Testeur certifié
Test d'IA (CT-AI)
Syllabus

Version 1.0

International Software Testing Qualifications Board

Fourni par

Alliance for Qualification, Artificial Intelligence United,
Chinese Software Testing Qualifications Board,
et Korean Software Testing Qualifications Board

Avis de copyright

Historique de révision

Version	Date	Remarques
1.0	2021/10/01	Release pour le GA
1.0 FR	2022/08/06	Version française
1.0.1 FR	2025/09/05	Correction de coquilles

Table des matières

Remerciements

Ce document a été officiellement publié par l'Assemblée générale de l'ISTQB® le 1er octobre 2021.

Il a été produit par une équipe de l'International Software Testing Qualifications Board: Klaudia Dussa-Zieger (chair), Werner Henschelchen, Vipul Kocher, Qin Liu, Stuart Reid, Kyle Siemens, and Adam Leon Smith.

L'équipe remercie les auteurs des trois syllabi qui ont contribué à l'élaboration de ce document ;

- A4Q: Rex Black, Bruno Legeard, Jeremias Rößler, Adam Leon Smith, Stephan Goericke, Werner Henschelchen
- AiU: Auteurs principaux Vipul Kocher, Saurabh Bansal, Srinivas Padmanabhuni and Sonika Bengani et co-auteurs Rik Marselis, José M. Diaz Delgado
- CSTQB/KSTQB: Qin Liu, Stuart Reid

L'équipe remercie les groupes de travail Examen, Glossaire et Marketing pour leur soutien tout au long de l'élaboration du syllabus, Graham Bath pour sa révision technique et les représentants issus des Membres de l'ISTQB® pour leurs suggestions et leurs contributions.

Les personnes suivantes ont participé à la révision et aux commentaires de ce syllabus :
Laura Albert, Reto Armuzzi, Árpád Beszédes, Armin Born, Géza Bujdosó, Renzo Cerquozzi, Sudeep Chatterjee, Seunghee Choi, Young-jae Choi, Piet de Roo, Myriam Christener, Jean-Baptiste Crouigneau, Guofu Ding,Erwin Engelsma, Hongfei Fan, Péter Földházi Jr., Tamás Gergely, Ferdinand Gramsamer, Attila Gyúri, Matthias Hamburg, Tobias Horn, Jarosław Hryszko, Beata Karpinska, Joan Killeen, Rik Kochuyt, Thomas Letzkus, Chunhui Li, Haiying Liu, Gary Mogyorodi, Rik Marselis, Imre Mészáros, Tetsu Nagata, Ingvar Nordström, Gábor Péterffy, Tal Pe'er, Ralph Pichler, Nishan Portoyan, Meile Posthuma, Adam Roman, Gerhard Runze, Andrew Rutz, Klaus Skafte, Mike Smith, Payal Sobti, Péter Sótér, Michael Stahl, Chris van Bael, Stephanie van Dijck, Robert Werkhoven, Paul Weymouth, Dong Xin, Ester Zabar, Claude Zhang.

0 Introduction

0.1 Objectif de ce syllabus.

Ce syllabus constitue la base du test ISTQB® Testeur Certifié Test d'IA. L'ISTQB® fournit ce syllabus de la manière suivante :

1. Aux Membres, pour la traduction dans leur langue respective et pour l'accréditation des organismes de formation. Les Membres peuvent adapter le syllabus à leurs besoins linguistiques particuliers et modifier les références pour les adapter à leurs publications locales.
2. Aux organismes de certification, pour élaborer des questions d'examen localisées et adaptées aux objectifs d'apprentissage de ce syllabus.
3. Aux organismes de formation, pour produire des cours et pour déterminer les méthodes d'enseignement appropriées.
4. Aux candidats à la certification, pour préparer l'examen de certification (dans le cadre d'une formation ou de manière indépendante).
5. A la communauté internationale de l'ingénierie des logiciels et des systèmes, pour faire progresser la profession des tests de logiciels et de systèmes, et comme base pour des livres et des articles.

0.2 La certification testeur certifié en test d'IA

La certification testeur certifié en test d'IA s'adresse à toute personne impliquée dans le test de systèmes basés sur l'IA et/ou l'IA pour le test. Il s'agit notamment de testeurs, d'analystes de test, d'analystes de données, d'ingénieurs de test, de consultants en test, de responsables de test, de testeurs / utilisateurs impliqués dans les tests d'acceptation et de développeurs de logiciels. Cette certification convient également à toute personne souhaitant acquérir une compréhension de base des tests de systèmes basés sur l'IA et/ou de l'IA pour les tests, comme les chefs de projet, les responsables qualité, les responsables du développement de logiciels, les analystes métier, les membres de l'équipe des opérations, les directeurs informatiques et les consultants en management.

La vue d'ensemble de la certification testeur certifié en test d'IA [I03] est fournie dans un document séparé qui comprend les informations suivantes :

- Objectifs métier pour le syllabus
- Matrice des objectifs métiers et lien avec les objectifs d'apprentissage
- Résumé du syllabus
- Relations entre les syllabus

0.3 Objectifs d'apprentissage examinables et niveau cognitif des connaissances

Les objectifs d'apprentissage soutiennent les résultats du métier et sont utilisés pour créer les examens de Testeur certifié - test d'IA.

Il peut être demandé aux candidats de reconnaître, se souvenir ou se rappeler d'un mot-clé ou d'un concept mentionné dans l'un des onze chapitres. Les niveaux d'objectifs d'apprentissage spécifiques sont indiqués au début de chaque chapitre, et classés comme suit :

- K1 : Se souvenir
- K2 : Comprendre
- K3 : Appliquer
- K4 : Analyser

Tous les termes énumérés comme mots-clés juste en dessous des titres de chapitre doivent être mémorisés (K1), même s'ils ne sont pas explicitement mentionnés dans les objectifs d'apprentissage.

0.4 Travaux pratiques Niveaux de compétence.

La certification Spécialiste des testeurs certifiés en test d'IA comprend des objectifs pratiques qui mettent l'accent sur les aptitudes et compétences concrètes.

Les niveaux suivants s'appliquent aux objectifs pratiques (comme indiqué) :

- H0 : Démonstration en direct d'un exercice ou vidéo enregistrée.
- H1 : Exercice guidé. Les participants suivent une séquence d'étapes réalisée par le formateur.
- H2 : Exercice avec conseils. L'étudiant reçoit un exercice avec des conseils pertinents afin que l'exercice puisse être résolu dans le temps imparti, ou les participants participent à une discussion.

Les compétences sont acquises en effectuant des exercices pratiques, tels que ceux présentés dans la liste suivante :

- Démontrer le sous-ajustement et le sur-ajustement (H0).
- Effectuer la préparation des données en vue de la création d'un modèle ML (H2).
- Identifier les ensembles de données d'apprentissage et de test et créer un modèle ML (H2).
- Évaluer le modèle ML créé à l'aide de métriques de performance fonctionnelle ML sélectionnées (H2).
- Expérience de l'implémentation d'un perceptron (H1).
- Utiliser un outil pour montrer comment l'explicabilité peut être utilisée par les testeurs (H2).
- Appliquer le test par paire pour dériver et exécuter des cas de test pour un système basé sur l'IA (H2).

- Appliquer le test métamorphique pour dériver et exécuter des cas de test pour un scénario donné (H2).
- Appliquer les tests exploratoires à un système basé sur l'IA (H2).
- Discuter, à l'aide d'exemples, des activités de test pour lesquelles l'IA est moins susceptible d'être utilisée (H2).
- Mettre en œuvre un système simple de prédiction des défauts basé sur l'IA (H2).

0.5 L'examen de testeur certifié en test d'IA.

L'examen de testeur certifié en test d'IA sera basé sur ce syllabus. Les réponses aux questions de l'examen peuvent nécessiter l'utilisation de matériel basé sur plus d'une section de ce syllabus. Toutes les sections du syllabus sont examinables, à l'exception de l'introduction et des annexes. Les normes et les livres sont inclus comme références, mais leur contenu n'est pas examinable, au-delà de ce qui est résumé dans le syllabus lui-même à partir de ces normes et livres.

Reportez-vous au document "Aperçu" du testeur certifié en test d'IA pour plus de détails dans la section "Structure de l'examen".

Note sur les conditions d'admission : le certificat ISTQB® de Niveau Fondation doit être obtenu avant de passer l'examen de testeur certifié en test d'IA.

0.6 Accréditation

Un Membre de l'ISTQB® peut accréditer les organismes de formation dont le matériel de cours suit ce syllabus. Les organismes de formation doivent obtenir les directives d'accréditation auprès du Membre ou de l'organisme qui effectue l'accréditation. Un cours accrédité est reconnu conforme à ce syllabus et est autorisé à intégrer un examen ISTQB® dans le cadre du cours.

Les directives d'accréditation pour ce syllabus suivent les directives générales d'accréditation publiées par le groupe de travail Processes Management and Compliance Working Group de l'ISTQB®.

0.7 Niveau de détail

Le niveau de détail de ce syllabus permet d'assurer la cohérence des cours et des examens au niveau international. Afin d'atteindre ce but, le syllabus se compose :

- Des objectifs pédagogiques généraux décrivant l'intention du testeur certifié en test d'IA.
- Une liste de termes que les participants doivent être capables de se rappeler.
- Des objectifs d'apprentissage et de mise en pratique pour chaque domaine de connaissances, décrivant les résultats d'apprentissage à atteindre.
- Une description des concepts clés, y compris des références à des sources telles que la littérature ou les normes acceptées.

Le contenu du syllabus n'est pas une description de l'ensemble du domaine de connaissances pour le test des systèmes basés sur l'IA ; il reflète le niveau de détail à couvrir dans les cours de formation de testeur certifié en test d'IA. Il se concentre sur l'introduction des concepts de base de l'intelligence artificielle (IA) et de l'apprentissage automatique en particulier, et sur la façon dont les systèmes basés

sur ces technologies peuvent être testés.

0.8 Organisation de ce syllabus.

Il y a onze chapitres dont le contenu peut être examiné. Le titre de niveau supérieur de chaque chapitre précise la durée du chapitre ; la durée n'est pas fournie en dessous du niveau du chapitre. Pour les cours de formation accrédités, le syllabus exige un minimum de 25,1 heures d'instruction, réparties entre les onze chapitres comme suit :

- Chapitre 1 : 105 minutes Introduction à l'IA
- Chapitre 2 : 105 minutes Caractéristiques de qualité des systèmes basés sur l'IA
- Chapitre 3 : 145 minutes Machine Learning (ML) - Vue d'ensemble
- Chapitre 4 : 230 minutes ML - Données
- Chapitre 5 : 120 minutes ML - Métriques de performance fonctionnelle
- Chapitre 6 : 65 minutes ML - Réseaux neuronaux et tests
- Chapitre 7 : 115 minutes Test des systèmes basés sur l'IA - Vue d'ensemble
- Chapitre 8 : 150 minutes Test des caractéristiques de qualité spécifiques à l'IA
- Chapitre 9 : 245 minutes Méthodes et techniques pour le test des systèmes basés sur l'IA
- Chapitre 10 : 30 minutes Environnements de test pour les systèmes basés sur l'IA
- Chapitre 11 : 195 minutes Utilisation de l'IA pour les tests

1 Introduction à l'IA – 105 minutes

Mots clés pour les tests

Aucun

Mots-clés spécifiques à l'IA

AI as a Service (AIaaS), framework de développement de l'IA, effet de l'IA, système basé sur l'IA, intelligence artificielle (IA), réseau neuronal, deep learning (DL), réseau neuronal profond, IA générale, règlement général sur la protection des données (RGPD), machine learning (ML), IA étroite, modèle pré entrainé, super IA, singularité technologique, apprentissage par transfert.

Objectifs d'apprentissage pour le chapitre 1 :

1.1 Définition de l'IA et de l'effet de l'IA

AI-1.1.1 K2 Décrire l'effet de l'IA et comment il influence la définition de l'IA.

1.2 IA Étroite, générale et super IA

AI-1.2.1 K2 Distinguer l'IA étroite, l'IA générale et la super IA.

1.3 Systèmes basés sur l'IA et systèmes conventionnels.

AI-1.3.1 K2 Faire la différence entre les systèmes basés sur l'IA et les systèmes conventionnels.

1.4 Les technologies de l'IA

AI-1.4.1 K1 Reconnaître les différentes technologies utilisées pour mettre en œuvre l'IA.

1.5 Frameworks de développement de l'IA

AI-1.5.1 K1 Identifier les frameworks de développement de l'IA les plus populaires.

1.6 Matériel pour les systèmes basés sur l'IA

AI-1.6.1 K2 Comparer les choix disponibles en matière de matériel pour la mise en œuvre de systèmes basés sur l'IA.

1.7 L'IA en tant que service (AIaaS)

AI-1.7.1 K2 Expliquer le concept d'IA en tant que service (AIaaS).

1.8 Modèles pré-entraînés

AI-1.8.1 K2 Expliquer l'utilisation de modèles d'IA pré-entraînés et les risques qui y sont associés.

1.9 Normes, réglementations et IA

AI-1.9.1 K2 Décrire comment les normes s'appliquent aux systèmes basés sur l'IA.

1.1 Définition de l'IA et de l'effet de l'IA.

Le terme d'intelligence artificielle (IA) remonte aux années 1950 et désigne l'objectif de construire et de programmer des machines "intelligentes" capables d'imiter les êtres humains. Aujourd'hui, la définition a beaucoup évolué et la définition suivante résume bien le concept [S01] :

La capacité d'un système technique à acquérir, traiter et appliquer des connaissances et des compétences.

La façon dont les gens comprennent la signification de l'IA dépend de leur perception actuelle. Dans les années 1970, l'idée d'un système informatique capable de battre un humain aux échecs semblait futuriste et la plupart des gens considéraient cela comme de l'IA. Aujourd'hui, plus de vingt ans après que le système informatique Deep Blue a battu le champion du monde d'échecs Garry Kasparov, l'approche de "force brute" mise en œuvre dans ce système n'est pas considérée par beaucoup comme une véritable intelligence artificielle (c'est-à-dire que le système n'a pas appris à partir de données et n'était pas capable d'auto-apprentissage). De même, les systèmes experts des années 1970 et 1980 incorporaient l'expertise humaine sous forme de règles qui pouvaient être exécutées de manière répétée sans la présence de l'expert. Ces systèmes étaient considérés comme de l'IA à l'époque, mais ne sont plus considérés comme tels aujourd'hui.

L'évolution de la perception de ce qui constitue l'IA est connue sous le nom d'"effet IA" [R01]. L'évolution de la perception de l'IA dans la société entraîne celle de sa définition. Par conséquent, toute définition donnée aujourd'hui est susceptible de changer à l'avenir et de ne pas correspondre à celles du passé.

1.2 IA étroite, générale et super IA.

À un niveau élevé, l'IA peut être divisée en trois catégories :

- L'IA étroite (également appelée IA faible) : les systèmes sont programmés pour effectuer une tâche spécifique dans un contexte limité. Actuellement, cette forme d'IA est largement disponible. Par exemple, les systèmes de jeu, les filtres anti-spam, les générateurs de cas de test et les assistants vocaux.
- Les systèmes d'IA générale (également appelée IA forte) possèdent des capacités cognitives générales (de grande envergure) similaires à celles des humains. Ces systèmes basés sur l'IA peuvent raisonner et comprendre leur environnement comme le font les humains, et agir en conséquence. En 2021, aucun système d'IA générale n'avait encore été réalisé.
- Les systèmes de super IA sont capables de reproduire la cognition humaine (IA générale) et utilisent une puissance de traitement massive, une mémoire pratiquement illimitée et un accès à toutes les connaissances humaines (par exemple, par l'accès au web). On pense que les systèmes de super IA deviendront rapidement plus sages que les humains. Le moment où les systèmes basés sur l'IA passent de l'IA générale à la super IA est communément appelé la singularité technologique [B01].

1.3 Systèmes basés sur l'IA et systèmes conventionnels.

Dans un système informatique conventionnel typique, le logiciel est programmé par l'homme à l'aide d'un langage impératif, qui comprend des constructions telles que « if-then-else » et des boucles. Il est relativement facile pour les humains de comprendre comment le système transforme les entrées en sorties. Dans un système basé sur l'IA utilisant l'apprentissage automatique (machine learning), le système utilise des modèles de données pour déterminer comment il doit réagir à l'avenir à de nouvelles données (voir le chapitre 3 pour une explication détaillée de ML). Par exemple, un processeur d'images basé sur l'IA conçu pour identifier des images de chats est entraîné avec un ensemble d'images connues pour contenir des chats. L'IA détermine d'elle-même quels modèles ou caractéristiques dans les données peuvent être utilisés pour identifier les chats. Ces modèles et règles sont ensuite appliqués à de nouvelles images afin de déterminer si elles contiennent des chats. Dans de nombreux systèmes

basés sur l'IA, la procédure de prédiction est donc moins facile à comprendre pour les humains (voir section 2.7).

En pratique, les systèmes basés sur l'IA peuvent être mis en œuvre par diverses technologies (voir section 1.4), et l'"effet IA" (voir section 1.1) peut déterminer ce qui est actuellement considéré comme un système basé sur l'IA et ce qui est considéré comme un système conventionnel.

1.4 Technologies d'IA.

L'IA peut être mise en œuvre à l'aide d'un large éventail de technologies (voir [B02] pour plus de détails), telles que :

- La logique floue
- Les algorithmes de recherche
- Les techniques de raisonnement
    - Moteurs de règles
    - Classificateurs déductifs
    - Raisonnement à base de cas
    - Raisonnement procédural
- Les techniques de machine learning
    - Réseaux neuronaux
    - Modèles bayésiens
    - Arbres de décision
    - Forêt aléatoire
    - Régression linéaire
    - Régression logistique
    - Algorithmes de mise en grappes
    - Algorithmes génétiques
    - Machine à vecteur de support (SVM)

Les systèmes basés sur l'IA mettent généralement en œuvre une ou plusieurs de ces technologies

1.5 Frameworks de développement IA

Il existe de nombreux frameworks de développement de l'IA, dont certains sont axés sur des domaines spécifiques. Ces frameworks prennent en charge une série d'activités, telles que la préparation des données, la sélection d'algorithmes et la compilation de modèles à exécuter sur divers processeurs, tels que les unités centrales de traitement (CPU), les unités de traitement graphique (GPU) ou les unités de traitement Tensor (TPU). Le choix d'un framework particulier peut également dépendre d'aspects particuliers tels que le langage de programmation utilisé pour la mise en œuvre et sa facilité d'utilisation. Les frameworks suivants sont parmi les plus populaires (en date d'avril 2021) :

- Apache MxNet : Un framework open-source de deep learning utilisé par Amazon pour Amazon Web Services (AWS) [R02].
- CNTK : La boîte à outils cognitive de Microsoft (CNTK) est une boîte à outils de deep learning en open source [R03].
- IBM Watson Studio : Une suite d'outils qui prennent en charge le développement de solutions d'IA [R04].
- Keras : Une API open source de haut niveau, écrite en langage Python, capable de fonctionner au-dessus de TensorFlow et CNTK [R06].
- PyTorch : Une bibliothèque ML open-source exploitée par Facebook et utilisée pour les applications de traitement d'images et de traitement du langage naturel (NLP). Le support est fourni pour les interfaces Python et C++ [R07].
- Scikit-learn : Une bibliothèque open-source de ML pour le langage de programmation Python [R08].
- TensorFlow : un framework ML open-source basé sur des graphes de flux de données pour un machine learning évolutif, fourni par Google [R05].

Notez que ces frameworks de développement sont en constante évolution, parfois combinés et parfois remplacés par de nouveaux frameworks.

1.6 Matériel pour les systèmes basés sur l'IA

Une variété de matériel est utilisée pour l'apprentissage des modèles ML (voir le chapitre 3) et la mise en œuvre des modèles. Par exemple, un modèle qui effectue la reconnaissance vocale peut fonctionner sur un smartphone bas de gamme, bien qu'un accès à la puissance du cloud computing puisse être nécessaire pour l'entraîner. Une approche courante utilisée lorsque le dispositif hôte n'est pas connecté à Internet consiste à entraîner le modèle dans le cloud et à le déployer ensuite sur le dispositif hôte.

Le ML bénéficie généralement d'un matériel qui prend en charge les attributs suivants :

- Arithmétique de basse précision : Cela utilise moins de bits pour le calcul (par exemple, 8 au lieu de 32 bits, ce qui est généralement suffisant pour le ML).
- La capacité de travailler avec de grandes structures de données (par exemple, pour supporter la multiplication de matrices).
- Traitement massivement parallèle (simultané).

Les CPU polyvalents prennent en charge des opérations complexes qui ne sont généralement pas nécessaires aux applications de ML et ne fournissent que quelques cœurs. Par conséquent, leur architecture est moins efficace pour l'apprentissage et l'exécution de modèles de ML que les GPU, qui ont des milliers de cœurs et qui sont conçus pour effectuer le traitement massivement parallèle mais relativement simple des images. Par conséquent, les GPU sont généralement plus performants que les CPU pour les applications de ML, même si les CPU ont généralement des vitesses d'horloge plus élevées. Pour les travaux de ML à petite échelle, les GPU constituent généralement la meilleure option.

Certains matériels sont spécialement conçus pour l'IA, comme les circuits intégrés spécifiques aux applications (ASIC - Application-Specific Integrated Circuits) et les systèmes sur puce (SoC - System on a Chip). Ces solutions spécifiques à l'IA possèdent des caractéristiques telles que des cœurs multiples,

une gestion spéciale des données et la possibilité d'effectuer un traitement en mémoire. Elles conviennent le mieux à l'informatique périphérique, tandis que l'apprentissage du modèle ML se fait dans le cloud.

Du matériel doté d'architectures d'IA spécifiques est actuellement (avril 2021) en cours de développement. Cela inclut les processeurs neuromorphiques [B03], qui n'utilisent pas l'architecture traditionnelle de von Neumann, mais plutôt une architecture qui imite dans une certaine mesure les neurones du cerveau.

Voici quelques exemples de fournisseurs de matériel d'IA et de leurs processeurs (en avril 2021) :

- NVIDIA : Ils fournissent une gamme de GPU et de processeurs spécifiques à l'IA, tels que le Volta [R09].
- Google : Ils ont développé des circuits intégrés spécifiques aux applications pour l'apprentissage et l'inférence. Les TPU de Google (Cloud Tensor Processing Units) [R10] sont accessibles aux utilisateurs sur le Google Cloud, tandis que le TPU Edge [R11] est un ASIC spécialement conçu pour exécuter l'IA sur des appareils individuels.
- Intel : Intel fournit les processeurs de réseaux neuronaux Nervana [R12] pour le deep learning (apprentissage et inférence) et les unités de traitement de la vision Movidius Myriad pour l'inférence dans les applications de vision par ordinateur et de réseaux neuronaux.
- Mobileye : Cette société produit la famille EyeQ de dispositifs SoC [R13] pour prendre en charge le traitement complexe et intense de la vision. Ils ont une faible consommation d'énergie pour une utilisation dans les véhicules.
- Apple : La société produit la puce Bionic pour l'IA intégrée dans les iPhones [B04].
- Huawei : sa puce Kirin 970 pour les smartphones intègre un traitement de réseau neuronal pour l'IA [B05].

1.7 AI as a Service (AIaaS)

Les composants d'IA, tels que les modèles ML, peuvent être créés au sein d'une organisation, téléchargés à partir d'une tierce partie ou utilisés comme un service sur le web (AIaaS – AI as a Service). Une approche hybride est également possible, dans laquelle une partie de la fonctionnalité d'IA est fournie à partir du système et une autre partie est fournie en tant que service.

Lorsque l'intelligence artificielle est utilisée comme un service, l'accès à un modèle d'intelligence artificielle est fourni sur le web et une assistance peut également être fournie pour la préparation et le stockage des données, l'apprentissage, l'évaluation, le réglage, les tests et le déploiement du modèle.

Des fournisseurs tiers (par exemple, AWS, Microsoft) proposent des services d'IA spécifiques, tels que la reconnaissance faciale et vocale. Cela permet aux individus et aux organisations de mettre en œuvre l'IA en utilisant des services basés sur le cloud, même lorsqu'ils ne disposent pas des ressources et de l'expertise suffisantes pour construire leurs propres services d'IA. En outre, les modèles de ML fournis dans le cadre d'un service tiers sont susceptibles d'avoir été entraînés sur un ensemble de données plus vaste et plus diversifié que celui auquel ont accès de nombreuses parties prenantes, notamment celles qui ont récemment fait leur entrée sur le marché de l'IA.

1.7.1 Contrats pour l'IA en tant que service.

Ces services d'intelligence artificielle sont généralement fournis dans le cadre de contrats similaires à ceux des logiciels en tant que service (SaaS – Software as a Service) basés sur le cloud et sans intelligence artificielle. Un contrat pour AIaaS comprend généralement un accord de niveau de service (SLA – Service Level Agreement) qui définit les engagements de disponibilité et de sécurité. Ces SLA couvrent généralement un temps de fonctionnement du service (par exemple, 99,99% de temps de fonctionnement) et un temps de réponse pour corriger les défauts, mais définissent rarement des métriques de performance fonctionnelle ML (comme la précision), de manière similaire (voir chapitre 5). L'AIaaS est souvent payé sur la base d'un abonnement, et si la disponibilité et/ou les temps de réponse contractuels ne sont pas respectés, le fournisseur de services offre généralement des crédits pour les services futurs. Outre ces crédits, la plupart des contrats AIaaS prévoient une responsabilité limitée (autre qu'en termes de frais payés), ce qui signifie que les systèmes basés sur l'IA qui dépendent de l'AIaaS sont généralement limités à des applications à risque relativement faible, où la perte de service ne serait pas trop dommageable.

Les services s'accompagnent souvent d'une période d'essai initiale gratuite au lieu d'une période d'acceptation. Pendant cette période, le consommateur de l'AIaaS est censé tester si le service fourni répond à ses besoins en termes de fonctionnalités et de performances (par exemple, la précision). Ceci est généralement nécessaire pour couvrir tout manque de transparence sur le service fourni (voir section 7.5).

1.7.2 Exemples d'AIaaS

Voici quelques exemples d'AIaaS (en date d'avril 2021) :

- IBM Watson Assistant : il s'agit d'un chatbot d'IA dont le prix est fixé en fonction du nombre d'utilisateurs actifs mensuels.
- Google Cloud AI and ML Products : Ceux-ci fournissent une IA basée sur les documents qui comprend un analyseur syntaxique de formulaires et une reconnaissance de caractères (OCR - Optical Character Recognition) de documents. Les prix sont basés sur le nombre de pages envoyées pour traitement.
- Amazon CodeGuru : Il s'agit d'un examen du code Java ML qui fournit aux développeurs des recommandations pour améliorer la qualité de leur code. Les prix sont basés sur le nombre de lignes de code source analysées.
- Microsoft Azure Cognitive Search : Ce service fournit une recherche en cloud d'IA. Les prix sont basés sur les unités de recherche (définies en termes de stockage et de débit utilisés).

1.8 Modèles pré-entraînés

1.8.1 Introduction aux modèles pré-entraînés.

L'apprentissage des modèles ML peut être coûteux (voir le chapitre 3). Tout d'abord, les données doivent être préparées, puis le modèle doit être entraîné. La première activité peut consommer de grandes quantités de ressources humaines, tandis que la seconde peut consommer beaucoup de ressources informatiques. De nombreuses organisations n'ont pas accès à ces ressources.

Une alternative moins coûteuse et souvent plus efficace consiste à utiliser un modèle pré-entraîné. Celui-ci offre des fonctionnalités similaires au modèle requis et sert de base à la création d'un nouveau modèle qui étend et/ou concentre les fonctionnalités du modèle pré-entraîné. De tels modèles ne sont disponibles que pour un nombre limité de technologies, telles que les réseaux neuronaux et les forêts aléatoires.

Si un classificateur d'images est nécessaire, il peut être entraîné à l'aide de l'ensemble de données publiques ImageNet, qui contient plus de 14 millions d'images classées en plus de 1000 catégories. Cela réduit le risque de consommer des ressources importantes sans garantie de succès. Il est également possible de réutiliser un modèle existant qui a déjà été entraîné sur ce jeu de données. En utilisant un tel modèle pré-entraîné, les coûts d'apprentissage sont économisés et le risque d'échec est largement éliminé.

Lorsqu'un modèle pré-entraîné est utilisé sans modification, il peut simplement être intégré dans le système basé sur l'IA ou être utilisé comme un service (voir la section 1.7).

1.8.2 Apprentissage par transfert.

Il est également possible de prendre un modèle pré-entraîné et de le modifier pour répondre à une deuxième exigence, différente. Il s'agit de l'apprentissage par transfert, utilisé sur les réseaux neuronaux profonds dans lesquels les premières couches (voir chapitre 6) du réseau neuronal effectuent généralement des tâches de base (par exemple, l'identification de la différence entre des lignes droites et courbes dans un classificateur d'images), tandis que les dernières couches effectuent des tâches plus spécialisées (par exemple, la différenciation des types architecturaux de bâtiments). Dans cet exemple, toutes les couches d'un classificateur d'images, sauf les dernières, peuvent être réutilisées, ce qui élimine le besoin d'entraîner les premières couches. Les couches ultérieures sont ensuite réentraînées pour répondre aux exigences uniques d'un nouveau classificateur. En pratique, le modèle pré-entraîné peut être affiné par un apprentissage supplémentaire sur de nouvelles données spécifiques au problème.

L'efficacité de cette approche dépend largement de la similitude entre la fonction exécutée par le modèle original et la fonction requise par le nouveau modèle. Par exemple, modifier un classificateur d'images qui identifie les espèces de chats pour ensuite identifier les races de chiens serait bien plus efficace que de le modifier pour identifier les accents des gens.

De nombreux modèles pré-entraînés sont disponibles, notamment auprès de chercheurs universitaires. Parmi ces modèles pré-entraînés, citons les modèles ImageNet [R14] tels que Inception, VGG, AlexNet et MobileNet pour la classification d'images et les modèles en traitement de la langue naturelle (NLP – Natural Language Processing) pré-entraînés tels que BERT de Google [R15].

