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

Version 2.0

International Software Testing Qualifications Board

Fourni par
Alliance for Qualification, Artificial Intelligence United, Chinese Software Testing Qualifications Board, and Korean Software Testing Qualifications Board

Avis de copyright

Historique des révisions

Version	Date	Remarques
1.0	10/01/2021	Release pour le GA
2.0 Alpha	15/08/2025	Revue alpha
2.0 Beta	07/01/2026	Revue bêta
2.0	17/04/2026	Release pour le GA
2.0 FR	31/05/2026	Version française

Table des matières

Remerciements

Ce document a été officiellement publié par l'Assemblée générale de l'ISTQB® le 17 avril 2026.

Taskforce IA ISTQB (v2.0): Klaudia Dussa-Zieger (chair), Stuart Reid, Vipul Kocher, Qin Liu, Jaroslaw Hryszko, Kyle Alexander Siemens, et Werner Henschelchen.

Les personnes suivantes ont participé à la révision et aux commentaires de ce syllabus : Marina Abratis, Tom Adams, Laura Albert, Abhishek Alladi, Menno van den Berg, Earl Burba, Simeone Chiumarulo, Marco Ciarlitto, Alessandro Collino, Jean-Baptiste Crouigneau, Yara Dalgamoni, Taz Daughtrey, Wim Decoutere, Dmitrii Degtiarenko, Iuliia Emelianova, Lozana Enbah, Tamás Gergely, David Hendrickx, David Janota, Sagar Joshi, Norbert Juhász, Willem Keesman, John Kurowski, Ine Lutterman, Niranjan Maharajh, Rik Marselis, Judy McKay, Gary Mogyorodi, Markus Niehammer, Tauhida Parveen, Arnd Pehl, Lukas Piska, Daniel Polan, Andrew Pollner, Nishan Portoyan, Meile Posthuma, Miroslav Renda, Randall Rice, Piet de Roo, Nicola de Rosa, Mark Rutz, Salvatore Sarno, Klaus Skafte, Giancarlo Tomasig, Yaron Tsubery, Rahul Verma, André Verschelling, Linda Vreeswijk, et Mario Winter.

Taskforce IA ISTQB (v1.0): Klaudia Dussa-Zieger (chair), Vipul Kocher, Qin Liu, Stuart Reid, Adam Leon Smith, Kyle Alexander Siemens, et Werner Henschelchen.

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: Main authors Vipul Kocher, Saurabh Bansal, Srinivas Padmanabhuni, and Sonika Bengani and co-authors 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.

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 :

- 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.
- Aux organismes de certification, pour élaborer des questions d'examen localisées et adaptées aux objectifs d'apprentissage de ce syllabus.
- Aux organismes de formation, pour produire des cours et pour déterminer les méthodes d'enseignement appropriées.
- Aux candidats à la certification, pour préparer l'examen de certification (dans le cadre d'une formation ou de manière indépendante).
- 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..

0.3 Parcours professionnel des testeurs

Le programme ISTQB® fournit du soutien aux professionnels du test à toutes les étapes de leur carrière. Les personnes ayant obtenu la certification ISTQB® Testeur Certifié de niveau Fondation peuvent également s'intéresser aux niveaux Avancés (Analyste de Test, Analyste Technique de Test et Management des Tests), puis au niveau Expert (Management des Tests ou Amélioration du processus de Test). La filière « Specialist » propose une immersion approfondie dans des approches de test spécifiques et des activités de test, par exemple les tests agiles, l'automatisation des tests, les tests d'IA, les tests avec l'IA générative ou les tests d'applications mobiles, ou encore dans le savoir-faire en matière de tests de groupe pour certains secteurs d'activité, par exemple l'automobile ou les jeux vidéo. Veuillez consulter www.istqb.org pour obtenir les dernières informations sur le programme Testeur Certifié de l'ISTQB.

0.4 Objectifs métier

Cette section présente les objectifs métier attendus d'un candidat ayant obtenu la certification CT-AI. Un testeur certifié en test d'IA est capable de:

BO1	Comprendre l'état actuel de l'IA, y compris l'IA générative.
BO2	Expérimenter la mise en œuvre et les tests de modèles de machine learning.
BO3	Comprendre le fonctionnement et le test des réseaux neuronaux simples.
BO4	Comprendre les caractéristiques de qualité spécifiques à l'IA définies par ISO/IEC 25059.
BO5	Calculer et interpréter les mesures de performance fonctionnelle ML pour les modèles de machine learning.
BO6	Reconnaître la portée et l'importance des deux niveaux de test spécifiques à l'évaluation des systèmes de machine learning.
BO7	Contribuer au développement d'une stratégie de test efficace pour un système de machine learning.
BO8	Concevoir et mettre en œuvre des cas de test pour les systèmes de machine learning.

0.5 Objectifs d'apprentissage, Objectifs pratiques et niveau cognitif des connaissances

Les objectifs d'apprentissage (LO) soutiennent les objectifs métier et servent à élaborer les examens CT-AI. Les niveaux spécifiques des objectifs d'apprentissage sont indiqués au début de chaque chapitre et classés comme suit:

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

Vous trouverez plus de détails et des exemples d'objectifs d'apprentissage à l'annexe A.

Pour tous les termes répertoriés comme mots-clés juste en dessous des titres de chapitre, il convient de retenir le nom et la définition corrects issus du glossaire ISTQB® ou du chapitre 9 (K1), même s'ils ne sont pas explicitement mentionnés dans l'objectif d'apprentissage. Les objectifs pratiques (HO) se concentrent sur l'application concrète des objectifs d'apprentissage et sont présentés au début de chaque chapitre. Le niveau d'un HO est classé comme suit:

- H0 : Cela peut inclure une démonstration en direct d'un exercice ou une vidéo enregistrée. Étant donné que le candidat ne réalise pas cet exercice lui-même, il ne s'agit pas à proprement parler d'un exercice.
- H1 : Exercice guidé. Les candidats suivent une série d'étapes effectuées par le formateur.
- H2 : Exercice avec des indices. Le candidat se voit proposer un exercice accompagné d'indices pertinents afin de lui permettre de le résoudre dans le temps imparti.

0.6 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.

Le principal critère d'admission pour toute personne souhaitant passer l'examen CT-AI est la détention de la certification ISTQB ® Testeur Certifié de niveau Fondation [CTFL].

Il est fortement recommandé aux candidats de :

- Posséder une expérience minimale dans le domaine du développement logiciel ou des tests logiciels, par exemple six mois d'expérience en tant que testeur de systèmes ou de tests d'acceptation des utilisateurs, data scientist ou développeur logiciel.
- Suivre une formation accréditée selon les normes ISTQB® (par l'un des Membres reconnus par l'ISTQB).

0.7 Accréditation

Un Membre de l'ISTQB® peut accréditer des organismes de formation dont le contenu de la formation est conforme au présent syllabus. Les organismes de formation doivent se procurer les directives d'accréditation auprès du Membre ou de l'organisme chargé de l'accréditation. Un cours accrédité est reconnu comme conforme au présent syllabus et permet d'intégrer un examen ISTQB® au programme de formation. Les directives d'accréditation relatives au présent syllabus suivent les directives générales d'accréditation publiées par le groupe de travail sur la gestion des processus et la conformité.

0.8 Gestion des normes

Des organismes internationaux de normalisation tels que l'IEEE et l'ISO ont publié des normes relatives aux caractéristiques de qualité et aux tests de logiciels. Ces références ont pour but de fournir un framework (comme dans les références aux normes ISO/IEC 25059 et ISO/IEC 25010 concernant un modèle de qualité pour les systèmes basés sur l'IA), ou d'offrir une source d'informations complémentaires si le lecteur le souhaite. Veuillez noter que les syllabi utilisent ces documents normatifs à titre de référence. Les documents de normes ne sont pas destinés à être examinés.

0.9 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.

Le syllabus utilise la terminologie (c'est-à-dire le nom et la signification) des termes employés dans le domaine des tests logiciels et de l'assurance qualité, conformément au glossaire de l'ISTQB®.

0.10 Organisation de ce syllabus

Il y a sept chapitres dont le contenu fait l'objet d'une évaluation. Le titre principal de chaque chapitre indique la durée de celui-ci ; aucun calendrier n'est précisé au niveau du chapitre. Pour les formations agréées, le syllabus prévoit un minimum de 19,5 heures d'enseignement, réparties entre les sept chapitres comme suit:

- Chapitre 1 : 120 minutes - Introduction à l'intelligence artificielle (IA)
  - Comprendre les principales différences entre les systèmes basés sur l'IA et les systèmes conventionnels, et explorer l'éventail des capacités de l'IA, allant de l'IA étroite à la super-IA.
  - Acquérir une compréhension fondamentale des technologies d'IA, y compris l'IA générative (GenAI), ainsi que du matériel, de l'hébergement et des frameworks de développement utilisés pour implémenter des systèmes de machine learning (MLS).
  - Découvrir comment les réglementations et les normes influencent le développement et le test des solutions basées sur l'IA.
- Chapitre 2 : 45 minutes - Caractéristiques de qualité des systèmes basés sur l'IA
  - Découvrir les caractéristiques de qualité propres aux systèmes basés sur l'IA, notamment celles définies dans la norme ISO/IEC 25059, et comprendre les considérations liées à la sûreté lors de l'utilisation de l'IA dans des systèmes critiques.
  - Explorer comment définir des critères d'acceptation adaptés au comportement et aux performances spécifiques des solutions basées sur l'IA.
- Chapitre 3 : 375 minutes - Machine learning
  - Comprendre les différents types de machine learning, les étapes clés du processus de développement, ainsi que la manière dont les modèles pré-entraînés, le réglage fin et la "retrieval-augmented generation (RAG)" contribuent aux systèmes modernes basés sur l'IA.
  - Découvrir la préparation des données de test, les rôles des jeux de données d'entraînement, de validation et de test, et leur influence sur le développement et les performances des modèles ML.

- Explorer les réseaux neuronaux, notamment leur structure et leurs mesures de couverture, et acquérir une expérience pratique des métriques de performance et de l'utilisation d'une matrice de confusion.

- Chapitre 4 : 195 minutes - Test des systèmes basés sur l'IA
  - Comprendre les défis spécifiques liés aux tests des systèmes basés sur l'IA, notamment les différences de testabilité entre les systèmes « figés » et les systèmes adaptatifs, la nécessité de recourir à des tests statistiques et les difficultés liées à la définition d'oracles de test.
  - Apprendre à tester les modèles d'IA générative (GenAI) et les grands modèles de langage (LLM), en utilisant des techniques telles que le « red teaming » et les tests exploratoires pour les tâches de test exécutées par l'IA.
  - Explorer les stratégies de test pour le MLS, couvrant différents niveaux de test et l'application de tests basés sur le risque.
- Chapitre 5 : 180 minutes - Test des données d'entrée pour les systèmes de Machine Learning
  - Apprendre à tester et à valider les données d'entrée pour les systèmes de machine learning, notamment grâce à des techniques permettant de détecter les biais, de vérifier l'exactitude des étiquettes, d'évaluer la représentativité des données et de tester le pipeline de données.
- Chapitre 6 : 225 minutes - Test modèle de MLdes modèles pour les systèmes de Machine Learning
  - Découvrir des approches de test visant à l'atténuation des risques liés aux modèles ML, notamment l'examen de la documentation, les tests fonctionnels de ML ainsi que la détection du surajustement, du sous-ajustement et de la dérive.
  - Apprendre des approches de test avancées, notamment les tests adverses, les tests A/B, les tests dos-à-dos et l'utilisation d'attaques pour mettre au jour les faiblesses des modèles.
  - Acquérir une compréhension pratique des tests métamorphiques, notamment comment dériver et appliquer des cas de test lorsque les oracles de test traditionnels sont insuffisants.
- Chapitre 7 : 30 minutes - Test du développement de Machine Learning
  - Découvrir les approches de test et les stratégies de test permettant d'atténuer les risques lors du développement et du déploiement de systèmes de machine learning, afin de garantir un fonctionnement robuste du système dans les environnements de production.

