDS) développés dans cette étude démontrent de très bonnes performances à plusieurs niveaux. Concernant les métriques opérationnelles, les deux modèles atteignent des résultats comparables aux références actuelles de la littérature [18] [**?**] [16], avec :

— Une exactitude (*Accuracy*) et une précision dépassant systématiquement 99%.
— Un rappel (Recall) supérieur à 98%.
— Un taux de faux positifs (FPR) inférieur à 0,1%.
— Des temps d'inférence compatibles avec des contraintes temps réel (<100ms).

Ces performances valident non seulement la robustesse de notre approche, mais aussi l'efficacité des phases préliminaires de collecte et de prétraitement des données. Bien que ces étapes ne

Tableau 8.1 Métriques d'évaluation pour les cas d'usage NSL-KDD et ADS-B

(a) Métriques de performance de l'IDS

Cas d'usage	**Exactitude**	**Précision**	**Rappel**	**FPR**	**Temps (ms)**
NSL-KDD	99.09%	99.08%	99.10%	0.78%	0.21 ms
ADS-B	99.88%	99.86%	98.92%	0.01%	0.14 ms

(b) Métriques d'explication

Cas d'usage	**Stabilité**	**Précision**	**Solidité**	**Densité**	**Efficacité**
NSL-KDD	100%	0.27	99.96%	92.7%	22 ms/instance
ADS-B	100%	0.88	99.3%	74.8%	0.88 ms/instance

constituent pas en elles-mêmes une contribution novatrice, elles ont été cruciales pour aboutir à des modèles performants.

L'évaluation des mécanismes d'explication révèle des comportements distincts entre nos deux applications :

1. *Modèle ADS-B :*
   - Attribution SHAP forte aux caractéristiques importantes (baisse de précision de 0,88 après suppression des 3 principales caractéristiques)
   - Densité modérée autour de zéro (74,8%), indiquant une prise en compte large des caractéristiques
   - Comportement "inclusif" privilégiant la sensibilité au détriment de la sélectivité.
2. *Modèle NSL-KDD :*
   - Densité élevée autour de zéro (92,7%), témoignant d'une sélection stricte
   - Précision limitée (0,27) révélant une attribution trop conservatrice des importances
   - Approche "parcimonieuse" pouvant négliger certaines caractéristiques pertinentes

En termes d'efficacité, le cas NSL-KDD est bien moins performant que le cas ADS-B. Cela s'explique par deux raisons. La première est la quantité de calcul à effectuer : dans le cas NSL-KDD il y a 122 caractéristiques et 5 classes à calculer soit près de 600 valeurs SHAP par instance là où il n'y en a que 50 pour ADS-B. La deuxième raison est que l'on a utilisé DeepSHAP pour ADS-B face à KernelSHAP pour NSL-KDD, illustrant encore la performance de DeepSHAP en termes de rapidité. Ce choix a été motivé par la volonté de comparer en pratique les deux théories étudiées en 3.

Les deux systèmes présentent par ailleurs une excellente stabilité théorique, inhérente aux propriétés formelles de SHAP, ainsi qu'une robustesse avérée aux entrées bruitées. Plus

fondamentalement, cette étude démontre la capacité d'un cadre XAI unifié à :

— S'adapter à des protocoles variés (ADS-B et NSL-KDD)

— Maintenir des performances temps-réel

— Fournir des explications interprétables malgré des contraintes opérationnelles fortes

Ces résultats ouvrent des perspectives intéressantes pour l'optimisation conjointe des métriques de performance et d'explicabilité dans les systèmes critiques.

8.2 Discussions des objectifs de recherche, limitations et perspectives d'amélioration

Au terme de nos applications, revenons sur les objectifs de recherche que je me suis fixé et voyons dans quelle mesure ceux-ci ont été atteints.

1 - Définir un cadre conceptuel pour l'XAI, structurant les concepts clés autour du principe des confiance et de déploiement en contexte avionique.

Cet objectif a nécessité une revue de littérature interdisciplinaire (IA éthique, ingénierie systèmes critiques...) pour identifier 11 concepts clés, validés par la communauté scientifique d'une part et adaptés à ma problématique d'autre part. Pour retenir un concept dans mon cadre, celui-ci devait être à la fois nécessaire à l'augmentation de la confiance en l'IA, et tout de même suffisamment indépendant des autres pour ne pas être considéré comme un sous-concept. Ce procédé de sélection a abouti au 8 concepts retenus jugés nécessaires pour obtenir la confiance en un modèle. Nous avons ajouté à cela 3 concepts, particulièrement pertinents dans un contexte avionique. Le choix d'une représentation visuelle 3.1 reflète l'équilibre requis dans ce domaine : aucun concept ne prime sur un autre, car une défaillance de l'un (ex : robustesse) compromettrait l'ensemble du système. La centralité de la "confiance en l'IA" et des "spécificités avioniques" n'est pas hiérarchique mais relationnelle : chaque concept y contribue via des mécanismes distincts (ex : la responsabilité par la traçabilité des décisions, l'intelligibilité par la clarté des sorties algorithmiques). Des concepts comme l'explicabilité, l'interprétabilité ou la transparence sont des notions phares que l'on retrouvera dans pratiquement tous les travaux d'XAI. Cependant d'autres concepts présents dans notre cadre sont plus propres aux infrastructures critiques. La responsabilité, la stabilité, l'efficacité ou encore la satisfaction permettent chacun de mettre l'accent sur une volonté de venir en aide aux travailleurs avioniques, sans les surcharger tout en leur promettant une solution robuste, rapide et viable. Cet objectif est donc rempli et nous permet de conclure que l'XAI est une discipline qui trouve entièrement sa place dans le domaine avionique prenant en considération

les contraintes lourdes imposées par cette industrie. Les validations empiriques, rendues possibles par ce cadre, constitueront une étape essentielle pour quantifier son impact opérationnel.