1.8.3 Risques liés à l'utilisation de modèles pré-entraînés et à l'apprentissage par transfert.

L'utilisation de modèles pré-entraînés et l'apprentissage par transfert sont deux approches courantes de la création de systèmes basés sur l'IA, mais elles comportent certains risques. En voici quelques-uns :

- Un modèle pré-entraîné peut manquer de transparence par rapport à un modèle généré en interne.
- Le niveau de similitude entre la fonction exécutée par le modèle pré-entraîné et la fonctionnalité requise peut être insuffisant. De plus, cette différence peut ne pas être comprise par l'expert en analyse de données pour l'IA (data scientist).

- Les différences entre les étapes de préparation des données (voir section 4.1) utilisées pour le modèle pré-entraîné lors de son développement initial et les étapes de préparation des données utilisées lorsque ce modèle est ensuite utilisé dans un nouveau système peuvent avoir un impact sur la performance fonctionnelle résultante.
- Les défauts d'un modèle pré-entraîné sont susceptibles d'être hérités par ceux qui le réutilisent et peuvent ne pas être documentés. Par exemple, les biais hérités (voir section 2.4) peuvent ne pas être apparents s'il y a un manque de documentation sur les données utilisées pour entraîner le modèle. De même, si le modèle pré-entraîné n'est pas largement utilisé, il est probable qu'il y ait plus de défauts inconnus (ou non documentés) et des tests plus rigoureux peuvent être nécessaires pour atténuer ce risque.
- Les modèles créés par apprentissage par transfert sont très probablement sensibles aux mêmes vulnérabilités que le modèle pré-entraîné sur lequel ils sont basés (par exemple, les attaques adverses, comme expliqué au point 9.1.1). En outre, si l'on sait qu'un système basé sur l'IA contient un modèle pré-entraîné spécifique (ou est basé sur un modèle pré-entraîné spécifique), les vulnérabilités qui lui sont associées peuvent déjà être connues des attaquants potentiels.

Il convient de noter que plusieurs des risques susmentionnés peuvent être plus facilement atténués en disposant d'une documentation complète sur le modèle pré-entraîné (voir la section 7.5).

1.9 Normes, règles et IA.

Le comité technique mixte de la IEC et de l'ISO sur les technologies de l'information (ISO/IEC JTC1) élabore des normes internationales qui contribuent à l'IA. Par exemple, un sous-comité sur l'IA (ISO/IEC JTC 1/SC42), a été créé en 2017. En outre, l'ISO/IEC JTC1/SC7, qui couvre l'ingénierie des logiciels et des systèmes, a publié un rapport technique sur le "Test des systèmes basés sur l'IA" [S01].

Des normes sur l'IA sont également publiées au niveau régional (par exemple, les normes européennes) et au niveau national.

Le règlement général sur la protection des données (RGPD), qui s'applique à l'ensemble de l'Union Européenne, est entré en vigueur en mai 2018 et fixe des obligations pour les responsables du traitement des données en ce qui concerne les données personnelles et la prise de décision automatisée [B06]. Il comprend des exigences pour évaluer et améliorer les performances fonctionnelles des systèmes d'IA, y compris l'atténuation des discriminations potentielles, et pour garantir les droits des personnes à ne pas être soumises à une prise de décision automatisée. L'aspect le plus important du RGPD du point de vue des tests est que les données personnelles (y compris les prédictions) doivent être exactes. Cela ne signifie pas que chaque prédiction faite par le système doit être exacte, mais que le système doit être suffisamment précis pour les fins auxquelles il est utilisé.

L'organisme national de normalisation allemand (DIN) a également développé le métamodèle de qualité de l'IA ([S02], [S03]).

Des normes sur l'IA sont également publiées par des organismes de secteurs professionnels. Par exemple, l'Institute of Electrical and Electronics Engineers (IEEE) travaille sur une série de normes sur l'éthique et l'IA (The IEEE Global Initiative for Ethical Considerations in Artificial Intelligence and Autonomous Systems). Bon nombre de ces normes sont encore en cours d'élaboration au moment de la rédaction du présent document.

Lorsque l'IA est utilisée dans des systèmes liés à la sécurité, les normes réglementaires pertinentes sont applicables, comme les normes ISO 26262 [S04] et ISO/PAS 21448 (SOTIF) [S05] pour les systèmes

automobiles. Ces normes réglementaires sont généralement imposées par des organismes gouvernementaux, et il serait illégal de vendre une voiture dans certains pays si le logiciel inclus n'était pas conforme à la norme ISO 26262. Les normes en tant que telles sont des documents de cadrage non-obligatoire sur le plan légal, et leur utilisation n'est normalement rendue obligatoire que par une législation ou un contrat. Cependant, de nombreux utilisateurs de normes le font pour bénéficier de l'expertise des auteurs et pour créer des produits de meilleure qualité.

2 Caractéristiques de qualité des systèmes basés sur l'IA – 105 minutes

Mots clés

Aucun

Mots clés spécifiques à l'IA

Adaptabilité, biais algorithmique, autonomie, biais, évolution, explicabilité, IA explicable (XAI), flexibilité, biais inapproprié, interprétabilité, système ML, machine learning, piratage de récompense (reward hacking), robustesse, biais d'échantillon, système d'auto-apprentissage, effets secondaires, transparence.

Objectifs d'apprentissage pour le chapitre 2 :

2.1 Flexibilité et adaptabilité

AI-2.1.1 K2 Expliquer l'importance de la flexibilité et de l'adaptabilité comme caractéristiques des systèmes basés sur l'IA.

2.2 Autonomie

AI-2.2.1 K2 Expliquer la relation entre l'autonomie et les systèmes basés sur l'IA.

2.3 Evolution

AI-2.3.1 K2 Expliquez l'importance de la gestion de l'évolution des systèmes basés sur l'IA.

2.4 Biais

AI-2.4.1 K2 Décrire les différentes causes et types de biais que l'on trouve dans les systèmes basés sur l'IA.

2.5 Ethique

AI-2.5.1 K2 Discuter des principes éthiques à respecter lors du développement, du déploiement et de l'utilisation de systèmes basés sur l'IA.

2.6 Effets secondaires et piratage de récompense

AI-2.6.1 K2 Expliquer l'apparition d'effets secondaires et de piratage de récompense dans les systèmes basés sur l'IA.

2.7 Transparence, interprétabilité et explicabilité

AI-2.7.1 K2 Expliquer comment la transparence, l'interprétabilité et l'explicabilité s'appliquent aux systèmes basés sur l'IA.

2.8 Sûreté et IA

AI-2.8.1 K1 Rappeler les caractéristiques qui rendent difficile l'utilisation de systèmes basés sur l'IA dans les applications liées à la sûreté.

2.1 Flexibilité and adaptabilité.

La flexibilité et l'adaptabilité sont des caractéristiques de qualité étroitement liées. Dans ce syllabus, la flexibilité est considérée comme la capacité du système à être utilisé dans des situations qui ne faisaient pas partie des exigences initiales du système, tandis que l'adaptabilité est considérée comme la facilité avec laquelle le système peut être modifié pour de nouvelles situations, comme un matériel différent et des environnements opérationnels changeants.

La flexibilité et l'adaptabilité sont toutes deux utiles si :

- L'environnement opérationnel n'est pas entièrement connu lorsque le système est déployé.
- Le système est censé faire face à de nouveaux environnements opérationnels.
- Le système est censé s'adapter à de nouvelles situations.
- Le système doit déterminer quand il doit modifier son comportement.

Les systèmes basés sur l'IA et l'auto-apprentissage sont censés présenter toutes les caractéristiques ci-dessus. Par conséquent, ils doivent être adaptables et avoir le potentiel d'être flexibles.

Les exigences de flexibilité et d'adaptabilité d'un système basé sur l'IA doivent inclure des détails sur les changements d'environnement auxquels le système doit s'adapter. Ces exigences doivent également préciser les contraintes relatives au temps et aux ressources que le système peut utiliser pour s'adapter (par exemple, combien de temps faut-il pour s'adapter à la reconnaissance d'un nouveau type d'objet...).

2.2 Autonomie

Lorsqu'on définit l'autonomie, il est important de reconnaître d'abord qu'un système entièrement autonome serait complètement indépendant de la surveillance et du contrôle humains. En pratique, l'autonomie complète n'est pas souvent souhaitée. Par exemple, les voitures entièrement autonomes, qui sont communément appelées "autonomes", sont officiellement classées comme ayant une "automatisation complète de la conduite" [B07].

Nombreux sont ceux qui considèrent que les systèmes autonomes sont "intelligents", ce qui suggère qu'ils comprennent des composants basés sur l'IA pour exécuter certaines fonctions. Par exemple, les véhicules autonomes qui doivent être conscients de la situation utilisent généralement plusieurs capteurs et le traitement d'images pour recueillir des informations sur l'environnement immédiat du véhicule. L'apprentissage automatique, et en particulier le deep learning (voir section 6.1), s'est avéré être l'approche la plus efficace pour remplir cette fonction. Les systèmes autonomes peuvent également inclure des fonctions de prise de décision et de contrôle. Ces deux fonctions peuvent être exécutées efficacement à l'aide de composants basés sur l'IA.

Même si certains systèmes basés sur l'IA sont considérés comme autonomes, cela ne s'applique pas à tous les systèmes basés sur l'IA. Dans ce syllabus, l'autonomie est considérée comme la capacité du système à fonctionner indépendamment de la surveillance et du contrôle humains pendant des périodes prolongées. Cela peut aider à identifier les caractéristiques d'un système autonome qui doivent être spécifiées et testées. Par exemple, la durée pendant laquelle un système autonome est censé fonctionner de manière satisfaisante sans intervention humaine doit être connue. En outre, il est important d'identifier les événements pour lesquels le système autonome doit rendre le contrôle à ses contrôleurs humains.

2.3 Evolution

Dans ce syllabus, l'évolution est considérée comme la capacité du système à s'améliorer en réponse à des contraintes externes changeantes. Certains systèmes d'IA peuvent être décrits comme étant auto-apprenants et les systèmes d'IA auto-apprenants réussis doivent intégrer cette forme d'évolution.

Les systèmes basés sur l'IA fonctionnent souvent dans un environnement en évolution. Comme pour les autres formes de systèmes informatiques, un système basé sur l'IA doit être suffisamment souple et adaptable pour faire face aux changements de son environnement opérationnel.

Les systèmes d'IA à apprentissage automatique doivent généralement gérer deux formes de changement :

- La première forme de changement est celle où le système apprend de ses propres décisions et de ses interactions avec son environnement.
- L'autre forme de changement est celle où le système apprend des changements apportés à son environnement opérationnel.

Dans les deux cas, le système évoluera idéalement pour améliorer son efficacité et son efficience. Toutefois, cette évolution doit être limitée pour éviter que le système ne développe des caractéristiques indésirables. Toute évolution doit continuer à répondre aux exigences et aux contraintes initiales du système. Si celles-ci font défaut, le système doit être géré de manière à ce que toute évolution reste dans les limites et qu'elle soit toujours conforme aux valeurs humaines. La section 2.6 fournit des exemples relatifs à l'impact des effets secondaires et du piratage de récompense sur les systèmes d'IA à apprentissage automatique.

2.4 Biais

Dans le contexte des systèmes basés sur l'IA, le biais est une mesure statistique de la distance entre les résultats fournis par le système et ce qui est considéré comme des "résultats équitables" qui ne montrent aucun favoritisme envers un groupe particulier. Les préjugés inappropriés peuvent être liés à des attributs tels que le sexe, la race, l'origine ethnique, l'orientation sexuelle, le niveau de revenu et l'âge. Des cas de biais inappropriés dans des systèmes basés sur l'IA ont été signalés, par exemple, dans des systèmes utilisés pour faire des recommandations en matière de prêts bancaires, dans des systèmes de recrutement et dans des systèmes de contrôle judiciaire.

Les préjugés peuvent être introduits dans de nombreux types de systèmes basés sur l'IA. Par exemple, il est difficile d'empêcher que le parti pris des experts soit intégré aux règles appliquées par un système expert. Cependant, la prévalence des systèmes ML signifie qu'une grande partie de la discussion relative à la partialité a lieu dans le contexte de ces systèmes.

Les systèmes ML sont utilisés pour prendre des décisions et faire des prédictions, à l'aide d'algorithmes qui utilisent des données collectées, et ces deux composants peuvent introduire un biais dans les résultats :

- Le biais algorithmique peut se produire lorsque l'algorithme d'apprentissage est mal configuré, par exemple lorsqu'il surévalue certaines données par rapport à d'autres. Cette source de biais peut être causée et gérée par le réglage des hyperparamètres des algorithmes ML (voir section 3.2).

- Un biais d'échantillon peut se produire lorsque les données d'apprentissage ne sont pas entièrement représentatives de l'espace de données auquel ML est appliqué.

Un biais inapproprié est souvent causé par un biais d'échantillon, mais occasionnellement il peut aussi être causé par un biais algorithmique.

2.5 Ethique

L'éthique est définie dans le dictionnaire de Cambridge comme :

> *« Un système de croyances acceptées qui contrôlent le comportement, en général basé sur la valeur morale »*.

Les systèmes basés sur l'IA et dotés de capacités accrues ont un effet largement positif sur la vie des gens. La généralisation de ces systèmes a suscité des inquiétudes quant à leur utilisation éthique. Ce qui est considéré comme éthique peut évoluer dans le temps et peut également changer selon les localités et les cultures. Il faut veiller à ce que le déploiement d'un système basé sur l'IA d'un endroit à un autre tienne compte des différences de valeurs des parties prenantes.
On trouve des politiques nationales et internationales sur l'éthique de l'IA dans de nombreux pays et régions. L'Organisation de coopération et de développement économiques a publié ses principes pour l'IA, les premières normes internationales convenues par les gouvernements pour le développement responsable de l'IA, en 2019 [B08]. Ces principes ont été adoptés par quarante-deux pays lors de leur publication et sont également soutenus par la Commission européenne. Ils comprennent des recommandations politiques pratiques ainsi que des principes fondés sur des valeurs pour une " gestion responsable de l'IA digne de confiance ". Ces principes sont résumés comme suit :

- L'IA devrait profiter aux personnes et à la planète en favorisant la croissance inclusive, le développement durable et le bien-être.
- Les systèmes d'IA devraient respecter l'état de droit, les droits de l'homme, les valeurs démocratiques et la diversité, et devraient inclure des garanties appropriées pour assurer une société équitable.
- L'IA doit être transparente afin que les gens comprennent les résultats et puissent les contester.
- Les systèmes d'IA doivent fonctionner de manière robuste, sûre et sécurisée tout au long de leur cycle de vie et les risques doivent être évalués en permanence.
- Les organisations et les individus qui développent, déploient ou exploitent des systèmes d'IA doivent être tenus pour responsables.

2.6 Effets secondaires et Piratage de récompense

Les effets secondaires et le piratage de récompense peuvent amener les systèmes basés sur l'IA à générer des résultats inattendus, voire nuisibles, lorsque le système tente d'atteindre ses objectifs [B09].

Des effets secondaires négatifs peuvent se produire lorsque le concepteur d'un système basé sur l'IA spécifie un objectif qui "se concentre sur l'accomplissement de certaines tâches spécifiques dans l'environnement mais ignore d'autres aspects de l'environnement (potentiellement très vaste), et exprime donc implicitement une indifférence à l'égard de variables environnementales qu'il pourrait, en fait, être nuisibles de modifier" [B09]. Par exemple, une voiture autonome dont l'objectif est de se rendre à destination "de la manière la plus efficace possible en termes de consommation de carburant et de

sécurité" peut atteindre cet objectif, mais avec l'effet secondaire que les passagers deviennent extrêmement ennuyés par le temps excessif qu'ils prennent.

Le piratage de récompense peut résulter du fait qu'un système basé sur l'IA atteint un objectif spécifique en utilisant une solution "intelligente" ou "facile" qui "pervertit l'esprit de l'intention du concepteur". En fait, l'objectif peut être détourné. Un exemple largement utilisé de piratage de récompense est celui d'un système basé sur l'IA qui apprend à jouer à un jeu d'arcade. On lui présente l'objectif de réaliser le "meilleur score" et, pour ce faire, il pirate simplement l'enregistrement de données qui stocke le meilleur score, au lieu de jouer au jeu pour l'atteindre.

2.7 Transparence, interprétabilité et explicabilité

Les systèmes basés sur l'IA sont généralement appliqués dans des domaines où les utilisateurs doivent avoir confiance en ces systèmes. Cela peut être pour des raisons de sûreté, mais aussi par respect de la vie privée et lorsqu'ils peuvent fournir des prédictions et des décisions susceptibles de changer leur vie.

La plupart des utilisateurs considèrent les systèmes basés sur l'IA comme des "boîtes noires" et ne savent pas vraiment comment ces systèmes arrivent à leurs résultats. Dans certains cas, cette ignorance peut même s'appliquer aux ingénieurs IA qui ont construit les systèmes. Parfois, les utilisateurs ne sont même pas conscients qu'ils interagissent avec un système basé sur l'IA.

La complexité inhérente aux systèmes basés sur l'IA a donné naissance au domaine de "l'IA explicable" (XAI – eXplainable Artificial Intelligence). L'objectif de l'IA explicable est de permettre aux utilisateurs de comprendre comment les systèmes basés sur l'IA parviennent à leurs résultats, ce qui augmente la confiance des utilisateurs à leur égard.

Selon la Royal Society [B10], il y a plusieurs raisons de vouloir l'XAI, notamment :

- Donner confiance aux utilisateurs dans le système.
- Se protéger contre les préjugés.
- Répondre aux normes réglementaires ou aux exigences politiques.
- Améliorer la conception du système.
- Évaluer les risques, la robustesse et la vulnérabilité.
- Comprendre et vérifier les résultats d'un système.
- L'autonomie, le pouvoir d'action (donner à l'utilisateur le sentiment de pouvoir agir) et le respect des valeurs sociales.

Cela conduit aux trois caractéristiques XAI de base, souhaitables pour les systèmes basés sur l'IA du point de vue d'une partie prenante (voir également la section 8.6) :

- Transparence : Il s'agit de la facilité avec laquelle l'algorithme et les données d'apprentissage utilisés pour générer le modèle peuvent être déterminés.
- Interprétabilité : Il s'agit de la compréhensibilité de la technologie d'IA par les différentes parties prenantes, y compris les utilisateurs.
- Explicabilité : Il s'agit de la facilité avec laquelle les utilisateurs peuvent déterminer comment le système basé sur l'IA aboutit à un résultat particulier.

2.8 Sûreté et IA

Dans ce syllabus, la sûreté est considérée comme l'attente qu'un système basé sur l'IA ne causera pas de dommages aux personnes, aux biens ou à l'environnement. Les systèmes basés sur l'IA peuvent être utilisés pour prendre des décisions qui affectent la sûreté. Par exemple, les systèmes basés sur l'IA qui travaillent dans les domaines de la médecine, de la fabrication industrielle, de la défense, de la sécurité et des transports ont le potentiel d'affecter la sûreté.

Les caractéristiques des systèmes basés sur l'IA qui font qu'il est plus difficile de s'assurer qu'ils sont sûrs (par exemple, qu'ils ne nuisent pas aux humains) sont les suivantes :

- La complexité
- Le non-déterminisme
- La nature probabiliste
- L'auto-apprentissage
- Le manque de transparence, d'interprétabilité et d'explicabilité
- Le manque de robustesse

Les défis liés au test de plusieurs de ces caractéristiques sont abordés au chapitre 8.

3 Machine Learning (ML) – Aperçu - 145 minutes

Mots-clés

Aucun

Mots-clés spécifiques à l'IA

Association, classification, clustering, préparation des données, algorithme ML, framework ML, critères de performance fonctionnelle ML, modèle ML, données d'apprentissage ML, workflow ML, évaluation du modèle, ajustement du modèle, aberration, surajustement (overfitting), régression, apprentissage par renforcement, apprentissage supervisé, sous-ajustement (underfitting), apprentissage non supervisé.

Objectifs d'apprentissage pour le chapitre 3 :

3.1 Formes de ML

AI-3.1.1 K2 Décrire la classification et la régression dans le cadre de l'apprentissage supervisé.

AI-3.1.2 K2 Décrire le regroupement et l'association dans le cadre de l'apprentissage non supervisé.

AI-3.1.3 K2 Décrire l'apprentissage par renforcement.

3.2 Workflow ML

AI-3.2.1 K2 Résumez le workflow utilisé pour créer un système ML.

3.3 Sélectionner une forme de ML

AI-3.3.1 K3 Compte tenu d'un scénario de projet, identifiez une forme appropriée de ML (classification, régression, regroupement (clustering), association ou apprentissage par renforcement).

3.4 Facteurs impliqués dans la sélection d'un algorithme ML

AI-3.4.1 K2 Expliquer les facteurs impliqués dans la sélection des algorithmes ML.

3.5 **Surajustement et sous-ajustement**.

AI-3.5.1 K2 Résumer les concepts de sous-ajustement et de surajustement.
HO-3.5.1 H0 Démontrer le sous-ajustement et le surajustement.

3.1 Formes de ML

Les algorithmes ML peuvent être catégorisés comme suit :

- Apprentissage supervisé,
- Apprentissage non supervisé, et
- Apprentissage par renforcement.

3.1.1 Apprentissage supervisé.

Dans ce type d'apprentissage, l'algorithme crée le modèle ML à partir de données étiquetées pendant la phase d'apprentissage. Les données étiquetées, qui comprennent généralement des paires d'entrées (par exemple, une image d'un chien et l'étiquette "chien"), sont utilisées par l'algorithme pour déduire la relation entre les données d'entrée (par exemple, des images de chiens) et les étiquettes de sortie (par exemple, "chien" et "chat") pendant l'apprentissage. Pendant la phase de test du modèle ML, un nouvel ensemble de données non vues est appliqué au modèle formé pour prédire la sortie. Le modèle est déployé lorsque le niveau de précision de la sortie est satisfaisant.

Les problèmes résolus par l'apprentissage supervisé sont divisés en deux catégories :

- La classification : La classification est utilisée lorsque le problème nécessite qu'une entrée soit classée dans l'une des quelques classes prédéfinies. La reconnaissance des visages ou la détection d'objets dans une image sont des exemples de problèmes qui utilisent la classification.
- La régression : C'est lorsque le problème nécessite que le modèle ML prédise une sortie numérique en utilisant la régression. La prédiction de l'âge d'une personne à partir de données d'entrée sur ses habitudes ou la prédiction de la future valeur boursière des actions sont des exemples de problèmes qui utilisent la régression.

Notez que le terme régression, tel qu'il est utilisé dans le contexte d'un problème de ML, est différent de son utilisation dans d'autres syllabus de l'ISTQB®, comme [I01], où la régression est utilisée pour décrire le problème des modifications de logiciels causant des défauts liés aux changements.

3.1.2 Apprentissage non supervisé.

Dans ce type d'apprentissage, l'algorithme crée le modèle ML à partir de données non étiquetées pendant la phase d'apprentissage. Les données non étiquetées sont utilisées par l'algorithme pour déduire des modèles dans les données d'entrée pendant l'apprentissage et affecter les entrées à différentes classes, en fonction de leurs points communs. Pendant la phase de test, le modèle formé est appliqué à un nouvel ensemble de données non identifiées pour prédire à quelles classes les données d'entrée doivent être affectées. Le modèle est déployé lorsque le niveau de précision des résultats est jugé satisfaisant.

Les problèmes résolus par l'apprentissage non supervisé sont divisés en deux catégories :

- Le clustering (regroupement) : Il s'agit du cas où le problème nécessite l'identification de similitudes entre les points de données d'entrée, ce qui permet de les regrouper en fonction de caractéristiques ou d'attributs communs. Par exemple, le clustering est utilisé pour catégoriser différents types de clients à des fins de marketing.

- L'association : Il s'agit du cas où le problème nécessite l'identification de relations ou de dépendances intéressantes entre les attributs des données. Par exemple, un système de recommandation de produits peut identifier des associations basées sur le comportement d'achat des clients.

3.1.3 Apprentissage par renforcement.

L'apprentissage par renforcement est une approche dans laquelle le système (un "agent intelligent") apprend en interagissant avec l'environnement de manière itérative et tire ainsi des enseignements de son expérience. L'apprentissage par renforcement n'utilise pas de données d'apprentissage. L'agent est récompensé lorsqu'il prend une décision correcte et pénalisé lorsqu'il prend une décision incorrecte.

La mise en place de l'environnement, le choix de la bonne stratégie pour que l'agent atteigne l'objectif souhaité et la conception d'une fonction de récompense sont des défis majeurs lors de la mise en œuvre de l'apprentissage par renforcement. La robotique, les véhicules autonomes et les chatbots sont des exemples d'applications qui utilisent l'apprentissage par renforcement.

3.2 Workflow ML

Les activités du workflow de machine learning sont les suivantes :

Comprendre les objectifs

L'objectif du modèle ML à déployer doit être compris et convenu avec les parties prenantes pour assurer l'alignement avec les professionnels du métier. Les critères d'acceptation (y compris les métriques de performance fonctionnelle du ML - voir le chapitre 5) doivent être définis pour le modèle développé.

Choisir un framework

Un framework de développement d'IA approprié doit être choisi en fonction des objectifs, des critères d'acceptation et des professionnels du métier (voir section 1.5).

Sélectionner et construire l'algorithme

Un algorithme d'intelligence artificielle est sélectionné en fonction de divers facteurs, notamment les objectifs, les critères d'acceptation et les données disponibles (voir section 3.4). L'algorithme peut être codé manuellement, mais il est souvent extrait d'une bibliothèque de codes pré-écrits. L'algorithme est ensuite compilé pour préparer l'apprentissage du modèle, si nécessaire.

Préparer et tester les données

La préparation des données (voir section 4.1) comprend l'acquisition des données, le prétraitement des données et l'ingénierie des caractéristiques. Une analyse exploratoire des données (EDA) peut être effectuée parallèlement à ces activités.

Les données utilisées par l'algorithme et le modèle seront basées sur les objectifs et sont utilisées par toutes les activités de la phase "génération et test du modèle" présentée sur la figure 1. Par exemple, si le système est un système de transaction boursière en temps réel, les données proviendront du marché boursier.

Les données utilisées pour entraîner, régler et tester le modèle doivent être représentatives des données opérationnelles qui seront utilisées par le modèle. Dans certains cas, il est possible d'utiliser des ensembles de données pré-collectées pour l'apprentissage initial du modèle (par

exemple, voir les ensembles de données Kaggle [R16]). Sinon, les données brutes nécessitent généralement un prétraitement et une ingénierie des caractéristiques.

Le test des données et de toute étape de préparation automatisée des données doit être effectué. Voir la section 7.2.1 pour plus de détails sur le test des données d'entrée.

Entraîner le modèle

L'algorithme ML sélectionné utilise des données d'apprentissage pour former le modèle.

Certains algorithmes, tels que ceux générant un réseau neuronal, lisent l'ensemble de données d'apprentissage plusieurs fois. Chaque itération d'apprentissage sur l'ensemble de données d'apprentissage est appelée une époque.

Les paramètres définissant la structure du modèle (par exemple, le nombre de couches d'un réseau neuronal ou la profondeur d'un arbre de décision) sont transmis à l'algorithme. Ces paramètres sont appelés hyperparamètres du modèle.

Les paramètres qui contrôlent l'apprentissage (par exemple, le nombre d'époques à utiliser pour l'entraînement d'un réseau neuronal) sont également transmis à l'algorithme. Ces paramètres sont appelés hyperparamètres de l'algorithme.

Évaluer le modèle

Le modèle est évalué par rapport aux métriques de performance fonctionnelles convenues du ML, en utilisant l'ensemble de données de validation, et les résultats sont ensuite utilisés pour améliorer (régler) le modèle. L'évaluation et la mise au point du modèle doivent ressembler à une expérience scientifique qui doit être soigneusement menée dans des conditions contrôlées, avec une documentation claire. En pratique, plusieurs modèles sont généralement créés et entraînés à l'aide de différents algorithmes (forêts aléatoires, SVM - support-vector machines - et réseaux neuronaux, par exemple), et le meilleur est choisi en fonction des résultats de l'évaluation et du réglage.

Ajuster le modèle

Les résultats de l'évaluation du modèle par rapport aux métriques de performance fonctionnelle convenues sont utilisés pour ajuster les paramètres du modèle aux données et améliorer ainsi ses performances. Le modèle peut être ajusté par réglage des hyperparamètres, où l'activité d'apprentissage est modifiée (par exemple, en changeant le nombre d'étapes d'apprentissage ou la quantité de données utilisées pour l'apprentissage), ou les attributs du modèle sont mis à jour (par exemple, le nombre de neurones dans un réseau neuronal ou la profondeur d'un arbre de décision).

Les trois activités d'apprentissage, d'évaluation et de réglage peuvent être considérées comme faisant partie de la génération du modèle, comme le montre la figure 1.

Tester le modèle

Une fois qu'un modèle a été généré (c'est-à-dire qu'il a été formé, évalué et réglé), il doit être testé par rapport à un ensemble de données de test indépendant pour s'assurer que les critères de performance fonctionnelle du ML sont respectés (voir section 7.2.2). Les mesures de performance fonctionnelle issues des tests sont également comparées à celles de l'évaluation, et si la performance du modèle avec des données indépendantes est significativement inférieure à celle de l'évaluation, il peut être nécessaire de sélectionner un autre modèle.

En plus des tests de performance fonctionnelle, des tests non fonctionnels, tels que le temps d'apprentissage du modèle, le temps et l'utilisation des ressources nécessaires pour fournir une prédiction, doivent également être effectués. En général, ces tests sont effectués par le data engineer / scientist, mais des testeurs ayant une connaissance suffisante du domaine et un accès aux ressources pertinentes peuvent également effectuer ces tests.

Déployer le modèle

Une fois le développement du modèle terminé, comme le montre la figure 1, le modèle mis au point doit généralement être réorganisé en vue de son déploiement avec les ressources associées, y compris le pipeline de données pertinent. Ceci est normalement réalisé par le biais du framework. Les cibles peuvent inclure les systèmes embarqués et le cloud, où le modèle est accessible via une API Web.