1 Introduction à l'Intelligence Artificielle – 120 minutes

Mots clés
Aucun

Mots-clés spécifiques à l'IA
Système basé sur l'IA, intelligence artificielle, IA générale, machine learning, framework de développement de machine learning, IA étroite, super-IA

Objectifs d'apprentissage du chapitre 1:

1.1 Introduction à l'IA

AI-1.1.1 (K2) Distinguer les systèmes basés sur l'IA des systèmes conventionnels

AI-1.1.2 (K2) Distinguer l'IA étroite, l'IA générale et la super-IA

AI-1.1.3 (K2) Expliquer les différents types de technologies d'IA

AI-1.1.4 (K2) Expliquer l'IA générative

AI-1.1.5 (K2) Comparer les choix disponibles en matière de matériel pour implémenter des systèmes de machine learning

AI-1.1.6 (K2) Comparer les options pour le développement et l'hébergement de modèles d'IA

AI-1.1.7 (K2) Résumer les fonctionnalités offertes par les frameworks de développement de machine learning

AI-1.1.8 (K2) Expliquer comment les réglementations et les normes influencent le développement et les tests des systèmes basés sur l'IA

1.1 Introduction à l'IA

Ce chapitre présente les principales différences entre les systèmes conventionnels et ceux basés sur l'IA, en mettant l'accent sur leurs approches de conception, leur adaptabilité et leur explicabilité. Il décrit l'éventail des capacités de l'IA, allant de l'IA étroite à l'IA générale et à la super-IA, et présente les technologies principales telles que le ML, le deep learning et l'IA générative. Le chapitre décrit le matériel et les environnements de développement courants pour les systèmes basés sur l'IA, ainsi que les frameworks de ML les plus utilisés. Enfin, il examine les normes réglementaires et techniques essentielles qui guident le développement et le déploiement responsables de l'IA dans divers domaines.

1.1.1 Systèmes basés sur l'IA et systèmes conventionnels

Les systèmes informatiques classiques sont généralement programmés à l'aide de langages impératifs, dans lesquels les développeurs définissent explicitement des instructions étape par étape, notamment des constructions telles que les instructions « if-then-else » et des boucles. Cette approche déterministe contribue à rendre le comportement du système prévisible et transparent, ce qui permet aux utilisateurs de comprendre plus facilement comment les entrées produisent des sorties différentes. En revanche, la plupart des systèmes basés sur l'IA, en particulier ceux qui exploitent le ML (machine learning), ne suivent pas de règles prédéfinies. Ils analysent plutôt les modèles présents dans les données pour déterminer comment réagir à de nouvelles entrées. Par exemple, un système de reconnaissance d'images basé sur l'IA et entraîné à identifier des chats ne s'appuie pas sur des règles explicitement codées. Il apprend plutôt à partir d'un jeu de données d'images de chats, en extrayant des modèles qu'il applique ensuite à des images inconnues pour les classer avec précision.

Une différence fondamentale entre les systèmes conventionnels et ceux basés sur l'IA réside dans leur approche de la résolution de problèmes. De nombreux systèmes basés sur l'IA s'appuient sur le raisonnement probabiliste, l'inférence statistique et la reconnaissance de formes pour générer des résultats. Cela permet aux modèles d'IA de gérer plus efficacement les formes complexes d'incertitude et d'ambiguïté, ce qui se traduit par des résultats qui ne sont pas toujours prévisibles.

L'explicabilité constitue un défi majeur pour les systèmes basés sur l'IA. De nombreux modèles d'IA, en particulier les architectures de deep learning, peuvent contenir des milliards de paramètres, ce qui rend leur fonctionnement interne difficile à interpréter pour les humains. Cette nature de « boîte noire » soulève des inquiétudes dans des domaines critiques tels que la santé, la finance, la défense et les transports, où il est crucial de comprendre pourquoi un modèle d'IA a pris une décision particulière. Assurer la transparence (voir 2.1.1) et la facilité de compréhension dans la prise de décision pilotée par l'IA est devenu un enjeu central de la réglementation de l'IA.

Une autre différence notable réside dans l'adaptabilité. Les systèmes traditionnels sont statiques et nécessitent généralement des mises à jour manuelles pour intégrer de nouvelles connaissances ou s'adapter aux changements environnementaux. Les systèmes basés sur l'IA, en revanche, sont capables d'auto-apprentissage et améliorent continuellement leurs performances à mesure qu'ils traitent de nouvelles données. Cette adaptabilité rend l'IA particulièrement performante dans les environnements dynamiques. Elle nécessite également un suivi continu afin de garantir une adéquation constante avec les exigences principales.

1.1.2 IA étroite, IA générale et super-IA

L'IA étroite, également appelée IA faible, est conçue pour accomplir des tâches spécifiques et regroupe tous les systèmes d'IA actuellement déployés. Les systèmes basés sur l'IA étroite fonctionnent dans un

domaine restreint et peuvent se montrer très efficaces pour résoudre des problèmes spécialisés, tels que la reconnaissance d'images, le traitement de la parole et la traduction linguistique. Cependant, ils sont incapables de généraliser au-delà des fonctions qu'ils ont apprises. Par exemple, un système basé sur l'IA qui excelle dans la reconnaissance faciale ne peut pas effectuer de traduction linguistique à moins d'avoir été explicitement réentraîné à cette fin. L'IA de pointe (Frontier AI), un sous-ensemble de l'IA étroite, représente la forme la plus avancée de ces systèmes, repoussant les limites des capacités actuelles grâce aux modèles GenAI. L'IA de pointe comprend des systèmes à grande échelle dotés de capacités de prise de décision hautement autonomes ; cependant, ils restent spécifiques à une tâche et n'ont pas encore atteint la polyvalence de l'IA générale.

L'IA générale, également appelée « IA forte », désigne un système basé sur l'IA capable d'accomplir la plupart des tâches intellectuelles dont est capable un être humain. L'IA générale serait capable de comprendre, d'apprendre et d'appliquer des connaissances dans un large éventail de tâches sans avoir besoin d'être réentraînée pour chaque nouvelle tâche. Ce type d'IA ferait preuve d'un raisonnement et d'une adaptabilité semblables à ceux des humains, capable de résoudre des problèmes inconnus dans divers domaines, tout comme le font les humains. Malgré des progrès significatifs en matière d'IA, aucun système basé sur l'IA ne possède aujourd'hui d'intelligence générale.

La super-IA (superintelligence artificielle) est une forme d'IA dans laquelle un système basé sur l'IA s'améliore en permanence sans nécessiter d'intervention ou de contrôle humain. Pour que la super-IA soit possible, l'accès à Internet n'est pas strictement nécessaire, mais un tel accès pourrait considérablement étendre ses capacités et son influence. Elle surpasserait l'intelligence humaine et l'IA générale, ce qui, selon beaucoup, constituerait un risque existentiel pour l'humanité. Le moment où les systèmes basés sur l'IA passeraient de l'IA générale à la super IA, si cela devait se produire, est communément appelé la singularité technologique.

1.1.3 Les différents types de technologies d'IA

L'intelligence artificielle englobe toute une gamme de technologies, chacune étant adaptée à des tâches et à des défis spécifiques. Le ML, l'une des principales branches de l'IA, permet aux systèmes d'apprendre à partir de données et de construire des modèles sans programmation explicite. Alors que certains systèmes de ML peuvent s'adapter et s'améliorer en continu grâce à de nouvelles données tout au long de leur cycle de vie, d'autres fonctionnent sur la base des connaissances initialement acquises et nécessitent un réentraînement explicite pour mettre à jour leurs capacités. Le ML comprend plusieurs approches :

- L'apprentissage supervisé utilise des données étiquetées et des algorithmes, tels que la régression linéaire et les arbres de décision, pour des tâches telles que la prédiction et la classification.
- L'apprentissage non supervisé met en évidence des modèles dans des données non étiquetées grâce à des techniques telles que le clustering (en utilisant des algorithmes de clustering).
- L'apprentissage par renforcement permet à des agents intelligents d'apprendre des comportements optimaux par le biais d'interactions par essais et erreurs avec leur environnement.

Les principales technologies de ML comprennent les réseaux neuronaux, les modèles bayésiens, les machines à vecteurs de support (SVM) et les forêts aléatoires. Pour en savoir plus sur le ML, voir le chapitre 3.

Le deep learning (DL), un sous-ensemble des modèles ML, utilise des réseaux neuronaux profonds pour résoudre des problèmes complexes. Par exemple:

- Les réseaux neuronaux convolutifs (CNN) sont très efficaces pour la reconnaissance d'images et la détection d'objets.
- Les réseaux neuronaux récurrents (RNN) sont spécialisés dans le traitement de données séquentielles, telles que le texte ou les séries chronologiques.
- Les Transformers gèrent les dépendances à longue portée dans les séquences, alimentant ainsi les modèles de traitement du langage naturel et les Transformers de vision pour les images.

L'IA générative s'appuie sur ces technologies pour créer de nouveaux contenus, notamment du texte, des images et des fichiers audio (voir 1.1.4). Elle repose sur des modèles tels que les LLM, qui associent des réseaux neuronaux profonds (DNN) au traitement du langage naturel (NLP) afin d'analyser et de générer un langage semblable à celui des humains.

Parmi les autres technologies d'IA spécialisées, on peut citer:

- Le traitement du langage naturel (NLP) pour l'analyse linguistique, notamment pour des tâches telles que l'analyse des sentiments et la traduction automatique.
- La vision par ordinateur pour l'analyse des données visuelles, au soutien d'applications telles que la reconnaissance faciale et la robotique.
- La logique floue pour le raisonnement en situation d'incertitude.
- Les algorithmes de recherche pour résoudre des problèmes d'optimisation, tels que la navigation et la prise de décision stratégique.
- Les systèmes de raisonnement basés sur des règles, ou systèmes experts, pour l'aide à la décision structurée.

Bien que l'intégration complète de ces technologies d'IA reste limitée, de nouveaux développements tels que les LLM démontrent le potentiel de combiner différentes technologies d'IA en systèmes intelligents unifiés. L'IA agentique étend ces technologies grâce à des agents autonomes qui planifient, raisonnent et agissent de manière indépendante pour atteindre des objectifs dans des environnements dynamiques.

1.1.4 IA générative

L'IA générative (GenAI) désigne les systèmes basés sur l'IA spécialisés dans la création de nouveaux contenus, tels que du texte, des images, des vidéos, de la musique ou des données de complexité élevée, même si bon nombre d'entre eux soutiennent également la classification et la prédiction. Ces systèmes apprennent à partir de vastes quantités de données pour produire des résultats qui ressemblent à leurs données d'entraînement, ce qui permet un large éventail d'applications créatives et pratiques.

Les principales technologies à la base de la GenAI comprennent les réseaux antagonistes génératifs (GAN), les modèles de diffusion et les Transformers. Les GAN utilisent deux réseaux neuronaux en compétition pour créer des données synthétiques très réalistes. Les modèles de diffusion génèrent du contenu en ajoutant puis en supprimant progressivement du bruit aux données, ce qui donne des résultats de haute qualité. Les modèles Transformers, qui sous-tendent les LLM, utilisent des mécanismes d'auto-attention pour générer du texte cohérent et pertinent sur le plan contextuel, et sont de plus en plus adaptés à des tâches multimodales. Au-delà de leurs fondements techniques, les systèmes GenAI soulèvent d'importantes préoccupations sociétales et éthiques. L'utilisation abusive est une préoccupation majeure : ces modèles peuvent être exploités pour créer des « deepfakes », diffuser de la désinformation ou générer du contenu frauduleux convaincant, sapant ainsi la confiance dans les médias

numériques et le discours public. La facilité de production de contenu synthétique amplifie les risques liés à la vie privée, à la sécurité et à la manipulation, mais elle peut apporter des bénéfices notables et ouvrir des opportunités d'innovation dans les domaines du divertissement, du marketing, de l'éducation et de la recherche.