Limitations et travaux futurs : Ce cadre révèle toutefois une focalisation inégale dans son exploitation. En choisissant SHAP, les concepts d'interprétabilité, d'explicabilité et de transparence ont été au cœur de l'étude. En ajoutant des métriques d'évaluation, on s'est assuré que notre solution prenait en considération les concepts de précision, stabilité, efficacité et robustesse. L'intelligibilité a été prise en compte dans la partie interprétation des résultats. En revanche, les concepts de satisfaction des utilisateurs, d'éthique et de responsabilité, bien qu'identifiés comme essentiels, n'ont pas été explorés empiriquement dans cette recherche. Cette limitation est motivée par deux raisons principales : d'une part, ces facteurs humains nécessitent des études approfondies et des analyses spécifiques qui dépassent le cadre de ce travail ; d'autre part, ces sujets sont actuellement traités par d'autres étudiants au sein du laboratoire, ce qui les positionne comme *out of scope* pour notre étude. Nous avons ainsi priorisé la validation méthodologique avant son évaluation complète. Les travaux de [49] soulignent néanmoins que leur intégration renforce la validité globale des systèmes XAI, ce qui en fait des axes prioritaires pour des recherches futures. Pour préparer ces travaux, j'ai développé un *plug-in* pour le simulateur de vol de Polytechnique, permettant d'accéder aux analyses SHAP d'un IDS utilisant les caractéristiques de vol (altitude, vitesse...) en temps réel. Cet outil ouvre la voie à des tests utilisateurs en conditions réelles, ciblant notamment l'intelligibilité et la satisfaction. D'autres études pourront ensuite s'appuyer sur cette base pour explorer les contraintes de responsabilité et d'éthique, qui restent des défis majeurs mais distincts de nos objectifs initiaux.

2 - Sélectionner des méthodes d'XAI adaptées aux contraintes avioniques, puis établir un formalisme mathématique unifié pour standardiser leur utilisation.

J'ai commencé par utiliser une taxonomie existante [48] pour recenser les catégories de techniques d'XAI existantes dans la littérature. J'ai donc fait le choix d'approfondir les techniques agnostiques, mieux adaptées au contexte avionique dans lequel nous voulons les appliquer. En effet, il n'existe pas un unique type de modèle utilisé pour les IDS, ceux-ci sont sans cesse améliorés et leurs structures changent régulièrement, d'où l'importance d'avoir recours à des techniques d'XAI agnostiques, avec une meilleure adaptabilité. J'ai ensuite décidé de me concentrer sur SHAP, la méthode d'XAI probablement la plus utilisée dans la littérature, s'appuyant sur des fondations théoriques solides : les valeurs de Shapley, LIME ou encore DeepLIFT. La motivation derrière ce choix est la solidité mathématique derrière ces méthodes permettant des garanties de stabilité et de reproductibilité des résultats. La réunification de

ces techniques sous un unique formalisme mathématique s'est avérée extrêmement fructueuse, j'ai exploré un univers mathématique bien plus grand qu'escompté initialement. La prise en main du formalisme a surtout porté ses fruits lors des études de cas, dans lesquels le fonctionnement des techniques n'a plus été ni un enjeu ni une source de doute. Le chapitre consacré à l'extension de SHAP, à des modèles de *Transformer* pour le protocole MIL-STD-1553 7, constitue l'aboutissement de mon travail mathématique. L'édification d'un socle mathématique a permis d'utiliser les notations SHAP pour obtenir des résultats théoriques satisfaisants.

Limitations et travaux futurs : L'application de SHAP à un modèle incluant un *Transformer* se heurte à une complexité algorithmique importante. Notre approche a permis de contourner cette limite en proposant des approximations linéaires des fonctions d'activation et en adaptant les poids d'attention pour propager les valeurs SHAP à travers l'architecture. Cependant, ces approximations restent simplificatrices : les couches *feed_forward*, pourtant déterminantes dans le mécanisme de contribution, n'ont pas été intégrées dans l'analyse, et l'agrégation des têtes d'attention masque certaines contributions intermédiaires. Une implémentation plus robuste, tenant compte de ces aspects, serait nécessaire pour évaluer pleinement la cohérence des résultats. Ces limitations ouvrent des perspectives claires pour des travaux futurs. L'objectif principal serait d'étendre la bibliothèque SHAP (Python) en y intégrant nos développements théoriques, afin de la rendre plus adaptable à des architectures variées, y compris les *Transformers*. Une telle flexibilité est essentielle dans le domaine aéronautique, où les modèles doivent être à la fois interprétables et généralisables. Cette extension pourrait notamment améliorer la prise en compte des couches *feed_forward* et préserver l'information des contributions intermédiaires dans les mécanismes d'attention, renforçant ainsi la pertinence des explications générées.

3 - Établir une méthodologie d'intégration de l'XAI aux IDS avioniques, puis évaluer son applicabilité via deux cas d'usage concrets.

J'ai développé une méthodologie protocolaire pour intégrer l'XAI dans le processus de détection d'intrusions pour la cybersécurité des communications. La méthodologie consiste à ajouter à l'IDS une couche d'explicabilité utilisant SHAP, fournissant des graphes d'explicabilité. Cette couche se décompose en 3 analyses (globale, locale, spécifique). A la suite de cette couche, notre méthodologie intègre une méthode d'évaluation pour vérifier la viabilité des explications générées, dans le contexte avionique. Cette évaluation intègre des concepts et sous-concepts de notre cadre comme la robustesse, la sélectivité ou la complexité, non exploités directement par SHAP.

Cette méthodologie a été appliquée à deux scénarios : le jeu de données NSL-KDD (générique) et une attaque par injection ADS-B (spécifique). Notre méthode a fourni des résultats probants notamment dans sa capacité à fournir des explications très cohérentes avec les menaces concrètes existantes dans le domaine de la sécurité des communications. De plus, elle a prouvé sa robustesse en s'appliquant à un modèle de classification multiple et plus seulement binaire et en s'illustrant dans deux scénarios très différents.

Limitations et travaux futurs : Si les garanties théoriques de SHAP sont mathématiquement solides, leur traduction pratique révèle une subtilité importante : bien que les résultats soient techniquement corrects, leur interprétation demeure contextuelle et nécessite un savoir-faire métier. Cette observation est cruciale dans le domaine avionique, où ces explications doivent ultimement guider les décisions des opérateurs. En pratique, SHAP ne se distingue pas fondamentalement d'autres méthodes probabilistes sur ce point – sa valeur ajoutée réside davantage dans sa granularité que dans une objectivité absolue.

Cette constatation rejoint les enjeux identifiés dans notre premier objectif : une explication mathématiquement rigoureuse ne suffit pas à elle seule à instaurer la confiance si sa représentation n'est pas intuitivement accessible. Ainsi, **les résultats SHAP ne sont pas directement exploitables par tout utilisateur**. Nos analyses l'ont montré : l'interprétation des graphes SHAP a requis un double accompagnement – à la fois en sécurité des systèmes et en expertise avionique. Il serait irréaliste d'intégrer ces visualisations brutes dans un cockpit sans filtre supplémentaire. Ces outils s'adressent avant tout à des spécialistes capables de les contextualiser.