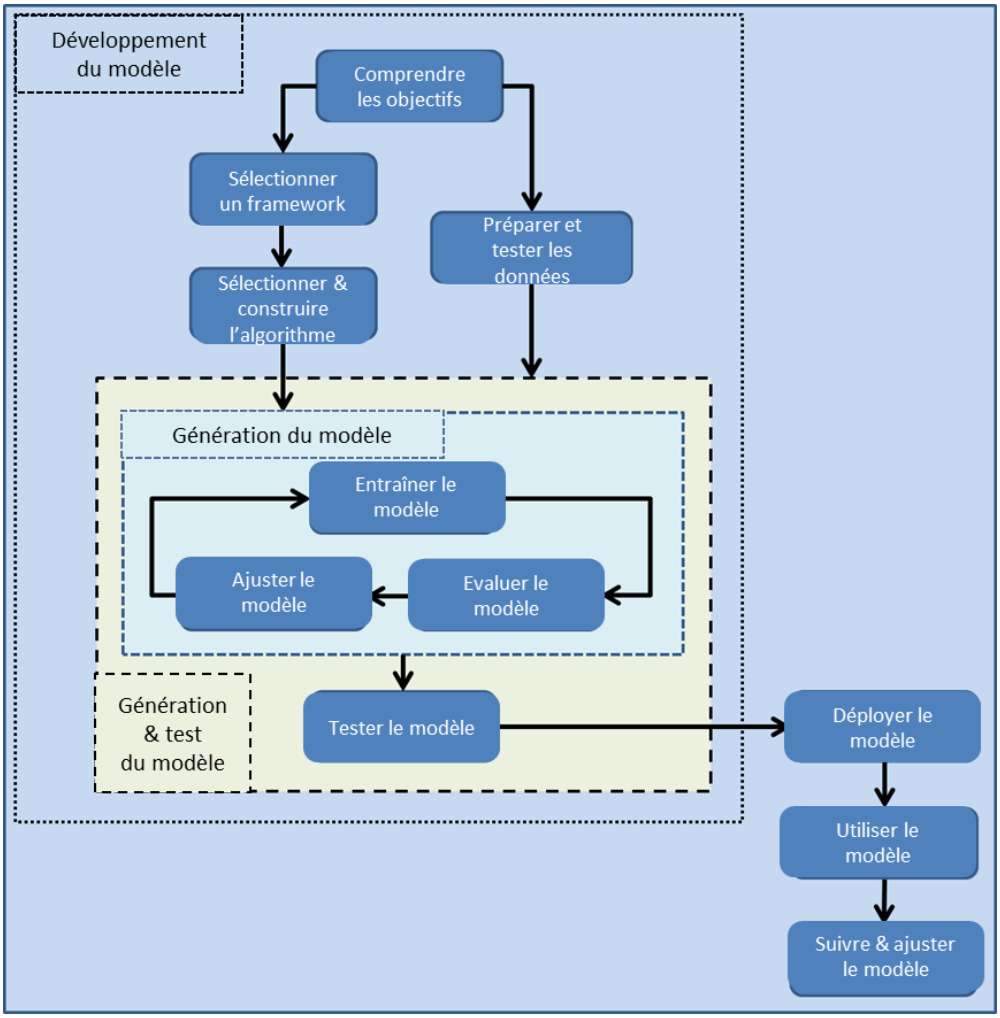

Figure 1: Workflow ML

Utiliser le modèle

Une fois déployé, le modèle fait généralement partie d'un système d'IA plus vaste et peut être utilisé de manière opérationnelle. Les modèles peuvent effectuer des prédictions programmées par lots à des intervalles de temps définis ou s'exécuter sur demande en temps réel.

Surveiller et ajuster le modèle

Pendant l'utilisation du modèle, la situation peut évoluer et le modèle peut s'éloigner des performances prévues (voir les sections 2.3 et 7.6). Pour s'assurer que toute dérive est identifiée et gérée, le modèle opérationnel doit être régulièrement évalué par rapport à ses critères d'acceptation.

Il peut être jugé nécessaire de mettre à jour les paramètres du modèle pour résoudre le problème de dérive ou il peut être décidé qu'un réapprentissage avec de nouvelles données est nécessaire pour créer un modèle plus précis ou plus robuste. Dans ce cas, un nouveau modèle peut être créé et entraîné avec des données d'apprentissage mises à jour. Le nouveau modèle peut alors être comparé au modèle existant en utilisant une forme de test A/B (voir section 9.4).

Le workflow de ML présenté dans la figure 1 est une séquence logique. Dans la pratique, le workflow est appliqué d'une manière où les étapes sont répétées de façon itérative (par exemple, lorsque le modèle est évalué, il est souvent nécessaire de revenir à l'étape de l'apprentissage, et parfois à la préparation des données).

Les étapes présentées dans la figure 1 ne comprennent pas l'intégration du modèle ML avec les parties non-ML du système global. En général, les modèles ML ne peuvent pas être déployés de manière isolée et doivent être intégrés avec les parties non-ML. Par exemple, dans les applications de vision, il existe un pipeline de données qui nettoie et modifie les données avant de les soumettre au modèle ML. Lorsque le modèle fait partie d'un système d'IA plus vaste, il devra être intégré à ce système avant son déploiement. Dans ce cas, des tests d'intégration, de système et d'acceptation peuvent être effectués, comme décrit dans la section 7.2.

3.3 Sélectionner une forme de ML

Lors de la sélection d'une approche ML appropriée, les directives suivantes s'appliquent :

- Il doit y avoir suffisamment de données d'apprentissage et de test disponibles pour l'approche ML sélectionnée.
- Pour l'apprentissage supervisé, il est nécessaire d'avoir des données correctement étiquetées.
- S'il existe une étiquette de sortie, il peut s'agir d'un apprentissage supervisé.
- Si la sortie est discrète et catégorielle, il peut s'agir d'une classification.
- Si la sortie est numérique et continue par nature, il peut s'agir d'une régression.
- Si aucune sortie n'est fournie dans l'ensemble de données donné, il peut s'agir d'un apprentissage non supervisé.
- Si le problème consiste à regrouper des données similaires, il peut s'agir de clustering.

- Si le problème consiste à trouver des éléments de données co-occurrents, il peut s'agir d'association.
- L'apprentissage par renforcement est mieux adapté aux contextes dans lesquels il y a une interaction avec l'environnement.
- Si le problème fait intervenir la notion d'états multiples et implique des décisions à chaque état, l'apprentissage par renforcement peut être applicable.

3.4 Facteurs impliqués dans la sélection de l'algorithme ML

Il n'existe pas d'approche définitive pour sélectionner l'algorithme ML optimal, les paramètres du modèle ML et les hyperparamètres du modèle ML. En pratique, cet ensemble est choisi en fonction d'une combinaison des facteurs suivants :

- La fonctionnalité requise (par exemple, si la fonctionnalité est la classification ou la prédiction d'une valeur discrète).
- Les caractéristiques de qualité requises, telles que
  - La précision (par exemple, certains modèles peuvent être plus précis, mais plus lents)
  - Les contraintes sur la mémoire disponible (par exemple, pour un système embarqué)
  - La vitesse d'apprentissage (et de réapprentissage) du modèle
  - La vitesse de prédiction (par exemple, pour les systèmes en temps réel)
  - Les exigences de transparence, d'interprétabilité et d'explicabilité
- Le type de données disponibles pour l'apprentissage du modèle (par exemple, certains modèles ne peuvent fonctionner qu'avec des données d'image).
- La quantité de données disponibles pour l'apprentissage et le test du modèle (certains modèles peuvent, par exemple, avoir tendance à se surajuster avec une quantité limitée de données, à un degré plus élevé que d'autres modèles)
- Le nombre de caractéristiques des données d'entrée que le modèle est censé utiliser (d'autres facteurs, tels que la vitesse et la précision, sont susceptibles d'être directement affectés par le nombre de caractéristiques)
- Le nombre prévu de classes pour le regroupement (par exemple, certains modèles peuvent ne pas convenir aux problèmes comportant plus d'une classe).
- Choix en fonction de l'expérience antérieure.
- Choix par essais successifs.

3.5 Surajustement et sous-ajustement

3.5.1 Surajustement

Le surajustement se produit lorsque le modèle s'adapte trop étroitement à un ensemble de points de données et ne parvient pas à généraliser correctement. Un tel modèle fonctionne très bien avec les

données utilisées pour le former mais peut avoir du mal à fournir des prédictions précises pour de nouvelles données. Le surajustement peut se produire lorsque le modèle essaie de s'adapter à chaque point de données, y compris les points de données qui peuvent être décrits comme du bruit ou des valeurs aberrantes. Il peut également se produire lorsque les données fournies dans l'ensemble de données d'apprentissage sont insuffisantes.

3.5.2 Sous-ajustement

Le sous-ajustement se produit lorsque le modèle n'est pas suffisamment sophistiqué pour s'adapter avec précision aux modèles des données d'apprentissage. Les modèles sous-ajustés ont tendance à être trop simplistes et peuvent avoir du mal à fournir des prédictions précises pour les nouvelles données et les données très similaires aux données d'apprentissage. Une cause de sous-ajustement peut être un ensemble de données d'apprentissage qui ne contient pas de caractéristiques reflétant des relations importantes entre les entrées et les sorties. Cela peut également se produire lorsque l'algorithme ne s'adapte pas correctement aux données (par exemple, en créant un modèle linéaire pour des données non linéaires).

3.5.3 Exercice pratique : Démonstration de surajustement et de sous-ajustement.

Démontrez les concepts de surajustement et de sous-ajustement d'un modèle. Pour ce faire, vous pouvez utiliser un ensemble de données contenant très peu de données (surajustement) et un ensemble de données présentant de faibles corrélations entre les caractéristiques (sous-ajustement).

4 ML - Data – 230 minutes

Mots-clés

Aucun

Mots-clés spécifiques à l'IA

Annotation, augmentation, modèle de classification, étiquetage des données, préparation des données, données d'apprentissage ML, apprentissage supervisé, ensemble de données de test, ensemble de données de validation.

Objectifs d'apprentissage pour le chapitre 4 :

4.1 Préparation des données dans le cadre du workflow ML.

AI-4.1.1 K2 Décrire les activités et les défis liés à la préparation des données.

HO-4.1.1 H2 Effectuer la préparation des données en vue de la création d'un modèle ML.

4.2 Ensembles de données d'apprentissage, de validation et de test dans le workflow ML

AI-4.2.1 K2 Décrire et comparer l'utilisation des ensembles de données d'apprentissage, de validation et de test dans le développement d'un modèle ML.

HO-4.2.1 H2 Identifier les ensembles de données d'apprentissage et de test et créer un modèle ML.

4.3 Problèmes de qualité des données.

AI-4.3.1 K2 Décrire les problèmes typiques de qualité des ensembles de données.

4.4 La qualité des données et son effet sur le modèle ML.

AI-4.4.1 K2 Reconnaître comment une mauvaise qualité des données peut causer des problèmes avec le modèle ML résultant.

4.5 Étiquetage des données pour l'apprentissage supervisé.

AI-4.5.1 K1 Rappeler les différentes approches de l'étiquetage des données dans les ensembles de données pour l'apprentissage supervisé.

AI-4.5.2 K1 Rappeler les raisons pour lesquelles les données des ensembles de données sont mal étiquetées.

4.1 Préparation des données dans le cadre du workflow ML.

La préparation des données représente en moyenne 43% de l'effort du workflow ML et est probablement l'activité la plus gourmande en ressources du workflow ML. En comparaison, la sélection et la construction de modèles ne représentent que 17% [R17]. La préparation des données fait partie du pipeline de données, qui reçoit les données brutes et les restitue sous une forme qui peut être utilisée à la fois pour former un modèle ML et pour la prédiction par un modèle ML formé.

La préparation des données peut être considérée comme comprenant les activités suivantes :

Acquisition des données

- Identification : Les types de données à utiliser pour l'apprentissage et les prédictions sont identifiés. Par exemple, dans le cas d'une voiture à conduite autonome, cela peut inclure l'identification des besoins en données radar, vidéo et d'imagerie laser, de détection et de télémétrie (LiDAR - Light Detection And Ranging).
- Collecte : La source des données est identifiée et les moyens de collecte des données sont déterminés. Par exemple, cela pourrait inclure l'identification du Fonds monétaire international (FMI) comme source de données financières et les canaux qui seront utilisés pour soumettre les données au système basé sur l'IA.
- Étiquetage : Voir la section 4.5.

Les données acquises peuvent se présenter sous diverses formes (par exemple, numérique, catégorielle, image, tableau, texte, séries chronologiques, capteur, géospatial, vidéo et audio).

Prétraitement des données

- Nettoyage : Lorsque des données incorrectes, des données en double ou des valeurs aberrantes sont identifiées, elles sont soit supprimées, soit corrigées. En outre, l'imputation des données peut être utilisée pour remplacer les valeurs de données manquantes par des valeurs estimées ou devinées (par exemple, en utilisant les valeurs moyennes, médianes et modales). La suppression ou l'anonymisation des informations personnelles peut également être effectuée.
- Transformation : Le format des données est modifié (par exemple, en décomposant une adresse présentée sous forme de chaîne en ses éléments constitutifs, en supprimant un champ contenant un identifiant aléatoire, en convertissant des données catégorielles en données numériques, en changeant le format des images). Certaines des transformations appliquées aux données numériques comprennent la mise à l'échelle afin de s'assurer que la même plage est utilisée. La normalisation, par exemple, redimensionne les données pour qu'elles prennent une moyenne de zéro et un écart-type de un. Cette normalisation garantit que les données ont une plage comprise entre zéro et un.
- Augmentation : Elle est utilisée pour augmenter le nombre d'échantillons dans un ensemble de données. L'augmentation peut également être utilisée pour inclure des exemples adverses dans les données d'apprentissage, ce qui permet de résister aux attaques adverses (voir 9.1).
- Échantillonnage : Il s'agit de la sélection d'une partie de l'ensemble des données disponibles afin de pouvoir observer des modèles dans l'ensemble des données. Cette opération est généralement effectuée pour réduire les coûts et le temps nécessaire à la création du modèle ML.

Notez que tout prétraitement comporte le risque de modifier des données valides utiles ou d'ajouter des données non valides.

Ingénierie des caractéristiques

- Sélection des caractéristiques : Une caractéristique est un attribut/propriété reflété dans les données. La sélection des caractéristiques implique la sélection des caractéristiques qui sont les plus susceptibles de contribuer à l'apprentissage et à la prédiction du modèle. En pratique, elle comprend souvent la suppression des caractéristiques qui ne sont pas censées (ou que l'on ne souhaite pas) avoir un effet sur le modèle résultant. En éliminant les informations non pertinentes (bruit), la sélection des caractéristiques peut réduire le temps d'apprentissage global, empêcher le surajustement (voir section 3.5.1), augmenter la précision et rendre les modèles plus généralisables.
- Extraction de caractéristiques : Il s'agit de dériver des caractéristiques informatives et non redondantes à partir des caractéristiques existantes. L'ensemble de données résultant est généralement plus petit et peut être utilisé pour générer un modèle ML de précision équivalente à moindre coût et plus rapidement.

Parallèlement à ces activités de préparation des données, une analyse exploratoire des données (EDA - Exploratory Data Analysis) est également effectuée pour soutenir la tâche globale de préparation des données. Il s'agit notamment d'effectuer une analyse des données pour découvrir les tendances inhérentes aux données et d'utiliser la visualisation des données pour représenter les données dans un format visuel en traçant des tendances dans les données.
Bien que les activités et sous-activités de préparation des données ci-dessus aient été présentées dans un ordre logique, différents projets peuvent les réorganiser ou n'en utiliser qu'un sous-ensemble. Certaines des étapes de préparation des données, telles que l'identification de la source de données, ne sont effectuées qu'une seule fois et peuvent être réalisées manuellement. D'autres étapes peuvent faire partie du pipeline de données opérationnel et fonctionnent normalement sur des données réelles. Ces tâches doivent être automatisées.

4.1.1 Les défis de la préparation des données.

Parmi les défis liés à la préparation des données figurent :

- La nécessité de connaître :
  - le domaine d'application,
  - les données et leurs propriétés,
  - les différentes techniques associées à la préparation des données.
- La difficulté d'obtenir des données de haute qualité à partir de sources multiples.
- La difficulté d'automatiser le pipeline de données et de s'assurer que le pipeline de données de production est à la fois évolutif et d'une efficience raisonnable (par exemple, le temps nécessaire pour terminer le traitement d'une donnée).
- Les coûts associés à la préparation des données.
- Ne pas accorder une priorité suffisante à la vérification des défauts introduits dans le pipeline de données lors de la préparation des données.

- L'introduction de biais d'échantillonnage (voir section 2.4.)

4.1.2 Exercice pratique : Préparation des données pour le ML.

Pour un ensemble donné de données brutes, effectuez les étapes de préparation des données applicables, comme indiqué à la section 4.1, afin de produire un ensemble de données qui sera utilisé pour créer un modèle de classification utilisant l'apprentissage supervisé.

Cette activité constitue la première étape de la création d'un modèle ML qui sera utilisé pour les exercices suivants.

Pour réaliser cette activité, les participants disposeront du matériel approprié (et spécifique à la langue), notamment :

- Bibliothèques,
- Frameworks ML,
- Outils,
- Un environnement de développement.

4.2 Ensembles de données d'apprentissage, de validation et de test dans le workflow du ML

Logiquement, trois ensembles de données équivalentes (c'est-à-dire sélectionnées au hasard à partir d'un seul ensemble de données initial) sont nécessaires pour développer un modèle ML :

- Un ensemble de données d'apprentissage, qui est utilisé pour entraîner le modèle.
- Un ensemble de données de validation, qui est utilisé pour évaluer et ensuite ajuster le modèle.
- Un ensemble de données de test, (également connu sous le nom d'ensemble de données de rétention), qui est utilisé pour tester le modèle ajusté.

Si un nombre illimité de données appropriées est disponible, la quantité de données utilisées dans le workflow ML pour l'apprentissage, l'évaluation et le test dépend généralement des facteurs suivants :

- L'algorithme utilisé pour entraîner le modèle.
- La disponibilité des ressources, telles que la mémoire RAM, l'espace disque, la puissance de calcul, la bande passante du réseau et le temps disponible.

En pratique, en raison du défi que représente l'acquisition d'un nombre suffisant de données appropriées, les ensembles de données d'apprentissage et de validation sont souvent dérivés d'un seul ensemble de données combiné. L'ensemble de données de test est conservé séparément et n'est pas utilisé pendant l'apprentissage Cela permet de garantir que le modèle développé n'est pas influencé par les données de test et que les résultats des tests reflètent fidèlement la qualité du modèle.

Il n'existe pas de ratio optimal pour diviser l'ensemble de données combiné en trois ensembles de données individuels, mais les ratios typiques qui peuvent être utilisés comme ligne directrice vont de 60:20:20 à 80:10:10 (apprentissage : validation : test). La division des données en ces ensembles de données est souvent effectuée de manière aléatoire, sauf si l'ensemble de données est petit ou s'il y a un

risque que les ensembles de données résultants ne soient pas représentatifs des données opérationnelles attendues.

Si les données disponibles sont limitées, le fait de les répartir en trois ensembles de données peut entraîner une insuffisance de données pour un apprentissage efficace. Pour résoudre ce problème, les ensembles de données d'apprentissage et de validation peuvent être combinés (en gardant l'ensemble de données de test séparé), puis utilisés pour créer des combinaisons multiples de cet ensemble de données (par exemple, 80 % d'apprentissage / 20 % de validation). Les données sont ensuite attribuées de manière aléatoire aux ensembles de données d'apprentissage et de validation. L'apprentissage, la validation et le réglage sont effectués en utilisant ces combinaisons divisées multiples pour créer des modèles réglés multiples, et la performance globale du modèle peut être calculée comme la moyenne de toutes les exécutions.
Plusieurs méthodes sont utilisées pour créer des combinaisons à fractionnement multiple, notamment le split-test, le bootstrap, la validation croisée K-fold et la validation croisée leave-one-out (voir [B02] pour plus de détails).

4.2.1 Exercice pratique : Identifier les données d'apprentissage et de test et créer un modèle ML.

Divisez les données précédemment préparées (voir section 4.1.2) en ensembles de données d'apprentissage, de validation et de test.

Entraîne et testez un modèle de classification utilisant l'apprentissage supervisé avec ces ensembles de données.

Expliquez la différence entre l'évaluation/l'ajustement et le test en comparant la précision obtenue avec les ensembles de données de validation et de test.

4.3 Problèmes de qualité des ensembles de données

Les problèmes de qualité typiques relatifs aux données d'un ensemble de données comprennent, sans s'y limiter, ceux qui figurent dans le tableau suivant :

Aspect qualité	Description
Données incorrectes	Les données capturées étaient incorrectes (par exemple, par un capteur défectueux) ou saisies incorrectement (par exemple, erreurs de copier-coller).
Données incomplètes	Des valeurs de données peuvent être manquantes (par exemple, un champ dans un enregistrement peut être vide, ou les données pour un intervalle de temps particulier peuvent avoir été omises). Les données incomplètes peuvent avoir plusieurs causes, notamment des problèmes de sécurité, des problèmes matériels et des erreurs humaines.
Données mal étiquetées	Il y a plusieurs raisons possibles pour que des données soient mal étiquetées (voir section 4.5.2).

Aspect qualité	Description
Données insuffisantes	Les données disponibles sont insuffisantes pour que les algorithmes d'apprentissage utilisés puissent reconnaître les modèles (il convient de noter que la quantité minimale de données requise varie selon les algorithmes).
Données non prétraitées	Les données doivent être prétraitées pour s'assurer qu'elles sont propres, dans un format cohérent et qu'elles ne contiennent pas de valeurs aberrantes indésirables (voir section 4.1).
Données obsolètes	Les données utilisées pour l'apprentissage et la prédiction doivent être aussi actuelles que possible (par exemple, l'utilisation de données financières datant de plusieurs années peut conduire à des résultats inexacts).
Données non équilibrées	Des données déséquilibrées peuvent résulter d'un biais inapproprié (par exemple, basé sur la race, le sexe ou l'origine ethnique), d'un mauvais placement des capteurs (par exemple, des caméras de reconnaissance faciale placées à hauteur de plafond), de la variabilité de la disponibilité des ensembles de données et des motivations différentes des fournisseurs de données.
Données iniques / injustes	L'équité est une caractéristique de qualité subjective, mais elle peut souvent être identifiée. Par exemple, pour soutenir la diversité ou l'équilibre entre les sexes, les données sélectionnées peuvent être positivement biaisées en faveur des minorités ou des groupes défavorisés (notez que de telles données peuvent être considérées comme justes mais ne sont pas nécessairement équilibrées).
Données dupliquées	Les enregistrements de données répétés peuvent influencer indûment le modèle ML résultant.
Données non pertinentes	Les données qui ne sont pas pertinentes pour le problème abordé peuvent influencer négativement les résultats et entraîner un gaspillage de ressources.
Questions de confidentialité / vie privée	Toute utilisation des données doit respecter les lois pertinentes sur la confidentialité des données (par exemple, le RGPD en ce qui concerne les informations personnelles des individus dans l'Union Européenne).
Questions de sécurité	Les données frauduleuses ou trompeuses qui ont été délibérément insérées dans les données d'apprentissage peuvent conduire à l'inexactitude du modèle formé.

4.4 La qualité des données et son effet sur le modèle ML.

La qualité du modèle ML dépend fortement de la qualité de l'ensemble de données à partir duquel il est créé. Des données de mauvaise qualité peuvent entraîner à la fois des modèles défectueux et des prédictions erronées.

Les catégories suivantes de défauts résultent de problèmes de qualité des données :

- Précision réduite : Ces défauts sont causés par des données erronées, incomplètes, mal étiquetées, insuffisantes, obsolètes, non pertinentes, et des données qui n'ont pas été prétraitées. Par exemple, si les données sont utilisées pour construire un modèle des prix attendus des maisons, mais que les données d'apprentissage contiennent peu ou pas de données sur les maisons individuelles avec véranda, les prix prédits pour ce type de maison spécifique seront probablement inexacts.
- Modèle biaisé : Ces défauts sont causés par des données incomplètes, déséquilibrées, injustes, manquant de diversité ou dupliquées. Par exemple, si les données d'une caractéristique particulière sont manquantes (par exemple, toutes les données médicales pour la prédiction des maladies sont recueillies auprès de sujets d'un sexe particulier), cela risque d'avoir un effet négatif sur le modèle résultant (à moins que le modèle ne soit utilisé que pour faire des prédictions pour ce sexe de manière opérationnelle).
- Modèle compromis : Ces défauts sont dus à des restrictions en matière de confidentialité et de sécurité des données. Par exemple, les problèmes de confidentialité des données peuvent conduire à des vulnérabilités en matière de sécurité, ce qui permettrait aux attaquants de faire de l'ingénierie inverse à partir des modèles et pourrait ensuite entraîner la fuite d'informations personnelles.

4.5 Étiquetage des données pour l'apprentissage supervisé.

L'étiquetage des données est l'enrichissement des données non étiquetées (ou mal étiquetées) par l'ajout d'étiquettes, afin qu'elles puissent être utilisées dans l'apprentissage supervisé. L'étiquetage des données est une activité gourmande en ressources qui, selon les rapports, occupe en moyenne 25 % du temps des projets ML [B11].

Dans sa forme la plus simple, l'étiquetage des données peut consister à placer des images ou des fichiers texte dans différents dossiers, en fonction de leurs classes. Par exemple, tous les fichiers texte d'avis positifs sur un produit sont placés dans un dossier et tous les avis négatifs dans un autre. L'étiquetage des objets dans les images en dessinant des rectangles autour d'eux est une autre technique d'étiquetage courante, souvent appelée annotation. Des annotations plus complexes peuvent être nécessaires pour étiqueter des objets 3D ou pour dessiner des boîtes de délimitation autour d'objets irréguliers. L'étiquetage et l'annotation des données sont généralement pris en charge par des outils.

4.5.1 Approches d'étiquetage des données.

L'étiquetage peut être effectué de plusieurs façons :

- En interne : L'étiquetage est effectué par des développeurs, des testeurs ou une équipe au sein de l'organisation qui est mise en place pour l'étiquetage.
- Externalisé : L'étiquetage est effectué par une organisation externe spécialisée.
- Crowdsourced (NDT : par la foule) : L'étiquetage est effectué par un grand groupe d'individus. En raison de la difficulté à gérer la qualité de l'étiquetage, il est possible de demander à plusieurs annotateurs d'étiqueter les mêmes données et de décider ensuite de l'étiquette à utiliser.
- Assisté par l'IA : Des outils basés sur l'IA sont utilisés pour reconnaître et annoter des données ou pour regrouper des données similaires. Les résultats sont ensuite confirmés ou

éventuellement complétés (par exemple, en modifiant la boîte de délimitation) par un humain, dans le cadre d'un processus en deux étapes.

- Hybride : Une combinaison des approches d'étiquetage ci-dessus pourrait être utilisée. Par exemple, l'étiquetage par la foule est généralement géré par une organisation externe qui a accès à des outils spécialisés de gestion de la foule basés sur l'IA.

Le cas échéant, il peut être possible de réutiliser un ensemble de données pré-étiquetées, évitant ainsi tout étiquetage des données. De nombreux ensembles de données de ce type sont accessibles au public, par exemple sur Kaggle [R16].

4.5.2 Données mal étiquetées dans les ensembles de données.

L'apprentissage supervisé suppose que les données sont correctement étiquetées par les annotateurs de données. Cependant, il est rare en pratique que tous les éléments d'un ensemble de données soient correctement étiquetés. Les données sont mal étiquetées pour les raisons suivantes :

- Des erreurs aléatoires peuvent être commises par les annotateurs (par exemple, appuyer sur le mauvais bouton).
- Des erreurs systémiques peuvent être commises (par exemple, lorsque les annotateurs reçoivent de mauvaises instructions ou une mauvaise formation).
- Des erreurs délibérées peuvent être commises par des annotateurs de données malveillants.
- Des erreurs de traduction peuvent prendre des données correctement étiquetées dans une langue et les étiqueter de manière erronée dans une autre.
- Lorsque le choix est ouvert à l'interprétation, les jugements subjectifs des annotateurs de données peuvent conduire à des étiquettes de données contradictoires de la part de différents annotateurs.
- Le manque de connaissances requises dans le domaine peut conduire à un étiquetage incorrect.
- Les tâches de classification complexes peuvent entraîner un plus grand nombre d'erreurs.
- Les outils utilisés pour soutenir l'étiquetage des données présentent des défauts qui conduisent à des étiquettes incorrectes.
- Les approches d'étiquetage basées sur le modèle ML sont probabilistes, ce qui peut conduire à des étiquettes incorrectes.

5 ML : Métriques de performance fonctionnelle – 120 minutes

Mots-clés

Aucun

Mots-clés spécifiques à l'IA

Exactitude, surface sous la courbe (AUC – Area Under the Curve), matrice de confusion, score F1, métrique inter-clusters, métrique intra-clusters, erreur quadratique moyenne (MSE - Mean Squared Error), suites de référence ML, métrique de performance fonctionnelle ML, précision, rappel, courbe ROC (Receiver Operating Characteristic), modèle de régression, R-squared, coefficient de silhouette.

Objectifs d'apprentissage pour le chapitre 5 :

5.1 Matrice de confusion

AI-5.1.1 K3 Calculer les métriques de performance fonctionnelle ML à partir d'un ensemble donné de données de matrice de confusion.

5.2 Métriques supplémentaires de performance fonctionnelle ML pour la classification, la régression et le clustering

AI-5.2.1 K2 Contraster et comparer les concepts derrière les métriques de performance fonctionnelle ML pour les méthodes de classification, de régression et de clustering.

5.3 Limites des métriques de performance fonctionnelle ML

AI-5.3.1 K2 Résumer les limites de l'utilisation des mesures de performance fonctionnelle de la ML pour déterminer la qualité du système ML.

5.4 Sélection des métriques de performance fonctionnelle de ML

AI-5.4.1 K4 Sélectionner les métriques de performance fonctionnelle appropriées et/ou leurs valeurs pour un modèle et un scénario de ML donnés.

HO-5.4.1 H2 Évaluer le modèle de ML créé en utilisant les métriques de performance fonctionnelle de ML sélectionnées.