L'impact sur l'emploi est également notable, en particulier parmi les professions intellectuelles. Alors que l'IA générative automatise des tâches telles que la rédaction, la conception, le codage et même la documentation juridique ou médicale, le débat sur les risques de suppression d'emplois s'intensifie. Si ces technologies peuvent stimuler la productivité et favoriser de nouvelles opportunités créatives, elles peuvent également entraîner des bouleversements sur le marché du travail et nécessiter une reconversion à grande échelle.

La durabilité est un autre enjeu urgent. L'entraînement et l'exécution de grands modèles de GenAI nécessitent d'importantes ressources informatiques, ce qui entraîne une forte consommation d'énergie et une empreinte carbone significative. Cet impact environnemental a suscité des appels en faveur de conceptions de modèles plus efficaces et d'infrastructures plus écologiques. La plupart des outils GenAI pratiques actuels reposent sur des modèles de base, qui sont ensuite affinés pour des applications spécifiques. Le domaine évolue également vers des modèles multimodaux capables de traiter et de générer du contenu sous forme de texte, d'images et d'audio, permettant ainsi des systèmes basés sur l'IA plus riches et plus flexibles. Des frameworks réglementaires, tels que la loi européenne sur l'IA [EU AI Act], voient le jour pour encadrer le développement et l'utilisation responsables de ces technologies.

1.1.5 Matériel pour les systèmes de machine learning

Divers matériels sont utilisés pour les MLS. Différents types de matériels peuvent être utilisés pour l'entraînement et l'inférence. Par exemple, un modèle de reconnaissance vocale peut fonctionner sur un smartphone d'entrée de gamme, même si l'accès à la puissance du cloud computing peut être nécessaire pour l'entraîner.
Le ML tire généralement bénéfice d'un matériel offrant le soutien suivant :

- La capacité à traiter de grandes structures de données.
- Le traitement massivement parallèle (concurrent), par exemple pour prendre en charge la multiplication matricielle.
- L'arithmétique à faible précision (quantification) : cette approche utilise moins de bits pour le calcul (par exemple, 4 bits au lieu de 32 bits), ce qui se traduit par un traitement plus rapide, une consommation d'énergie réduite, des puces plus petites et plus rentables, ainsi que des exigences en bande passante réduites.

Les processeurs centraux (CPU) à usage général fournissent un soutien pour l'exploitation de tâches complexes avec une grande précision, ce qui n'est généralement pas une exigence pour les applications ML, mais ils ne disposent généralement que de quelques cœurs. De ce fait, leur architecture est moins efficace pour l'entraînement et l'exécution de modèles ML que celle des processeurs graphiques (GPU), qui disposent de milliers de cœurs et sont conçus pour effectuer un traitement graphique massivement parallèle, mais relativement simple. Par conséquent, les GPU surpassent généralement les CPU dans les applications ML, même si ces derniers fonctionnent généralement à des fréquences d'horloge plus élevées. Pour les tâches ML à petite échelle, les GPU constituent généralement la meilleure option. Certains matériels sont spécialement conçus pour l'IA, tels que les circuits intégrés spécifiques à une application (ASIC) et les dispositifs « système sur puce » (SoC). Ces solutions spécifiques à l'IA disposent de plusieurs cœurs, d'une gestion de données spécialisée et de la capacité d'effectuer un

traitement en mémoire. Elles sont particulièrement adaptées à l'informatique en périphérie (NDT: edge computing), tandis que l'entraînement du modèle de ML est effectué dans le cloud à l'aide de matériel spécialisé.

Le développement d'architectures matérielles spécifiques à l'IA se poursuit. Cela inclut les processeurs neuromorphiques, qui ne reposent pas sur l'architecture traditionnelle de von Neumann, mais sur des conceptions inspirées du cerveau et imitant les structures neuronales.

1.1.6 Développement et hébergement des modèles IA

Les systèmes basés sur l'IA peuvent être achetés auprès de fournisseurs tiers ou développés en interne au sein d'une organisation. Ces systèmes s'appuient généralement sur des modèles pré-entraînés et peuvent être déployés sur site ou dans le cloud, les modèles eux-mêmes étant hébergés soit sur site, soit dans le cloud, où les options basées sur le cloud sont souvent accessibles sous forme de service (AIaaS). Les systèmes basés sur l'IA tiers se présentent généralement sous forme de modèles pré-entraînés ou d'IA en tant que service (AIaaS), ce qui permet un déploiement et une mise sur le marché plus rapides. À l'inverse, les systèmes basés sur l'IA développés en interne (configurations sur site ou cloud personnalisées) peuvent être mieux adaptés à des exigences spécifiques, mais leur développement nécessitera probablement des compétences spécialisées, qu'elles proviennent d'experts internes ou d'équipes externalisées. Le développement local (codage et entraînement sur site) permet un contrôle direct et garantit la confidentialité. Les petits modèles, tels que les arbres de décision ou les réseaux neuronaux compacts, peuvent être développés sur des ordinateurs personnels, tandis que les modèles de taille moyenne peuvent nécessiter des GPU dédiés. Pour les modèles à grande échelle, des regroupements de serveurs sur site haute performance deviennent nécessaires, avec les coûts d'énergie, de refroidissement et de matériel qui y sont associés.

Le développement dans le cloud offre une grande flexibilité. Les clouds publics, en particulier, proposent des environnements préconfigurés avec une tarification à l'utilisation, ce qui limite l'investissement initial en matériel et permet une évolutivité aisée. À l'inverse, les clouds privés peuvent offrir une sécurité et une confidentialité accrues pour les applications qui en ont besoin, mais ce niveau de contrôle nécessite un investissement initial plus important dans l'infrastructure.

De nombreuses organisations adoptent des approches hybrides, notamment en développant des prototypes localement avant de passer à une infrastructure cloud, en conservant les composants sensibles sur site (comme la préparation des données privées) et en exploitant les ressources cloud pour les tâches nécessitant une grande puissance de calcul.

Les modèles d'IA peuvent être hébergés dans divers environnements, allant des configurations locales aux plateformes basées sur le cloud. L'hébergement local consiste à exécuter des modèles de petite taille sur des ordinateurs personnels ou des smartphones, ce qui garantit la confidentialité et élimine les coûts de licence liés au cloud, mais offre des capacités matérielles limitées. Pour les modèles d'IA plus volumineux, les entreprises peuvent mettre en place des serveurs dédiés, qui nécessitent un investissement initial important mais offrent un contrôle accru. L'hébergement des modèles d'IA sur le cloud peut se faire sur des clouds publics ou privés. Les services de cloud public offrent une évolutivité pour accéder à une infrastructure robuste et puissante, éliminant les soucis de maintenance et les rendant idéaux pour les charges de travail fluctuantes. Les clouds privés offrent des bénéfices similaires, avec une sécurité renforcée et des options de personnalisation, et sont gérés en interne ou par des fournisseurs dédiés, bien qu'à un coût plus élevé.

Les approches hybrides combinent ces méthodes, permettant ainsi aux organisations d'exécuter certaines opérations en local tout en tirant parti de l'élasticité du cloud pour les tâches gourmandes en ressources. Les solutions optimales de développement et d'hébergement, qui sont généralement choisies

séparément, dépendent de facteurs tels que la taille du modèle, sa complexité, les exigences de performance, les contraintes budgétaires, les considérations en matière de sécurité et de confidentialité des données, les besoins de déploiement et les exigences réglementaires. Certaines organisations adoptent des stratégies multi-tiers afin de trouver un équilibre entre efficacité et contrôle.

1.1.7 Frameworks de développement de ML

Les frameworks de développement de ML fournissent une boîte à outils permettant de créer et d'entraîner des modèles ML. Parmi les fonctionnalités typiques offertes par ces frameworks, on trouve :

- Traitement des données : ils facilitent le chargement, le prétraitement et la gestion des données utilisées pour entraîner et tester le modèle. Cela peut impliquer le nettoyage, le formatage et la transformation des données dans un format adapté au modèle choisi.
- Construction du modèle de ML : ces frameworks proposent des bibliothèques d'algorithmes d'apprentissage automatique et des outils permettant de réaliser la conception de l'architecture du modèle de ML construit. Cela inclut la spécification du type de modèle (par exemple, réseau neuronal, arbre de décision), du nombre de couches et de connexions, ainsi que des opérations mathématiques effectuées au sein du modèle.
- Entraînement et optimisation : les frameworks fournissent des algorithmes qui ajustent de manière itérative les paramètres internes du modèle en fonction des données d'entraînement et du résultat souhaité. L'objectif est d'optimiser les performances du modèle dans l'accomplissement de la tâche souhaitée (par exemple, classification, régression ML). Certains frameworks peuvent offrir du soutien pour l'entraînement distribué et améliorer ou affiner des modèles pré-entraînés.
- Évaluation : ils offrent des outils permettant d'évaluer les performances du modèle entraîné sur des données non vues. Cela peut impliquer de réaliser des mesures de l'exactitude, de la précision et du rappel pour les tâches de classification, ou des taux d'erreur pour les tâches de régression ML (voir 3.1.1).
- Déploiement : certains frameworks offrent des fonctionnalités permettant de déployer le modèle entraîné pour une utilisation en conditions réelles. Cela peut impliquer de convertir le modèle dans un format adapté à l'intégration avec des applications web, des appareils mobiles, des appareils en périphérie ou des systèmes embarqués.

Ces frameworks peuvent fonctionner à différents niveaux d'abstraction. Certains proposent une interface de programmation d'application (API) de bas niveau, offrant aux développeurs un plus grand contrôle sur la construction des modèles, mais nécessitant davantage de compétences en programmation. D'autres proposent une API de haut niveau, simplifiant la création de modèles mais offrant moins d'options de personnalisation.

Différents frameworks peuvent se concentrer sur différents domaines d'application. Certains sont polyvalents et offrent un large soutien pour de nombreux domaines d'application. À l'inverse, d'autres sont plus spécialisés, se concentrant sur des domaines spécifiques tels que la reconnaissance d'images, la reconnaissance vocale et la traduction linguistique.
Le choix du framework le plus approprié peut dépendre de plusieurs facteurs, tels que :

- Le domaine d'application ;
- La nécessité d'une interface utilisateur conviviale pour un prototypage rapide ;

- La configurabilité pour les modèles complexes ;
- L'expertise des utilisateurs ;
- Les considérations de déploiement, car certains frameworks sont mieux adaptés aux environnements aux ressources limitées ;
- Le niveau de soutien (de la communauté) ;
- La maturité de l'écosystème.

1.1.8 Réglementations et normes pour l'IA

Les réglementations et les normes en matière d'IA sont essentielles au développement, au déploiement et à l'utilisation responsables de l'IA. L'objectif principal est de renforcer la confiance dans l'IA et de contribuer à la concrétisation de ses bénéfices tout en atténuant ses risques potentiels. Idéalement, la conformité avec ces réglementations et normes devrait garantir que les systèmes basés sur l'IA soient sûrs, équitables, transparents, durables, responsables, éthiques et utilisés de manière responsable.