Cette orientation ouvre la voie à des travaux futurs passionnants, notamment :

— Le développement d'une postanalyse intelligente des graphes SHAP, potentiellement via un LLM spécialisé formé sur le domaine avionique.

— L'enrichissement du *plug*-in pour automatiser la conversion des données SHAP en recommandations priorisées et contextualisées.

— Des tests utilisateurs en environnement simulé pour valider l'efficacité cognitive de ces adaptations.

Le simulateur servira de plateforme idéale pour explorer ces pistes, transformant une approche théoriquement robuste en un outil opérationnellement pertinent.

CHAPITRE 9 CONCLUSION

Ce mémoire a exploré l'intégration de méthodes d'XAI, et plus particulièrement de SHAP, dans les systèmes de détection d'intrusions (IDS) appliqués à la cybersécurité avionique. Nos travaux ont permis d'atteindre trois objectifs principaux : la définition d'un cadre conceptuel visuel structurant les concepts clés de l'XAI en contexte avionique, l'établissement d'un formalisme mathématique unifié pour SHAP, et la proposition d'une méthodologie d'intégration de l'XAI aux IDS, évaluée sur deux cas d'usage concrets (ADS-B et NSL-KDD).

Les résultats obtenus démontrent que notre approche permet de concilier performance opérationnelle et explicabilité. Les modèles d'IDS développés atteignent des métriques compétitives (exactitude $> 99\%$, FPR $<0{,}1\%$, temps d'inférence <1 ms), tout en fournissant des explications stables et robustes grâce à SHAP. Cependant, l'analyse approfondie des sorties explicatives révèle une limite majeure : bien que mathématiquement rigoureuses, ces explications restent difficilement interprétables sans expertise en analyse de données. Cette observation souligne la nécessité d'une couche supplémentaire de traduction entre les sorties brutes de SHAP et les besoins opérationnels des utilisateurs finaux (pilotes, contrôleurs aériens, etc...).

Les perspectives identifiées ouvrent plusieurs voies de recherche prometteuses. D'une part, l'amélioration des mécanismes d'explication pour les architectures complexes (comme les *Transformers*) nécessite des travaux théoriques approfondis. D'autre part, l'évaluation des concepts négligés dans cette étude (satisfaction utilisateur, éthique, responsabilité) via des tests en conditions réelles (simulateur de vol) constituera une étape clé pour valider l'adoption opérationnelle de ces systèmes. Enfin, le développement d'interfaces adaptées, convertissant les explications techniques en alertes actionnables, apparaît comme un prérequis indispensable à un déploiement à grande échelle.

En conclusion, ce travail confirme la pertinence de l'XAI pour renforcer la confiance dans les IDS avioniques, tout en identifiant des défis pratiques cruciaux. Les résultats obtenus posent les bases d'une intégration progressive de ces méthodes dans les systèmes critiques, où l'équilibre entre performance, transparence et simplicité d'usage reste un enjeu central.

RÉFÉRENCES

[1] Y. Xie, N. Pongsakornsathien, A. Gardi et R. Sabatini, "Explanation of machine-learning solutions in air-traffic management," *Aerospace*, vol. 8, p. 224, 08 2021.

[2] J. Persinos, "The nexus of avionics, artificial intelligence, and aircraft values," Rapport technique, 2024. [En ligne]. Disponible : https://www.aviationtoday.com/2024/10/02/the-nexus-of-avionics-artificial-intelligence-and-aircraft-values

[3] M. R. Manesh et N. Kaabouch, "Analysis of vulnerabilities, attacks, countermeasures and overall risk of the automatic dependent surveillance-broadcast (ads-b) system," *International Journal of Critical Infrastructure Protection*, vol. 19, 10 2017.

[4] F. Onodueze et D. Josyula, "Anomaly detection on mil-std-1553 dataset using machine learning algorithms," dans *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2020, p. 592–598.

[5] "Sita 2024 air transport it insights," SITA, Rapport technique, 2024. [En ligne]. Disponible : https://www.sita.aero/globalassets/docs/surveys--reports/2024-air-transport-it-insights.pdf

[6] "Transparent artificial intelligence and automation to air traffic management systems," Cordis, Rapport technique, 2022. [En ligne]. Disponible : https://cordis.europa.eu/article/id/442199-building-trust-in-air-traffic-management-ai/fr

[7] "Single pilot decision making," EASA, Rapport technique, 2012. [En ligne]. Disponible : https://www.easa.europa.eu/sites/default/files/dfu/HE4_Single-Pilot-Decision-Making-v1.pdf

[8] A. Degas, M. Islam, C. Hurter, S. Barua, H. Rahman, M. Poudel, D. Ruscio, M. Ahmed, S. Begum, M. Rahman, G. Cartocci, G. Di Flumeri, G. Borghini, F. Babiloni et P. Aricò, "A survey on artificial intelligence (ai) and explainable ai in air traffic management : Current trends and development with future research trajectory," *Applied Sciences*, vol. Computing and Artificial Intelligence, 01 2022.

[9] "Explainable artificial intelligence (xai) for air traffic management," Tech Port, Rapport technique, 2024. [En ligne]. Disponible : https://techport.nasa.gov/projects/154535

[10] J.-S. Marrocco, "Tript-ids : Triplet loss pre-trained transformer for avionic intrusion detection system," Mémoire de maîtrise, Polytechnique Montréal, décembre 2023. [En ligne]. Disponible : https://publications.polymtl.ca/56994/

[11] S. M. Lundberg et S.-I. Lee, "A unified approach to interpreting model predictions," dans *Proceedings of the 31st International Conference on Neural Information Processing*

Systems, ser. NIPS'17. Red Hook, NY, USA : Curran Associates Inc., 2017, p. 4768–4777.

[12] M. T. Ribeiro, S. Singh et C. Guestrin, ""why should i trust you ?" : Explaining the predictions of any classifier," dans *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA : Association for Computing Machinery, 2016, p. 1135–1144. [En ligne]. Disponible : https://doi.org/10.1145/2939672.2939778

[13] J. Orasanu et L. Martin, "Hessd '98 100 errors in aviation decision making : A factor in accidents and incidents," 12 2008.

[14] A. B. Garcia, R. F. Babiceanu et R. Seker, "Artificial intelligence and machine learning approaches for aviation cybersecurity : An overview," dans *2021 Integrated Communications Navigation and Surveillance Conference (ICNS)*, 2021, p. 1–8.