5.5 Suites de benchmarks pour ML

AI-5.5.1 K2 Expliquer l'utilisation des benchmarks dans le contexte de la ML

5.1 Matrice de confusion

Dans un problème de classification, il est rare qu'un modèle prédise toujours correctement les résultats. Pour tout problème de ce type, une matrice de confusion peut être créée avec les possibilités suivantes :

		Réel	
		Positif	**Négatif**
Prédit	**Positif**	**Vrai Positif (TP)**	**Faux Positif (FP)**
	Négatif	**Faux Négatif (FN)**	**Vrai Négatif (TN)**

Figure 2: Matrice de confusion

Notez que la matrice de confusion illustrée à la figure 2 peut être présentée différemment, mais qu'elle générera toujours des valeurs pour les quatre situations possibles : vrai positif (TP), vrai négatif (TN), faux positif (FP) et faux négatif (FN).

Sur la base de la matrice de confusion, les métriques suivantes sont définies :

- Exactitude

  Exactitude = (TP + TN) / (TP +TN + FP + FN) * 100%.

  L'exactitude mesure le pourcentage de toutes les classifications correctes.

- Précision

  Précision = TP / (TP + FP) * 100 %.

  La précision mesure la proportion de positifs qui ont été correctement prédits. Il s'agit d'une mesure de la certitude que l'on peut avoir sur les prédictions positives.

- Rappel

  Rappel = TP / (TP + FN) * 100 %.

  Le rappel (également appelé sensibilité) mesure la proportion de positifs réels qui ont été prédits correctement. Il s'agit d'une mesure de la certitude que l'on peut avoir de ne manquer aucun positif.

- Score F1

  Score F1 = 2* (Précision * Rappel) / (Précision + Rappel)

Le score F1 est calculé comme la moyenne harmonique de la précision et du rappel. Il aura une valeur comprise entre zéro et un. Un score proche de un indique que les données erronées ont peu d'influence sur le résultat. Un score F1 faible indique que le modèle détecte mal les données positives.

5.2 Métriques supplémentaires de performance fonctionnelle ML pour la classification, la régression et le clustering.

Il existe de nombreuses métriques pour différents types de problèmes de ML (en plus de celles liées à la classification décrites dans la section 5.1). Certaines des métriques les plus couramment utilisées sont décrites ci-dessous.

Métriques de classification supervisée

- La courbe ROC (receiver operating characteristic) est un graphique qui illustre la capacité d'un classificateur binaire à faire varier son seuil de discrimination. Cette méthode a été développée à l'origine pour les radars militaires, d'où son nom.
  La courbe ROC représente le taux de vrais positifs (TPR) (également connu sous le nom de rappel) par rapport au taux de faux positifs (FPR = FP / (TN + FP)), avec le TPR sur l'axe des y et le FPR sur l'axe des x.
- L'aire sous la courbe (AUC) est l'aire sous la courbe ROC. Elle représente le degré de séparabilité d'un classificateur, montrant comment le modèle distingue les classes. Avec une AUC plus élevée, les prédictions du modèle sont meilleures.

Métriques de régression supervisée

Pour les modèles de régression supervisée, les métriques représentent la qualité de l'ajustement de la ligne de régression aux points de données réels.

- L'erreur quadratique moyenne (MSE - Mean Squared Error) est la moyenne des différences au carré entre la valeur réelle et la valeur prédite. La valeur de l' erreur quadratique moyenne est toujours positive, et une valeur proche de zéro suggère un meilleur modèle de régression. En élevant la différence au carré, on s'assure que les erreurs positives et négatives ne s'annulent pas.
- Le R-carré (également connu sous le nom de coefficient de détermination) est une mesure de l'adéquation du modèle de régression aux variables dépendantes.

Métriques du clustering non supervisé

Pour le clustering non supervisé, il existe plusieurs métriques qui représentent les distances entre les différents clusters et la proximité des points de données au sein d'un cluster donné.

- Les métriques intra-clusters mesurent la similarité des points de données au sein d'un cluster.
- Les métriques inter-clusters mesurent la similarité des points de données dans différents clusters.
- Le coefficient de silhouette (également connu sous le nom de score de silhouette) est une mesure (entre -1 et +1) basée sur les distances moyennes inter-cluster et intra-cluster. Un score de +1 signifie que les clusters sont bien séparés, un score de zéro implique un clustering aléatoire, et un score de -1 signifie que les clusters sont mal assignés.

5.3 Limites des métriques de performance fonctionnelle ML.

Les métriques de performance fonctionnelle ML sont limitées à la mesure de la fonctionnalité du modèle, par exemple en termes d'exactitude, de précision, de rappel, de MSE, d'AUC et de coefficient de silhouette. Elles ne mesurent pas d'autres caractéristiques de qualité non fonctionnelles, telles que celles définies dans la norme ISO 25010 [S06] (par exemple, l'efficience des performances) et celles décrites au chapitre 2 (par exemple, l'explicabilité, la flexibilité et l'autonomie). Dans ce syllabus, le terme "Métriques de performance ML" est utilisé en raison de l'utilisation répandue du terme "métriques de performance" pour faire référence à ces métriques fonctionnelles. L'ajout de "fonctionelle ML" souligne que ces métriques sont spécifiques au machine learning et n'ont aucun rapport avec les métriques d'efficience des performances.

Les métriques de performance fonctionnelle ML sont limitées par plusieurs autres facteurs :

- Pour l'apprentissage supervisé, les métriques de performance fonctionnelle ML sont calculées sur la base de données étiquetées, et l'exactitude des métriques résultantes dépend d'un étiquetage correct (voir section 4.5).
- Les données utilisées pour la mesure peuvent ne pas être représentatives (par exemple, elles peuvent être biaisées) et les métriques de performance fonctionnelle ML générées dépendent de ces données (voir section 2.4).
- Le système peut comprendre plusieurs composants, mais la métrique de performance fonctionnelle ML ne s'applique qu'au modèle ML. Par exemple, le pipeline de données n'est pas considéré par la métrique de performance fonctionnelle ML pour évaluer le modèle.
- La plupart des métriques de performance fonctionnelle du ML ne peuvent être mesurées qu'avec l'aide d'outils.

5.4 Sélection des métriques de performance fonctionnelle ML.

Il n'est normalement pas possible de construire un modèle ML qui obtient le score le plus élevé pour toutes les métriques de performance fonctionnelle ML générées à partir d'une matrice de confusion. Au lieu de cela, les métriques de performance fonctionnelle les plus appropriées sont sélectionnées comme critères d'acceptation, sur la base de l'utilisation attendue du modèle (par exemple, pour minimiser les faux positifs, une valeur élevée de précision est requise, tandis que pour minimiser les faux négatifs, la métrique de rappel doit être élevée). Les critères suivants peuvent être utilisés pour sélectionner les métriques de performance fonctionnelle du ML décrites dans les sections 5.1 et 5.2 :

- Exactitude : Cette métrique est susceptible d'être applicable si les ensembles de données sont symétriques (par exemple, le nombre de faux positifs et de faux négatifs et les coûts sont similaires). Cette métrique devient un mauvais choix si une classe de données domine sur les autres, auquel cas le score F1 doit être considéré.
- Précision : Cette métrique peut convenir lorsque le coût des faux positifs est élevé et que la confiance dans les résultats positifs doit être élevée. Un filtre anti-spam (où la classification d'un courriel comme spam est considérée comme positive) est un exemple où une précision élevée est nécessaire, car la plupart des utilisateurs n'accepteront pas de placer dans le dossier spam un trop grand nombre de courriels qui ne sont pas réellement des spams. Lorsque le classificateur traite des situations où un très grand pourcentage de cas sont positifs, l'utilisation de la précision seule ne sera probablement pas un bon choix.

- Rappel : Lorsqu'il est essentiel de ne pas manquer de résultats positifs, un score de rappel élevé est important. Par exemple, il est probablement inacceptable de manquer des résultats positifs dans la détection du cancer et de les marquer comme négatifs (c'est-à-dire qu'aucun cancer n'a été détecté).
- Score F1 - Le score F1 est le plus utile lorsqu'il y a un déséquilibre dans les classes attendues et lorsque la précision et le rappel sont d'importance similaire.

En plus des métriques ci-dessus, plusieurs métriques sont décrites dans la section 5.2. Elles peuvent être applicables à des problèmes de ML donnés, par exemple :

- L'AUC de la courbe ROC peut être utilisée pour les problèmes de classification supervisée.
- MSE et R-carré peuvent être utilisés pour les problèmes de régression supervisée.
- Les métriques inter-cluster, intra-cluster et le coefficient de silhouette peuvent être utilisés pour les problèmes de clustering non supervisé.

5.4.1 Exercice pratique : Évaluer le modèle ML créé.

En utilisant le modèle de classification entraîné dans l'exercice précédent, calculez et affichez les valeurs d'exactitude, de précision, de rappel et de score F1. Le cas échéant, utilisez les fonctions de bibliothèque fournies par votre framework de développement pour effectuer les calculs.

5.5 Suites de Benchmark pour ML

De nouvelles technologies d'IA, telles que de nouveaux ensembles de données, algorithmes, modèles et matériels, sont publiées régulièrement, et il peut être difficile de déterminer l'efficacité relative de chaque nouvelle technologie.

Afin de fournir des comparaisons objectives entre ces différentes technologies, il existe des suites de benchmark ML standard dans l'industrie. Elles couvrent un large éventail de domaines d'application et fournissent des outils pour évaluer les plateformes matérielles, les frameworks logiciels et les plateformes clouds en termes de performances d'IA et de ML.

Les suites de benchmark ML peuvent fournir diverses mesures, y compris les temps d'apprentissage (par exemple, la vitesse à laquelle un framework peut entraîner un modèle ML en utilisant un ensemble de données d'apprentissage défini à une métrique de qualité cible spécifiée, comme une exactitude de 75%), et les temps d'inférence (par exemple, la vitesse à laquelle un modèle ML formé peut effectuer une inférence).

Les suites de benchmark ML sont fournies par plusieurs organisations différentes, telles que :

- MLCommons [R18] : Il s'agit d'une organisation à but non lucratif formée en 2020 et précédemment nommée ML Perf, qui fournit des benchmarks pour les frameworks logiciels, les processeurs spécifiques à l'IA et les plateformes cloud ML.
- DAWNBench [R19] : Il s'agit d'une suite de benchmark ML de l'Université de Stanford.
- MLMark [R20] : Il s'agit d'une suite de benchmark ML conçue pour mesurer la performance et l'exactitude de l'inférence embarquée du Consortium de benchmark des microprocesseurs embarqués (Embedded Microprocessor Benchmark Consortium).

6 ML – Réseaux neuronaux et test – 65 minutes

Mots clés

Aucun

Mots-clés spécifiques à l'IA

Valeur d'activation, réseau neuronal profond (DNN – Deep Neural Network), données d'apprentissage ML, perceptron multicouche, réseau neuronal, couverture des neurones, perceptron, couverture des changements de signe, couverture signe-signe, apprentissage supervisé, couverture des seuils, données d'apprentissage, couverture des changements de valeur.

Objectifs d'apprentissage pour le chapitre 6 :

6.1 Réseaux neuronaux

AI-6.1.1 K2 Expliquer la structure et la fonction d'un réseau neuronal, y compris un DNN.

HO-6.1.1 H1 Expérimenter la mise en œuvre d'un perceptron.

6.2 Mesures de couverture pour les réseaux neuronaux

AI-6.2.1 K2 Décrire les différentes mesures de couverture pour les réseaux neuronaux.

6.1 Réseaux neuronaux

Les réseaux neuronaux artificiels étaient initialement destinés à imiter le fonctionnement du cerveau humain, qui peut être considéré comme un ensemble de neurones biologiques connectés. Le perceptron monocouche est l'un des premiers exemples de mise en œuvre d'un réseau neuronal artificiel. Il s'agit d'un réseau neuronal à une seule couche (c'est-à-dire un seul neurone). Il peut être utilisé pour l'apprentissage supervisé de classificateurs, qui décident si une entrée appartient ou non à une classe spécifique.

La plupart des réseaux neuronaux actuels sont considérés comme des réseaux neuronaux profonds car ils comportent plusieurs couches et peuvent être considérés comme des perceptrons multicouches (cf. Figure 3).

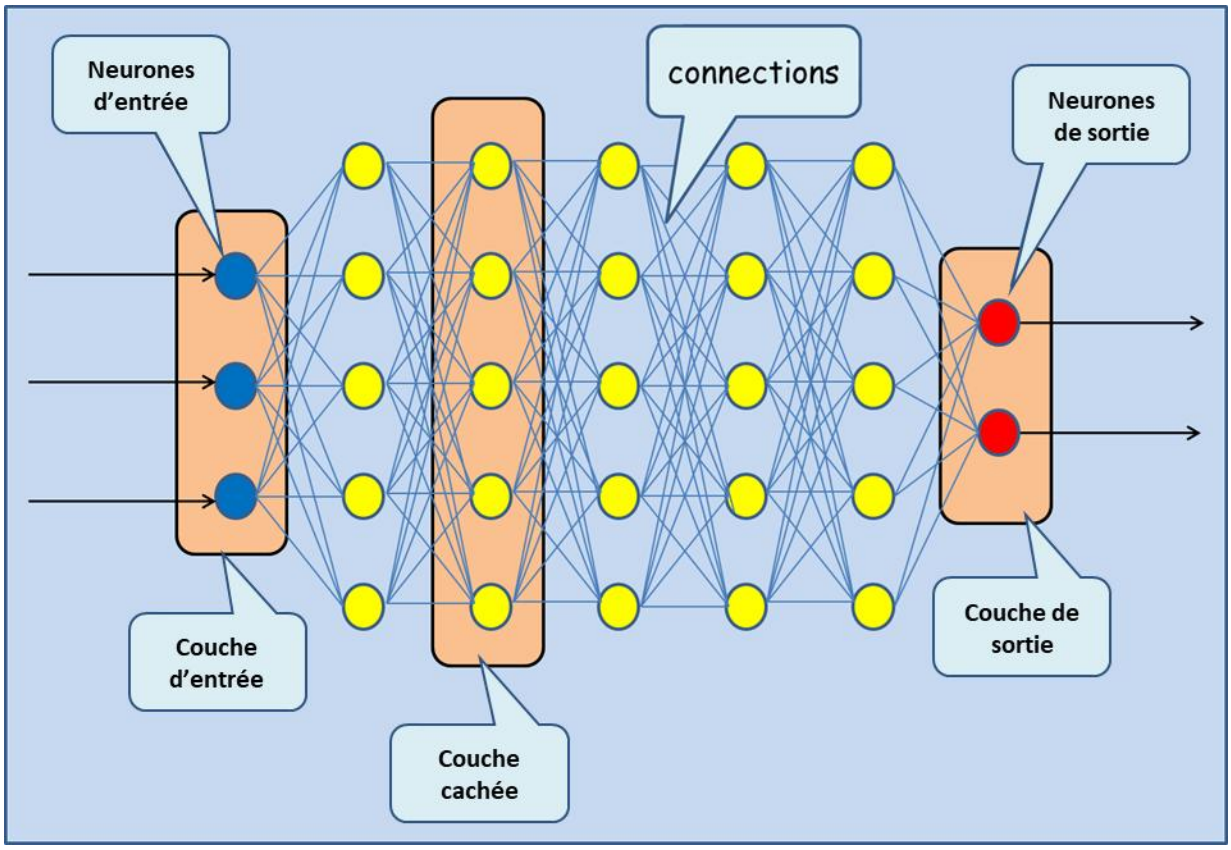

Figure 3 Structure d'un réseau neuronal profond

Un réseau neuronal profond comprend trois types de couches. La couche d'entrée reçoit les entrées, par exemple les valeurs des pixels d'une caméra. La couche de sortie fournit des résultats au monde extérieur. Il peut s'agir, par exemple, d'une valeur indiquant la probabilité que l'image d'entrée soit un chat. Entre les couches d'entrée et de sortie se trouvent des couches cachées composées de neurones artificiels, également appelés nœuds. Les neurones d'une couche sont connectés à chacun des neurones de la couche suivante et il peut y avoir un nombre différent de neurones dans chaque couche

successive. Les neurones effectuent des calculs et transmettent les informations à travers le réseau, des neurones d'entrée aux neurones de sortie.

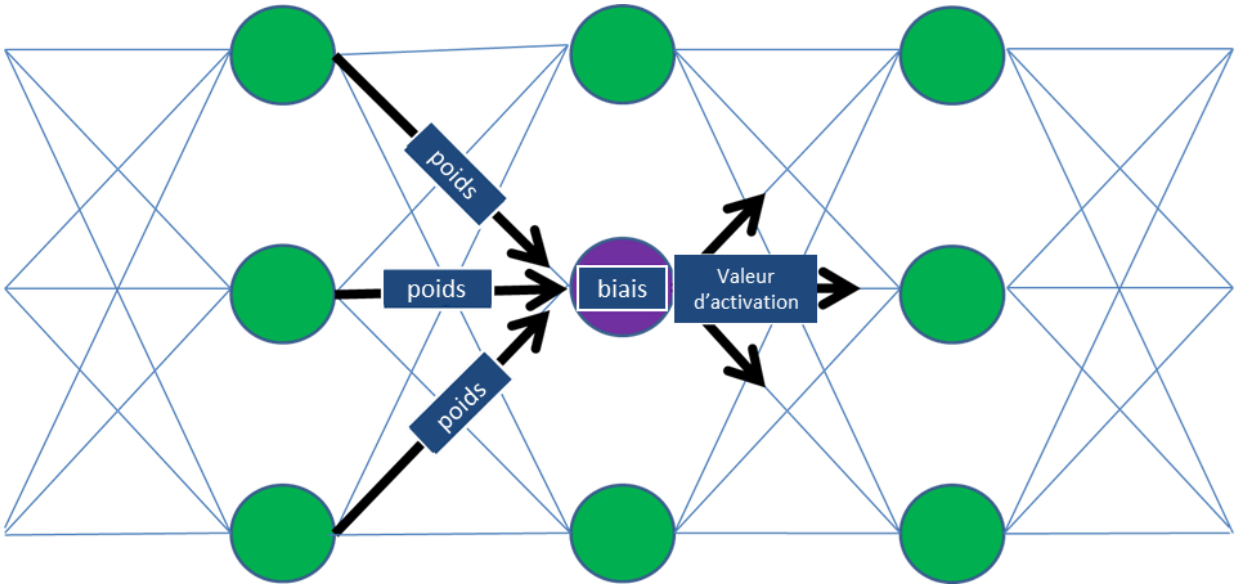

Figure 4 Calcul effectué par chaque neurone

Comme le montre la figure 4, le calcul effectué par chaque neurone (sauf ceux de la couche d'entrée) génère ce que l'on appelle la valeur d'activation. Cette valeur est calculée en exécutant une formule (la fonction d'activation) qui reçoit en entrée les valeurs d'activation de tous les neurones de la couche précédente, les poids attribués aux connexions entre les neurones (ces poids changent au fur et à mesure que le réseau apprend), et le biais individuel de chaque neurone. Notez que ce biais est une valeur constante prédéfinie et n'est pas lié au biais considéré précédemment dans la section 2.4). L'exécution de différentes fonctions d'activation peut entraîner le calcul de différentes valeurs d'activation. Ces valeurs sont généralement centrées autour de zéro et comprises entre -1 (signifiant que le neurone est " désintéressé ") et +1 (signifiant que le neurone est " très intéressé ").

Lors de l'apprentissage du réseau neuronal, chaque neurone est préréglé sur une valeur de biais et les données d'apprentissage sont transmises au réseau, chaque neurone exécutant la fonction d'activation, pour générer finalement une sortie. La sortie générée est ensuite comparée au résultat correct connu (des données étiquetées sont utilisées dans cet exemple d'apprentissage supervisé). La différence entre la sortie réelle et le résultat correct connu est ensuite réinjectée dans le réseau pour modifier les valeurs des poids des connexions entre les neurones afin de minimiser cette différence. Au fur et à mesure que des données d'apprentissage sont transmises au réseau, les poids sont progressivement ajustés au fur et à mesure que le réseau apprend. Finalement, les sorties produites sont considérées comme suffisamment bonnes pour mettre fin à l'apprentissage.

6.1.1 Exercice pratique, implémenter un Perceptron simple

Les participants seront guidés à travers un exercice démontrant qu'un Perceptron apprend une fonction simple, telle qu'une fonction AND.

L'exercice doit montrer comment un perceptron apprend en modifiant les poids pendant un certain nombre d'époques jusqu'à ce que l'erreur soit nulle. Divers mécanismes peuvent être utilisés pour cette activité (par exemple, feuille de calcul, simulation, etc.).

6.2 Mesures de couverture pour les réseaux neuronaux

Le respect des critères de couverture de test de boîte blanche (par exemple, la couverture des instructions, des branches, des conditions/décisions modifiées (MC/DC) [I01] est obligatoire pour la conformité à certaines normes liées à la sûreté [S07] lors de l'utilisation de code source impératif traditionnel, et est recommandé par de nombreux praticiens du test pour d'autres applications critiques. La surveillance et l'amélioration de la couverture aident à la conception de nouveaux cas de test, conduisant à une confiance accrue dans l'objet de test.

L'utilisation de telles mesures pour mesurer la couverture des réseaux neuronaux présente peu d'intérêt, car le même code tend à être exécuté à chaque fois que le réseau neuronal est exécuté. En revanche, des mesures de couverture ont été proposées sur la base de la couverture de la structure du réseau neuronal lui-même, et plus précisément des neurones qui le composent. La plupart de ces mesures sont basées sur les valeurs d'activation des neurones.

La couverture des réseaux neuronaux est un nouveau domaine de recherche. Les articles académiques n'ont été publiés que depuis 2017 et, à ce titre, il existe peu de preuves objectives (par exemple, des résultats de recherche dupliqués) qui montrent que les mesures proposées sont efficaces. Il convient toutefois de noter que, bien que la couverture des déclarations et des décisions soit utilisée depuis plus de 50 ans, il existe également peu de preuves objectives de leur efficacité relative, même si elles sont obligatoires pour mesurer la couverture des logiciels dans les applications liées à la sûreté, telles que les dispositifs médicaux et les systèmes avioniques.

Les critères de couverture suivants pour les réseaux neuronaux ont été proposés et appliqués par les chercheurs à une variété d'applications :

- Couverture des neurones : La couverture complète des neurones exige que chaque neurone du réseau neuronal atteigne une valeur d'activation supérieure à zéro [B12]. Ceci est très facile à réaliser dans la pratique et la recherche a montré qu'une couverture de près de 100% est obtenue avec très peu de cas de test sur une variété de réseaux neuronaux profonds. Cette mesure de couverture peut être très utile comme signal d'alarme lorsqu'elle n'est pas atteinte.
- Couverture du seuil : La couverture complète du seuil exige que chaque neurone du réseau neuronal atteigne une valeur d'activation supérieure à un seuil spécifié. Les chercheurs qui ont créé le framework DeepXplore ont suggéré que la couverture des neurones soit mesurée sur la base de la valeur d'activation dépassant un seuil qui changerait en fonction de la situation. Ils ont effectué leurs recherches avec un seuil de 0,75 lorsqu'ils ont déclaré avoir trouvé efficacement des milliers de comportements incorrects dans certains cas de figure en utilisant cette approche en boîte blanche. Ce type de couverture a été renommé ici pour le distinguer plus facilement de la couverture des neurones avec un seuil fixé à zéro, car certains autres chercheurs utilisent le terme "couverture des neurones" pour désigner la couverture des neurones avec un seuil de zéro.

- Couverture de changement de signe : Pour obtenir une couverture complète du changement de signe, les cas de test doivent amener chaque neurone à atteindre des valeurs d'activation positives et négatives [B13].
- Couverture du changement de valeur : Pour obtenir une couverture complète du changement de valeur, les cas de test doivent faire en sorte que chaque neurone atteigne deux valeurs d'activation, la différence entre les deux valeurs étant supérieure à une valeur choisie [B13].
- Couverture signe-signe : Cette couverture considère les paires de neurones dans les couches adjacentes et le signe pris par leurs valeurs d'activation. Pour qu'une paire de neurones soit considérée comme couverte, un cas de test doit montrer que le changement de signe d'un neurone de la première couche entraîne le changement de signe du neurone de la deuxième couche, tandis que les signes de tous les autres neurones de la deuxième couche restent inchangés [B13]. Il s'agit d'un concept similaire à la couverture MC/DC pour le code source impératif.

Les chercheurs ont fait état d'autres mesures de couverture basées sur les couches (bien que plus simples que la couverture par le signe), et une approche réussie utilisant les algorithmes du plus proche voisin pour identifier les changements significatifs dans les ensembles de neurones voisins a été mise en œuvre dans l'outil TensorFuzz [B14].

7 Tester les systèmes basés sur l'IA - aperçu – 115 minutes

Mots clés

Test des données d'entrée, test des modèles ML.

Mots-clés spécifiques à l'IA

Composant IA, biais d'automatisation, big data, dérive des concepts, pipeline de données, métriques de performance fonctionnelle ML, données d'apprentissage

Objectifs d'apprentissage pour le chapitre 7:

7.1 Spécification des systèmes basés sur l'IA

AI-7.1.1 K2 Expliquer comment les spécifications des systèmes basés sur l'IA peuvent créer des difficultés en matière de test.

7.2 Niveaux de test des systèmes basés sur l'IA

AI-7.2.1 K2 Décrire comment les systèmes basés sur l'IA sont testés à chaque niveau de test.

7.3 Données de test pour tester les systèmes basés sur l'IA

AI-7.3.1 K1 Rappeler les facteurs associés aux données de test qui peuvent rendre difficile le test des systèmes basés sur l'IA.

7.4 Test des biais d'automatisation des systèmes basés sur l'IA

AI-7.4.1 K2 Expliquer le biais d'automatisation et comment il affecte les tests.

7.5 Documenter un modèle ML

AI-7.5.1 K2 Décrire la documentation d'un composant d'intelligence artificielle et comprendre comment la documentation aide au test des systèmes basés sur l'intelligence artificielle.

7.6 Tester la dérive du concept

AI-7.6.1 K2 Expliquer la nécessité de tester fréquemment le modèle formé pour gérer la dérive des concepts.

7.7 Choisir une approche de test pour un système ML

AI-7.7.1 K4 Pour un scénario donné, déterminer une approche de test à suivre lors du développement d'un système ML.

7.1 Spécification des systèmes basés sur l'IA.

Les exigences du système et les spécifications de conception sont tout aussi importantes pour les systèmes basés sur l'IA que pour les systèmes conventionnels. Ces spécifications servent de base aux testeurs pour vérifier si le comportement réel du système est conforme aux exigences spécifiées. Cependant, si les spécifications sont incomplètes et manquent de testabilité, cela introduit un problème d'oracle de test (voir section 8.7).

Il y a plusieurs raisons pour lesquelles la spécification des systèmes basés sur l'IA peut être particulièrement difficile :

- Dans de nombreux projets de systèmes basés sur l'IA, les exigences sont spécifiées uniquement en termes d'objectifs métier de haut niveau et de prédictions requises. Cela s'explique notamment par la nature exploratoire du développement des systèmes basés sur l'IA. Souvent, les projets de systèmes basés sur l'IA commencent avec un ensemble de données, et l'objectif est de déterminer quelles prédictions peuvent être obtenues à partir de ces données. C'est une différence par rapport à la spécification de la logique requise dès le début d'un projet conventionnel.
- L'exactitude du système basé sur l'IA est souvent inconnue jusqu'à ce que les résultats des tests indépendants soient disponibles. Avec l'approche de développement exploratoire, cela conduit souvent à des spécifications inadéquates, car la mise en œuvre est déjà en cours au moment où les critères d'acceptation souhaités sont déterminés.
- La nature probabiliste de nombreux systèmes basés sur l'IA peut rendre nécessaire la spécification de tolérances pour certaines des exigences de qualité attendues, telles que l'exactitude des prédictions.
- Lorsque les objectifs du système demandent de reproduire le comportement humain, plutôt que de fournir une fonctionnalité spécifique, cela conduit souvent à des exigences de comportement mal spécifiées, basées sur le fait que le système est aussi bon ou meilleur que les activités humaines qu'il vise à remplacer. Cela peut rendre difficile la définition d'un oracle de test, en particulier lorsque les humains qu'il remplace ont des capacités très différentes.
- Lorsque l'IA est utilisée pour mettre en œuvre des interfaces utilisateur, par exemple par la reconnaissance du langage naturel, la vision par ordinateur ou l'interaction physique avec les humains, les systèmes doivent faire preuve d'une flexibilité accrue. Cependant, une telle flexibilité peut également créer des difficultés pour identifier et documenter toutes les différentes façons dont ces interactions peuvent se produire.
- Les caractéristiques de qualité propres aux systèmes basés sur l'IA, telles que l'adaptabilité, la flexibilité, l'évolution et l'autonomie, doivent être prises en compte et définies dans le cadre de la spécification des exigences (voir chapitre 2). La nouveauté de ces caractéristiques peut les rendre difficiles à définir et à tester.

7.2 Niveaux de test pour les systèmes basés sur l'IA.

Les systèmes basés sur l'IA comprennent généralement des composants IA et non-IA. Les composants non-IA peuvent être testés à l'aide d'approches classiques [I01], tandis que les composants IA et les systèmes contenant des composants IA peuvent devoir être testés différemment à certains égards, comme décrit ci-dessous. Pour tous les niveaux de test qui

incluent le test des composants d'IA, il est important que le test soit étroitement soutenu par des data engineers/scientists et des experts du domaine.

Une différence majeure par rapport aux niveaux de test utilisés pour les logiciels conventionnels est l'inclusion de deux nouveaux niveaux de test spécialisés pour gérer explicitement le test des données d'entrée et des modèles utilisés dans les systèmes basés sur l'IA [B15]. La majeure partie de cette section est applicable à tous les systèmes basés sur l'IA, bien que certaines parties soient spécifiquement axées sur les systèmes à base de ML.