Au niveau international, les Principes de l'OCDE sur l'IA [OCDE IA] et le rapport des Nations unies sur la gouvernance de l'IA au service de l'humanité [ONU Gov IA] constituent des instruments de droit non contraignant influents qui favorisent une compréhension commune de la gestion responsable de l'IA. Ils servent de boussole aux gouvernements nationaux et aux organisations dans l'élaboration de leurs propres stratégies en matière d'IA. Ces principes mettent l'accent sur une IA centrée sur l'humain, sur les considérations éthiques et sur l'importance de la coopération internationale.

La loi européenne sur l'IA marque une étape réglementaire décisive, illustrant une approche fondée sur les risques en matière de réglementation de l'IA. En classant les systèmes basés sur l'IA par niveau de risque, allant de minimal à inacceptable, la réglementation est adaptée en conséquence. Les systèmes à haut risque, en particulier ceux qui ont une incidence sur les droits fondamentaux ou la sûreté, sont soumis à des exigences strictes qui englobent des tests rigoureux, la gouvernance des données et un contrôle humain. Les sanctions financières substantielles en cas de non-conformité, calculées en pourcentage du chiffre d'affaires mondial, soulignent l'engagement de l'UE en matière d'application de la réglementation. En revanche, de nombreux pays hors de l'UE adoptent une approche plus permissive, privilégiant des réglementations moins contraignantes afin d'encourager l'innovation.

Les normes techniques, développées par des organisations telles que l'ISO et l'IEEE, sont essentielles pour traduire les aspirations de haut niveau en implémentations concrètes. Elles fournissent des spécifications techniques concrètes et des bonnes pratiques, comblant ainsi le fossé entre la politique et la pratique. Par exemple, la norme ISO/IEC TR 29119-11 fournit des orientations détaillées sur les tests des systèmes basés sur l'IA, un élément essentiel pour démontrer la conformité réglementaire. Parallèlement, la série de normes ISO/IEC 42119 est en cours de développement afin de couvrir divers aspects des tests des systèmes basés sur l'IA. En outre, des réglementations spécifiques à certains secteurs voient le jour dans des domaines tels que la santé et la finance, reconnaissant les risques particuliers posés par l'IA dans ces domaines.

Pour s'y retrouver efficacement dans ce paysage en constante évolution de la gouvernance de l'IA, un dialogue et une collaboration permanents sont essentiels. Les pouvoirs publics, le secteur privé, le monde universitaire et la société civile doivent s'engager activement afin de parvenir à une approche harmonisée et efficace de la gouvernance de l'IA à l'échelle mondiale. De plus, compte tenu de la nature dynamique de l'IA, les réglementations et les normes doivent être régulièrement révisées et mises à jour afin de rester pertinentes et efficaces pour orienter le développement, le déploiement et l'utilisation responsables de l'IA.

2 Caractéristiques de qualité pour les systèmes basés sur l'IA – 45 minutes

Mots clés

Adaptabilité fonctionnelle, exactitude fonctionnelle de l'IA, intervenabilité, robustesse de l'IA, sûreté, atténuation des risques sociétaux et éthiques, transparence, contrôle par l'utilisateur

Mots-clés spécifiques à l'IA

Aucun

Objectifs d'apprentissage du chapitre 2 :

2.1 Caractéristiques de qualité des systèmes basés sur l'IA

AI-2.1.1 (K2) Classifier les comportements des systèmes basés sur l'IA selon les caractéristiques de qualité définies dans la norme ISO/IEC 25059

AI-2.1.2 (K2) Expliquer les considérations particulières qui se posent lorsque l'IA est utilisée dans des systèmes liés à la sûreté

2.2 Critères d'acceptation pour les systèmes basés sur l'IA

AI-2.2.1 (K2) Donner des exemples de critères d'acceptation pour les systèmes basés sur l'IA

2.1 Caractéristiques de qualité des systèmes basés sur l'IA

Cette section traite des caractéristiques de qualité spécifiques à l'IA décrites dans la norme ISO/IEC 25059, qui étend les modèles traditionnels de qualité logicielle afin de prendre en compte les aspects propres aux systèmes basés sur l'IA. Elle présente des caractéristiques nouvelles et adaptées, notamment l'exactitude fonctionnelle de l'IA, l'adaptabilité fonctionnelle, la contrôlabilité par l'utilisateur, la transparence, la robustesse de l'IA et l'intervenabilité, ainsi que l'atténuation des risques sociétaux et éthiques. Les principaux défis en matière de sûreté de l'IA, tels que les spécifications vagues, le non-déterminisme, l'auto-apprentissage, l'explicabilité limitée et l'évolution des normes, sont également abordés, l'accent étant mis sur leur impact sur les tests et la réglementation.

2.1.1 Caractéristiques qualité spécifique à l'IA

La norme ISO/IEC 25059 étend le modèle de qualité de la norme ISO/IEC 25010 afin de prendre en compte les spécificités de l'IA. Cette extension évalue les systèmes basés sur l'IA sous deux angles : la qualité du produit et la qualité d'utilisation. Du point de vue des tests, ces caractéristiques de qualité influencent directement la manière dont les objectifs de test sont définis, dont les critères d'acceptation sont formulés et dont les résultats des tests sont interprétés pour les systèmes basés sur l'IA. Les caractéristiques nouvelles et modifiées, par rapport à la norme ISO/IEC 25010, comprennent:

- Exactitude fonctionnelle de l'IA (qualité du produit) : les systèmes basés sur l'IA, en particulier ceux qui utilisent le ML probabiliste, ne peuvent garantir une précision parfaite. Étant donné qu'un certain taux d'erreur est prévisible dans les résultats de l'IA, le concept d'exactitude fonctionnelle a été adapté en conséquence. La norme ISO/IEC 25059 évalue l'exactitude fonctionnelle en prenant en compte à la fois les résultats corrects et incorrects et en définissant des seuils acceptables pour les résultats incorrects, reflétant ainsi la variabilité inhérente aux résultats des systèmes basés sur l'IA (voir 3.3).
- Adaptabilité fonctionnelle (qualité du produit) : une nouvelle sous-caractéristique de l'adéquation fonctionnelle. La capacité du système basé sur l'IA à s'adapter de manière autonome aux changements de son environnement opérationnel après son déploiement.
- Contrôlabilité par l'utilisateur (qualité du produit) : une nouvelle sous-caractéristique de la capacité d'interaction (à noter que la capacité d'interaction est elle-même un nouveau terme qui remplace l'utilisabilité dans la version 2023 de la norme ISO/IEC 25010). Une propriété d'un système basé sur l'IA permettant à un humain ou à un autre agent externe d'intervenir dans son fonctionnement en temps opportun.
- Transparence (qualité du produit et qualité d'utilisation) : nouvelle sous-caractéristique de la capacité d'interaction et nouvelle sous-caractéristique de la satisfaction. Elle concerne le degré auquel des informations pertinentes sur le système basé sur l'IA sont communiquées aux parties prenantes (voir 6.1.2).
- Robustesse de l'IA (qualité du produit) : nouvelle sous-caractéristique de la fiabilité. Elle décrit la capacité d'un système basé sur l'IA à maintenir son niveau d'exactitude fonctionnelle de l'IA quelles que soient les circonstances, telles que la présence de données d'entrée biaisées, adverses ou invalides, d'interférences externes, de conditions environnementales défavorables et d'une mauvaise utilisation par l'opérateur.

- Intervenabilité (qualité du produit) : une nouvelle sous-caractéristique de la sécurité. Le degré auquel un opérateur peut intervenir en temps opportun dans le fonctionnement d'un système basé sur l'IA afin de prévenir tout préjudice ou danger.
- Atténuation des risques sociétaux et éthiques (qualité d'utilisation) : une nouvelle sous-caractéristique de l'« absence de risque ». Prend en compte de nombreux domaines pour atténuer les risques sociétaux et éthiques, notamment la responsabilité, l'équité et la non-discrimination, la responsabilité professionnelle, la promotion des valeurs humaines, la vie privée, la sûreté et la sécurité, le contrôle humain de la technologie, l'implication et le développement communautaires, la conception centrée sur l'humain, le respect de l'État de droit, le respect des normes internationales de comportement, la durabilité environnementale et les pratiques de travail.

2.1.2 IA et sûreté

Les systèmes liés à la sûreté peuvent potentiellement causer des blessures ou des dommages aux personnes, aux biens ou à l'environnement. Le développement et les tests de systèmes liés à la sûreté ne reposant pas sur l'IA peuvent demander beaucoup d'efforts, mais restent réalisables ; en revanche, pour les systèmes basés sur l'IA, plusieurs défis supplémentaires se posent:

- Spécifications : dans les systèmes traditionnels liés à la sécurité, les exigences sont définies pour l'ensemble du système et réaménagées jusqu'à ce que le développeur puisse les transformer en code. Les exigences de nombreux systèmes basés sur l'IA commencent souvent par des objectifs vagues, puis sont implicitement fournies via les données d'entraînement qui codifient des modèles, des règles et des objectifs, sans formaliser entièrement chaque détail dès le départ. Cela peut signifier que la traçabilité nécessaire entre les exigences et l'implémentation est insuffisante pour les systèmes basés sur l'IA.
- Non-déterminisme : cette caractéristique de nombreux systèmes basés sur l'IA rend intrinsèquement difficile la garantie d'un comportement précis de ces systèmes. Même des modèles rigoureusement testés peuvent présenter un comportement inattendu en raison de facteurs tels que la génération de nombres aléatoires ou de légères variations dans les valeurs d'entrée.
- Auto-apprentissage : des tests rigoureux sont utilisés pour démontrer l'intégrité de sûreté d'un système avant son déploiement. Pour les systèmes basés sur l'IA et l'auto-apprentissage, cela est compromis car le comportement du système s'éloigne progressivement du comportement initialement testé. La gestion de la manière dont le modèle apprend et des données qu'il utilise peut parfois aider à éviter l'émergence de nouveaux comportements problématiques. Sinon, des mesures de sûreté peuvent être implémentées pour empêcher le modèle d'apprendre ou de prendre des décisions susceptibles de compromettre la sûreté (par exemple, un composant de modération de contenu pour filtrer les invites).
- Explicabilité et transparence : pour les systèmes liés à la sûreté, il est essentiel de comprendre comment et pourquoi le système prend ses décisions. Cependant, les processus décisionnels des systèmes basés sur l'IA manquent souvent de transparence. Les techniques d'IA explicable, telles que LIME (Local interpretable model-agnostic explanations), peuvent fournir des informations sur le raisonnement du système basé sur l'IA ; toutefois, leur disponibilité est limitée et elles peuvent nuire aux performances du système.

- Évolution de la réglementation : Le paysage réglementaire des systèmes basés sur l'IA liés à la sécurité est en constante évolution. L'utilisation de l'IA n'est actuellement pas incluse dans les normes internationales matures de sûreté de fonctionnement, et certaines de ces normes interdisent même son utilisation dans de tels systèmes. La loi européenne sur l'IA [EU AI Act] (voir 1.1.8) classe les systèmes d'IA utilisés comme composants de sûreté (par exemple dans l'aviation, les dispositifs médicaux ou l'automobile) comme présentant un risque élevé et impose des exigences strictes en matière de développement et de tests.

2.2 Critères d'acceptation pour les systèmes basés sur l'IA

Cette section présente les critères d'acceptation relatifs aux caractéristiques de qualité spécifiées dans la norme ISO/IEC 25059, ainsi qu'à la sûreté. Pour les systèmes basés sur l'IA, les critères d'acceptation doivent souvent être de nature statistique, probabiliste ou basés sur des seuils plutôt que binaires, ce qui pose des défis supplémentaires en matière de tests.