[15] D. McCallie, J. Butts et R. Mills, "Security analysis of the ads-b implementation in the next generation air transportation system," *International Journal of Critical Infrastructure Protection*, vol. 4, p. 78–87, 08 2011.

[16] X. Ying, J. Mazer, G. Bernieri, M. Conti, L. Bushnell et R. Poovendran, "Detecting ads-b spoofing attacks using deep neural networks," dans *2019 IEEE Conference on Communications and Network Security (CNS)*, 2019, p. 187–195.

[17] R. Karam, M. Salomon et R. Couturier, "Supervised ads-b anomaly detection using a false data generator," dans *2022 2nd International Conference on Computer, Control and Robotics (ICCCR)*, 2022, p. 218–223.

[18] A. Fried et M. Last, "Facing airborne attacks on ads-b data with autoencoders," *Computers Security*, vol. 109, p. 102405, 07 2021.

[19] N. G. Markevich et A. Wool, "Hardware fingerprinting for the arinc 429 avionic bus," 2020. [En ligne]. Disponible : https://arxiv.org/abs/2003.12456

[20] M. Wolf, M. Minzlaff et M. Moser, "Information technology security threats to modern e-enabled aircraft : A cautionary note," *J. Aerosp. Inf. Syst.*, vol. 11, p. 447–457, 2014. [En ligne]. Disponible : https://api.semanticscholar.org/CorpusID:2609971

[21] C.-H. Cheng, F. Diehl, G. Hinz, Y. Hamza, G. Nuehrenberg, M. Rickert, H. Ruess et M. Truong-Le, "Neural networks for safety-critical applications — challenges, experiments and perspectives," dans *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2018, p. 1005–1006.

[22] F. Tambon, G. Laberge, L. An, A. Nikanjam, P. S. N. Mindom, Y. Pequignot, F. Khomh, G. Antoniol, E. Merlo et F. Laviolette, "How to certify machine learning based safety-critical systems ? a systematic literature review," *Automated Software Engineering*, vol. 29, n°. 2, avr. 2022. [En ligne]. Disponible : http://dx.doi.org/10.1007/s10515-022-00337-x

[23] C. S. Hernandez, S. Ayo et D. Panagiotakopoulos, “An explainable artificial intelligence (xai) framework for improving trust in automated atm tools,” dans *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, 2021, p. 1–10.

[24] G. Yadam, A. K. Moharir et I. Srivastava, “Explainable and visually interpretable machine learning for flight sciences,” dans *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2020, p. 1–6.

[25] R. Dalmau-Codina, F. Ballerini, H. Naessens, S. Belkoura et S. Wangnick, “An explainable machine learning approach to improve take-off time predictions,” *Journal of Air Transport Management*, vol. 95, p. 102090, 08 2021.

[26] K. Nor, S. R. Pedapati et M. Muhammad, “Application of explainable ai (xai) for anomaly detection and prognostic of gas turbines with uncertainty quantification.” 09 2021.

[27] S. ten Zeldam, “Automated failure diagnosis in aviation maintenance using explainable artificial intelligence (xai),” July 2018. [En ligne]. Disponible : http://essay.utwente.nl/75381/

[28] S. Sutthithatip, S. Perinpanayagam, S. Aslam et A. Wileman, “Explainable ai in aerospace for enhanced system performance,” dans *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, 2021, p. 1–7.

[29] S. Sutthithatip, S. Perinpanayagam et S. Aslam, “(explainable) artificial intelligence in aerospace safety-critical systems,” dans *2022 IEEE Aerospace Conference (AERO)*, 2022, p. 1–12.

[30] L. He, A. Nabil et B. Song, “Explainable deep reinforcement learning for uav autonomous navigation,” 2021. [En ligne]. Disponible : https://arxiv.org/abs/2009.14551

[31] L. Viganò et D. Magazzeni, “Explainable security,” 2018. [En ligne]. Disponible : https://arxiv.org/abs/1807.04178

[32] G. Srivastava, R. H. Jhaveri, S. Bhattacharya, S. Pandya, Rajeswari, P. K. R. Maddikunta, G. Yenduri, J. G. Hall, M. Alazab et T. R. Gadekallu, “Xai for cybersecurity : State of the art, challenges, open issues and future directions,” 2022. [En ligne]. Disponible : https://arxiv.org/abs/2206.03585

[33] W. Guo, “Explainable artificial intelligence for 6g : Improving trust between human and machine,” *IEEE Communications Magazine*, vol. 58, n°. 6, p. 39–45, 2020.

[34] S. Mane et D. Rao, “Explaining network intrusion detection system using explainable ai framework,” 2021. [En ligne]. Disponible : https://arxiv.org/abs/2103.07110

[35] S. Patil, V. Varadarajan, S. Mazhar, A. Sahibzada, N. Ahmed, O. Sinha, S. Kumar V C, K. Shaw et K. Kotecha, “Explainable artificial intelligence for intrusion detection system,” *Electronics*, vol. 11, p. 3079, 09 2022.

[36] C. I. Nwakanma, L. A. C. Ahakonye, T. Jun, J. M. Lee et D.-S. Kim, “Explainable scada-edge network intrusion detection system : Tree-lime approach,” dans *2023 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 2023, p. 1–7.

[37] R. Alenezi et S. A. Ludwig, “Explainability of cybersecurity threats data using shap,” dans *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2021, p. 01–10.

[38] A. Kuppa et N.-A. Le-Khac, “Black box attacks on explainable artificial intelligence(xai) methods in cyber security,” dans *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, p. 1–8.

[39] ——, “Adversarial xai methods in cybersecurity,” *IEEE Transactions on Information Forensics and Security*, vol. PP, p. 1–1, 10 2021.

[40] D. Slack, S. Hilgard, E. Jia, S. Singh et H. Lakkaraju, “Fooling lime and shap : Adversarial attacks on post hoc explanation methods,” 2020. [En ligne]. Disponible : https://arxiv.org/abs/1911.02508

[41] A. Warnecke, D. Arp, C. Wressnegger et K. Rieck, “Evaluating explanation methods for deep learning in security,” 2020. [En ligne]. Disponible : https://arxiv.org/abs/1906.02108

[42] R. Cugny, J. Aligon, M. Chevalier, G. Roman Jimenez et O. Teste, “AutoXAI : Un cadre pour sélectionner automatiquement la solution d’XAI la plus adaptée,” dans *Revue des Nouvelles Technologies de l’Information*, vol. RNTI-E-39, Lyon, France, janv. 2023, p. 491–498. [En ligne]. Disponible : https://hal.science/hal-04258658

[43] P. Lopes, E. Silva, C. Braga, T. Oliveira et L. Rosado, “Xai systems evaluation : A review of human and computer-centred methods,” *Applied Sciences*, vol. 12, n°. 19, 2022. [En ligne]. Disponible : https://www.mdpi.com/2076-3417/12/19/9423

[44] O. Arreche, T. R. Guntur, J. W. Roberts et M. Abdallah, “E-xai : Evaluating black-box explainable ai frameworks for network intrusion detection,” *IEEE Access*, vol. 12, p. 23 954–23 988, 2024.