7.2.1 Test des données d'entrée.

L'objectif du test des données d'entrée est de s'assurer que les données utilisées par le système pour l'apprentissage et la prédiction sont de la plus haute qualité (voir section 4.3). Il comprend les éléments suivants :

- Revues.
- Techniques statistiques (par exemple, test des données pour détecter les biais).
- EDA (analyse exploratoire des données) des données d'apprentissage.
- Tests statiques et dynamiques du pipeline de données.

Le pipeline de données comprend généralement plusieurs composants chargés de la préparation des données (voir section 4.1), et le test de ces composants comprend à la fois des tests de composants et des tests d'intégration. Le pipeline de données pour l'apprentissage peut être très différent du pipeline de données utilisé pour réaliser la prédiction en situation opérationnelle. Pour l'apprentissage, le pipeline de données peut être considéré comme un prototype, par rapport à la version entièrement conçue et automatisée utilisée de manière opérationnelle. C'est pourquoi les tests de ces deux versions du pipeline de données peuvent être très différents. Cependant, le test de l'équivalence fonctionnelle des deux versions doit également être envisagé.

7.2.2 Test des modèles de ML.

L'objectif du test des modèles de ML est de s'assurer que le modèle sélectionné répond à tous les critères de performance qui ont pu être spécifiés. Cela comprend :

- Les critères de performance fonctionnels du ML (voir les sections 5.1 et 5.2).

- Les critères d'acceptation non fonctionnels du ML qui sont appropriés pour le modèle en isolation, tels que la vitesse d'apprentissage, la vitesse de prédiction, les ressources informatiques utilisées, l'adaptabilité et la transparence.

Le test des modèles de ML vise également à déterminer que le choix du framework, de l'algorithme, du modèle, des paramètres du modèle et des hyperparamètres de ML est aussi proche de l'optimum que possible. Le cas échéant, le test du modèle ML peut également inclure des tests pour atteindre les critères de couverture de type boîte blanche (voir section 6.2). Le modèle sélectionné est ensuite intégré à d'autres composants, IA et non-IA.

7.2.3 Test des composants

Le test des composants est un niveau de test conventionnel qui s'applique à tous les composants qui ne font pas partie du modèle, tels que les interfaces utilisateur et les composants de communication.

7.2.4 Test d'intégration des composants

Le test d'intégration des composants est un niveau de test conventionnel qui est effectué pour s'assurer que les composants du système (IA et non-IA) interagissent correctement. Il teste que les entrées du pipeline de données sont reçues comme prévu par le modèle, et que toutes les prédictions produites par le modèle sont échangées avec les composants du système concernés (par exemple, l'interface utilisateur) et utilisées correctement par ceux-ci. Lorsque l'IA est fournie en tant que service (voir section 1.7), il est normal d'effectuer des tests API du service fourni dans le cadre des tests d'intégration des composants.

7.2.5 Test système

Le test système est un niveau de test conventionnel qui est effectué pour s'assurer que le système complet de composants intégrés (IA et non-IA) fonctionne comme prévu, tant du point de vue fonctionnel que non fonctionnel, dans un environnement de test qui est étroitement représentatif de l'environnement opérationnel. Selon le système, ces tests peuvent prendre la forme d'essais sur le terrain dans l'environnement opérationnel prévu ou de tests dans un simulateur (par exemple, si les scénarios de test sont dangereux ou difficiles à reproduire dans un environnement opérationnel).
Pendant les tests système, les critères de performance fonctionnelle du ML sont testés à nouveau pour s'assurer que les résultats des tests initiaux du modèle de ML ne sont pas altérés lorsque le modèle est intégré dans un système complet. Ce test est particulièrement important lorsque le composant d'intelligence artificielle a été délibérément modifié (par exemple, en compressant un DNN pour réduire sa taille).
Le test système est également le niveau de test dans lequel de nombreuses exigences non fonctionnelles du système sont testées. Par exemple, des tests adverses peuvent être effectués pour tester la robustesse et le système peut être testé pour l'explicabilité. Le cas échéant, les interfaces avec les composants matériels (par exemple, les capteurs) peuvent être testées dans le cadre des tests du système.

7.2.6 Tests d'acceptation

Le test d'acceptation est un niveau de test classique et sert à déterminer si le système complet est acceptable pour le client. Pour les systèmes basés sur l'IA, la définition des critères d'acceptation peut être difficile (voir section 8.8). Lorsque l'IA est fournie en tant que service (voir section 1.7), des tests d'acceptation peuvent être nécessaires pour déterminer l'adéquation du service au système prévu et si, par exemple, les critères de performance fonctionnelle de ML ont été suffisamment atteints.

7.3 Données de test pour tester les systèmes basés sur l'IA.

Selon la situation et le système en cours de test (SUT – System Under Test), l'acquisition de données de test peut représenter un défi. Il existe plusieurs défis potentiels dans le traitement des données de test pour les systèmes basés sur l'IA, notamment :

- Les big data (données à fort volume, à grande vitesse et à grande variété) peuvent être difficiles à créer et à gérer. Par exemple, il peut être difficile de créer des données de test représentatives pour un système qui consomme de grands volumes d'images et d'audio à une vitesse élevée.
- Les données d'entrée peuvent devoir évoluer dans le temps, en particulier si elles représentent des événements du monde réel. Par exemple, les photographies enregistrées pour tester un

système de reconnaissance faciale peuvent devoir être "vieillies" pour représenter le vieillissement des personnes sur plusieurs années dans la vie réelle.

- Les données personnelles ou confidentielles peuvent nécessiter des techniques spéciales d'anonymisation, de cryptage ou de rédaction. Une autorisation légale d'utilisation peut également être requise.
- Lorsque les testeurs utilisent la même implémentation que les data scientists pour l'acquisition et le prétraitement des données, les défauts dans ces étapes peuvent être masqués.

7.4 Test des biais d'automatisation dans les systèmes basés sur l'IA.

Une catégorie de systèmes basés sur l'IA aide les humains à prendre des décisions. Cependant, l'homme a parfois tendance à faire trop confiance à ces systèmes. Cette confiance mal placée peut être appelée biais d'automatisation ou biais de complaisance, et prend deux formes.

- La première forme de biais d'automatisation/de complaisance est celle où l'humain accepte les recommandations fournies par le système et ne tient pas compte des apports d'autres sources (y compris lui-même). Par exemple, une procédure dans laquelle un humain saisit des données dans un formulaire peut être améliorée en utilisant l'apprentissage machine pour préremplir le formulaire, et l'humain valide ensuite ces données. Il a été démontré que cette forme de biais d'automatisation réduit généralement la qualité des décisions prises de 5 %, mais ce chiffre pourrait être beaucoup plus élevé en fonction du contexte du système [B16]. De même, la correction automatique du texte tapé (par exemple, dans les messages des téléphones portables) est souvent défectueuse et peut en modifier le sens. Souvent, les utilisateurs ne le remarquent pas et n'annulent pas l'erreur.
- La deuxième forme de biais d'automatisation/complaisance est celle où l'humain rate une défaillance du système parce qu'il ne le surveille pas suffisamment. Par exemple, les véhicules semi-autonomes deviennent de plus en plus autonomes, mais ils comptent toujours sur un humain pour prendre le relais en cas d'accident imminent. En général, l'occupant du véhicule fait progressivement trop confiance aux capacités du système à contrôler le véhicule et il commence à faire moins attention. Cela peut conduire à une situation où il est incapable de réagir de manière appropriée en cas de besoin.

Dans les deux scénarios, les testeurs doivent comprendre comment la prise de décision humaine peut être compromise, et tester à la fois la qualité des recommandations du système et la qualité des données humaines correspondantes fournies par des utilisateurs représentatifs.

7.5 Documenter un composant IA

Le contenu typique de la documentation d'un composant d'IA comprend :

- Généralités : Identifiants, description, détails du développeur, exigences matérielles, détails de la licence, version, date et point de contact.
- Conception : Hypothèses et décisions techniques.
- Utilisation : Cas d'utilisation primaires et secondaires, utilisateurs types, approche de l'auto-apprentissage, biais connus, questions éthiques, questions de sûreté, transparence, seuils de décision, dérive de la plateforme et du concept.

- Ensembles de données : Caractéristiques, collecte, disponibilité, exigences de prétraitement, utilisation, contenu, étiquetage, taille, confidentialité, sécurité, partialité/équité et restrictions/contraintes.
- Test : Ensemble de données de test (description et disponibilité), indépendance des tests, résultats des tests, approche des tests pour la robustesse, l'explicabilité, la dérive des concepts et la portabilité.
- Apprentissage et performance fonctionnelle de ML : Algorithme de ML, pondérations, ensemble de données de validation, sélection des métriques de performance fonctionnelle de ML, seuils pour les métriques de performance fonctionnelle de ML, et métriques de performance fonctionnelle de ML réelles.

Une documentation claire permet d'améliorer les tests en assurant la transparence de la mise en œuvre du système basé sur l'IA. Les domaines clés de la documentation qui sont importants pour les tests sont :

- L'objectif du système, et la spécification des exigences fonctionnelles et non fonctionnelles. Ces types de documentation font généralement partie de la base de test.
- Les informations sur l'architecture et la conception, qui décrivent comment les différents composants IA et non-IA interagissent. Ces informations permettent d'identifier les objectifs des tests d'intégration et peuvent servir de base à des tests boîte blanche de la structure du système.
- La spécification de l'environnement d'exploitation. Ceci est nécessaire pour tester l'autonomie, la flexibilité et l'adaptabilité du système.
- La source de toutes les données d'entrée, y compris les métadonnées associées. Ceci doit être clairement compris lors des tests des aspects suivants :
    - L'exactitude fonctionnelle des entrées non fiables.
    - Biais explicite ou implicite de l'échantillon
    - Flexibilité, y compris l'apprentissage erroné dû à de mauvaises entrées de données pour les systèmes d'auto-apprentissage.
- La manière dont le système est censé s'adapter aux changements dans son environnement opérationnel. Ceci est nécessaire comme base de test lors des tests d'adaptabilité.
- Les détails sur les utilisateurs prévus du système. Ces informations sont nécessaires pour garantir la représentativité des tests.

7.6 Test de dérive du concept.

L'environnement opérationnel peut changer au fil du temps sans que le modèle entraîné ne change en conséquence. Ce phénomène est connu sous le nom de dérive conceptuelle et entraîne généralement une diminution de la précision et de l'utilité des sorties du modèle. Par exemple, l'impact des campagnes de marketing peut entraîner un changement de comportement des clients potentiels sur une période donnée. Il peut s'agir de changements saisonniers ou de changements brusques dus à des changements culturels, moraux ou sociétaux qui sont externes au système. Un exemple d'un tel changement abrupt est l'impact de la pandémie de COVID-19 et son effet sur l'exactitude des modèles utilisés pour les projections de ventes et les marchés boursiers.

Les systèmes susceptibles d'être sujets à la dérive conceptuelle doivent être régulièrement testés par rapport aux critères de performance fonctionnelle convenus dans le cadre du programme de contrôle de la qualité, afin de s'assurer que toute dérive conceptuelle est détectée suffisamment tôt pour que le problème puisse être atténué. Les mesures d'atténuation typiques peuvent comprendre la mise hors service du système ou un nouvel apprentissage du système. Dans le cas d'un réapprentissage, celui-ci sera effectué à l'aide de données d'apprentissage actualisées et sera suivi de tests de confirmation, de tests de régression et éventuellement d'une forme de test A/B (voir le paragraphe 9.4), dans lequel le système B mis à jour doit être plus performant que le système A original.

7.7 Sélection d'une approche de test pour un système ML.

Un système basé sur l'IA comprendra généralement des composants IA et non-IA. L'approche de test est basée sur une analyse des risques pour un tel système et comprendra à la fois des tests conventionnels et des tests plus spécialisés pour traiter les facteurs spécifiques aux composants IA et aux systèmes basés sur l'IA.

La liste suivante présente certains risques typiques et les mesures d'atténuation correspondantes, spécifiques aux systèmes ML. Il convient de noter que cette liste ne fournit qu'un nombre limité d'exemples et qu'il existe de nombreux autres risques spécifiques aux systèmes ML qui doivent être atténués par des tests.

Aspect du risque	**Description et mesures d'atténuation possibles**
La qualité des données peut être inférieure à celle attendue.	Ce risque peut devenir un problème de plusieurs façons, chacune pouvant être prévenue de différentes manières (voir section 4.4). Les mesures d'atténuation courantes comprennent le recours aux revues, à l'EDA et aux tests dynamiques.
Le pipeline de données opérationnelles peut être défectueux.	Ce risque peut être partiellement atténué par les tests dynamiques des différents composants du pipeline et les tests d'intégration du pipeline complet.
Le workflow ML utilisé pour développer le modèle peut être sous-optimal. (voir Section 3.2)	Ce risque pourrait être dû aux éléments suivants : - Un manque d'accord initial sur le workflow de ML à suivre. - Un mauvais choix de workflow. - Les data engineers ne suivent pas le workflow. Des revues avec des experts peuvent atténuer le risque de choisir le mauvais workflow, tandis qu'une gestion ou des audits plus pratiques pourraient résoudre les problèmes d'accord sur le workflow et de mise en œuvre de celui-ci.
Le choix du framework, de l'algorithme, du modèle, des paramètres du modèle et/ou des hyperparamètres du ML peut être sous-optimal.	Ce risque peut être dû à un manque d'expertise des décideurs, ou à une mauvaise mise en œuvre des étapes d'évaluation et de réglage (ou de test) du workflow de ML. Des revues avec des experts peuvent atténuer le risque de prendre de mauvaises décisions, et une meilleure gestion peut garantir le

Aspect du risque	Description et mesures d'atténuation possibles
	respect des étapes d'évaluation et de réglage (et de test) du workflow.
Il se peut que les critères de performance fonctionnelle souhaités pour le ML ne soient pas remplis sur le plan opérationnel, même si le composant ML répond à ces critères de manière isolée.	Ce risque pourrait être dû au fait que les ensembles de données utilisés pour l'apprentissage et le test du modèle ne sont pas représentatifs des données rencontrées dans la pratique. Les revues des ensembles de données sélectionnés par des experts (ou des utilisateurs) peuvent atténuer le risque que les données sélectionnées ne soient pas représentatives.
Les critères de performance fonctionnelle de ML sont respectés, mais les utilisateurs peuvent être mécontents des résultats obtenus.	Ce risque pourrait être dû à la sélection de mauvais critères de performance (par exemple, un rappel élevé a été sélectionné alors qu'une précision élevée était nécessaire). Des revues avec des experts peuvent atténuer le risque de choisir les mauvais critères de performance fonctionnels du ML, ou des tests basés sur l'expérience pourraient également identifier les critères inappropriés. Le risque pourrait également être dû à une dérive du concept, auquel cas des tests plus fréquents du système opérationnel pourraient atténuer le risque.
Les critères de performance fonctionnelle souhaités pour le ML sont respectés, mais les utilisateurs peuvent être mécontents du service fourni.	Ce risque pourrait être dû à un manque d'attention portée aux exigences non fonctionnelles du système). Il convient de noter que l'éventail des caractéristiques de qualité des systèmes basés sur l'IA va au-delà de celles énumérées dans la norme ISO/CEI 25010 (voir le chapitre 2). L'utilisation d'une approche fondée sur les risques pour hiérarchiser les caractéristiques de qualité et la réalisation des tests non fonctionnels pertinents pourraient atténuer ce risque. Par ailleurs, le problème pourrait être dû à une combinaison de facteurs qui pourraient être identifiés par des tests basés sur l'expérience, dans le cadre des tests du système. Le chapitre 8 fournit des conseils sur la manière de tester ces caractéristiques.
Le système d'auto-apprentissage peut ne pas fournir le service attendu par les utilisateurs.	Ce risque peut être dû à diverses raisons, par exemple : - Les données utilisées par le système pour l'auto-apprentissage peuvent être inadaptées. Dans ce cas, des revues d'experts pourraient identifier les données problématiques. - Le système peut échouer parce que la nouvelle fonctionnalité d'auto-apprentissage est inacceptable. Ce problème pourrait être atténué par des tests de régression automatisés comprenant une comparaison des performances avec la fonctionnalité précédente.

Aspect du risque	Description et mesures d'atténuation possibles
	- Le système peut apprendre d'une manière qui n'est pas attendue par les utilisateurs, ce qui pourrait être détecté par des tests basés sur l'expérience.
Les utilisateurs peuvent être frustrés de ne pas comprendre comment le système détermine ses décisions.	Ce risque pourrait être dû à un manque d'explicabilité, d'interprétabilité et/ou de transparence. Voir la section 8.6 pour plus de détails sur la manière de tester ces caractéristiques.
Les utilisateurs peuvent constater que le modèle fournit d'excellentes prédictions lorsque les données sont similaires aux données d'apprentissage, mais qu'il donne de mauvais résultats dans le cas contraire.	Ce risque peut être dû à un surajustement (voir section 3.5.1), qui peut être détecté en testant le modèle avec un ensemble de données totalement indépendant de l'ensemble de données d'apprentissage ou en effectuant des tests basés sur l'expérience.

8 Tester les caractéristiques de qualité spécifiques à l'IA – 150 minutes

Mots clés

Oracle de test

Mots-clés spécifiques à l'IA

Biais algorithmique, système autonome, autonomie, système expert, explicabilité, biais inapproprié, interprétabilité, méthode LIME - Local interpretable model-agnostic explanations, données d'apprentissage ML, système non déterministe, système probabiliste, biais d'échantillonnage, système d'auto-apprentissage, transparence.

Objectifs d'apprentissage pour le chapitre 8:

8.1 Les défis du test des systèmes d'auto-apprentissage

AI-8.1.1 K2 Expliquer les défis du test créés par l'auto-apprentissage des systèmes basés sur l'IA.

8.2 Test des systèmes autonomes basés sur l'IA

AI-8.2.1 K2 Décrire comment les systèmes autonomes basés sur l'IA sont testés.

8.3 Tester les biais algorithmiques, les biais d'échantillonnage et les biais inappropriés

AI-8.3.1 K2 Expliquer comment tester la présence de biais dans un système basé sur l'IA.

8.4 Les défis du test des systèmes probabilistes et non-déterministes basés sur l'IA

AI-8.4.1 K2 Expliquer les défis du test créés par la nature probabiliste et non déterministe des systèmes basés sur l'IA.

8.5 Les défis du test des systèmes complexes basés sur l'IA

AI-8.5.1 K2 Expliquer les défis du test créés par la complexité des systèmes basés sur l'IA.

8.6 Tester la transparence, l'interprétabilité et l'explicabilité des systèmes basés sur l'IA

AI-8.6.1 K2 Décrire comment la transparence, l'interprétabilité et l'explicabilité des systèmes basés sur l'IA peuvent être testées.

HO-8.6.1 H2 Utiliser un outil pour montrer comment l'explicabilité peut être utilisée par les testeurs.

8.7 Test d'oracles pour les systèmes basés sur l'IA

AI-8.7.1 K2 Expliquer les défis de la création d'oracles de test résultant des caractéristiques spécifiques des systèmes basés sur l'IA.

8.8 Objectifs des tests et critères d'acceptation

AI-8.8.1 K4 Sélectionner les objectifs de test et les critères d'acceptation appropriés pour les caractéristiques de qualité spécifiques à l'IA d'un système basé sur l'IA donné.

8.1 8.1 Les défis du test des systèmes d'auto-apprentissage.

Il existe plusieurs défis potentiels à surmonter lors du test des systèmes d'auto-apprentissage (voir le chapitre 2 pour plus de détails sur ces systèmes), notamment :

- Changement inattendu : Les exigences et contraintes initiales dans lesquelles le système doit fonctionner sont généralement connues, mais il peut y avoir peu ou pas d'informations disponibles sur les changements effectués par le système lui-même. Il est normalement possible d'effectuer des tests par rapport aux exigences et à la conception initiales (et à toutes les contraintes spécifiées), mais si le système a conçu une mise en œuvre innovante ou a joué avec une solution (dont la mise en œuvre ne peut pas être vue), il peut être difficile de concevoir des tests qui sont appropriés pour cette nouvelle mise en œuvre. En outre, lorsque les systèmes changent eux-mêmes (et leurs résultats), les résultats des tests précédemment réussis peuvent changer. C'est un défi pour la conception des tests. Il peut être résolu en concevant des tests appropriés qui restent pertinents lorsque le système change de comportement, évitant ainsi un problème potentiel de test de régression. Cependant, cela peut également nécessiter la conception de nouveaux tests basés sur les nouveaux comportements observés du système.
- Critères d'acceptation complexes : Il peut être nécessaire de définir les attentes en matière d'amélioration par le système lorsqu'il s'auto-apprend. Par exemple, on peut supposer que si le système se modifie lui-même, sa performance fonctionnelle globale devrait s'améliorer. En outre, spécifier autre chose qu'une simple "amélioration" peut rapidement devenir complexe. Par exemple, une amélioration minimale peut être attendue (plutôt que n'importe quelle amélioration), ou l'amélioration requise peut être liée à des facteurs environnementaux (par exemple, une amélioration minimale de 10 % de la fonctionnalité X est requise si le facteur environnemental F change de plus de Y). Ces problèmes peuvent être résolus par la spécification et les tests par rapport à des critères d'acceptation plus complexes, et par le maintien d'un enregistrement continu de la performance fonctionnelle de base du système actuel.
- Temps de test insuffisant : Il peut être nécessaire de savoir à quelle vitesse le système est censé apprendre et s'adapter, compte tenu de différents scénarios. Ces critères d'acceptation peuvent être difficiles à spécifier et à acquérir. Si un système s'adapte rapidement, le temps peut être insuffisant pour exécuter manuellement de nouveaux tests après chaque changement, il peut donc être nécessaire d'écrire des tests qui peuvent être exécutés automatiquement lorsque le système change lui-même. Ces défis peuvent être relevés par la spécification de critères d'acceptation appropriés (voir section 8.8) et par des tests continus automatisés.
- Exigences en matière de ressources : Les exigences du système peuvent inclure des critères d'acceptation pour les ressources que le système est autorisé à utiliser lors de l'auto-apprentissage ou de l'adaptation. Il peut s'agir, par exemple, de la quantité de temps de traitement et de mémoire autorisée à être utilisée pour s'améliorer. En outre, il faut se demander si l'utilisation de ces ressources doit être liée à une amélioration mesurable de la fonctionnalité ou de l'exactitude. Ce défi affecte la spécification des critères d'acceptation.
- Spécifications insuffisantes de l'environnement opérationnel : Un système d'auto-apprentissage peut changer si les entrées environnementales qu'il reçoit sont en dehors des plages attendues, ou si elles ne sont pas reflétées dans les données de formation. Ces entrées peuvent être des attaques sous forme d'altérations des données (voir section 9.1.2). Il peut être difficile de prévoir toute la gamme des environnements opérationnels et des changements environnementaux, et donc d'identifier l'ensemble complet des cas de test représentatifs et des exigences

environnementales. Idéalement, l'ensemble des changements possibles de l'environnement opérationnel auxquels le système est censé répondre sera défini comme critère d'acceptation.

- Environnement de test complexe : La gestion de l'environnement de test pour s'assurer qu'il peut imiter tous les changements potentiels à haut risque de l'environnement opérationnel est un défi et peut impliquer l'utilisation d'outils de test (par exemple, un outil d'injection de fautes). Selon la nature de l'environnement opérationnel, on peut effectuer des tests en manipulant les entrées et les capteurs, ou en obtenant l'accès à différents environnements physiques dans lesquels le système peut être testé.
- Modifications du comportement indésirable : Un système d'auto-apprentissage modifie son comportement en fonction de ses entrées et il peut être impossible pour les testeurs d'empêcher que cela se produise. Cela peut se produire, par exemple, si un système tiers est utilisé, ou si le système de production est testé. En répétant les mêmes tests, un système d'auto-apprentissage peut devenir plus efficace pour répondre à ces tests, ce qui peut ensuite influencer le comportement à long terme du système. Il est donc important d'éviter que les tests n'entraînent une modification négative du comportement du système d'auto-apprentissage. C'est un défi pour la conception des cas de test et la gestion des tests.

8.2 Test des systèmes autonomes basés sur l'IA.

Les systèmes autonomes doivent être capables de déterminer quand ils ont besoin d'une intervention humaine et quand ils n'en ont pas besoin. Par conséquent, pour tester l'autonomie des systèmes basés sur l'IA, il faut créer les conditions permettant au système d'exercer cette prise de décision.

Les tests d'autonomie peuvent nécessiter :

- Tester si le système demande une intervention humaine pour un scénario spécifique alors que le système devrait abandonner le contrôle. Ces scénarios peuvent inclure un changement de l'environnement opérationnel ou le dépassement des limites de l'autonomie du système.
- Tester si le système demande l'intervention humaine lorsque le système devrait renoncer au contrôle après une période de temps donnée.
- Tester si le système demande inutilement une intervention humaine alors qu'il devrait continuer à fonctionner de manière autonome.

Il peut être utile d'utiliser l'analyse des valeurs limites appliquée à l'environnement d'exploitation pour générer les conditions nécessaires à ces tests. Il peut être difficile de définir comment les paramètres qui déterminent l'autonomie se manifestent dans l'environnement d'exploitation et de créer les scénarios de test qui dépendent de la nature de l'autonomie.

8.3 Test pour le biais algorithmique, le biais d'échantillonnage et le biais inapproprié.

Un système de ML doit être évalué par rapport aux différents biais et des mesures doivent être prises pour éliminer les biais inappropriés. Cela peut impliquer l'introduction délibérée d'un biais positif pour contrer le biais inapproprié.

Le test avec un ensemble de données indépendant permet souvent de détecter les biais. Cependant, il peut être difficile d'identifier toutes les données à l'origine du biais, car l'algorithme de ML peut utiliser des combinaisons de caractéristiques apparemment sans rapport pour créer un biais indésirable.

Les systèmes basés sur l'IA doivent être testés pour détecter les biais algorithmiques, les biais d'échantillonnage et les biais inappropriés (voir section 2.4). Cela peut impliquer :

- Une analyse pendant les activités d'apprentissage, d'évaluation et de réglage du modèle pour identifier si un biais algorithmique est présent.
- La revue de la source des données d'apprentissage et des processus utilisés pour les acquérir, afin d'identifier la présence d'un biais d'échantillon.
- De revoir le prétraitement des données dans le cadre du workflow du ML afin d'identifier si les données ont été affectées d'une manière qui pourrait causer un biais d'échantillon.
- De mesurer comment les changements dans les entrées du système affectent les sorties du système sur un grand nombre d'interactions, et d'examiner les résultats sur la base des groupes de personnes ou d'objets pour lesquels le système peut être biaisé de manière inappropriée. Cette méthode est similaire à la méthode LIME (Local Interpretable Model-Agnostic Explanations) abordée en 8.6, et peut être réalisée dans un environnement de production ainsi que dans le cadre de tests avant la mise en production.
- D'obtenir des informations supplémentaires concernant les attributs des données d'entrée potentiellement liées au biais et les corréler aux résultats. Il peut s'agir de données démographiques, par exemple, qui peuvent être appropriées pour tester un biais inapproprié affectant des groupes de personnes, lorsque l'appartenance à un groupe est pertinente pour évaluer le biais mais ne constitue pas une entrée du modèle. En effet, le biais peut être basé sur des variables "cachées" qui ne sont pas explicitement présentes dans les données d'entrée, mais qui sont déduites par l'algorithme.

8.4 Les défis du test des systèmes probabilistes et non-déterministes basés sur l'IA.

La plupart des systèmes probabilistes sont également non-déterministes, et la liste suivante de défis de test s'applique donc généralement aux systèmes basés sur l'IA ayant l'un de ces attributs :

- Il peut y avoir plusieurs résultats valides d'un test avec le même ensemble de préconditions et d'entrées.
  Cela rend la définition des résultats attendus plus difficile et peut causer des difficultés :
  - Lorsque les tests sont réutilisés pour des tests de confirmation.
  - Lorsque les tests sont réutilisés pour des tests de régression.
  - Lorsque la reproductibilité des tests est importante.
  - Lorsque les tests sont automatisés.
- Le testeur a généralement besoin d'une connaissance plus approfondie du comportement requis du système afin de pouvoir proposer des vérifications raisonnables pour savoir si le test a réussi plutôt que de simplement énoncer une valeur exacte pour le résultat attendu du test. Par exemple, les testeurs peuvent avoir besoin de définir des résultats attendus plus sophistiqués par

rapport aux systèmes conventionnels. Ces résultats attendus peuvent inclure des tolérances (par exemple, "le résultat réel se situe-t-il à moins de 2 % de la solution optimale ?").

- Lorsqu'il n'est pas possible d'obtenir un seul résultat définitif d'un test en raison de la nature probabiliste du système, il est souvent nécessaire que le testeur exécute un test plusieurs fois afin de générer un résultat statistiquement valide.

8.5 Les défis du test des systèmes complexes basés sur l'IA.

Les systèmes basés sur l'IA sont souvent utilisés pour mettre en œuvre des tâches qui sont trop complexes pour les humains. Cela peut conduire à un problème d'oracle de test car les testeurs sont incapables de déterminer les résultats attendus comme ils le feraient normalement (voir section 8.7). Par exemple, les systèmes basés sur l'IA sont souvent utilisés pour identifier des modèles dans de grands volumes de données. Ces systèmes sont utilisés parce qu'ils peuvent trouver des modèles que les humains, même après de nombreuses analyses, ne peuvent tout simplement pas trouver manuellement. Il peut être difficile de comprendre le comportement requis de ces systèmes de manière suffisamment approfondie pour pouvoir générer les résultats attendus.

Un problème similaire se pose lorsque la structure interne d'un système basé sur l'IA est générée par un logiciel, ce qui la rend trop complexe pour être comprise par les humains. Cela conduit à une situation où le système basé sur l'IA ne peut être testé que comme une boîte noire. Même lorsque la structure interne est visible, elle ne fournit aucune information supplémentaire utile pour faciliter les tests.

La complexité des systèmes basés sur l'IA augmente lorsqu'ils fournissent des résultats probabilistes et sont non déterministes par nature (voir section 8.4).