2.2.1 Critères d'acceptation pour les systèmes basés sur l'IA

Lors de l'évaluation de la qualité d'un système basé sur l'IA, il est essentiel de prendre en compte à la fois les caractéristiques de qualité fonctionnelles et non fonctionnelles. Cela permet de s'assurer que le système fonctionne comme prévu et qu'il répond à des exigences de qualité plus générales. Les normes ISO/IEC 25010 et ISO/IEC 25059 fournissent un framework complet pour définir la qualité des logiciels. Dans cette section, l'accent est mis sur les critères d'acceptation associés aux caractéristiques de qualité spécifiques à l'IA (c'est-à-dire celles définies dans la norme ISO/IEC 25059) et à la sûreté (voir 2.1.2). Le tableau suivant présente des exemples de critères d'acceptation pour la sûreté et pour chacune des caractéristiques de qualité définies dans la norme ISO/IEC 25059.

Caractéristique	**Exemples de critères d'acceptation**
Exactitude fonctionnelle de l'IA (voir 3.3)	• Précision de 95 % pour un système de reconnaissance d'images. • Taux de rappel de 90 % pour un système de prédiction des défauts.
Adaptabilité fonctionnelle	• Le système de gestion du moteur dispose d'un délai maximal de 20 secondes pour s'adapter lorsqu'il franchit un seuil d'altitude défini. • Un service de streaming vidéo doit adapter sa page d'accueil de manière à recommander au moins 40 % de documentaires lorsqu'un utilisateur a regardé trois documentaires dans leur intégralité au cours d'une même session.
Contrôlabilité par l'utilisateur	• Un superviseur peut prendre le contrôle d'un drone autonome en moins de 0,5 seconde lorsque celui-ci envoie un signal de détresse en raison d'une perte de sa position GPS. • Le système de contrôle agricole avertit l'agriculteur lorsque les performances visuelles du capteur se dégradent de plus de 30 %, ce qui permet une prise de contrôle manuelle immédiate ; il se désactive

	complètement si la dégradation dépasse 50 % sans réaction de l'utilisateur.
Transparence	• Des informations suffisantes sont fournies concernant le modèle de ML tiers et la provenance de ses données d'entraînement afin de satisfaire aux exigences de la norme d'entreprise applicable. • Le tableau de bord opérationnel et l'API du système doivent fournir un point de terminaison qui renvoie l'identifiant de version unique du modèle de prédiction actuellement déployé ainsi qu'un lien vers la documentation correspondante.
Robustesse de l'IA	• Le temps de réponse des prévisions du système d'alerte de pénétration de sécurité basé sur l'IA reste inférieur à 1 seconde lorsque l'accès à la base de données centrale des vulnérabilités est interrompu pendant 30 secondes. • Le dispositif IA en périphérie doit passer automatiquement à un mode d'inférence à moindre fidélité et à faible consommation d'énergie (au lieu de s'arrêter) lorsque sa température de fonctionnement interne dépasse 85 °C pendant une période continue de 10 secondes.
Intervenabilité	• Si un robot franchit sa zone de sûreté, la chaîne de production peut être arrêtée dans les 0,5 seconde suivant le déclenchement de l'arrêt. • Afin d'éviter tout risque de coupure de courant, le système de gestion du réseau électrique doit prévoir un délai de confirmation de 30 secondes, pendant lequel un ingénieur peut opposer son veto à toute action proposée par l'IA et classée comme « critique » avant qu'elle ne soit exécutée automatiquement.
Atténuation des risques sociétaux et éthiques	• Le système automatisé de détermination des peines ne fait aucune distinction entre les groupes raciaux selon la métrique d'équité définie. • Le chatbot doit réussir un audit interne de type « red teaming » avec un score d'au moins 95 %, démontrant ainsi qu'il ne génère pas de contenu incitant à la violence, à l'automutilation ou à la haine.
Sûreté	• Les composants non liés à l'IA du système de commande de direction basé sur l'IA sont conformes à la norme ISO 26262-6 au niveau ASIL (niveau d'intégrité de la sûreté automobile) C. • 100 % des relations entre les entrées et les sorties du modèle de ML utilisé dans le système de commande de la centrale nucléaire peuvent être cartographiées avec une précision moyenne d'au moins 99,9 % à l'aide d'un outil d'explicabilité.

	• Les signaux de commande qui dépassent les limites de sûreté spécifiées de plus de 10 % sont analysés et régulés dans les 0,15 seconde suivant leur détection par le sous-système de suivi de la sûreté.

3 Machine Learning – 375 minutes

Mots clés

Couverture des neurones à K multisections, critères de performance fonctionnelle de ML, performance fonctionnelle de ML, mesures de performance fonctionnelle de ML, modèle de ML, couverture des limites des neurones, couverture des neurones, perceptron

Mots-clés spécifiques à l'IA

Association, classification, regroupement, préparation des données, machine learning, algorithme de machine learning, framework de développement de machine learning, workflow de machine learning, modèle pré-entraîné, régression en machine learning, apprentissage par renforcement, apprentissage supervisé, apprentissage non supervisé

Objectifs d'apprentissage du chapitre 3:

3.1 Introduction au Machine Learning

AI-3.1.1 (K2) Distinguer les différentes formes de ML

AI-3.1.2 (K2) Résumer le workflow utilisé pour créer un système de ML

HO-3.1.3 (H2) Créer un modèle de ML

AI-3.1.4 (K2) Résumer l'utilisation des modèles pré-entraînés, du réglage fin et de la retrieval-augmented generation (RAG)

3.2 Données pour le Machine Learning

AI-3.2.1 (K2) Expliquer les activités liées à la préparation des données

HO-3.2.2 (H2) Effectuer la préparation des données afin de fournir le soutien nécessaire à la création d'un modèle de ML

AI-3.2.3 (K2) Comparer l'utilisation des jeux de données d'entraînement, de validation et de test dans le développement d'un modèle de ML

3.3 Mesures de performance fonctionnelle ML pour la classification

AI-3.3.1 (K3) Calculer des mesures de performance fonctionnelle de ML à partir d'un ensemble donné de données de matrice de confusion

HO-3.3.2 (H2) Évaluer un modèle de ML en utilisant des mesures de performance fonctionnelle de ML sélectionnées

HO-3.3.3 (H2) Montrer l'impact de différentes combinaisons de modèles ML et des jeux de données sur l'entraînement et le comportement des modèles

3.4 Réseaux neuronaux

AI-3.4.1 (K2) Expliquer la structure et le fonctionnement d'un réseau neuronal profond

HO-3.4.2 (H1) Se familiariser avec l'implémentation d'un perceptron

AI-3.4.3 (K2) Décrire les différentes mesures de couverture pour les réseaux neuronaux

3.1 Introduction au Machine Learning

Cette section présente les principales catégories d'algorithmes ML : apprentissage supervisé, non supervisé et par renforcement, en distinguant leurs types de problèmes respectifs et leurs applications typiques. Elle décrit le workflow standard de développement de modèles ML, depuis la définition des objectifs et la préparation des données jusqu'à l'entraînement, l'évaluation et le déploiement des modèles, en mettant l'accent sur l'itération et l'intégration système. Des aspects pratiques tels que la création de modèles par la pratique, l'utilisation de modèles pré-entraînés, le réglage fin et la retrieval-augmented generation (RAG) sont également abordés, mettant en avant des approches permettant d'adapter et d'améliorer efficacement les modèles d'IA pour de nouvelles tâches tout en gérant les limitations héritées. La compréhension de ce workflow est essentielle pour les testeurs, car différentes activités de test s'appliquent à différentes étapes, et les défaillances trouvent souvent leur origine dans des étapes antérieures telles que la préparation des données de test ou la sélection des modèles.

3.1.1 Différentes formes de Machine Learning

Les algorithmes de ML se répartissent en trois catégories : l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement.
Dans l'apprentissage supervisé, les algorithmes entraînent des modèles à l'aide de données étiquetées, où chaque ensemble d'entrées est associé à une étiquette de sortie correspondante (par exemple, des images étiquetées « chien » ou « chat »). Le modèle apprend à cartographier les entrées vers les sorties en identifiant des modèles dans les données d'entraînement. L'apprentissage supervisé se divise généralement en :

- Classification : cela consiste à attribuer des entrées à des classes prédéfinies, comme classer les e-mails comme spam ou non, ou reconnaître des objets dans des images.
- Régression en ML : cela consiste à prédire des valeurs numériques continues, comme l'estimation de l'âge d'une personne à partir de données sur son mode de vie ou la prévision des cours boursiers.

Notez que le terme « régression en ML », lorsqu'il est utilisé dans le contexte du ML, diffère de son utilisation dans d'autres syllabi ISTQB®, où la régression décrit le problème des modifications logicielles causant des défauts liés aux changements.

Dans l'apprentissage non supervisé, l'algorithme entraîne des modèles en utilisant des données non étiquetées, en déduisant des modèles ou des structures sans étiquettes de sortie explicites !. Le modèle regroupe les entrées similaires en fonction de caractéristiques communes. L'apprentissage non supervisé est généralement classé en:

- Re regroupement : il s'agit de regrouper des points de données en fonction de leurs similitudes, par exemple en segmentant la clientèle en différents groupes à des fins de marketing ciblé.
- Association : il s'agit d'identifier des relations ou des dépendances entre les attributs des données, par exemple en trouvant des tendances dans le comportement d'achat des clients afin de recommander des produits.

Dans l'apprentissage par renforcement, le système basé sur l'IA (un « agent intelligent ») apprend en interagissant avec son environnement. L'agent reçoit un retour positif (récompenses) ou négatif (pénalités) en fonction du résultat de ses actions, ce qui lui permet d'apprendre à partir de l'expérience

plutôt que d'un jeu de données. Les défis liés à l'apprentissage par renforcement comprennent la mise en place de l'environnement, la conception de la fonction de récompense et la sélection de la meilleure stratégie pour atteindre l'objectif souhaité. Les applications comprennent la robotique, les véhicules autonomes et les systèmes adaptatifs tels que les chatbots. Chaque approche de ML traite différents types de problèmes, le choix dépendant de la nature des données disponibles et de la tâche spécifique à accomplir.

3.1.2 Workflow de Machine Learning

Les étapes du workflow de ML, illustrées à la figure 1, sont les suivantes:

Compréhension des objectifs

L'objectif du modèle de ML est compris et approuvé par les parties prenantes afin de vérifier sa cohérence avec les priorités opérationnelles. Des critères d'acceptation (y compris des mesures de performance fonctionnelle ML – voir 3.3) sont définis pour le modèle développé.

Sélection du framework

Un framework de développement d'apprentissage automatique adapté (voir la section 1.1.7 pour plus de détails sur les fonctionnalités fournies) est sélectionné en fonction des objectifs, des critères d'acceptation (voir la section 2.2) et des priorités opérationnelles.

Sélection & conception de l'algorithme

Le choix d'un algorithme de ML repose sur divers facteurs, notamment les objectifs, les critères d'acceptation et les données disponibles (voir 3.2). L'algorithme peut être codé manuellement, mais il est souvent extrait d'une bibliothèque logicielle. Il est ensuite compilé, si nécessaire.

Préparation & test des données

La préparation des données (voir 3.2) comprend l'acquisition des données, leur prétraitement et l'ingénierie des caractéristiques. Une analyse exploratoire des données (EDA) peut être réalisée parallèlement à ces activités.

Les données utilisées par l'algorithme et le modèle sont déterminées en fonction des objectifs et sont utilisées dans toutes les activités de la section « Génération et test du modèle » illustrée à la figure 1.

Les données utilisées pour l'entraînement, l'évaluation, le réglage et le test du modèle doivent être représentatives des données qui seront utilisées par le modèle en production.