[45] S. Ali, T. Abuhmed, S. El-Sappagh, K. Muhammad, J. M. Alonso-Moral, R. Confalonieri, R. Guidotti, J. Del Ser, N. Díaz-Rodríguez et F. Herrera, “Explainable artificial intelligence (xai) : What we know and what is left to attain trustworthy artificial intelligence,” *Inf. Fusion*, vol. 99, n°. C, nov. 2023. [En ligne]. Disponible : https://doi.org/10.1016/j.inffus.2023.101805

[46] C. Molnar, *Interpretable Machine Learning : A Guide for Making Black Box Models Explainable*. Lulu.com : Lulu.com, 2020, chapter (in type field) 10, p. 120–145, online Edition. [En ligne]. Disponible : https://christophm.github.io/interpretable-ml-book/

[47] T. Miller, “Explanation in artificial intelligence : Insights from the social sciences,” *Artificial Intelligence*, vol. 267, 06 2017.

[48] A. Abusitta, M. Q. Li et B. C. Fung, “Survey on explainable ai : Techniques, challenges and open issues,” *Expert Syst. Appl.*, vol. 255, n°. PC, nov. 2024. [En ligne]. Disponible : https://doi.org/10.1016/j.eswa.2024.124710

[49] J. H. Hsiao, H. H. T. Ngai, L. Qiu, Y. Yang et C. C. Cao, “Roadmap of designing cognitive metrics for explainable artificial intelligence (XAI),” *CoRR*, vol. abs/2108.01737, 2021. [En ligne]. Disponible : https://arxiv.org/abs/2108.01737

[50] I. J. Goodfellow, J. Shlens et C. Szegedy, “Explaining and harnessing adversarial examples,” 2015. [En ligne]. Disponible : https://arxiv.org/abs/1412.6572

[51] C. Rudin, “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead,” 2019. [En ligne]. Disponible : https://arxiv.org/abs/1811.10154

[52] L. S. Shapley, “A value for n-person games,” dans *Contributions to the Theory of Games II*, H. W. Kuhn et A. W. Tucker, édit. Princeton : Princeton University Press, 1953, p. 307–317.

[53] H. P. Young, “Monotonic solutions of cooperative games,” vol. 14, n°. 2, 1985. [En ligne]. Disponible : https://doi.org/10.1007/BF01769885

[54] S. Lundberg et S.-I. Lee, “An unexpected unity among methods for interpreting model predictions,” 11 2016.

[55] I. Covert et S.-I. Lee, “Improving kernelshap : Practical shapley value estimation using linear regression,” dans *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Banerjee et K. Fukumizu, édit., vol. 130. PMLR, 13–15 Apr 2021, p. 3457–3465. [En ligne]. Disponible : https://proceedings.mlr.press/v130/covert21a.html

[56] A. Shrikumar, P. Greenside, A. Shcherbina et A. Kundaje, “Not just a black box : Learning important features through propagating activation differences,” 2017. [En ligne]. Disponible : https://arxiv.org/abs/1605.01713

[57] A. Shrikumar, P. Greenside et A. Kundaje, “Learning important features through propagating activation differences,” 2019. [En ligne]. Disponible : https://arxiv.org/abs/1704.02685

[58] H. Chen, S. Lundberg et S.-I. Lee, “Explaining models by propagating shapley values of local components,” 2019. [En ligne]. Disponible : https://arxiv.org/abs/1911.11888

[59] H. Chen, S. M. Lundberg et S.-I. Lee, “Explaining a series of models by propagating shapley values,” *Nature Communications*, vol. 13, n°. 1, août 2022. [En ligne]. Disponible :

http://dx.doi.org/10.1038/s41467-022-31384-3

[60] S. Lindbergh. An introduction to explainable ai with shapley values. [En ligne]. Disponible : https://shap.readthedocs.io/en/latest/example_notebooks/overviews/An%20introduction%20to%20explainable%20AI%20with%20Shapley%20values.html

[61] R. ZHAO, "Nsl-kdd," 2022. [En ligne]. Disponible : https://dx.doi.org/10.21227/8rpg-qt98

[62] F. W. L. W. P. A. Stolfo, Salvatore et P. Chan, "KDD Cup 1999 Data," UCI Machine Learning Repository, 1999, DOI : https ://doi.org/10.24432/C51C7N. [En ligne]. Disponible : https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[63] C. de Malefette, J.-Y. Ouattara, F. Gohring de Magalhaes, A. Shanejat Bushehri, A. Abusitta et G. N. Nicolescu, "An xai-based framework for trustworthy communications in avionics," *IEEE TRANSACTIONS ON AEROSPACE AND ELECTRONIC SYSTEMS [Soumis]*, 2025.

[64] A. Fried, "ADS-B Air Traffic for Anomalous Trajectory Detection," 2020. [En ligne]. Disponible : https://doi.org/10.17632/4x578h29f6.1

[65] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser et I. Polosukhin, "Attention is all you need," 2023. [En ligne]. Disponible : https://arxiv.org/abs/1706.03762

[66] J. Devlin, M.-W. Chang, K. Lee et K. N. Toutanova, "Bert : Pre-training of deep bidirectional transformers for language understanding," 2018. [En ligne]. Disponible : https://arxiv.org/abs/1810.04805

[67] H. Chefer, S. Gur et L. Wolf, "Transformer interpretability beyond attention visualization," 2021. [En ligne]. Disponible : https://arxiv.org/abs/2012.09838

[68] S. Jain et B. C. Wallace, "Attention is not explanation," 2019. [En ligne]. Disponible : https://arxiv.org/abs/1902.10186

ANNEXE A DÉMONSTRATION DES CONCEPTS D'XAI

Preuves des concepts d'XAI

Preuve de 3.3.3

Lemma A.0.1. *Si N est un support de v, alors : $\forall i \notin N, \varphi_i(v) = 0$*

Démonstration. Soit $N' = N \cup \{i\}$. N' est donc également un support de v. Par définition d'un support, on sait que $v(N') = v(N' \cap N) = v(N)$. L'axiome d'efficacité stipule que $v(N') = \sum_{j \in N'} \varphi_j(v) = \varphi_i(v) + \sum_{j \in N} \varphi_j(v) = \varphi_i(v) + v(N)$. Ainsi $\varphi_i(v) = 0$. □

Introduisons maintenant la notion de jeu primitif :

Definition A.0.2. *Soit $R \subset U$ non vide, on appelle jeu primitif de v selon R le jeu :*

$$v_R(S) = \begin{cases} 1 & si\ S \supseteq R, \\ 0 & si\ S \not\supseteq R. \end{cases}$$

Pour tout nombre positif c, cv_R est également un jeu.