Les problèmes liés aux systèmes non déterministes sont exacerbés lorsqu'un système basé sur l'IA est constitué de plusieurs composants en interaction, chacun fournissant des résultats probabilistes. Par exemple, un système de reconnaissance faciale est susceptible d'utiliser un modèle pour identifier un visage dans une image, et un second modèle pour reconnaître quel visage a été identifié. Les interactions entre les composants de l'IA peuvent être complexes et difficiles à comprendre, ce qui rend difficile l'identification de tous les risques et la conception de tests qui vérifient le système de manière adéquate.

8.6 Test de la transparence, de l'interprétabilité et de l'explicabilité des systèmes basés sur l'IA.

Les informations sur la manière dont le système a été mis en œuvre peuvent être fournies par les développeurs du système. Il peut s'agir des sources de données d'apprentissage, de la façon dont l'étiquetage a été effectué et de la façon dont les composants du système ont été conçus. Lorsque ces informations ne sont pas disponibles, cela peut rendre la conception des tests difficile. Par exemple, si les informations sur les données d'apprentissage ne sont pas disponibles, il devient difficile d'identifier les lacunes potentielles dans ces données et de tester l'impact de ces lacunes. Cette situation peut être comparée aux tests boîte noire et boîte blanche, et présente les mêmes avantages et inconvénients. La transparence peut être testée en comparant les informations documentées sur les données et l'algorithme à la mise en œuvre réelle et en déterminant dans quelle mesure elles correspondent.

Avec le ML, il est souvent plus difficile d'expliquer le lien entre une entrée spécifique et une sortie spécifique qu'avec les systèmes conventionnels. Ce faible niveau d'explicabilité est principalement dû au fait que le modèle générant la sortie est lui-même généré par du code (l'algorithme) et ne reflète pas la

façon dont les humains pensent à un problème. Les différents modèles de ML offrent différents niveaux d'explicabilité et doivent être sélectionnés en fonction des exigences du système, qui peuvent inclure l'explicabilité et la testabilité.

Une méthode pour comprendre l'explicabilité consiste à tester dynamiquement le modèle ML en appliquant des perturbations aux données de test. Il existe des méthodes permettant de quantifier l'explicabilité de cette manière et d'en fournir des explications visuelles. Certaines de ces méthodes sont indépendantes du modèle, tandis que d'autres sont spécifiques à un type particulier de modèle et nécessitent d'y avoir accès. Les tests exploratoires peuvent également être utilisés pour mieux comprendre la relation entre les entrées et les sorties d'un modèle.

La méthode LIME est agnostique par rapport au modèle et utilise des perturbations d'entrée injectées dynamiquement et l'analyse des sorties pour fournir aux testeurs une vue de la relation entre les entrées et les sorties. Cette méthode peut être efficace pour fournir une explication du modèle. Cependant, elle se limite à fournir des raisons possibles pour les sorties, plutôt qu'une raison définitive, et n'est pas applicable à tous les types d'algorithmes.

L'interprétabilité d'un système basé sur l'IA dépend fortement du public auquel il s'adresse. Les différentes parties prenantes peuvent avoir des exigences différentes en termes de degré de compréhension de la technologie sous-jacente.

Mesurer et tester le niveau de compréhension pour l'interprétabilité et l'explicabilité peut s'avérer difficile car les parties prenantes n'ont pas toutes le même niveau de capacité et peuvent ne pas être d'accord. En outre, il peut être difficile d'identifier le profil des parties prenantes types pour de nombreux types de systèmes. Lorsqu'ils sont réalisés, ces tests prennent généralement la forme d'enquêtes et/ou de questionnaires auprès des utilisateurs.

8.6.1 Exercice pratique : Explicabilité du modèle.

Utiliser un outil approprié pour fournir une explicabilité basée sur le modèle précédemment créé. Par exemple, pour un modèle de classification d'images ou un modèle de classification de textes, une méthode agnostique, telle que LIME, peut être appropriée.

Les candidats devraient utiliser l'outil pour générer des explications sur les décisions du modèle, en particulier sur la manière dont les caractéristiques des entrées influencent les sorties.

8.7 Oracles de test pour les systèmes basés sur l'IA

Un problème majeur dans les tests des systèmes basés sur l'IA peut être la spécification des résultats attendus. Un oracle de test est la source utilisée pour déterminer le résultat attendu d'un test [I01]. Ce défi dans la détermination des résultats attendus est connu sous le nom de problème de l'oracle de test.

Avec des systèmes complexes, non déterministes ou probabilistes, il peut être difficile d'établir un oracle de test sans connaître la "vérité du terrain" (c'est-à-dire le résultat réel dans le monde réel que le système basé sur l'IA essaie de prédire). Cette "vérité de base" est différente d'un oracle de test, dans la mesure où un oracle de test ne fournit pas nécessairement une valeur attendue, mais seulement un mécanisme permettant de déterminer si le système fonctionne correctement ou non.

Les systèmes basés sur l'IA peuvent évoluer (voir section 2.3), et le test des systèmes d'auto-apprentissage (voir section 8.1) peut également souffrir de problèmes d'oracle de test car ils se modifient eux-mêmes et peuvent ainsi rendre nécessaire une mise à jour fréquente des attentes fonctionnelles du système.

Une autre cause de difficulté pour obtenir un oracle de test efficace est que, dans de nombreux cas, l'exactitude du comportement du logiciel est subjective. Les assistants virtuels (par exemple Siri et Alexa) sont un exemple de ce problème dans la mesure où différents utilisateurs ont souvent des attentes très différentes et peuvent obtenir des résultats différents en fonction de leur choix de mots et de la clarté de leur discours.

Dans certaines situations, il peut être possible de définir le résultat attendu avec des limites ou des tolérances. Par exemple, le point d'arrêt d'une voiture autonome pourrait être défini comme se trouvant à une distance maximale d'un point spécifique. Dans le contexte des systèmes experts, la détermination des résultats attendus peut être réalisée en consultant un expert (en notant que l'avis de l'expert peut toujours être erroné). Il y a plusieurs facteurs importants à prendre en compte dans de telles circonstances :

- Les experts humains varient dans leurs niveaux de compétence. Les experts impliqués doivent être au moins aussi compétents que les experts que le système est censé remplacer.
- Les experts peuvent ne pas être d'accord entre eux, même lorsqu'on leur présente les mêmes informations.
- Les experts humains peuvent ne pas approuver l'automatisation de leur jugement. Dans ce cas, leur évaluation des résultats potentiels doit se faire en double aveugle (c'est-à-dire que ni les experts ni les évaluateurs des résultats ne doivent savoir quelles évaluations ont été automatisées).
- Les humains sont plus susceptibles de relativiser les réponses (par exemple, avec des phrases comme "Je ne suis pas sûr, mais..."). Si ce type d'avertissement n'est pas disponible pour le système basé sur l'IA, il faut en tenir compte lors de la comparaison des réponses.

Il existe des techniques de test qui peuvent atténuer le problème de l'oracle des tests, comme les tests A/B (voir section 9.4), les tests dos-à-dos (voir section 9.3) et les tests métamorphiques (voir section 9.5).

8.8 Objectifs de test et critères d'acceptation

Les objectifs de test et les critères d'acceptation d'un système doivent être basés sur les risques perçus du produit. Ces risques peuvent souvent être identifiés à partir d'une analyse des caractéristiques de qualité requises. Les caractéristiques de qualité d'un système basé sur l'IA comprennent celles qui sont traditionnellement prises en compte dans la norme ISO/CEI 25010 [S06] (c'est-à-dire l'adéquation fonctionnelle, l'efficience des performances, la compatibilité, la facilité d'utilisation, la fiabilité, la sécurité, la maintenabilité et la portabilité), mais elles doivent également prendre en compte les aspects suivants :

Aspect	**Critères d'acceptation**
Adaptabilité	• Vérifier que le système fonctionne toujours correctement et répond aux exigences non fonctionnelles lorsqu'il s'adapte à un changement dans son environnement. Cela peut être mis en œuvre sous la forme de tests de régression automatisés.

Aspect	Critères d'acceptation
	• Vérifier le temps que prend le système pour s'adapter à un changement dans son environnement. • Vérifier les ressources utilisées lorsque le système s'adapte à un changement d'environnement.
Flexibilité	• Examiner comment le système se comporte dans des contextes extérieurs à la spécification initiale. Cela peut prendre la forme de tests de régression automatisés exécutés dans l'environnement opérationnel modifié. • Vérifier le temps que prend le système et/ou les ressources utilisées pour se modifier afin de gérer un nouveau contexte.
Evolution	• Vérifier dans quelle mesure le système apprend de sa propre expérience. • Vérifier la capacité du système à faire face aux changements de profil des données (c'est-à-dire la dérive des concepts).
Autonomie	• Vérifier comment le système réagit lorsqu'il est contraint de sortir de l'enveloppe opérationnelle dans laquelle il est censé être totalement autonome. • Vérifier si le système peut être "persuadé" de demander une intervention humaine alors qu'il devrait être totalement autonome.
Transparence, interprétabilité et explicabilité	• Vérifier la transparence en examinant la facilité d'accès à l'algorithme et à l'ensemble des données. • Vérifier l'interprétabilité et l'explicabilité en interrogeant les utilisateurs du système ou, si ces derniers ne sont pas disponibles, des personnes ayant une expérience similaire.
Absence de préjugés inappropriés	• Lorsque les systèmes sont susceptibles d'être affectés par un biais, on peut le tester en utilisant une suite de tests indépendants sans biais, ou en faisant appel à des revues d'experts. • Comparer les résultats des tests à l'aide de données externes, telles que les données de recensement, afin de vérifier l'absence de biais indésirable sur les variables déduites (test de validité externe).
Ethique	• Vérifier le système à l'aide d'une liste de contrôle appropriée, telle que la liste d'évaluation de la Commission Européenne pour l'intelligence artificielle digne de confiance [R21], qui prend en charge les exigences clés décrites dans les lignes directrices en matière d'éthique pour l'intelligence artificielle digne de confiance [R22].

Aspect	Critères d'acceptation
Systèmes probabilistes et systèmes non déterministes	• Ceci ne peut pas être évalué avec des critères d'acceptation précis. Lorsqu'il fonctionne correctement, le système peut donner des résultats légèrement différents pour les mêmes tests.
Effets secondaires	• Identifier les effets secondaires potentiellement dangereux et tenter de générer des tests qui amènent le système à présenter ces effets secondaires.
Piratage de récompense	• Des tests indépendants peuvent identifier le piratage de récompense lorsque ces tests utilisent un moyen différent de mesurer le succès par rapport à l'agent intelligent testé.
Sûreté	• Cela doit être soigneusement évalué, peut-être dans un environnement de test virtuel (voir section 10.2). Il peut s'agir de tentatives pour forcer un système à se mettre en défaut.

Pour les systèmes ML, les métriques de performance fonctionnelle requises pour le modèle ML doivent être spécifiées (voir chapitre 5).

9 Méthodes et techniques pour le test des systèmes basés sur l'IA – 245 minutes

Mots clés

A/B testing, test adverse, test dos à dos, estimation d'erreur, test base sur l'expérience, test exploratoire, relation métamorphique (MR – Metamorphic Relation), test métamorphique (MT - Metamorphic Testing), test par paire, pseudo-oracle, problème de l'oracle de test, tours

Mots-clés spécifiques à l'IA

Attaque adverse, exemple adverse, empoisonnement des données, système ML, modèle entraîné

Objectifs d'apprentissage pour le chapitre 9 :

9.1 Attaques adverses et empoisonnement des données

AI-9.1.1 K2 Expliquer comment le test des systèmes ML peut aider à prévenir les attaques adverses et l'empoisonnement des données.

9.2 Test par paires

AI-9.2.1 K2 Expliquer comment les tests par paires sont utilisés pour les systèmes basés sur l'IA.

HO-9.2.1 H2 Appliquer le test par paires pour dériver et exécuter des cas de test pour un système basé sur l'IA.

9.3 Les tests dos à dos

AI-9.3.1 K2 Expliquer comment les tests dos à dos sont utilisés pour les systèmes basés sur l'IA.

9.4 Le test A/B

AI-9.4.1 K2 Expliquer comment le test A/B est appliqué au test des systèmes basés sur l'IA.

9.5 Le test métamorphique

AI-9.5.1 K3 Appliquer le test métamorphique pour le test des systèmes basés sur l'IA.

HO-9.5.1 H2 Appliquer le test métamorphique pour dériver des cas de test pour un scénario donné et les exécuter.

9.6 Test basé sur l'expérience des systèmes basés sur l'IA

AI-9.6.1 K2 Expliquer comment le test basé sur l'expérience peut être appliqué au test des systèmes basés sur l'IA.

HO-9.6.1 H2 Appliquer les tests exploratoires à un système basé sur l'IA.

9.7 Sélectionner les techniques de test pour les systèmes basés sur l'IA

AI-9.7.1 K4 Pour un scénario donné, choisir les techniques de test appropriées pour tester un système basé sur l'IA.

.

9.1 Attaques adverses et empoisonnement des données.

9.1.1 Attaques adverses

Une attaque adverse consiste pour un attaquant à perturber subtilement les entrées valides qui sont transmises au modèle formé pour l'amener à fournir des prédictions incorrectes. Ces entrées perturbées, connues sous le nom d'exemples adverse, ont été remarquées pour la première fois avec les filtres anti-spam, qui pouvaient être trompés en modifiant légèrement un courriel sans en perdre la lisibilité. Récemment, ils ont été davantage associés aux classificateurs d'images. En changeant simplement quelques pixels invisibles à l'œil humain, il est possible de persuader un réseau neuronal de changer sa classification d'image pour un objet très différent et avec un haut degré de confiance.

Les exemples adverses sont généralement transférables [B17], ce qui signifie qu'un exemple adverse qui fait échouer un système ML fera souvent échouer un autre système ML entraîné pour effectuer la même tâche. Même si le deuxième système ML a été entraîné avec des données différentes et est basé sur des architectures différentes, il reste souvent sujet à l'échec avec les mêmes exemples adverses.

Les attaques adverses de type "boîte blanche" sont celles où l'attaquant sait quel algorithme a été utilisé pour entraîner le modèle et quels paramètres ont été utilisés (il y a un niveau raisonnable de transparence). L'attaquant utilise ces connaissances pour générer des exemples adverses, par exemple en apportant de petites perturbations aux entrées et en surveillant celles qui entraînent des changements importants dans les sorties du modèle.

Les attaques adverses de type "boîte noire" impliquent que l'attaquant explore le modèle pour déterminer sa fonctionnalité, puis qu'il construise un modèle dupliqué offrant une fonctionnalité similaire. L'attaquant utilise ensuite une approche en boîte blanche pour identifier des exemples adverses pour ce modèle dupliqué. Comme les exemples adverses sont généralement transférables, les mêmes exemples adverses fonctionneront normalement aussi sur le modèle original.

S'il n'est pas possible de créer un modèle dupliqué, il est possible d'utiliser des tests automatisés à grande échelle pour découvrir différents exemples adverses et observer les résultats.

Le test adverse consiste simplement à effectuer des attaques adverses dans le but d'identifier les vulnérabilités afin de prendre des mesures préventives pour se protéger contre de futures défaillances. Les exemples adverses identifiés sont ajoutés aux données d'apprentissage afin que le modèle soit entraîné à les reconnaître correctement.

9.1.2 Empoisonnement des données

Les attaques par empoisonnement des données consistent pour un attaquant à manipuler les données d'apprentissage pour obtenir l'un des deux résultats suivants. L'attaquant peut insérer des portes dérobées ou des chevaux de Troie de réseau neuronal pour faciliter de futures intrusions, ou plus souvent, il utilisera des données d'apprentissage corrompues (par exemple, des données mal étiquetées) pour inciter le modèle formé à fournir des prédictions incorrectes.

Les attaques par empoisonnement peuvent être ciblées dans le but d'amener le système ML à mal classer dans des situations spécifiques. Elles peuvent également être indiscriminées, comme dans le cas d'une attaque par déni de service. Un exemple bien connu d'attaque par empoisonnement est l'altération du chatbot Microsoft Tay, dans lequel un nombre relativement faible de conversations Twitter nuisibles a entraîné le système, par le biais du retour d'information, à fournir des conversations altérées à l'avenir. Une forme courante d'attaque par empoisonnement des données consiste à signaler

faussement des millions de courriels non sollicités comme n'étant pas des pourriels (spam), dans le but de fausser les logiciels de filtrage des pourriels. Le risque d'empoisonnement des données est lié à la possibilité que des ensembles de données d'IA publics et largement utilisés soient empoisonnés.

Il est possible d'effectuer des tests pour détecter l'empoisonnement des données à l'aide de l'EDA, car les données empoisonnées peuvent apparaître comme des valeurs aberrantes. En outre, les politiques d'acquisition de données peuvent être revues pour s'assurer de la provenance des données d'apprentissage. Lorsqu'un système de ML opérationnel peut être attaqué en lui fournissant des données empoisonnées, des tests A/B (voir section 9.4) peuvent être utilisés pour vérifier que la version actualisée du système est toujours étroitement alignée sur la version précédente. Par ailleurs, les tests de régression d'un système mis à jour à l'aide d'une suite de tests fiables peuvent également permettre de déterminer si un système a été empoisonné.

9.2 Test par paires.

Le nombre de paramètres d'intérêt pour un système basé sur l'IA peut être extrêmement élevé, notamment lorsque le système utilise des données volumineuses ou interagit avec le monde extérieur, comme une voiture à conduite autonome. Pour effectuer des tests exhaustifs, il faudrait tester toutes les combinaisons possibles de ces paramètres réglés sur toutes les valeurs possibles. Cependant, comme cela entraînerait un nombre pratiquement infini de tests, des techniques de test sont utilisées pour sélectionner un sous-ensemble qui peut être exécuté dans le temps limité disponible.

Lorsqu'il est possible de combiner de nombreux paramètres, chacun d'entre eux pouvant avoir de nombreuses valeurs discrètes, les tests combinatoires peuvent être appliqués pour réduire de manière significative le nombre de cas de test requis, idéalement sans compromettre la capacité de détection des défauts de la suite de tests. Il existe plusieurs techniques de tests combinatoires (voir [I02] et [S08]). Cependant, dans la pratique, le test par paires est la technique la plus utilisée car elle est facile à comprendre et dispose d'un large support d'outils. En outre, la recherche a montré que la plupart des défauts sont causés par des interactions impliquant peu de paramètres [B33].

En pratique, même l'utilisation des tests par paires peut donner lieu à des suites de tests étendues pour certains systèmes, et l'utilisation de l'automatisation et des environnements de test virtuels (voir section 10.2) devient souvent nécessaire pour permettre l'exécution du nombre nécessaire de tests. Par exemple, en ce qui concerne les voitures à conduite autonome, les scénarios de test de haut niveau pour les tests de systèmes doivent porter à la fois sur les différents environnements dans lesquels les voitures sont censées fonctionner et sur les diverses fonctions du véhicule. Ainsi, les paramètres doivent inclure l'ensemble des contraintes environnementales (par exemple, les types de routes et leurs revêtements, les conditions météorologiques et de circulation, et la visibilité) et les différentes fonctions de conduite autonome (par exemple, le régulateur de vitesse adaptatif, l'assistance au maintien dans la voie et l'assistance au changement de voie). Outre ces paramètres, les données fournies par les capteurs pourraient être prises en compte à différents niveaux d'efficacité (par exemple, les données fournies par une caméra vidéo se dégradent au fur et à mesure de la progression du trajet et de la saleté).

À l'heure actuelle, les recherches ne permettent pas de déterminer le niveau de rigueur nécessaire à l'utilisation de tests combinatoires avec des systèmes basés sur l'intelligence artificielle critiques pour la sûreté, tels que les voitures à conduite autonome. Même si les tests par paires ne sont peut-être pas suffisants, on sait que cette approche est efficace pour détecter les défauts.

.

9.2.1 Exercice pratique : tests par paires

Pour un système d'IA implémenté avec un minimum de cinq paramètres et au moins cinq cents combinaisons possibles, utiliser un outil de test par paires pour identifier un ensemble réduit de combinaisons par paires et exécuter des tests pour ces combinaisons. Comparez le nombre de combinaisons par paires testées avec le nombre requis si toutes les combinaisons théoriquement possibles devaient être testées.

9.3 Test dos à dos

Une des solutions potentielles au problème de l'oracle de test (voir Section 8.7) lors du test de systèmes basés sur l'IA est d'utiliser le test dos à dos. Cette méthode est également connue sous le nom de test différentiel. Avec le test dos à dos, une version alternative du système est utilisée comme pseudo-oracle et ses sorties sont comparées aux résultats de test produits par le SUT. Le pseudo-oracle peut être un système existant ou être développé par une autre équipe, éventuellement sur une autre plate-forme, avec une autre architecture et un autre langage de programmation. Lors du test de l'adéquation fonctionnelle (par opposition aux exigences non fonctionnelles), le système utilisé comme pseudo-oracle n'est pas contraint d'atteindre les mêmes critères d'acceptation non fonctionnels que le SUT. Par exemple, il peut ne pas avoir à s'exécuter aussi rapidement, auquel cas il peut être beaucoup moins coûteux à construire.

Dans le contexte du ML, il est possible d'utiliser différents frameworks, algorithmes et paramètres de modèle pour créer un pseudo-oracle ML. Dans certaines situations, il peut également être possible de créer un pseudo-oracle à l'aide d'un logiciel conventionnel, non IA.

Pour que les pseudo-oracles soient efficaces dans la détection des défauts, il ne doit pas y avoir de logiciel commun dans le pseudo-oracle et le SUT. Sinon, il serait possible qu'un même défaut dans les deux entraîne la concordance des résultats des deux tests alors que les deux sont défectueux. Étant donné le grand nombre de logiciels d'IA immatures, réutilisables et libres utilisés pour développer des systèmes basés sur l'IA, la réutilisation du code entre le pseudo-oracle et le SUT peut compromettre le pseudo-oracle. Une mauvaise documentation des solutions d'IA réutilisables peut également rendre difficile la reconnaissance de ce problème par les testeurs.

9.4 Test A/B

Le test A/B est une méthode qui consiste à comparer la réponse de deux variantes d'un programme (A et B) aux mêmes entrées dans le but de déterminer laquelle des deux variantes est la meilleure. Il s'agit d'une approche de test statistique qui nécessite généralement la comparaison des résultats de plusieurs tests pour déterminer la différence entre les programmes.

Un exemple simple de cette méthode est celui où deux offres promotionnelles sont envoyées par e-mail à une liste de marketing divisée en deux ensembles. La moitié de la liste reçoit l'offre A, l'autre moitié l'offre B, et le succès de chaque offre permet de décider laquelle utiliser à l'avenir. De nombreuses entreprises de commerce électronique et de sites web utilisent le test A/B en production, en dirigeant différents consommateurs vers différentes fonctionnalités, pour aider à identifier les préférences des consommateurs.

Le test A/B est une approche permettant de résoudre le problème de l'oracle de test, où le système existant est utilisé comme un oracle partiel. Le test A/B ne génère pas de cas de test et ne fournit aucune indication sur la manière dont les tests doivent être conçus, bien que les entrées opérationnelles soient souvent utilisées dans les tests.

Le test A/B peut être utilisé pour tester les mises à jour d'un système basé sur l'IA lorsqu'il existe des critères d'acceptation convenus, tels que les métriques de performance fonctionnelle de ML, comme décrit au chapitre 5. Chaque fois que le système est mis à jour, le test A/B est utilisé pour vérifier que la variante mise à jour fonctionne aussi bien, voire mieux, que la variante précédente. Une telle approche peut être utilisée pour un simple classificateur, mais aussi pour tester des systèmes beaucoup plus complexes. Par exemple, une mise à jour visant à améliorer l'efficacité d'un système d'acheminement des transports d'une ville intelligente peut également être testée à l'aide de tests A/B (par exemple, en comparant les temps de trajet moyens de deux variantes du système sur des semaines consécutives).

Le test A/B peut également être utilisé pour tester les systèmes d'auto-apprentissage. Lorsque le système apporte un changement, des tests automatisés sont exécutés et les caractéristiques du système qui en résultent sont comparées à celles d'avant le changement. Si le système est amélioré, alors le changement est accepté, sinon le système revient à son état précédent.

Une différence majeure entre les tests A/B et les tests dos à dos concerne l'utilisation des tests A/B pour comparer deux variantes d'un même système et l'utilisation des tests dos à dos pour détecter les défauts.

9.5 Test métamorphique (MT)

Le test métamorphique [B18] est une technique visant à générer des cas de test qui sont basés sur un cas de test source qui a réussi. Un ou plusieurs cas de test de suivi sont générés en modifiant (métamorphosant) le cas de test source sur la base d'une relation métamorphique (MR).
La MR est basée sur une propriété d'une fonction requise de l'objet de test, de sorte qu'elle décrit comment un changement dans les entrées de test d'un scénario de test se reflète dans les résultats attendus du même scénario de test.

Par exemple, considérons un programme qui détermine la moyenne d'un ensemble de nombres. Un scénario de test source est généré, comprenant un ensemble de nombres et une moyenne attendue, et le scénario de test est exécuté pour confirmer qu'il réussit. Il est maintenant possible de générer de nouveaux cas de test basés sur ce que l'on sait de la fonction moyenne du programme. Au départ, l'ordre des nombres dont la moyenne est calculée peut simplement être modifié. Étant donné la fonction de moyenne, on peut prévoir que le résultat attendu restera le même. Ainsi, un scénario de test de suivi avec les nombres dans un ordre différent peut être généré sans avoir à calculer le résultat attendu. Avec un grand ensemble de nombres, cela pourrait conduire à générer un grand nombre d'ensembles de nombres différents dans lesquels les mêmes nombres sont utilisés dans des séquences différentes et chacun d'entre eux pourrait être utilisé pour créer un scénario de test distinct. Tous ces cas de test seraient basés sur le même cas de test source et auraient le même résultat attendu.

Il est courant d'avoir des MR et des nouveaux cas de test dont le résultat attendu est différent du résultat attendu original du cas de test source. Par exemple, en utilisant la même fonction moyenne, on peut obtenir une MR dans laquelle chaque élément de l'ensemble d'entrée est multiplié par deux. Le résultat attendu pour un tel ensemble est simplement le résultat attendu original multiplié par deux. De même, toute autre valeur peut être utilisée comme multiplicateur pour générer potentiellement un nombre infini de cas de test de suivi basés sur cette MR.

Le MT peut être utilisée pour la plupart des objets de test et peut être appliquée aux tests fonctionnels et non fonctionnels (par exemple, les tests d'installabilité couvrent différentes configurations cibles où les paramètres d'installation peuvent être sélectionnés dans différentes séquences). Il est particulièrement utile lorsque la génération des résultats attendus est problématique, en raison de l'absence d'un oracle

de test peu coûteux. C'est le cas de certains systèmes basés sur l'IA qui reposent sur l'analyse de données volumineuses, ou de ceux pour lesquels les testeurs ne savent pas très bien comment l'algorithme ML obtient ses prédictions. Dans le domaine de l'IA, le test métamorphique a été utilisé pour tester la reconnaissance d'images, les moteurs de recherche, l'optimisation d'itinéraires et la reconnaissance vocale, entre autres.

Comme expliqué ci-dessus, le MT peut être basée sur un cas de test source réussi, mais il est également utile s'il n'est pas possible de vérifier qu'un cas de test source est correct. Cela peut être le cas, par exemple, lorsque le programme met en œuvre une fonction qui est trop complexe pour qu'un testeur humain puisse la reproduire et l'utiliser comme oracle de test, comme c'est le cas avec certains systèmes basés sur l'IA. Dans cette situation, le MT peut être utilisé pour générer un ou plusieurs cas de test qui, lorsqu'ils sont exécutés, créent un ensemble de résultats dont la validité des relations entre les résultats peut être vérifiée. Avec cette forme de MT, on ne sait pas si les tests individuels sont corrects, mais les relations entre eux doivent être vraies, ce qui améliore la confiance dans le programme. Un exemple pourrait être un programme actuariel basé sur l'IA qui prédit l'âge au décès sur la base d'un large ensemble de données, où l'on sait, par exemple, que si le nombre de cigarettes fumées est augmenté, l'âge au décès prédit devrait diminuer (ou, au moins, rester le même).

Le MT est une technique de test relativement nouvelle, proposée pour la première fois en 1998. Elle diffère des techniques de test traditionnelles en ce que les résultats attendus des cas de test de suivi ne sont pas décrits en termes de valeurs absolues, mais sont relatifs aux résultats attendus dans le cas de cas de test sources. Elle est basée sur un concept facile à comprendre, qui peut être appliquée par des testeurs ayant peu d'expérience dans l'application de la technique mais qui comprennent le domaine d'application, et a des coûts similaires à ceux des techniques traditionnelles. Elle est également efficace pour révéler les défauts, les recherches montrant que seulement trois à six MR diverses peuvent révéler plus de 90% des défauts qui pourraient être détectés en utilisant des techniques basées sur un oracle de test traditionnel [B19]. Il est possible de générer automatiquement des cas de test à partir de MRs bien spécifiées et d'un cas de test source. Cependant, aucun outil commercial n'est actuellement disponible, bien que Google applique déjà le MT automatisé pour tester les pilotes graphiques d'Android à l'aide de l'outil GraphicsFuzz, dont le code source est ouvert (voir [R23]).

9.5.1 Exercice pratique : Test métamorphique

Dans cet exercice, les participants acquerront une expérience pratique des points suivants :

- Déterminer plusieurs relations métamorphiques (MRs) pour une application ou un programme d'IA donné. Ces MRs devraient inclure des cas où les résultats attendus des cas de test source et de suivi sont les mêmes et d'autres où ils sont différents.
- Générer des cas de test source pour l'application ou le programme basé sur l'IA. Il n'est pas nécessaire de garantir la réussite de ces tests, mais il convient de rappeler aux participants les limites du MT lorsqu'il n'existe pas d'"étalon-or".
- Utiliser les MRs dérivées et les cas de test source générés pour dériver les nouveaux cas de test.
- Exécution des nouveaux cas de test.

9.6 Test basé sur l'expérience des systèmes basés sur l'IA.

Les tests basés sur l'expérience comprennent l'estimation d'erreurs, les tests exploratoires et les tests basés sur des checklists [I01], qui peuvent tous être appliqués aux tests de systèmes basés sur l'IA.