Des tests des données et de toutes les étapes automatisées de préparation des données de test sont effectués (voir chapitre 5).

Entraînement du modèle

L'algorithme de ML sélectionné utilise des données d'entraînement pour entraîner le modèle de ML.

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 passés à l'algorithme. Ces paramètres sont appelés hyperparamètres du modèle.

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

Evaluation du modèle

Le modèle est évalué à l'aide des mesures de performance fonctionnelle ML convenues (voir 3.3), à partir du jeu de données de validation, et les résultats sont utilisés pour améliorer le modèle lors de l'étape de « réglage du modèle ». En pratique, on crée et on entraîne généralement plusieurs modèles à l'aide de différents algorithmes (par exemple, les forêts aléatoires, les machines à vecteurs de support et les réseaux neuronaux) et de divers jeux de données d'entraînement, puis on choisit la meilleure combinaison en fonction des résultats de l'évaluation.

Ajustement du modèle

Les résultats de l'évaluation servent à ajuster les hyperparamètres du modèle et ceux de l'algorithme. Le modèle est ensuite réentraîné avec ces paramètres ajustés afin d'améliorer sa performance fonctionnelle.

Ces trois étapes – entraînement, évaluation et réglage – constituent la « génération de modèle », comme le montre la figure 1.

Test du modèle

Une fois qu'un modèle acceptable a été généré dans le cadre des activités de « génération de modèle », il est testé en utilisant un jeu de données de test indépendant afin de vérifier que les critères de performance fonctionnelle de ML convenus sont respectés. Les résultats des tests sont également comparés à ceux de l'évaluation. Si les performances du modèle avec les données de test indépendantes sont nettement inférieures à celles observées lors de l'évaluation, il peut s'avérer nécessaire de revenir aux activités de « génération de modèle », voire à l'activité de « préparation et test des données », afin d'entraîner un nouveau modèle.

Outre les tests de performance fonctionnelle de ML, des tests non fonctionnels, tels que ceux portant sur le temps nécessaire pour fournir une prédiction, peuvent également être effectués. En règle générale, les tests dans le cadre de cette activité sont réalisés par des data engineers ou des data scientists ; toutefois, des testeurs disposant de connaissances suffisantes du domaine et d'un accès aux ressources pertinentes peuvent également effectuer ces tests.

Déploiement du modèle

Une fois la phase de « génération et de test du modèle » terminée, le modèle optimisé est généralement réorganisé en vue de son déploiement, avec son pipeline de données. Cette opération s'effectue généralement via le framework de développement ML. Les plateformes cibles peuvent inclure des systèmes embarqués et le cloud, où le modèle est accessible via une API Web. Le modèle déployé et réorganisé est testé afin de vérifier qu'il répond toujours à ses critères d'acceptation.

Utilisation du modèle

Une fois déployé, le modèle est généralement intégré à un système d'exploitation plus vaste basé sur l'IA. Les modèles peuvent effectuer des prédictions par lots à intervalles réguliers ou fonctionner en temps réel sur demande.

Suivi et ajustement du modèle

Pendant l'utilisation du modèle, sa situation peut évoluer, et le modèle peut s'écarter des performances prévues (voir 6.1.7). Afin de s'assurer que tout écart est détecté et géré, le modèle opérationnel est régulièrement évalué par rapport à ses critères d'acceptation.

Il peut s'avérer nécessaire de créer un nouveau modèle en le réentraînant avec de nouvelles données, en le réentraînant avec de nouveaux hyperparamètres, ou les deux. Le nouveau modèle peut ensuite être comparé au modèle existant en utilisant des tests A/B (voir 6.1.9).

Le workflow ML illustré à la figure 1 est une séquence logique ; cependant, dans la pratique, ce workflow est appliqué de manière itérative, les étapes étant répétées.

Les étapes présentées à la figure 1 n'incluent pas l'intégration du modèle de ML aux composantes non ML du système global. En général, les modèles ML ne peuvent pas être déployés de manière isolée et doivent être intégrés à des composants non ML (par exemple, dans les applications de vision, un pipeline de données est utilisé pour nettoyer et modifier les données avant de les soumettre au modèle de ML). Lorsque le modèle fait partie d'un système de grande envergure, il doit être intégré à ce système avant son déploiement. Dans ce cas, des niveaux de test d'intégration, de système et d'acceptation peuvent être effectués.

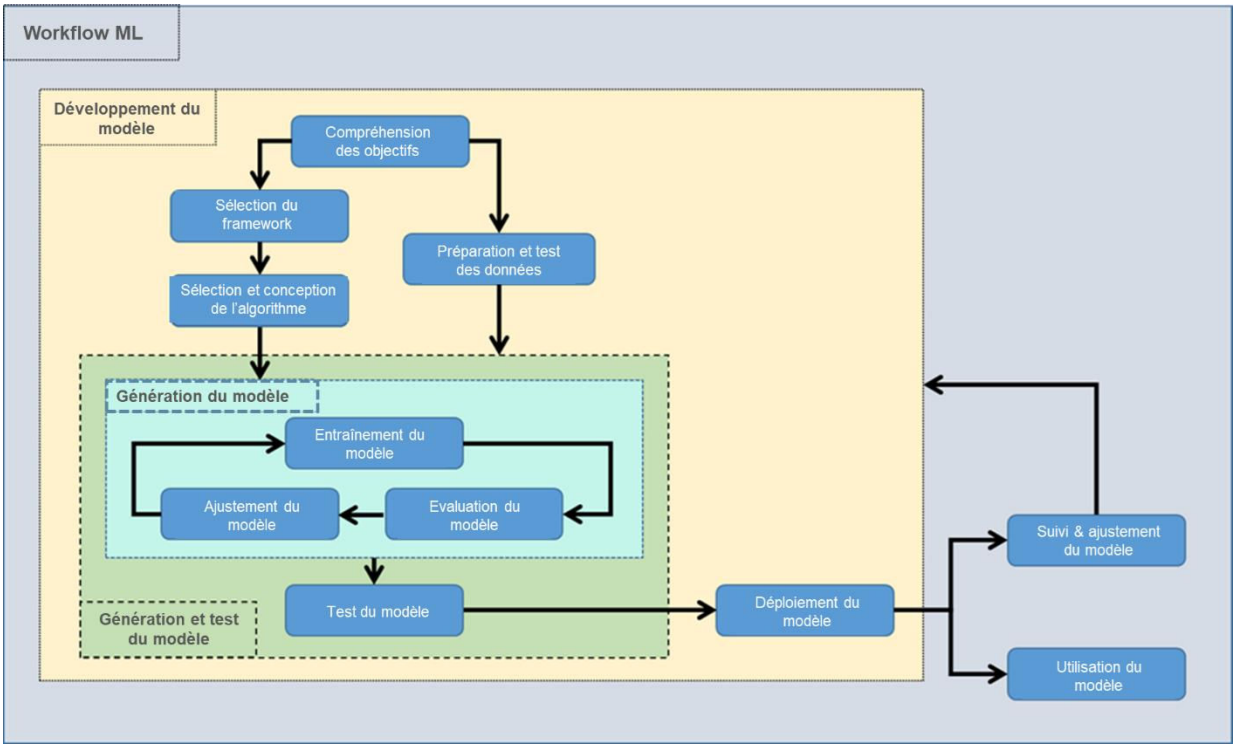

Figure 1: Workflow ML

3.1.3 Exercice pratique: Créer un modèle de Machine Learning

Sélectionnez, entraînez et testez un modèle de classification en utilisant l'apprentissage supervisé. Expliquez la différence entre l'évaluation/l'ajustement et le test en comparant la précision obtenue avec les jeux de données de validation et de test.

3.1.4 Modèles pré-entraînés, ajustement fin et Retrieval-Augmented Generation

L'entraînement d'un nouveau modèle d'IA à partir de zéro est à la fois coûteux et chronophage. Pour remédier à cela, une solution courante consiste à procéder à un « fine-tuning » (ajustement fin), qui consiste à prendre un réseau neuronal pré-entraîné et à l'adapter pour qu'il accomplisse une nouvelle tâche différente. L'un des principaux bénéfices est que cette méthode nécessite beaucoup moins de données d'entraînement (et d'efforts d'entraînement) que la construction d'un modèle à partir de zéro.

Le modèle pré-entraîné est affiné en effectuant un entraînement supplémentaire avec des données spécifiques à la nouvelle tâche. L'ajustement peut s'appliquer à l'ensemble du réseau neuronal, uniquement à des couches spécifiques (généralement proches de la sortie du réseau neuronal) ou à des couches supplémentaires. Après l'entraînement, les performances fonctionnelles de ML du modèle sont évaluées et, sur la base de ces résultats, un ajustement supplémentaire peut être effectué jusqu'à ce que le modèle réponde aux critères d'acceptation requis.

La réussite de l'ajustement fin dépend de la similitude entre les tâches d'origine et les nouvelles tâches. De légères différences peuvent conduire à un ajustement fin très efficace. Par exemple, adapter un classificateur d'images de races de chats pour identifier des races de chiens a de bonnes chances de fonctionner. En revanche, l'adapter pour reconnaître des accents linguistiques est moins efficace en raison de la différence plus importante. De même, l'ajustement fin d'un LLM pour l'analyse des valeurs limites définie par l'ISTQB nécessite une modification mineure du LLM et est facilement réalisable avec de bonnes données d'entraînement.

Une alternative à l'ajustement fin est la retrieval-augmented generation (RAG), qui consiste à fournir au LLM des sources de données spécifiques à la tâche requise. Ces sources de données sont transformées en un format consultable qui permet de les comparer au sujet de la requête.

Une fois les documents pertinents identifiés, ceux-ci sont intégrés à une invite améliorée, qui est passée au LLM. Comme des informations plus pertinentes sont désormais fournies au LLM, sa réponse correspondante est susceptible d'être plus précise. Avec la RAG, aucune modification n'est apportée au modèle pré-entraîné.

Un modèle pré-entraîné peut recourir à la méthode RAG, au réajustement, ou aux deux à la fois, pour améliorer ses performances. En général, les biais ou les vulnérabilités présents dans le modèle pré-entraîné se répercutent sur le nouveau modèle ; il est donc nécessaire de procéder à des tests pour s'assurer qu'il fonctionne de manière fiable et équitable dans le cadre de la nouvelle tâche.

3.2 Données pour Machine Learning

La préparation des données est reconnue comme l'une des activités les plus cruciales et les plus gourmandes en ressources du workflow d'apprentissage automatique. Si les données opérationnelles diffèrent sensiblement des données d'entraînement, les hypothèses relatives à la performance fonctionnelle de ML et à la sûreté de fonctionnement risquent de ne plus être valables. La préparation des données représente généralement une part nettement plus importante de l'effort global que d'autres étapes, telles que la sélection et la construction des modèles ML. Elle est intrinsèquement liée au pipeline de données, qui traite les données brutes et les transforme en un format utilisable tant pour l'entraînement que pour la prédiction par les modèles ML.

3.2.1 Activités de préparation des données

La préparation des données soutient la garantie de la qualité et de l'adéquation des données pour l'entraînement des modèles. Elle comprend plusieurs activités clés:

- Acquisition des données :
    - Identification des types de données pertinents (par exemple, numériques, catégorielles, images, texte).
    - Collecte de données provenant de diverses sources, telles que des bases de données, des API ou des capteurs en temps réel.
    - Étiquetage des données pour les tâches d'apprentissage supervisé, en vérifiant leur exactitude et leur cohérence.