Pour la suite, on note r, s et n les cardinaux des ensembles R, S et N.

Lemma A.0.3. *Soit $c \geq 0$ et $0 < r < \infty$, on a :* $\varphi_i(cv_R) = \begin{cases} c/r & si\ i \in R, \\ 0 & si\ i \notin R. \end{cases}$

Démonstration. Soient i, j dans R et choisissons une permutation $\pi \in \Pi(U)$ tel que $\pi i = j$ et $\pi R \equiv R$. Le symbole $\equiv$ signifie ici que les deux ensembles ont les mêmes éléments. Pour $U = \{1, 2, 3, 4\}$ et $R = (1, 2), i = 1, j = 2$, on peut par exemple choisir $\pi = (2, 1, 4, 3)$. Par définition d'un jeu primitif, on a donc $v_R = v_{\pi R}$. Par axiome de symétrie :

$$\begin{aligned} \varphi_j(cv_R) &= \varphi_{\pi i}(cv_R) && \text{(A.1)} \\ &= \varphi_{\pi i}(c\pi v_R) && \text{(A.2)} \\ &= \varphi_i(cv_R) && \text{(A.3)} \end{aligned}$$

Comme $v_R(R) = 1$, l'axiome d'efficacité permet de conclure que : $c = cv_r = \sum_{j \in R} \varphi_j(cv_R) = r\varphi_i(cv_R), \quad \forall i \in R$ □

Lemma A.0.4. *Tout jeu v s'écrit comme une combinaison linéaire de jeu primitifs v_R :*

$$v = \sum_{R \subset N, R \neq \emptyset} c_R(v) v_R \tag{A.4}$$

avec $c_R(v) = \sum_{T \subseteq R} (-1)^{r-t} v(T) \quad (0 < r < \infty)$, et N n'importe quel support de v

Démonstration. Soit N un support de v et $S \subset u$.
Si $S \subset N$,

$$\sum_{R \subset N, R \neq \emptyset} c_R(v) v_R(S) = \sum_{R \subset N, R \neq \emptyset} \sum_{T \subseteq R} (-1)^{r-t} v(T) v_R(S) \tag{A.5}$$

$$= \sum_{R \subset S, R \neq \emptyset} \sum_{T \subseteq R} (-1)^{r-t} v(T) \underbrace{v_R(S)}_{1} + \sum_{R \subset N \setminus S, R \neq \emptyset} \sum_{T \subseteq R} (-1)^{r-t} v(T) \underbrace{v_R(S)}_{0} \tag{A.6}$$

$$= \sum_{R \subset S, R \neq \emptyset} \sum_{T \subseteq R} (-1)^{r-t} v(T) \tag{A.7}$$

$$= \sum_{T \subseteq S} [\sum_{r=t}^{s} (-1)^{r-t} \binom{s-t}{r-t}] v(T) \tag{A.8}$$

$$= \sum_{T \subseteq S} [\sum_{r'=0}^{s'} (-1)^{r'} \binom{s'}{r'}] v(T) \tag{A.9}$$

Dans la dernière égalité, on a posé $s' = s - t$ et ré-indexé la variable muette $r' = r - t$. Pour tout s' strictement positif, le terme entre crochet s'annule. Le seul terme non nul est donc lorsque $s = t$, soit $S = T$. Ainsi :

$$\sum_{R \subset N, R \neq \emptyset} c_R(v) v_R(S) = \sum_{T \subseteq S} v(T) \tag{A.10}$$

$$= v(S) \tag{A.11}$$

Dans le cas général, pour un S quelconque, par définition d'un support on a :

$$v(S) = v(S \cap N) = \sum_{R \subset N} c_R(v) v_R(S \cap N) = \sum_{R \subset N, R \neq \emptyset} c_R(v) v_R(S) \tag{A.12}$$

Ce qui complète la preuve. □

En injectant le A.0.3 dans le A.0.4 en utilisant l'axiome d'agrégation on obtient, pour tout

$i \in N$:

$$\varphi_i(v) = \sum_{R\subset N, R\neq\emptyset} \varphi_i(c_R(v)v_R \tag{A.13}$$

$$= \sum_{R\subset N, i\in R} c_R(v)/r \tag{A.14}$$

$$= \sum_{R\subset N, i\in R} \sum_{S\subseteq R} (-1)^{r-s} v(S)/r \tag{A.15}$$

$$= \sum_{R\subset N, i\in R} \sum_{S\subseteq R, i\in S} (-1)^{r-s} v(S)/r + \sum_{R\subset N, i\in R} \sum_{S\subseteq R, i\notin S} (-1)^{r-s} v(S)/r \tag{A.16}$$

$$= \sum_{S\subseteq N, i\in S} [\sum_{S\subseteq R}^{n} \frac{(-1)^{r-s}}{r}] v(S) + \sum_{S\subseteq N, i\notin S} [\sum_{r=s}^{n} \frac{(-1)^{r-s}\binom{n-s-1}{r-s-1}}{r}] v(S) \tag{A.17}$$

$$= \sum_{S\subseteq N, i\in S} \frac{(s-1)!(n-s)!}{n!} v(S) - \sum_{S\subseteq N, i\notin S} \frac{s!(n-s-1)!}{n!} v(S) \tag{A.18}$$

$$= \sum_{S\subseteq N, i\in S} \gamma_n(s)[v(S) - v(S\backslash\{i\})] \tag{A.19}$$

où $\gamma_n(s) = \frac{(s-1)!(n-s)!}{n!}$.

Preuve de 3.3.6

Démonstration. **La valeur de Shapley est monotone :**
Soit $i \in U$, on suppose que $v^i(S) \geq w^i(S) \quad \forall S \subseteq U$. Alors :

$$\varphi_i(v) = \sum_{S\subseteq U, S\ni i} \gamma_U(S) v^i(S), \tag{A.20}$$

$$\geq \sum_{S\subseteq U, S\ni i} \gamma_U(S) w^i(S) \tag{A.21}$$

$$\geq \varphi_i(w) \tag{A.22}$$

Donc la valeur de Shapley est strictement monotone.