L'estimation d'erreurs est typiquement basée sur les connaissances des testeurs, les erreurs typiques des développeurs, et les défaillances dans des systèmes similaires (ou des versions précédentes). Un exemple d'estimation d'erreur appliquée aux systèmes basés sur l'IA pourrait être l'utilisation de connaissances sur la façon dont les systèmes ML ont échoué dans le passé en raison de l'utilisation de données d'apprentissage systématiquement biaisées.

Dans les tests exploratoires, les tests sont conçus, générés et exécutés de manière itérative, avec la possibilité de dériver des tests ultérieurs, sur la base des résultats des tests précédents. Les tests exploratoires sont particulièrement utiles en cas de spécifications médiocres ou de problèmes d'oracle de test, ce qui est souvent le cas pour les systèmes basés sur l'IA. Par conséquent, les tests exploratoires sont souvent utilisés dans ce contexte et devraient être utilisés pour compléter les tests plus systématiques basés sur des techniques, telles que les tests métamorphiques (voir Section 9.5).

Un tour est une métaphore utilisée pour un ensemble de stratégies et d'objectifs auxquels les testeurs peuvent se référer lorsqu'ils effectuent des tests exploratoires organisés autour d'un centre d'intérêt particulier [B20]. Des sessions typiques pour le test exploratoire de systèmes basés sur l'IA peuvent se concentrer sur les concepts de biais, de sous-ajustement et de surajustement dans les systèmes ML. Par exemple, une session basée sur les données pourrait être appliquée pour tester le modèle. Au cours de cette session, le testeur pourrait identifier les différents types de données utilisées pour l'apprentissage, leur distribution, leurs variations, leur format et leurs plages, etc. et ensuite utiliser les types de données pour tester le modèle.

Les systèmes ML dépendent fortement de la qualité des données d'apprentissage, et le domaine existant de l'EDA est étroitement lié à l'approche des tests exploratoires. L'EDA consiste à examiner les données à la recherche de modèles, de relations, de tendances et de valeurs aberrantes. Elle implique l'exploration interactive et pilotée par des hypothèses des données et est décrite dans [B21] comme : "Nous explorons les données avec des attentes. Nous révisons nos attentes en fonction de ce que nous voyons dans les données. Et nous itérons ce processus". L'EDA nécessite généralement un support d'outils dans deux domaines ; pour l'interaction avec les données, afin de permettre aux analystes de mieux comprendre les données complexes, et pour la visualisation des données, afin de leur permettre d'afficher facilement les résultats d'analyse. L'utilisation de techniques exploratoires, principalement pilotées par la visualisation des données, peut aider à valider l'algorithme ML utilisé, à identifier les changements qui permettent d'obtenir des modèles efficaces et à tirer parti de l'expertise du domaine [B22].

Google dispose d'un ensemble de vingt-huit tests ML écrits sous forme d'assertions, dans les domaines des données, du développement de modèles, de l'infrastructure et de la surveillance, qui est utilisé comme checklist de test au sein de Google pour les systèmes ML [B23]. La "checklist de test ML" de Google est présentée ici telle que publiée par Google :

Données ML:

1. Les attentes en matière de fonctionnalités sont capturées dans un schéma.
2. Toutes les fonctionnalités sont bénéfiques.
3. Le coût d'aucune fonctionnalité n'est trop élevé.

4. Les fonctionnalités sont conformes aux exigences du méta-niveau.
5. Le pipeline de données comporte des contrôles de confidentialité appropriés.
6. De nouvelles fonctionnalités peuvent être ajoutées rapidement.
7. Le code de toutes les fonctionnalités d'entrée est testé.

Développement du modèle :

1. Les spécifications du modèle sont revues et soumises.
2. Corrélation des métriques hors ligne et en ligne.
3. Tous les hyperparamètres ont été réglés.
4. L'impact de l'instabilité du modèle est connu.
5. Un modèle plus simple n'est pas meilleur.
6. La qualité du modèle est suffisante sur les tranches de données importantes.
7. Le modèle est testé pour des considérations d'inclusion.

Infrastructure ML :

1. L'apprentissage est reproductible.
2. Les spécifications du modèle sont testées lors des tests unitaires.
3. Le pipeline ML est testé lors des tests d'intégration.
4. La qualité du modèle est validée avant d'être mise en service.
5. Le modèle est débuggable.
6. Les modèles sont jaugés avant d'être mis en service. *(NDT : « canaried » en référence à la détection par ces oiseaux des gaz toxiques dans les mines de charbon)*
7. Les modèles mis en service peuvent être ramenés en arrière.

Tests de surveillance :

1. Les changements de dépendances entraînent une notification.
2. Les invariants de données sont maintenus pour les entrées.
3. L'apprentissage et le service ne sont pas biaisés.
4. Les modèles ne sont pas trop vieux.
5. Les modèles sont numériquement stables.
6. Les performances de calcul n'ont pas régressé.
7. La qualité de la prédiction n'a pas régressé.

9.6.1 Exercice pratique : Test exploratoire et analyse exploratoire des données (EDA).

Pour un modèle et un ensemble de données sélectionnés, les participants effectueront une session de test exploratoire basée sur les données, en considérant les différents types de données et leur distribution pour divers paramètres.

Les participants effectueront une analyse des données pour identifier les données manquantes et/ou les biais potentiels dans les données.

9.7 Sélection des techniques de test pour les systèmes basés sur l'IA.

Un système basé sur l'IA comprendra généralement des composants IA et non-IA. Le choix des techniques de test pour tester les composants non-IA est généralement le même que pour tout test conventionnel. Pour les composants basés sur l'IA, le choix peut être plus contraignant. Par exemple, lorsqu'un problème d'oracle de test est perçu (c'est-à-dire qu'il est difficile de générer les résultats attendus), il est possible, en fonction des risques perçus, d'atténuer ce problème en utilisant les techniques suivantes :

- Test dos à dos : Cela nécessite que des cas de test soient disponibles ou générés et qu'un système équivalent serve de pseudo-oracle, qui pour les tests de régression peut être une version antérieure du système. Pour une détection efficace des défauts, un système développé indépendamment peut être nécessaire.
- Test A/B : Ce test utilise souvent des entrées opérationnelles comme cas de test et est normalement utilisé pour comparer deux variantes du même système en utilisant une analyse statistique. Le test A/B peut être utilisé pour vérifier l'empoisonnement des données d'une nouvelle variante, ou pour le test de régression automatisé d'un système d'auto-apprentissage.
- Les tests métamorphiques : Cette méthode peut être utilisée par des testeurs inexpérimentés pour trouver des défauts de manière rentable, bien qu'ils doivent comprendre le domaine d'application. Le MT ne convient pas pour fournir des résultats définitifs car les résultats attendus ne sont pas absolus, mais plutôt relatifs aux cas de test source. Aucun outil commercial n'est actuellement disponible, mais de nombreux tests peuvent être générés manuellement.

Le test adverse est généralement approprié pour les modèles ML où la mauvaise manipulation d'exemples adverses pourrait avoir un impact significatif, ou lorsque le système peut être attaqué. De même, le test par empoisonnement des données peut être approprié pour les systèmes ML où le système peut être attaqué.

Lorsque les systèmes basés sur l'IA sont complexes et ont plusieurs paramètres, les tests par paires sont souvent appropriés.

Les tests basés sur l'expérience sont souvent appropriés pour tester les systèmes basés sur l'IA, en particulier pour prendre en compte les données utilisées pour l'apprentissage et les données opérationnelles. L'EDA peut être utilisée pour valider l'algorithme ML utilisé, identifier les améliorations de l'efficience et tirer parti de l'expertise du domaine. Google a constaté que sa checklist de test des ML est une approche efficace pour les systèmes ML.

Dans le domaine spécifique des réseaux neuronaux, la couverture du réseau de neurones est souvent adaptée aux systèmes critiques, certains critères de couverture nécessitant une couverture plus rigoureuse que d'autres.

10 Environnements de test pour les systèmes basés sur l'IA – 30 minutes

Mots clés
Environnement de test virtuel

Mots clés spécifiques à l'IA
Processeur spécifique à l'IA, système autonome, big data, explicabilité, système multi-agents, système d'auto-apprentissage

Objectifs d'apprentissage pour le chapitre 10 :

10.1 Environnements de test pour les systèmes basés sur l'IA

AI-10.1.1 K2 Décrire les principaux facteurs qui différencient les environnements de test pour les systèmes basés sur l'IA de ceux requis pour les systèmes conventionnels.

10.2 Environnements de test virtuels pour le test des systèmes basés sur l'IA

AI-10.2.1 K2 Décrire les avantages offerts par les environnements de test virtuels pour le test des systèmes basés sur l'IA.

10.1 Environnements de test pour les systèmes basés sur l'IA.

Les systèmes basés sur l'IA peuvent être utilisés dans une grande variété d'environnements opérationnels, ce qui signifie que les environnements de test sont tout aussi diversifiés. Les caractéristiques des systèmes basés sur l'IA qui peuvent faire que les environnements de test diffèrent de ceux des systèmes conventionnels sont les suivantes :

- L'auto-apprentissage : Les systèmes d'auto-apprentissage, et certains systèmes autonomes, sont censés s'adapter à des environnements opérationnels changeants qui peuvent ne pas avoir été entièrement définis lors du déploiement initial du système (voir section 2.1). Par conséquent, la définition d'environnements de test capables d'imiter ces changements environnementaux non définis est intrinsèquement difficile et peut nécessiter à la fois de l'imagination de la part des testeurs et un niveau d'aléatoire intégré dans l'environnement de test.
- Autonomie : On attend des systèmes autonomes qu'ils répondent aux changements de leur environnement sans intervention humaine, et qu'ils reconnaissent également les situations où l'autonomie doit être cédée aux opérateurs humains (voir section 2.2). Pour certains systèmes, l'identification et la simulation des circonstances dans lesquelles l'autonomie doit être cédée peuvent exiger que les environnements de test poussent les systèmes à l'extrême. Certains systèmes autonomes sont destinés à fonctionner dans des environnements dangereux, et il peut être difficile de mettre en place des environnements de test représentatifs et dangereux.
- Multi-agents : Lorsque des systèmes multi-agents basés sur l'IA sont censés travailler de concert avec d'autres systèmes basés sur l'IA, l'environnement de test peut devoir intégrer un niveau de non-déterminisme afin de pouvoir imiter le non-déterminisme des systèmes basés sur l'IA avec lesquels le SUT interagit.
- Explicabilité : La nature de certains systèmes basés sur l'IA peut rendre difficile de déterminer comment le système a pris ses décisions (voir section 2.7). Lorsque cela est important à comprendre avant le déploiement, l'environnement de test peut devoir intégrer des outils pour expliquer comment les décisions sont prises.
- Le matériel : Une partie du matériel utilisé pour héberger les systèmes basés sur l'IA est spécifiquement conçu à cet effet, comme les processeurs spécifiques à l'IA (voir section 1.6). La nécessité d'inclure ce matériel dans l'environnement de test doit être envisagée dans le cadre de la planification des tests.
- Données volumineuses : Lorsque l'on s'attend à ce qu'un système basé sur l'IA consomme des données volumineuses (big data) (par exemple, des données à haut volume, à haute vitesse et/ou à haute variabilité), la mise en place de ces données dans un environnement de test doit être soigneusement planifiée et mise en œuvre (voir section 7.3).

10.2 Environnements de test virtuels pour le test des systèmes basés sur l'IA.

L'utilisation d'un environnement de test virtuel pour tester un système basé sur l'IA présente les avantages suivants :

- Scénarios dangereux : Ils peuvent être testés sans mettre en danger le SUT, les autres systèmes en interaction, y compris les humains, ou l'environnement opérationnel (par exemple, les arbres, les bâtiments).
- Scénarios inhabituels : Ils peuvent être testés lorsqu'il serait autrement très long ou coûteux de mettre en place ces scénarios pour des opérations réelles (par exemple, en attendant un événement rare, tel qu'une éclipse solaire complète ou quatre bus entrant simultanément dans le même carrefour). De même, les cas limites, qui sont difficiles à créer dans le monde réel, peuvent être créés plus facilement, de manière répétée et reproductible dans un environnement de test virtuel.
- Scénarios extrêmes : Ils peuvent être testés alors qu'il serait coûteux ou impossible de les mettre en place dans la réalité (par exemple, pour une catastrophe nucléaire ou une exploration de l'espace lointain).
- Scénarios fortement consommateurs de temps : Ils peuvent être testés à des échelles de temps réduites (par exemple, plusieurs fois par seconde) dans un environnement virtuel. En revanche, leur mise en place et leur exécution en temps réel peuvent prendre des heures ou des jours. Un autre avantage est que plusieurs environnements de test virtuels peuvent être exécutés en parallèle. Cela se fait généralement dans le cloud et permet d'exécuter de nombreux scénarios simultanément, ce qui n'est pas toujours possible avec le matériel réel du système.
- Observabilité et contrôlabilité : Les environnements de test virtuels offrent une bien meilleure contrôlabilité de l'environnement de test. Par exemple, ils peuvent garantir qu'un ensemble inhabituel de conditions de transactions financières est reproduit. En outre, ils offrent une bien meilleure observabilité, car toutes les parties numériques de l'environnement peuvent être surveillées et enregistrées en permanence.
- Disponibilité : La simulation du matériel par des environnements de test virtuels permet de tester des systèmes avec des composants matériels (simulés) qui ne seraient pas disponibles autrement, peut-être parce qu'ils n'ont pas encore été développés ou parce qu'ils sont trop chers.

Les environnements de test virtuels peuvent être construits spécifiquement pour un système donné, être génériques ou être développés pour prendre en charge des domaines d'application spécifiques. Des environnements de test virtuels commerciaux et libres sont disponibles pour faciliter le test de systèmes basés sur l'IA. En voici quelques exemples :

- Morse : The Modular Open Robots Simulation Engine, est un simulateur pour la simulation générique de robots mobiles, simples ou multiples, basé sur le moteur de jeu Blender [R24].
- AI Habitat : Il s'agit d'une plateforme de simulation créée par Facebook AI, conçue pour former des agents incarnés (tels que des robots virtuels) dans des environnements 3D photoréalistes [R25].
- DRIVE Constellation : Il s'agit d'une plate-forme ouverte et évolutive pour les voitures à conduite autonome de NVIDIA. Elle est basée sur une plateforme cloud et est capable de générer des milliards de kilomètres de tests de véhicules autonomes [R26].
- MATLAB et Simulink : Ils permettent de préparer des données d'apprentissage, de produire des modèles ML et de simuler l'exécution de systèmes basés sur l'IA, y compris les modèles utilisant des données synthétiques [R27].

11 Utilisation de l'IA pour les tests – 195 minutes

Mots clés

Test visuel

Mots-clés spécifiques à l'IA

Techniques bayésiennes, classification, algorithme de clustering, prédiction de défauts, interface utilisateur graphique (GUI)

Objectifs d'apprentissage pour le chapitre 11 :

11.1 Technologies d'IA pour le test

AI-11.1.1 K2 Catégoriser les technologies d'IA utilisées dans le test de logiciels.

HO-11.1.1 H2 Discuter, à l'aide d'exemples, des activités de test pour lesquelles l'IA est moins susceptible d'être utilisée.

11.2 Utiliser l'IA pour analyser le reporting des défauts

AI-11.2.1 K2 Expliquer comment l'IA peut aider à soutenir l'analyse de nouveaux défauts.

11.3 Utiliser l'IA pour la génération de cas de test

AI-11.3.1 K2 Expliquer comment l'IA peut aider à la génération de cas de test.

11.4 Utiliser l'IA pour l'optimisation des suites de tests de régression

AI-11.4.1 K2 Expliquer comment l'IA peut aider à l'optimisation des suites de tests de régression.

11.5 Utiliser l'IA pour la prédiction des défauts

AI-11.5.1 K2 Expliquer comment l'IA peut aider à la prédiction des défauts.

HO-11.5.1 H2 Mettre en œuvre un système simple de prédiction des défauts basé sur l'IA.

11.6 Utiliser l'IA pour le test des interfaces utilisateurs

AI-11.6.1 K2 Expliquer l'utilisation de l'IA pour tester les interfaces utilisateurs.

11.1 Technologies d'IA pour les tests.

Plusieurs technologies d'IA sont énumérées dans la section 1.4, et toutes peuvent être utilisées pour soutenir un aspect spécifique des tests logiciels. Selon Harman [B24], la communauté du génie logiciel utilise trois grands domaines de technologies d'IA :

- La logique floue et les méthodes probabilistes : Celles-ci impliquent l'utilisation de techniques d'IA pour traiter des problèmes du monde réel qui sont eux-mêmes probabilistes. Par exemple, l'IA peut être utilisée pour analyser et prévoir les défaillances possibles d'un système à l'aide de techniques bayésiennes. Celles-ci peuvent estimer la probabilité de défaillance de composants ou de fonctions, ou refléter la nature potentiellement aléatoire des interactions humaines avec le système.
- Classification, apprentissage et prédiction : Ces techniques peuvent être utilisées pour divers cas d'utilisation, comme la prévision des coûts dans le cadre de la planification d'un projet ou la prévision des défauts. S'appuyant sur des technologies de ML, ce domaine est utilisé pour de nombreuses tâches de test de logiciels, notamment la gestion des défauts (voir section 11.2), la prédiction des défauts (voir section 11.5) et les tests d'interface utilisateur (voir section 11.6).
- Techniques de recherche et d'optimisation par ordinateur : Celles-ci peuvent être utilisées pour résoudre des problèmes d'optimisation dans des espaces de recherche potentiellement vastes et complexes (par exemple, à l'aide d'algorithmes de recherche). Les exemples incluent la génération de cas de test (voir Section 11.3), l'identification du plus petit nombre de cas de test permettant d'atteindre un critère de couverture donné, et l'optimisation des cas de test de régression (voir Section 11.4).

La catégorisation ci-dessus est nécessairement large, car il existe un chevauchement considérable entre les tâches de test qui peuvent être mises en œuvre par l'IA et les différentes technologies d'IA. Il ne s'agit également que d'une catégorisation, et d'autres pourraient être créées qui seraient tout aussi valables.

11.1.1 Exercice pratique : utilisation de l'IA dans les tests.

Dans le cadre d'une discussion, les participants identifieront les activités et les tâches de test qui ne sont actuellement pas pratiques à réaliser avec un système à base d'IA. Celles-ci peuvent inclure

- Spécifier des oracles de test.
- Communiquer avec les parties prenantes pour clarifier les ambiguïtés et récupérer les informations manquantes.
- Suggérer des améliorations de l'expérience utilisateur
- Remettre en question les hypothèses des parties prenantes et poser des questions gênantes.
- Comprendre les besoins des utilisateurs.

Il convient d'établir une distinction entre l'IA faible, qui pourrait être utilisée pour certaines tâches limitées, et l'IA générale, qui n'est actuellement pas disponible (voir la section 1.2.).

11.2 Utiliser l'IA pour analyser le reporting des défauts.

Les défauts signalés sont généralement catégorisés, classés par ordre de priorité et les éventuels doublons sont identifiés. Cette activité est souvent appelée triage ou analyse des défauts et vise à optimiser le temps passé pour la résolution des défauts. L'IA peut être utilisée pour soutenir cette activité de diverses manières, par exemple :

- Catégorisation : Le NLP [NDT : Natural Language Processing – traitement naturel du langage] [B25] peut être utilisée pour analyser le texte des rapports de défauts et extraire des sujets, tels que la zone de fonctionnalité affectée, qui peuvent ensuite être fournis, avec d'autres métadonnées, à des algorithmes de regroupement, tels que les k-voisins les plus proches ou les machines à vecteurs de support. Ces algorithmes peuvent identifier des catégories de défauts appropriées et mettre en évidence les défauts similaires ou dupliqués. La catégorisation basée sur l'IA est particulièrement utile pour les systèmes automatisés de reporting des défauts (par exemple, pour Microsoft Windows et pour Firefox) et pour les grands projets comptant de nombreux ingénieurs logiciels.
- Criticité : Les modèles ML entraînés sur les caractéristiques des défauts les plus critiques peuvent être utilisés pour identifier les défauts les plus susceptibles de provoquer les défaillances du système qui représentent un pourcentage important des défauts signalés [B26].
- Affectation : Les modèles ML peuvent suggérer quels développeurs sont les plus aptes à corriger des défauts particuliers, sur la base du contenu du défaut et des affectations précédentes du développeur.

11.3 Utilisation de l'IA pour la génération de cas de test.

L'utilisation de l'IA pour générer des tests peut être une technique très efficace pour créer rapidement des ressources de test et maximiser la couverture (par exemple, la couverture du code ou des exigences). La base de la génération de ces tests comprend le code source, l'interface utilisateur et un modèle de test lisible par une machine. Certains outils basent également les tests sur l'observation du comportement de bas niveau du système par le biais de l'instrumentation ou des fichiers de logs [B27].

Cependant, à moins qu'un modèle de test définissant les comportements requis soit utilisé comme base des tests, cette forme de génération de tests souffre généralement d'un problème d'oracle de test car l'outil basé sur l'IA ne sait pas quels devraient être les résultats attendus pour un ensemble donné de données de test. Une solution consiste à utiliser des tests dos à dos si un système approprié est disponible pour servir de pseudo-oracle (voir section 9.3). Une autre solution consiste à exécuter les tests en s'attendant à ce que ni une "application qui ne répond pas" ni une panne du système ne se produisent, ou d'autres indicateurs de défaillance simples similaires.

Des recherches comparant des outils de génération de tests basés sur l'IA avec des outils de test fuzzing non-IA [NDT: tests à données aléatoire] similaires montrent que les outils basés sur l'IA peuvent atteindre des niveaux de couverture équivalents et trouver plus de défauts tout en réduisant la séquence moyenne d'étapes nécessaires pour provoquer une défaillance d'une moyenne d'environ 15 000 étapes à environ 100 étapes. Cela facilite grandement le débogage [B27].

11.4 Utilisation de l'IA pour l'optimisation des suites de tests de régression.

Au fur et à mesure que des modifications sont apportées à un système, de nouveaux tests sont créés, exécutés et deviennent des candidats pour une suite de tests de régression. Pour éviter que les suites de tests de régression ne deviennent trop volumineuses, elles doivent être fréquemment optimisées pour sélectionner, hiérarchiser et même augmenter les cas de test afin de créer une suite de tests de régression plus efficace et efficiente.

Un outil basé sur l'IA peut optimiser la suite de tests de régression en analysant, par exemple, les informations provenant des résultats des tests précédents, les défauts associés et les dernières modifications apportées, comme les fonctionnalités qui sont cassées plus fréquemment et les tests qui exercent le code affecté par les modifications récentes.

Des recherches montrent qu'il est possible de réduire de 50 % la taille d'une suite de tests de régression tout en continuant à détecter la plupart des défauts [B28], et que des réductions de 40 % de la durée d'exécution des tests peuvent être atteintes sans réduction significative de la détection des défauts pour les tests d'intégration continue [B29].

11.5 Utilisation de l'IA pour la prédiction des défauts.

La prédiction des défauts peut être utilisée pour prédire si un défaut est présent, combien de défauts sont présents, ou si des défauts peuvent être trouvés. Cette capacité dépend de la sophistication de l'outil utilisé. Les résultats sont normalement utilisés pour prioriser les tests (par exemple, plus de tests pour les composants pour lesquels plus de défauts sont prédits).

La prédiction des défauts est généralement basée sur les métriques du code source, les métriques des processus et/ou les métriques des personnes et de l'organisation. En raison du grand nombre de facteurs potentiels à prendre en compte, la détermination de la relation entre ces facteurs et la probabilité de défauts dépasse les capacités humaines. Par conséquent, l'utilisation d'une approche basée sur l'IA, qui utilise généralement le ML, est une nécessité. La prédiction des défauts est plus efficace lorsqu'elle est basée sur des expériences antérieures dans une situation similaire (par exemple, avec la même base de code et/ou les mêmes développeurs).

La prédiction des défauts à l'aide de ML a été utilisée avec succès dans plusieurs situations différentes (par exemple, [B30] et [B31]). Les meilleurs prédicteurs se sont avérés être les personnes et les mesures organisationnelles plutôt que les métriques de code source plus largement utilisées, telles que les lignes de code et la complexité cyclomatique [B32].

11.5.1 Exercice pratique : Construire un système de prédiction des défauts.

Les participants utiliseront un ensemble de données approprié (comprenant par exemple des mesures de code source et des données sur les défauts correspondants) pour construire un modèle simple de prédiction des défauts et l'utiliser pour prédire la probabilité de défauts en utilisant des mesures de code source provenant d'un code similaire.

Le modèle doit utiliser au moins quatre caractéristiques de l'ensemble de données, et le groupe de participants doit explorer les résultats en utilisant plusieurs caractéristiques différentes pour mettre en évidence la façon dont les résultats changent en fonction des caractéristiques sélectionnées.

11.6 11.6 Utiliser l'IA pour tester les interfaces utilisateur.

11.6.1 Utiliser l'IA pour tester à travers l'interface utilisateur graphique (GUI).

Tester à travers l'interface graphique est l'approche typique des tests manuels (autres que les tests de composants) et est souvent le point de départ des initiatives d'automatisation des tests. Les tests qui en résultent émulent l'interaction humaine avec l'objet de test. Cette automatisation des tests scriptés peut être mise en œuvre en appliquant une approche de capture/rejeu, en utilisant soit les coordonnées réelles des éléments de l'interface utilisateur, soit les objets/widgets définis par logiciel de l'interface. Toutefois, cette approche présente plusieurs inconvénients en ce qui concerne l'identification des objets, notamment la sensibilité aux changements d'interface, de code et de plate-forme.

L'IA peut être utilisée pour réduire la fragilité de cette approche, en employant des outils basés sur l'IA pour identifier les objets corrects à l'aide de divers critères (par exemple, XPath, label, id, classe, coordonnées X/Y), et pour choisir les critères d'identification historiquement les plus stables. Par exemple, l'identifiant d'un bouton dans une zone particulière de l'application peut changer à chaque version, et l'outil d'IA peut donc accorder moins d'importance à cet identifiant au fil du temps et se fier davantage à d'autres critères. Cette approche permet de classer les objets de l'interface utilisateur comme correspondant au test ou ne correspondant pas au test.

Par ailleurs, le test visuel utilise la reconnaissance d'image pour interagir avec les objets de l'interface graphique par le biais de la même interface qu'un utilisateur réel, et n'a donc pas besoin d'accéder au code sous-jacent et aux définitions de l'interface. Cela le rend complètement non intrusif et indépendant de la technologie sous-jacente. Les scripts ne doivent travailler qu'à travers l'interface utilisateur visible. Cette approche permet au testeur de créer des scripts qui interagissent directement avec les images, les boutons et les champs de texte à l'écran de la même manière qu'un utilisateur humain, sans être affecté par la disposition générale de l'écran. L'utilisation de la reconnaissance d'images dans l'automatisation des tests peut être limitée par les ressources informatiques nécessaires. Cependant, la disponibilité d'une IA abordable qui prend en charge la reconnaissance d'images sophistiquée rend maintenant cette approche possible pour une utilisation courante.

11.6.2 Utilisation de l'IA pour tester l’interface graphique.

Des modèles ML peuvent être utilisés pour déterminer l'acceptabilité des écrans d'interface utilisateur (par exemple, en utilisant des heuristiques et un apprentissage supervisé). Les outils basés sur ces modèles peuvent identifier les éléments mal rendus, déterminer si certains objets sont inaccessibles ou difficiles à détecter, et détecter divers autres problèmes liés à l'apparence visuelle de l'interface graphique.

Si la reconnaissance d'images est une forme d'algorithme de vision par ordinateur, d'autres formes de vision par ordinateur basées sur l'IA peuvent être utilisées pour comparer des images (par exemple, des captures d'écran) afin d'identifier des modifications involontaires de la disposition, de la taille, de la position, de la couleur, de la police ou d'autres attributs visibles des objets. Les résultats de ces comparaisons peuvent être utilisés à l'appui des tests de régression pour vérifier que les modifications apportées à l'objet de test n'ont pas eu d'effet négatif sur l'interface utilisateur.

La technologie de vérification de l'acceptabilité des écrans peut être combinée avec des outils de comparaison pour créer des outils de test de régression basés sur l'IA plus sophistiqués, capables de dire si les changements d'interface utilisateur détectés sont susceptibles d'être acceptés par les utilisateurs ou si ces changements doivent être signalés pour être vérifiés par un humain. Ces outils basés sur l'IA

peuvent également être utilisés pour prendre en charge les tests de compatibilité sur différents navigateurs, appareils ou plateformes, afin de vérifier que l'interface utilisateur d'une même application fonctionne correctement sur différents navigateurs/appareils/plateformes.

12 Références

12.1 Standards

[S01] ISO/IEC TR 29119-11:2020, Software and systems engineering — Software testing — Part 11 Guidelines on the testing of AI-based systems

[S02] DIN SPEC 92001-1, Artificial Intelligence - Life Cycle Processes and Quality Requirements - Part 1: Quality Meta Model, https://www.din.de/en/wdc-beuth:din21:303650673 (accessed May 2021).

[S03] DIN SPEC 92001-2, Artificial Intelligence - Life Cycle Processes and Quality Requirements - Part 2: Technical and Organizational Requirements, https://www.din.de/en/innovation-and-research/din-spec-en/projects/wdc-proj:din21:298702628 (accessed May 2021).

[S04] ISO 26262 - https://www.iso.org/standard/68383.html (accessed May 2021)

[S05] ISO/PAS 21448:2019, Road vehicles – Safety of the intended functionality (SOTIF) - https://www.iso.org/standard/70939.html (accessed May 2021)

[S06] ISO/IEC 25010:2011, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models, 2011.

[S07] ISO 26262-6:2018 - Road vehicles - Functional safety - Part 6: Product development at the software level.

[S08] ISO/IEC/IEEE 29119-4:2015, Software and systems engineering — Software testing — Part 4: Test techniques.

12.2 Documents de l'ISTQB®

[I01] ISTQB® Testeur Certifié de Niveau Fondation Syllabus, Version 2018 V3.1
https://www.istqb.org/downloads/category/2-foundation-level-documents.html (accessed May 2021).
Documents associés aux certifications – CFTL – Comité Français des Tests Logiciels

[I02] ISTQB® Certified Tester Advanced Level Test Analyst Syllabus, Version 3.1, section 3.2.6
https://www.istqb.org/downloads/category/75-advanced-level-test-analyst-v3-1.html (accessed August 2021).