  Les données acquises peuvent prendre diverses formes (par exemple, numériques, catégorielles, images, tableaux, texte, séries chronologiques, capteurs, géospatiales, vidéo et audio).
- Prétraitement des données :
    - Nettoyage des données, notamment :
        - suppression des défauts, des doublons et des valeurs aberrantes afin de vérifier l'exactitude et la cohérence des données ;
        - imputation des valeurs manquantes à l'aide de techniques telles que la moyenne, la médiane ou le mode afin de garantir la complétude des données ;
        - anonymisation ou suppression des informations personnelles afin de protéger la vie privée et de se conformer à la réglementation.
    - Transformation des formats de données, mise à l'échelle et normalisation pour assurer la cohérence.
    - Enrichissement des données pour augmenter la taille de l'échantillon, intégration d'exemples adverses pour améliorer la robustesse face aux attaques adverses, et génération de données synthétiques.
    - Échantillonnage de sous-ensembles pour réduire les temps d'entraînement et les coûts de calcul.
- Ingénierie des caractéristiques :
    - Sélection des caractéristiques pertinentes en fonction de leur contribution aux performances du modèle de ML.
    - Extraction d'un sous-ensemble de caractéristiques informatives et non redondantes à partir des caractéristiques existantes afin de réduire les temps d'entraînement et les coûts de calcul.

Parallèlement à ces activités de préparation des données, on procède généralement à une analyse exploratoire des données (EDA) afin d'en tirer des enseignements. Cela comprend:

- L'identification des tendances, des schémas et des anomalies dans les données ;
- la visualisation des données en utilisant des graphiques et des diagrammes pour une meilleure compréhension.

La préparation des données d'entraînement est généralement un processus itératif, souvent effectué manuellement, et les différentes étapes de préparation peuvent être réorganisées ou omises en fonction des exigences spécifiques du projet. Les données opérationnelles doivent correspondre aux caractéristiques des données d'entraînement (par exemple, les distributions des données et les plages des caractéristiques) afin que le modèle fonctionne comme prévu en production. Cependant, les étapes de préparation elles-mêmes peuvent être adaptées pour garantir l'efficience et l'évolutivité de la production.

3.2.2 Exercice pratique: Préparation des données en soutien à la création d'un modèle de Machine Learning

Pour un jeu de données donné, effectuez les étapes de préparation des données applicables, telles que décrites à la section 3.2.1, afin de produire un jeu de données qui servira à créer un modèle de ML à l'aide de l'apprentissage supervisé.

Cette activité constitue la première étape de la création d'un modèle de ML qui sera utilisé dans les exercices à venir.

Pour mener à bien cette activité, les candidats disposeront du matériel approprié (et spécifique au langage), notamment :

- des bibliothèques ;
- un framework de développement ML ;
- des outils.

3.2.3 Entraînement, Validation, et jeux de données de test

En toute logique, trois jeux de données équivalents (par exemple, sélectionnés au hasard à partir d'un seul jeu de données représentatif) sont nécessaires pour développer un modèle de ML :

- Un jeu de données d'entraînement qui sert à entraîner le modèle.
- Un jeu de données de validation qui sert à évaluer puis à affiner le modèle.
- Un jeu de données de test, également appelé jeu de données de validation, qui sert à tester le modèle affiné.

Si l'on dispose d'une grande quantité de données pertinentes, le volume de données utilisé dans le workflow ML pour l'entraînement, l'évaluation et les tests dépend généralement des facteurs suivants :

- La complexité attendue du modèle.
- L'algorithme utilisé pour entraîner le modèle.
- La disponibilité des ressources, telles que la mémoire vive (RAM), l'espace disque, la puissance de calcul, la bande passante réseau et le temps disponible.
- Le niveau de confiance souhaité dans le modèle obtenu.

Lorsque les données sont limitées, une division en trois parties (entraînement, validation et test) peut entraîner un manque de données pour un entraînement efficace du modèle, augmentant ainsi le risque

de sous-ajustement. Pour remédier à cela, une stratégie courante consiste à mettre de côté un petit ensemble de test final, si cela est possible. Les données restantes (l'ensemble combiné d'entraînement et de validation) sont ensuite utilisées pour des techniques telles que la validation croisée k-fold (où k est un nombre entier spécifié par l'utilisateur, généralement 5 ou 10).

Dans la validation croisée k-fold, ces données sont divisées en k « plis (ou folds) ». Pour chaque pli, le modèle est entraîné sur k-1 plis et validé sur le pli de test. Ce processus est répété k fois, chaque pli servant une fois d'ensemble de validation. Cela permet un réglage robuste des hyperparamètres et une estimation fiable de la performance fonctionnelle de ML. Les données sont généralement attribuées de manière aléatoire aux plis, souvent à l'aide d'un échantillonnage stratifié afin de rendre chaque pli représentatif, en particulier avec des données déséquilibrées ou des jeux de données de petite taille.

Les métriques de performance (par exemple, la précision, le score F1 – voir 3.3.1) issues de la validation de chaque pli sont ensuite moyennées afin de fournir une estimation plus fiable de la capacité de généralisation du modèle. Une fois les hyperparamètres optimaux identifiés par validation croisée, un modèle final est généralement entraîné sur l'ensemble des données d'entraînement et de validation (toutes les données à l'exception de l'ensemble de test de validation) à l'aide de ces hyperparamètres. Ce modèle final est ensuite évalué une seule fois sur l'ensemble de tests de validation pour une évaluation finale et impartiale des performances. Si un ensemble de tests de validation n'est pas réalisable en raison d'une rareté extrême des données, la performance moyenne de la validation croisée est biaisée de manière optimiste et ne peut servir d'estimation impartiale. D'autres méthodes de rééchantillonnage pour les données limitées incluent la validation croisée « leave-one-out » (un cas particulier de la validation k-fold où k est égal au nombre d'échantillons) et les techniques de bootstrap.

3.3 Métriques de performance fonctionnelle ML pour la classification

Dans les tâches de classification (voir 3.1.1), une matrice de confusion peut être utilisée pour évaluer les prédictions d'un modèle, en les classant en vrais positifs, vrais négatifs, faux positifs ou faux négatifs. Des métriques clés, telles que l'exactitude, la précision, le rappel et le score F1, en découlent. Ces métriques mesurent la qualité de la classification, mettant en évidence les forces et les faiblesses du modèle de ML. Cette section explore le calcul et l'interprétation de ces métriques afin d'évaluer la performance fonctionnelle de ML.

3.3.1 Calcul des métriques de performance fonctionnelle pour le Machine Learning

Dans un problème de classification, un modèle prédit rarement les résultats correctement à chaque fois, en partie en raison de la nature probabiliste des modèles ML et du bruit dans les données. Pour tout problème de ce type, on peut établir une matrice de confusion présentant 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

Il convient de noter que la matrice de confusion illustrée à la figure 2 peut être présentée différemment (par exemple, en inversant les valeurs « prédites » et « réelles »), mais elle fournira toujours des valeurs pour les quatre cas de figure 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, avec des valeurs comprises entre 0 et 100. Un score proche de 100 signifie que le modèle atteint à la fois une précision et un rappel élevés, ce qui indique que les erreurs de classification (faux positifs et faux négatifs) ont un impact minime. À l'inverse, un score F1 faible indique que le modèle peine à identifier correctement les cas positifs, soit en omettant des cas réels, soit en générant de nombreuses fausses alertes.

3.3.2 Exercice pratique: Evaluer un modèle de Machine Learning en utilisant une sélection de métriques de performance fonctionnelle ML

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

3.3.3 Exercice pratique: Montrer l'impact de différents modèles de Machine Learning et des combinaisons de données

Dans la continuité de l'exercice précédent, utilisez différentes combinaisons de modèles ML et de jeux de données afin d'observer leur incidence sur l'entraînement du modèle de ML, ainsi que sur le comportement final de celui-ci. Notez les durées d'entraînement et les mesures de performance fonctionnelle ML.

3.4 Réseaux neuronaux

La conception des réseaux neuronaux artificiels a été initialement orientée vers l'imitation du fonctionnement du cerveau humain, que l'on peut considérer comme un réseau de neurones biologiques interconnectés.

Le perceptron à une seule couche est l'un des tout premiers exemples d'implémentation d'un réseau neuronal artificiel, ne comportant qu'une seule couche. Il peut être utilisé pour l'apprentissage supervisé de classificateurs binaires dans le cadre de problèmes linéairement séparables, qui déterminent si une entrée appartient ou non à une classe spécifique. Par exemple, un perceptron peut distinguer les e-mails qui sont des spams de ceux qui ne le sont pas, en apprenant à séparer les caractéristiques des deux catégories à l'aide d'une ligne droite en entrée.

La plupart des réseaux neuronaux actuels sont considérés comme des réseaux neuronaux profonds, car ils comportent plusieurs couches. Les réseaux entièrement connectés peuvent être considérés comme des perceptrons multicouches (voir figure 3).

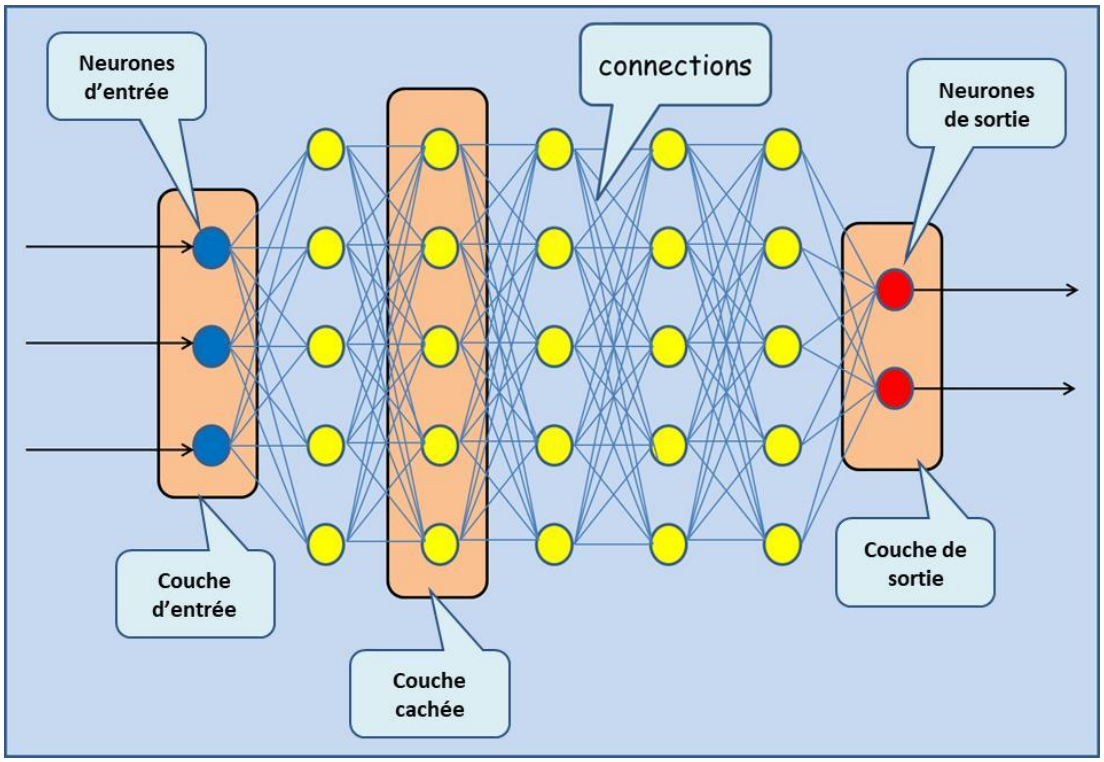

Figure 3: Structure d'un réseau neuronal profond

3.4.1 Structure et fonctionnement d'un réseau neuronal profond

Un réseau neuronal profond est généralement décrit comme étant composé de trois types principaux de couches. La couche d'entrée reçoit les données d'entrée, par exemple les valeurs de pixels provenant d'une caméra. La couche de sortie fournit les résultats au monde extérieur. Il peut s'agir, par exemple, d'une valeur indiquant la probabilité que l'image d'entrée représente un chat. Entre les couches d'entrée et de sortie se trouvent des couches cachées composées de neurones artificiels, également appelés nœuds. Dans de nombreuses architectures courantes, telles que les réseaux entièrement connectés, les neurones d'une couche sont connectés à chacun des neurones de la couche suivante, et le nombre de neurones peut varier d'une couche à l'autre.