Une allocation symétrique strictement monotone ne peut être que la valeur de Shapley :
Supposons maintenant que nous disposons d'une allocation ψ symétrique et strictement monotone. D'abord, en appliquant la définition de la monotonicité forte dans les deux sens, on montre que :

$$v^i(S) = w^i(S) \quad \forall S \subseteq U \implies \psi_i(v) = \psi_i(w). \tag{A.23}$$

Considérons le jeu w donnant une valeur nulle pour toute coalition i.e : $w(S) = 0 \quad \forall S \subseteq U$. Par extension, $w^i(S) = 0 \quad \forall i \in U$. Pour tout $j \neq i$, on peut choisir une permutation $\pi \in \Pi(U)$ tel que $\pi i = j$. Cette permutation vérifie forcément $\pi w = w$ puisque w attribue toujours une valeur nulle donc une permutation quelconque n'y changera rien. Ainsi, par symétrie :

$$\psi_j(w) = \psi_{\pi i}(\pi w) = \psi_i(w) \tag{A.24}$$

Or par définition d'une allocation :

$$\sum_{j \in U} \psi_j(w) = w(U) = 0 = |U|\psi_i(w) \implies \psi_i(w) = 0. \tag{A.25}$$

Il s'ensuit donc que :

$$\forall i \in U, \quad \forall v, \quad (v^i(S) = 0 \quad \forall S \subseteq U \implies \psi_i(v) = 0) \tag{A.26}$$

Autrement dit, le joueur nul a une contribution nulle, montrant ainsi que l'axiome du joueur nul n'est plus nécessaire car déductible des trois autres. On utilise maintenant le fait que tout jeu v se décompose en somme de jeu primitif, comme détaillé en annexe A :

$$v = \sum_{R \subset N, R \neq \emptyset} c_R(v) v_R \tag{A.27}$$

avec $c_R(v) = \sum_{T \subseteq R} (-1)^{r-t} v(T) \quad (0 < r < \infty)$, et N n'importe quel support de v

Dans le cas de l'allocation de Shapley, on a $\varphi_i(v) = \sum_{R, i \in R} c_R / r$. Soit I, l'indice de v correspondant au nombre minimum de coefficient non nul dans l'expression de v ci-dessus.
Si $I = 0$ alors $(v(S) = 0 \quad \forall S \subseteq U \implies v^i(S) = 0 \quad \forall S \subseteq U \implies \psi_i(v) = 0)$.
Si $I = 1$, alors il existe un $R \subseteq U$ tel que que $v = c_R v_R$. Pour $i \notin R$, $v^i(S) = c_R v_R^i(S) = 0$ impliquant $\psi_i(v) = 0$. Pour tout $i, j \in R$, la symétrie de v_R implique que $\psi_i(v) = \psi_j(v)$. Par définition d'une allocation, $\sum_{j \in R} \psi_j(v) = \sum_{j \in N} \psi_j(v) = v(N) = c_R v_R(N) = c_R$ soit $\psi_i(v) = c_R / r$. Ainsi, ψ et φ coïncident lorsque I vaut 0 ou 1. Supposons que ψ coïncident avec φ jusqu'à l'indice $I - 1$. Considérons que v a un indice de I. Récrivons la décomposition du jeu v en ne considérant que ses coefficients non nuls :

$$v = \sum_{k=1}^{I} c_{R_k} v_{R_k}, c_{R_k} \neq 0 \forall k \tag{A.28}$$

Soit $R = \bigcap_{k=1}^{I} c_{R_k} v_{R_k}$ et $i \notin R$. Soit le jeu :

$$w = \sum_{k=1, i \in R_k} c_{R_k} v_{R_k} \tag{A.29}$$

L'indice de w est au plus $I - 1$ car sinon i serait dans R. Ainsi par hypothèse de récurrence, $\psi_i(w) = \varphi_i(w)$. De plus, tout R_k ne contenant pas i n'ajoute aucune contribution à v puisque $v^i_{R_k} = 0$. Ainsi $\forall S \subseteq U, v^i(S) = w^i(S)$ et par monotonicité forte on conclut :

$$\psi_i(v) = \psi_i(w) = \sum_{k=1, i \in R_k} c_{R_k}/r = \varphi_i(v) \tag{A.30}$$

Ainsi pour $i \notin R$, $\psi(v)$ et $\varphi(v)$ coïncident. Supposons maintenant que $i \in R$. Par symétrie, $\psi_i(v)$ est égale à une constante c sur tous les R_k. De la même manière, $\varphi_i(v) = c'$. Ainsi, par efficacité :

$$\begin{aligned}
\sum_{j \in N} \psi_j(v) &= \sum_{j \in R} \psi_j(v) + \sum_{i \notin R} \psi_i(v) && \text{(A.31)} \\
&= \sum_{j \in R} \varphi_j(v) + \sum_{i \notin R} c && \text{(A.32)} \\
&= v(N) && \text{(A.33)} \\
&= \sum_{j \in R} \varphi_j(v) + \sum_{i \notin R} \varphi_i(v) && \text{(A.34)} \\
&= \sum_{j \in R} \varphi_j(v) + \sum_{i \notin R} c' && \text{(A.35)} \\
&\implies \sum_{i \notin R} c = \sum_{i \notin R} c' \implies c = c' && \text{(A.36)}
\end{aligned}$$

□

L'axiome de monotonicité forte implique l'axiome de symétrie

Démonstration. Soient v et w deux jeux. Une autre façon d'écrire l'axiome de monotonicité forte est :

$$v(S \cup \{i\}) - v(S) \geq w(S \cup \{i\}) - w(S), \quad \forall S \subset U \backslash \{i\} \implies \varphi_i(v) \geq \varphi_i(w) \tag{A.37}$$

Soient $i, j \in U$ tels que $i \neq j$. Considérons que w est le même jeu que v mais où i et j sont inversés. Ainsi, pour tout S ne contenant ni i ni j, $v(S \cup \{i\}) = w(S \cup \{j\})$ et $v(S) = w(S)$.