[I03] ISTQB® Certified Tester AI Testing, Overview of Syllabus, Version 1.0

12.3 Livres et articles

[B01] Cadwalladr, Carole (2014). "Are the robots about to rise? Google's new director of engineering thinks so…" The Guardian. Guardian News and Media Limited., https://www.theguardian.com/technology/2014/feb/22/robots-google-ray-kurzweil-terminator-singularity-artificial-intelligence (accessed May 2021).

[B02] Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, 4th Edition, Pearson, 2020.

[B03] M. Davies et al., "Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook," Proceedings of the IEEE, vol. 109, no. 5, pp. 911–934, May 2021, doi: 10.1109/JPROC.2021.3067593.

[B04] Chris Wiltz, Can Apple Use Its Latest AI Chip for More Than Photos?, Electronics & Test, Artificial Intelligence, https://www.designnews.com/electronics-test/can-apple-use-its-latest-ai-chip-more-photos/153617253461497 (accessed May 2021).

[B05] HUAWEI Reveals the Future of Mobile AI at IFA 2017, Huawei Press Release, https://consumer.huawei.com/en/press/news/2017/ifa2017-kirin970/ (accessed May 2021).

[B06] REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), April 2016,https://eur-lex.europa.eu/eli/reg/2016/679/oj (accessed May 2021)

[B07] Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles J3016_201806, SAE, ,https://www.sae.org/standards/content/j3016_201806/ (accessed May 2021).

[B08] G20 Ministerial Statement on Trade and Digital Economy: Annex. Available from: https://www.mofa.go.jp/files/000486596.pdf (accessed May 2021).

[B09] Concrete Problems in AI Safety, Dario Amodei (Google Brain), Chris Olah (Google Brain), Jacob Steinhardt (Stanford University), Paul Christiano (UC Berkeley), John Schulman (OpenAI), Dan Man´e (Google Brain), March 2016. https://arxiv.org/pdf/1606.06565 (accessed May 2021).

[B10] Explainable AI: the basics, Policy briefing, Issued: November 2019 DES6051, ISBN: 978-1-78252-433-5, The Royal Society.

[B11] The Ultimate Guide to Data Labeling for Machine Learning, www.cloudfactory.com/data-labeling-guide (accessed May 2021).

[B12] Pei et al, DeepXplore: Automated Whitebox Testing of Deep Learning Systems, Proceedings of ACM Symposium on Operating Systems Principles (SOSP '17), Jan 2017.

[B13] Sun et al, Testing Deep Neural Networks, https://www.researchgate.net/publication/323747173_Testing_Deep_Neural_Networks, accessed Nov 2019 (accessed May 2021).

[B14] A. Odena and I. Goodfellow, TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing, ArXiv e-prints, Jul. 2018, https://arxiv.org/pdf/1807.10875 (accessed May 2021)

[B15] Riccio, V. et al, Testing Machine Learning based Systems: A Systematic Mapping. Empirical Software Engineering, https://link.springer.com/article/10.1007/s10664-020-09881-0 (accessed May 2021)

[B16] Baudel, Thomas et al, Addressing Cognitive Biases in Augmented Business Decision Systems., https://arxiv.org/abs/2009.08127 (accessed May 2021)

[B17] Papernot, N. et al, Transferability in machine learning: from phenomena to black-box attacks using adversarial samples, arXiv preprint arXiv:1605.07277, 2016. https://arxiv.org/pdf/1605.07277 (accessed May 2021).

[B18] Chen et al, Metamorphic Testing: A Review of Challenges and Opportunities, ACM Comput. Surv. 51, 1, Article 4, January 2018. https://www.researchgate.net/publication/322261865_Metamorphic_Testing_A_Review_of_Challenges_and_Opportunities (accessed May 2021).

[B19] Huai Liu, Fei-Ching Kuo, Dave Towey, and Tsong Yueh Chen., How effectively does metamorphic testing alleviate the oracle problem?, IEEE Transactions on Software Engineering 40, 1, 4–22, 2014

[B20] James Whittaker, Exploratory Software Testing: Tips, Tricks, Tours and Techniques to Guide Test Design, 1. Edition, Addison-Wesley Professional, 2009.

[B21] L. Wilkinson, A. Anand, and R. Grossman. High-dimensional visual analytics: Interactive exploration guided by pairwise views of point distributions. Visualization and Computer Graphics, IEEE Transactions on, 12(6):1363–1372, 2006, https://www.cs.uic.edu/~wilkinson/Publications/sorting.pdf (accessed May 2021).

[B22] Ryan Hafen and Terence Critchlow, EDA and ML – A Perfect Pair for Large-Scale Data Analysis, IEEE 27th International Symposium on Parallel and Distributed Processing, 2013, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6651091 (accessed May 2021).

[B23] Breck, Eric, Shanqing Cai, Eric Nielsen, Michael Salib, and D. Sculley., The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction, IEEE International Conference on Big Data (Big Data), 2017, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8258038 (accessed May 2021).

[B24] Harman, The Role of Artificial Intelligence in Software Engineering, In First International Workshop on Realizing AI Synergies in Software Engineering (RAISE), pp. 1-6. IEEE, June 2012, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6227961 (accessed May 2021).

[B25] Nilambri et al, A Survey on Automated Duplicate Detection in a Bug Repository, International Journal of Engineering Research & Technology (IJERT), 2014, https://www.ijert.org/research/a-survey-on-automated-duplicate-detection-in-a-bug-repository-IJERTV3IS041769.pdf (accessed May 2021)

[B26] Kim, D.; Wang, X.; Kim, S.; Zeller, A.; Cheung, S.C.; Park, S. (2011). "Which Crashes Should I Fix First? Predicting Top Crashes at an Early Stage to Prioritize Debugging Efforts," in the IEEE Transactions on Software Engineering, volume 37, https://ieeexplore.ieee.org/document/5711013 (accessed May 2021).

[B27] Mao et al, Sapienz: multi-objective automated testing for Android applications, Proceedings of the 25th International Symposium on Software Testing and Analysis, July 2016, http://www0.cs.ucl.ac.uk/staff/K.Mao/archive/p_issta16_sapienz.pdf (accessed May 2021).

[B28] Rai et al, Regression Test Case Optimization Using Honey Bee Mating Optimization Algorithm with Fuzzy Rule Base, World Applied Sciences Journal 31 (4): 654-662, 2014, https://www.researchgate.net/publication/336133351_Regression_Test_Case_Optimization_Using_Honey_Bee_Mating_Optimization_Algorithm_with_Fuzzy_Rule_Base (accessed May 2021).

[B29] Dusica Marijan, Arnaud Gotlieb, Marius Liaaen. A learning algorithm for optimizing continuous integration development and testing practice, Journal of Software : Practice and Experience, Nov 2018.

[B30] Tosun et al, AI-Based Software Defect Predictors: Applications and Benefits in a Case Study, Proceedings of the Twenty-Second Innovative Applications of Artificial Intelligence Conference (IAAI-10), 2010.

[B31] Kim et al, Predicting Faults from Cached History, 29th International Conference on Software Engineering (ICSE'07), 2007.

[B32] Nagappan 2008 Nagappan et al, The Influence of Organizational Structure on Software Quality: An Empirical Case Study, Proceedings of the 30th international conference on Software engineering (ICSE'08), May 2008.

[B33] Kuhn et al, Software Fault Interactions and Implications for Software Testing, IEEE Transactions on Software Engineering vol. 30, no. 6, (June 2004) pp. 418-421.

12.4 Autres références

Les références suivantes renvoient à des informations disponibles sur Internet. Bien que ces références aient été vérifiées au moment de la publication, l'ISTQB® ne peut être tenu responsable si les références ne sont plus disponibles.

[R01] Wikipedia contributors, "AI effect," Wikipedia, https://en.wikipedia.org/wiki/AI_effect (accessed May 2021).

[R02] https://mxnet.apache.org/ (accessed May 2021).

[R03] https://docs.microsoft.com/en-us/cognitive-toolkit/ (accessed May 2021).

[R04] IBM Watson, https://www.ibm.com/watson/ai-services

[R05] https://www.tensorflow.org/ (accessed May 2021).

[R06] https://keras.io/ (accessed May 2021).

[R07] https://pytorch.org/ (accessed May 2021).

[R08] https://scikit-learn.org/stable/whats_new/v0.23.html (accessed May 2021).

[R09] NVIDIA VOLTA, https://www.nvidia.com/en-us/data-center/volta-gpu-architecture/ (accessed May 2021).

[R10] Cloud TPU, https://cloud.google.com/tpu/ (accessed May 2021).

[R11] Edge TPU, https://cloud.google.com/edge-tpu/ (accessed May 2021).

[R12] Intel® Nervana™ Neural Network processors deliver the scale and efficiency demanded by deep learning model evolution, https://www.intel.ai/nervana-nnp/ (accessed May 2021).

[R13] The Evolution of EyeQ, https://www.mobileye.com/our-technology/evolution-eyeq-chip/ (accessed May 2021).

[R14] ImageNet - http://www.image-net.org/ (accessed May 2021).

[R15] Google's BERT - https://github.com/google-research/bert (accessed May 2021).

[R16] https://www.kaggle.com/datasets (accessed May 2021).

[R17] https://www.kaggle.com/paultimothymooney/2018-kaggle-machine-learning-data-science-survey (accessed May 2021).

[R18] MLCommons - https://mlcommons.org/ (accessed May 2021).

[R19] DAWNBench – https://dawn.cs.stanford.edu/benchmark (accessed May 2021).

[R20] MLMark – https://www.eembc.org/mlmark (accessed May 2021).

[R21] https://digital-strategy.ec.europa.eu/en/library/assessment-list-trustworthy-artificial-intelligence-altai-self-assessment| Shaping Europe's digital future (europa.eu) (accessed August 2021)

[R22] https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai (accessed August 2021)

[R23] Google GraphicsFuzz, https://github.com/google/graphicsfuzz (accessed May 2021).

[R24] http://www.openrobots.org/morse/doc/0.2.1/morse.html (accessed May 2021).

[R25] https://ai.facebook.com/blog/open-sourcing-ai-habitat-a-simulation-platform-for-embodied-ai-research/ (accessed May 2021).

[R26] https://www.nvidia.com/en-gb/self-driving-cars/drive-constellation/ (accessed May 2021).

[R27] https://uk.mathworks.com/discovery/artificial-intelligence.html#ai-with-matlab (accessed May 2021).

13 Appendice A – Abbréviations

Abbréviation	Description
IA	Intelligence Artificielle
AIaaS	AI as a service
API	Application programming interface
AUC	Area under curve (aire sous le courbe)
DL	Deep learning
DNN	Deep neural network (réseau neuronal profond)
EDA	Exploratory data analysis (analyse exploratoire des données)
EU	European Union (UE)
FN	False negative (faux négatif)
FP	False positive (faux positif)
RGPD	Règlement Général pour la Protection des Données
GPU	Graphical processing unit
GUI	Graphical user interface
LIME	Local interpretable model-agonistic explanations
MC/DC	Modified condition decision coverage
ML	Machine learning
MR	Metamorphic relation (relation métamorphique)
MSE	Mean square error (erreur quadratique moyenne)
MT	Metamorphic testing (test métamorphique)
NLP	Natural language processing (traitement du langage naturel)
ROC	Receiver operating characteristic (Caractéristique d'exploitation du récepteur)
SUT	System under test (système sous test)
SVM	Support vector machine (Machine à vecteur de support)
TN	True negative (vrai négatif)
TP	True positive (vrai positif)
XAI	Explainable AI (IA explicable)

14 *Appendice B* – Termes spécifiques à l'IA et autres termes

Terme	Définition
exactitude	Métrique de performance fonctionnelle ML utilisée pour évaluer un classificateur, qui mesure la proportion de prédictions correctes. (d'après ISO/IEC TR 29119-11)
fonction d'activation	La formule associée à un neurone dans un réseau neuronal qui détermine la sortie du neurone à partir des entrées du neurone.
valeur d'activation	La sortie d'une fonction d'activation d'un neurone dans un réseau neuronal.
attaque adverse	L'utilisation délibérée d'exemples adverses pour faire échouer un modèle ML
AI as a Service (AIaaS)	Un modèle de licence et de fourniture de logiciels dans lequel les services d'IA et de développement de l'IA sont hébergés de manière centralisée.
Composant IA	Un composant qui fournit une fonctionnalité d'IA
effet AI	La situation dans laquelle un système d'IA précédemment étiqueté n'est plus considéré comme de l'IA à mesure que la technologie progresse (ISO/IEC TR 29119-11)
Système basé sur l'IA	Un système qui intègre un ou plusieurs composants d'IA
Processeur spécifique à l'IA	Un type de matériel spécialisé conçu pour accélérer les applications d'IA.
biais algorithmique	Un type de biais causé par l'algorithme ML
annotation	Activité consistant à identifier des objets dans des images avec des boîtes de délimitation afin de fournir des données étiquetées pour la classification.
area under curve (AUC) aire sous la courbe	Une mesure de la capacité d'un classificateur à distinguer deux classes.
l'intelligence artificielle (IA)	La capacité d'un système technique à acquérir, traiter, créer et appliquer des connaissances et des compétences (ISO/IEC TR 29119-11).
association	Une technique d'apprentissage non supervisée qui identifie les relations et les dépendances entre les échantillons.
augmentation	Activité consistant à créer de nouveaux points de données à partir d'un ensemble de données existant.

Terme	Définition
biais d'automatisation	Type de biais causé par une personne favorisant les recommandations d'un système automatisé de prise de décision par rapport à d'autres sources. Synonyme : biais de complaisance
système autonome	Un système capable de fonctionner sans intervention humaine pendant de longues périodes.
autonomie	L'aptitude d'un système à fonctionner pendant des périodes prolongées sans intervention humaine (ISO/IEC TR 29119-11)
Modèle bayésien	Un modèle statistique qui utilise la probabilité pour représenter l'incertitude des entrées et des sorties du modèle.
Technique bayésienne	Une technique qui considère les distributions de probabilité avant et après comme des paramètres d'un modèle statistique.
biais	La différence systématique de traitement de certains objets, personnes ou groupes par rapport à d'autres (ISO/IEC DIS 22989).
big data	Des ensembles de données étendus dont les caractéristiques en termes de volume, de variété, de vélocité et/ou de variabilité nécessitent des technologies et des techniques spécialisées pour être traités.
raisonnement à partir de cas	technique consistant à résoudre un nouveau problème en se basant sur les solutions de problèmes antérieurs similaires.
chatbot	Une application utilisée pour mener une conversation par le biais d'un texte ou d'une synthèse vocale.
classification	Un type de fonction ML qui prédit la classe de sortie pour une entrée donnée (d'après ISO/IEC TR 29119-11)
classificateur	Un modèle ML utilisé pour la classification Synonyme : modèle de classification
clustering	Un type de fonction ML qui regroupe les points de données similaires.
algorithme de clustering	Un type d'algorithme ML utilisé pour regrouper des objets similaires en clusters.
dérive du concept	Une modification de l'exactitude perçue des prédictions d'un modèle ML au fil du temps, causée par des changements dans les attentes et le comportement des utilisateurs et dans l'environnement opérationnel.
matrice de confusion	Une technique pour résumer la performance fonctionnelle ML d'un algorithme de classification

Terme	Définition
acquisition de données	L'activité d'acquisition de données pertinentes pour le problème métier à résoudre par un modèle ML.
étiquetage des données	L'activité consistant à ajouter des balises significatives aux objets dans les données brutes pour soutenir la classification dans le ML
pipeline de données	La mise en œuvre d'activités de préparation des données afin de fournir des données d'entrée pour soutenir l'apprentissage par un algorithme ML ou la prédiction par un modèle ML.
point de données	Un ensemble d'une ou plusieurs mesures comprenant une seule observation utilisée comme partie d'un ensemble de données.
empoisonnement des données	La manipulation délibérée et malveillante des données d'apprentissage ou d'entrée d'un modèle ML.
préparation des données	Les activités d'acquisition des données, de prétraitement des données et d'ingénierie des caractéristiques dans le workflow ML.
prétraitement des données	Les activités de nettoyage des données, de transformation des données, d'augmentation des données et d'échantillonnage des données dans le workflow ML.
visualisation des données	Une technique pour représenter graphiquement les relations, les tendances et les modèles de données.
jeu de données	Une collection de données utilisées pour l'apprentissage, l'évaluation, le test et la prédiction en ML
seuil de décision	Valeur qui transforme le résultat d'une fonction de prédiction en un résultat binaire supérieur ou inférieur à cette valeur. Synonyme : seuil de discrimination
arbre de décision	Un modèle ML arborescent dont les nœuds représentent les décisions et les branches les résultats possibles.
classificateur déductif	Un classificateur basé sur l'application de l'inférence et de la logique aux données d'entrée.
deep learning (DL)	ML utilisant des réseaux neuronaux à couches multiples
deep neural network	Réseau neuronal composé de plusieurs couches de neurones. Synonyme : perceptron multicouche
Prévision des défauts	Une technique pour prédire les zones de l'objet de test dans lesquelles des défauts se produiront ou la quantité de défauts présents.
système déterministe	Un système qui produira le même ensemble de sorties et le même état final à partir d'un ensemble donné d'entrées et d'un état de départ.

Terme	Définition
edge computing	La partie d'une architecture distribuée dans laquelle le traitement de l'information est effectué à proximité de l'endroit où cette information est utilisée.
époque	Une itération d'apprentissage d'un modèle ML sur l'ensemble des données d'apprentissage.
évolution	Le processus de changement continu d'un état inférieur, plus simple ou plus mauvais à un état supérieur, plus complexe ou meilleur.
système expert	Un système basé sur l'IA pour résoudre des problèmes dans un domaine ou un champ d'application particulier en tirant des conclusions d'une base de connaissances développée à partir de l'expertise humaine.
explicabilité	Le niveau de compréhension de la manière dont le système basé sur l'IA est parvenu à un résultat donné (ISO/IEC TR 29119-11).
IA explicable (XAI)	Le domaine d'étude lié à la compréhension des facteurs qui influencent les résultats des systèmes d'IA.
exploratory data analysis (EDA)	L'exploration visuelle, interactive et pilotée par des hypothèses, des données utilisées pour soutenir l'ingénierie des fonctionnalités.
Score-F1	Une métrique de performance fonctionnelle ML utilisée pour évaluer un classificateur qui fournit un équilibre entre le rappel et la précision.
Faux négatif (FN)	Une prédiction de modèle ML dans laquelle le modèle prédit par erreur la classe négative.
Faux positif (FP)	Une prédiction de modèle ML dans laquelle le modèle prédit par erreur la classe positive.
Function (feature)	Un attribut individuel mesurable des données d'entrée utilisé pour l'apprentissage par un algorithme ML et pour la prédiction par un modèle ML.
ingénierie des fonctions	L'activité dans laquelle les attributs des données brutes qui représentent le mieux les relations sous-jacentes qui devraient apparaître dans le modèle ML sont identifiés pour être utilisés dans les données d'apprentissage (ISO/IEC TR 29119-11).
flexibilité	La capacité d'un système à fonctionner dans des contextes extérieurs à sa spécification initiale (après ISO/IEC TR 29119-11)
fuzzy logic (logique floue)	Un type de logique basé sur le concept de vérité partielle représentée par des facteurs de certitude entre 0 et 1.

Terme	Définition
IA générale	IA qui présente un comportement intelligent comparable à celui d'un humain dans toute la gamme des capacités cognitives (ISO/CEI TR 29119-11). Synonyme : IA forte
Règlement général sur la protection des données (RGPD)	Le règlement de l'Union européenne (UE) sur la protection des données et la vie privée qui s'applique aux données des citoyens de l'UE et de l'Espace économique européen.
graphical processing unit (GPU)	Circuit intégré spécifique à une application, conçu pour manipuler et modifier la mémoire afin d'accélérer la création d'images dans un tampon d'images destiné à être transmis à un dispositif d'affichage.
vérité de terrain (ground truth)	L'information fournie par l'observation directe et la mesure qui est connue pour être réelle ou vraie.
hyperparamètre	Un paramètre utilisé pour contrôler l'apprentissage d'un modèle ML ou pour définir la configuration d'un modèle ML.
ajustement des hyperparamètres	L'activité consistant à déterminer les hyperparamètres optimaux en fonction d'objectifs particuliers.
biais inapproprié	Un type de biais qui amène un système à produire des résultats qui entraînent des effets négatifs pour un groupe particulier.
agent intelligent	Un programme autonome qui dirige son activité vers la réalisation d'objectifs en utilisant des observations et des actions.
métrique inter-clusters	Une métrique qui mesure la similarité des points de données dans différents clusters.
interprétabilité	Le niveau de compréhension du fonctionnement de la technologie d'IA sous-jacente (ISO/IEC TR 29119-11)
Métrique intra-cluster	Une métrique qui mesure la similarité des points de données au sein d'un cluster.
k-nearest neighbor (k voisin le plus proche)	Une approche de la classification qui estime la probabilité d'appartenance à un groupe pour un point de données en fonction de l'appartenance à un groupe des points de données les plus proches de celui-ci.
algorithme d'apprentissage	Un programme qui produit un modèle ML basé sur les propriétés de l'ensemble de données d'apprentissage
méthode LIME	Le programme Local Interpretable Model-Agnostic Explanations pour expliquer les prédictions d'un modèle ML

Terme	Définition
régression linéaire	Une technique statistique qui modélise la relation entre les variables en ajustant une équation linéaire aux données observées lorsque la variable cible est numérique.
régression logistique	Technique statistique qui modélise la relation entre les variables lorsque la variable cible est catégorielle plutôt que numérique.
machine learning (ML)	Le processus utilisant des techniques informatiques pour permettre aux systèmes d'apprendre à partir de données ou d'expériences (ISO/IEC TR 29119-11).
mean square error (MSE) (erreur quadratique Moyenne)	La mesure statistique de la différence quadratique moyenne entre les valeurs estimées et la valeur réelle.
algorithme ML	Un algorithme utilisé pour créer un modèle ML à partir d'un ensemble de données d'apprentissage.
Suite de benchmarks ML	Un ensemble de données utilisé pour comparer les modèles et les algorithmes de ML sur une gamme de métriques d'évaluation.
framework ML	Un outil ou une bibliothèque qui prend en charge la création d'un modèle ML
fonction ML	Fonctionnalité mise en œuvre par un modèle ML, telle que la classification, la régression ou le regroupement.
Modèle d'évaluation ML	Le processus de comparaison des mesures de performance fonctionnelle des modèles de ML avec les critères requis et ceux d'autres modèles de ML.
Entraînement du modèle ML	Le processus d'application de l'algorithme ML à l'ensemble de données d'apprentissage pour créer un modèle ML.
Ajustement du modèle ML	Le processus de test des hyperparamètres pour obtenir des performances optimales.
système ML	Un système qui intègre un ou plusieurs modèles ML
workflow ML	Une séquence d'activités utilisées pour gérer le développement et le déploiement d'un modèle de ML.
système multi-agent	Un système qui comprend plusieurs agents intelligents
IA étroite	IA axée sur une seule tâche bien définie pour résoudre un problème spécifique (ISO/IEC TR 29119-11) Synonyme : IA faible

Terme	Définition
natural language processing (NLP)	Domaine de l'informatique qui permet de lire, de comprendre et de déduire le sens des langues naturelles.
réseau neuronal	Réseau d'éléments de traitement primitifs reliés par des liens pondérés dont les poids sont ajustables, dans lequel chaque élément produit une valeur en appliquant une fonction non linéaire à ses valeurs d'entrée, et la transmet à d'autres éléments ou la présente comme une valeur de sortie (ISO/IEC 2382). Synonyme : réseau neuronal artificiel
cheval de Troie du réseau neuronal	Une vulnérabilité injectée dans un réseau neuronal à l'aide d'une attaque par empoisonnement des données dans le but de l'exploiter ultérieurement.
processeur neuromorphique	Un circuit intégré conçu pour imiter les neurones biologiques du cerveau humain.
neurone	Un nœud dans un réseau neuronal, recevant habituellement des valeurs d'entrée multiples et générant une valeur d'activation
bruit	Une distorsion ou une corruption des données
système non déterministe	Un système qui ne produira pas toujours le même ensemble de sorties et le même état final pour un ensemble particulier d'entrées et un état de départ.
Aberration (outlier)	Une observation qui se situe en dehors de la tendance générale de la distribution des données.
surajustement	La génération d'un modèle ML qui correspond trop étroitement à l'ensemble de données d'entraînement, ce qui entraîne un modèle qui a du mal à se généraliser à de nouvelles données (d'après ISO/IEC TR 29119-11).
perceptron	Un réseau neuronal avec une seule couche et un seul neurone.
précision	Une métrique de performance fonctionnelle ML utilisée pour évaluer un classificateur, qui mesure la proportion de positifs prédits qui sont corrects (d'après ISO/IEC TR 29119-11).
modèle pré-entraîné	Un modèle ML déjà entraîné lorsqu'il a été obtenu
système probabiliste	Un système dont le comportement est décrit en termes de probabilités ; ses résultats ne peuvent donc pas être parfaitement prédits.
raisonnement procédural	Technologie d'IA utilisée pour construire des systèmes de raisonnement en temps réel capables d'exécuter des tâches complexes dans des environnements dynamiques.

Terme	Définition
forêt aléatoire (random forest)	Ensemble de technologies de ML pour la classification, la régression et d'autres tâches qui fonctionnent en construisant et en exécutant de nombreux arbres de décision, puis en produisant le mode de la classe ou la prédiction moyenne des arbres individuels.
technique de raisonnement	IA qui génère des conclusions à partir des informations disponibles en utilisant des techniques logiques (Après ISO/IEC TR 29119-11)
rappel	Une métrique de performance fonctionnelle ML utilisée pour évaluer un classificateur, qui mesure la proportion de positifs réels qui ont été prédits correctement (d'après ISO/IEC TR 29119-11). Synonyme : sensibilité
Courbe ROC (receiver operating characteristic curve)	Un graphique qui illustre la capacité d'un classificateur binaire à faire varier son seuil de discrimination.
régression	Un type de fonction ML qui donne une valeur de sortie numérique ou continue pour une entrée donnée (d'après ISO/IEC TR 29119-11).
modèle de régression	Un modèle ML dont la sortie attendue pour une entrée numérique donnée est une variable continue (d'après ISO/IEC DIS 23053).
apprentissage par renforcement	L'activité de construction d'un modèle ML en utilisant un processus d'essai et de récompense pour atteindre un objectif (d'après ISO/IEC TR 29119-11)
fonction de récompense	Une fonction qui définit le succès de l'apprentissage par renforcement.
piratage de récompense	Activité réalisée par un agent intelligent pour maximiser sa fonction de récompense au détriment de la réalisation de l'objectif initial (d'après ISO/IEC TR 29119-11).
R-carré (R-squared)	Une mesure statistique de la proximité des points de données par rapport à la ligne de régression ajustée. Synonyme : coefficient de détermination
moteur de règles	Un ensemble de règles qui déterminent les actions à entreprendre lorsque certaines conditions sont remplies.
sûreté	L'attente selon laquelle un système ne conduit pas, dans des conditions définies, à un état dans lequel la vie humaine, la santé, les biens ou l'environnement sont mis en danger (ISO/IEC/IEEE 12207).
biais de l'échantillon	Un type de biais où l'ensemble de données n'est pas entièrement représentatif de l'espace de données auquel ML est appliqué.

Terme	Définition
algorithme de recherche	Un algorithme qui visite systématiquement un sous-ensemble de tous les états ou structures possibles jusqu'à ce que l'état ou la structure cible soit atteint (après ISO/IEC TR 29119-11).
système d'auto-apprentissage	Un système adaptatif qui modifie son comportement en fonction de l'apprentissage par essais et erreurs (d'après ISO/IEC TR 29119-11).
Coefficient de silhouette	Une mesure de classification entre -1 et +1 basée sur la moyenne des différences inter-cluster et intra-cluster. Synonyme : score de silhouette
super IA	Un système basé sur l'intelligence artificielle qui dépasse de loin les capacités humaines.
apprentissage supervisé	Entraînement d'un modèle ML à partir de données d'entrée et des étiquettes correspondantes.
machine à vecteur de support (SVM)	Une technique ML dans laquelle les points de données sont considérés comme des vecteurs dans un espace multidimensionnel séparés par un hyperplan.
singularité technologique	Un moment dans l'avenir où les progrès technologiques ne sont plus contrôlables par l'homme (d'après ISO/IEC TR 29119-11)
problème d'oracle de test	Le défi de déterminer si un test a réussi ou échoué pour un ensemble donné d'entrées de test et d'états.
ensemble de données d'apprentissage	Un jeu de données utilisé pour entraîner un modèle ML.
apprentissage par transfert	Une technique pour modifier un modèle ML pré-entraîné afin d'effectuer une tâche différente.
transparence	Le niveau de visibilité de l'algorithme et des données utilisées par le système basé sur l'IA (d'après ISO/IEC TR 29119-11)
vrai négatif (TN)	Une prédiction dans laquelle le modèle prédit correctement la classe négative.
vrai positif (TP)	Une prédiction dans laquelle le modèle prédit correctement la classe positive.
sous-ajustement	La génération d'un modèle ML qui ne reflète pas la tendance sous-jacente de l'ensemble de données d'entraînement, ce qui entraîne un modèle qui a du mal à faire des prédictions précises (ISO/IEC TR 29119-11).
apprentissage non supervisé	Apprentissage d'un modèle ML à partir de données d'entrée en utilisant un ensemble de données non étiquetées

Terme	Définition
jeu de données de validation	Un jeu de données utilisé pour évaluer un modèle ML formé dans le but d'ajuster le modèle.
Architecture de von Neumann	Une architecture informatique qui se compose de cinq éléments principaux : la mémoire, une unité centrale de traitement, une unité de contrôle, des entrées et des sorties.
poids	Variable interne d'une connexion entre neurones dans un réseau neuronal qui affecte la façon dont il calcule ses sorties et qui change au fur et à mesure que le réseau neuronal est formé.

15 Index