$$\forall S \subset U \backslash \{i\}, v(S \cup \{i\}) - v(S) \geq w(S \cup \{i\}) - w(S), \implies \varphi_i(v) \geq \varphi_i(w) \tag{A.38}$$

$$\implies \forall S \subset U \backslash \{i, j\}, v(S \cup \{i\}) \geq w(S \cup \{i\}), \implies \varphi_i(v) \geq \varphi_i(w) \tag{A.39}$$

$$\implies \forall S \subset U \backslash \{i, j\}, v(S \cup \{i\}) \geq v(S \cup \{j\}), \implies \varphi_i(v) \geq \varphi_i(w) \tag{A.40}$$

En effectuant le même procédé en inversant i et j, on montre que :

$$\forall S \subset U \backslash \{i, j\}, v(S \cup \{i\}) = v(S \cup \{j\}), \implies \varphi_i(v) = \varphi_i(w) = \varphi_j(v) \tag{A.41}$$

qui est l'équivalent de l'axiome de symétrie. Pour rappel, l'idée principale de cet axiome est que la contribution d'un joueur est une propriété intrinsèque du jeu. Ainsi, deux joueurs jouant de la même manière au jeu, c'est-à-dire donnant le même résultat dans toutes les coalitions, ont une contribution identique. □

Preuve de 3.3.13

Soient $C_{\Delta x_i \Delta y_j}$ et $C_{\Delta y_j \Delta t}$ vérifiant tout deux la sommation à delta et le multiplicateur $m_{\Delta x_i \Delta t}$ suivant la règle de chaînage, alors :

Démonstration.

$$\sum_i C_{\Delta x_i \Delta t} = \sum_i m_{\Delta x_i \Delta t} \Delta x_i \tag{A.42}$$

$$= \sum_i (\sum_j m_{\Delta x_i \Delta y_j} m_{\Delta y_j \Delta t}) \Delta x_i \tag{A.43}$$

$$= \sum_j (\sum_i m_{\Delta x_i \Delta y_j} \Delta x_i) m_{\Delta y_j \Delta t} \tag{A.44}$$

$$= \sum_j (\sum_i C_{\Delta x_i \Delta y_j}) m_{\Delta y_j \Delta t} \tag{A.45}$$

$$= \sum_j \Delta y_j m_{\Delta y_j \Delta t} \tag{A.46}$$

$$= \sum_i C_{\Delta y_j \Delta t} \tag{A.47}$$

$$= \Delta t \tag{A.48}$$

□

Preuve de 3.3.14

Démonstration. Soit N l'ensemble des caractéristiques du modèle et P l'ensemble des permutations de $N\backslash\{i\}$. Soit $\chi(x, x', S)$ le vecteur de $\mathbb{R}^N$ tel que :

$$\chi(x, x', S)_i = \begin{cases} x_i & \text{si } i \in S; \\ x'_i & \text{sinon} \end{cases} \tag{A.49}$$

$$\varphi_i(f, x) = \frac{1}{n} \sum_{S \subseteq N\backslash\{i\}} \binom{n-1}{s}^{-1} (f_x(x'_S) - f_x(x'_S\backslash i)) \text{d'après 3.28} \tag{A.50}$$

$$\varphi_i(f, x) = \frac{1}{n} \sum_{S \subseteq N\backslash\{i\}} \binom{n-1}{s}^{-1} (E_D[f(X)|x_{S\cup\{i\}}] - E_D[f(X)|x_S]) \text{d'après 3.29} \tag{A.51}$$

$$= \frac{1}{n!} \sum_{R \subseteq P} (E_D[f(X)|x_{P_R^i \cup \{i\}}] - E_D[f(X)|x_{P_R^i}]) \text{ en utilisant la formulation en permutation 3.3.4} \tag{A.52}$$

$$= \frac{1}{n!} \sum_{R \subseteq P} \frac{1}{|D|} \sum_{x^b \in D} (E_{\{x^b\}}[f(X)|x_{P_R^i \cup \{i\}}] - E_{\{x^b\}}[f(X)|x_{P_R^i}]) \text{ en décomposant sur toutes les } \textit{baseline} \tag{A.53}$$

$$= \frac{1}{n!} \sum_{R \subseteq P} \frac{1}{|D|} \sum_{x^b \in D} (f(\chi(x, x^b, P_R^i \cup \{i\})) - f(\chi(x, x^b, P_R^i))) \tag{A.54}$$

$$= \frac{1}{|D|} \sum_{x' \in D} \frac{1}{n!} \sum_{R \subseteq P} (f(\chi(x, x^b, P_R^i \cup \{i\})) - f(\chi(x, x', P_R^i))) \tag{A.55}$$

$$= \frac{1}{|D|} \sum_{x^b \in D} \underbrace{\frac{1}{n} \sum_{S \subseteq N\backslash\{i\}} \binom{n-1}{s}^{-1} (f(\chi(x, x^b, S \cup \{i\})) - f(\chi(x, x^b, S)))}_{\varphi_i(f, x, x^b)} \tag{A.56}$$

$$= \frac{1}{|D|} \sum_{x^b \in D} \varphi_i(f, x, x^b) \tag{A.57}$$

□

Preuve de 3.3.16

Démonstration. Par simplicité, on notera $\varphi = \varphi^i(h_i, x, x^b)$ Soit $\hat{1}_{a\times b}$ la matrice de taille $a \times b$ contenant que des 1. On sait déja que les φ_i respectent l'axiome d'efficacité soit :

$$\hat{1}_{1\times m_k} \varphi_i = f_i(x) - f_i(x^b) \tag{A.58}$$

Puisque $\psi^k = \varphi^k$ donc

$$\hat{1}_{1\times m_k}\psi_k = f_k(x) - f_k(x^b) \tag{A.59}$$

ce qui valide l'initialisation de notre récurrence. Soit $i \in 2...k$. Supposons maintenant que l'hypothèse est vrai au rang i soit $\hat{1}_{1\times m_i}\psi_i = f_k(x) - f_k(x^b)$. Alors :

$$\begin{aligned}
\psi^i &= \hat{1}_{1\times m_i}\varphi^i(\psi^{i+1} \oslash (f_i(x) - f_i(x^b)) && \text{(A.60)}\\
&= (f_i(x) - f_i(x^b))(\psi^{i+1} \oslash (f_i(x) - f_i(x^b)) && \text{(A.61)}\\
&\text{par efficacité de } \varphi^i && \text{(A.62)}\\
&= \hat{1}_{1\times o_i}\psi^{i+1} \text{car } f_i : \mathbb{R}^m \to \mathbb{R}^{o_i} && \text{(A.63)}\\
&= \hat{1}_{1\times m_{i+1}}\psi^{i+1} && \text{(A.64)}\\
&= f_k(x) - f_k(x^b) && \text{(A.65)}
\end{aligned}$$

□