Les neurones effectuent des calculs et transmettent des informations à travers le réseau, des neurones en entrée vers les neurones de sortie, transformant progressivement les données d'entrée en représentations de plus en plus abstraites jusqu'à atteindre la sortie.

Le calcul effectué par chaque neurone (après ceux en entrée) génère ce que l'on appelle la valeur d'activation. Cette valeur est calculée en effectuant d'abord une somme pondérée des valeurs d'activation de tous les neurones connectés de la couche précédente, chaque connexion ayant son propre poids indépendant, puis en y ajoutant le biais individuel du neurone. Cette somme est ensuite passée par une formule non linéaire appelée fonction d'activation. Notez que ce biais n'a aucun rapport avec le biais considéré au point 5.1.2. En utilisant différentes fonctions d'activation, les valeurs d'activation sont différentes.

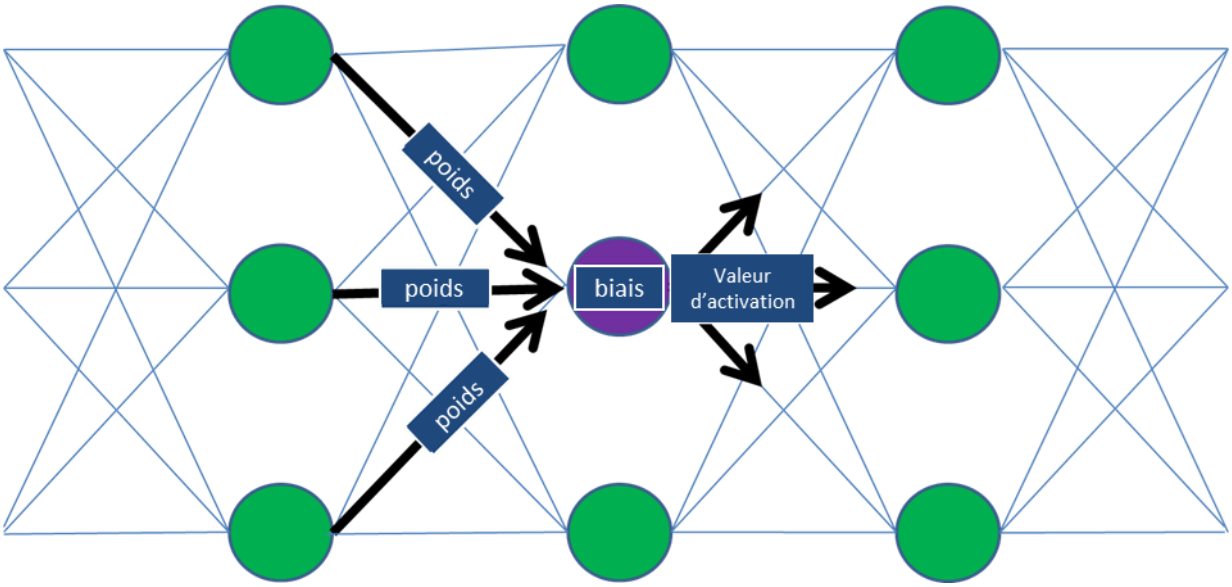

Figure 4: Calcul effectué par chaque neurone

Les poids reliant les neurones et la valeur de biais de chaque neurone sont généralement initialisés à de petites valeurs aléatoires (les biais étant parfois fixés à zéro) au début de l'entraînement. Les données d'entraînement sont passées à travers le réseau, chaque neurone appliquant la fonction d'activation, afin de générer une sortie finale. La sortie générée est ensuite comparée au résultat correct connu. L'erreur (ou perte) qui en résulte, et qui quantifie cette différence, est ensuite réinjectée dans le réseau afin d'ajuster les valeurs des poids et des biais, minimisant ainsi cette différence. À mesure que davantage de données d'entraînement sont transmises au réseau (chaque passage à travers le jeu de données d'entraînement est appelé une époque), les poids et les valeurs de biais sont progressivement ajustés au fur et à mesure que le réseau apprend. Après un certain temps, idéalement, la sortie produite est considérée comme suffisamment bonne pour mettre fin à l'entraînement.

3.4.2 Exercice pratique: Approche de l'implémentation d'un Perceptron

Les étudiants seront guidés à travers un exercice illustrant l'entraînement d'une fonction simple, telle qu'une fonction AND, par un perceptron.

Cet exercice doit montrer comment un perceptron apprend en modifiant ses poids et ses valeurs de biais au fil de plusieurs époques jusqu'à ce que l'erreur soit ramenée à zéro. Divers mécanismes (par exemple, un tableur ou un simulateur) peuvent être utilisés pour cette activité.

3.4.3 Mesures de couverture pour les réseaux neuronaux

Des métriques de couverture structurelle des réseaux neuronaux ont été développées pour évaluer dans quelle mesure les entrées de test sollicitent les mécanismes internes d'un modèle. Étant donné que les réseaux neuronaux ne suivent pas de chemins explicitement codés, mais que leur comportement est dicté par des poids, des valeurs de biais et des activations appris, il est nécessaire de disposer de mesures de couverture spécialisées pour évaluer à quel point les différentes parties du réseau ont été activées dans les conditions de test.

Les approches courantes comprennent [COV_REF]:

- Couverture des neurones : mesure la proportion de neurones du réseau dont la sortie dépasse un seuil spécifié lors du test.
- Couverture des neurones à k multisections (kMNC) : la plage de sortie possible de chaque neurone est divisée en k sections. La kMNC correspond à la proportion de ces sections activées lors du test.
- Couverture des limites des neurones (NBC) : mesure la proportion de neurones du réseau dont la sortie dépasse le maximum atteint lors de l'entraînement ou est inférieure au minimum atteint lors de l'entraînement pendant le test.

Ces métriques de couverture peuvent s'avérer utiles lors des tests d'IA, principalement en mettant en évidence les zones du modèle qui n'ont pas été testées et qui pourraient dissimuler des défauts ou des comportements non appris. Par exemple, si certains neurones ou certaines couches ne s'activent jamais pendant les tests, les performances du modèle aux limites de décision correspondantes pourraient être remises en question. Ces informations aident les testeurs à concevoir des entrées ou des conditions de test supplémentaires afin d'exploiter les parties du réseau qui n'ont pas été suffisamment explorées. À l'heure actuelle, les outils commerciaux prenant en charge ces mesures de couverture spécifiques sont limités. Cependant, la couverture structurelle à elle seule ne garantit pas qu'un réseau neuronal sera capable de bien généraliser ou de gérer les variations du monde réel. Les réseaux neuronaux peuvent apprendre des corrélations fallacieuses, conduisant à des activations correctes pour des raisons erronées. Par conséquent, les testeurs devraient également utiliser d'autres techniques de test, telles que les tests adverses et les tests métamorphiques, comme décrits dans les chapitres 5, 6 et 7.

4 Test des systèmes basés sur l'IA – 195 minutes

Mots-clés

Attaque, test exploratoire, test basé sur les risques, oracle de test

Mots-clés spécifiques à l'IA

Système adaptatif basé sur l'IA, système basé sur l'IA, IA générative, grand modèle de langage, système figé basé sur l'IA

Objectifs d'apprentissage du chapitre 4:

4.1 Introduction au test des systèmes basés sur l'IA

AI-4.1.1 (K2) Comparer la testabilité des systèmes basés sur l'IA de type « figé » et « adaptatif »

AI-4.1.2 (K2) Expliquer pourquoi une approche statistique est souvent nécessaire lors du test de systèmes basés sur l'IA

AI-4.1.3 (K2) Expliquer les défis et les solutions liés aux oracles de test pour les systèmes basés sur l'IA

4.2 Test de l'IA générative et des grands modèles de langage (LLMs)

AI-4.2.1 (K2) Expliquer comment tester l'IA générative

AI-4.2.2 (K3) Implémenter des tests de type « red teaming » pour les systèmes d'IA générative

HO-4.2.3 (H2) Appliquer des tests exploratoires à un grand modèle de langage (LLM) en effectuant une analyse des valeurs limites

4.3 Niveaux de test et systèmes de Machine Learning

AI-4.3.1 (K2) Résumer les niveaux de test utilisés pour développer des systèmes de Machine Learning

AI-4.3.2 (K2) Expliquer comment le test basé sur le risque est appliqué aux systèmes de Machine Learning

4.1 Introduction au test des systèmes basés sur l'IA

Le test des systèmes basés sur l'IA présente des défis particuliers par rapport au test des logiciels classiques. Tout d'abord, les systèmes basés sur l'IA peuvent être classés en deux grandes catégories : les systèmes « figés » et les systèmes « adaptatifs ». Les systèmes figés basés sur l'IA, dont le comportement est fixe après le déploiement, sont plus faciles à tester en raison de leur nature essentiellement déterministe, tandis que les systèmes adaptatifs, qui évoluent et apprennent, introduisent une certaine complexité, car leur comportement peut changer de manière imprévisible. De plus, la nature probabiliste de nombreux modèles d'IA nécessite souvent de tester avec des méthodes statistiques, car les méthodes déterministes peuvent s'avérer insuffisantes pour évaluer des résultats influencés par les données et les probabilités. Cela nécessite une évaluation des distributions de performances et des niveaux de confiance dans divers scénarios.

L'un des principaux défis en matière de tests, connu sous le nom de « problème de l'oracle de test », se pose lorsqu'il s'agit de déterminer si la sortie produite par le système sous test est correcte pour une entrée donnée. Dans les logiciels traditionnels, des spécifications bien définies permettent de préciser assez facilement les résultats attendus et d'en vérifier l'exactitude. Cependant, avec les systèmes basés sur l'IA, en particulier ceux qui traitent des tâches complexes ou subjectives, il peut être difficile, voire impossible, de définir clairement les résultats attendus. Cette incertitude est aggravée pour les tâches qui dépassent les capacités humaines, impliquent un certain flou ou ne disposent pas d'une « vérité de référence » objective, ce qui rend difficile l'implémentation d'oracles de test automatisés et nécessite parfois le jugement d'un expert ou un raisonnement statistique.

Des exigences système vagues ou incomplètes exacerbent encore davantage le problème des oracles de test dans les tests d'IA. Les solutions peuvent inclure le recours à des évaluations statistiques, la consultation d'experts du domaine ou la définition d'une « vérité de référence » par rapport à laquelle évaluer les résultats. Ces approches visent à établir des attentes fiables et à évaluer efficacement les systèmes basés sur l'IA.

4.1.1 Systèmes figés ou adaptatifs basés sur l'IA

De nombreux systèmes basés sur l'IA utilisés aujourd'hui sont des systèmes « figés » qui ne modifient pas leur comportement une fois déployés. Un exemple de système « figé » est un modèle de ML déployé basé sur un réseau neuronal profond (DNN), dans lequel les poids et les biais du DNN sont fixés après le développement et ne peuvent être modifiés que si le DNN est réentraîné. La technologie des voitures autonomes liée à la sûreté s'appuie souvent sur des systèmes figés basés sur l'IA pour des tâches spécifiques, telles que la détection de voie ou la reconnaissance des panneaux de signalisation. En revanche, un système adaptatif basé sur l'IA, tel qu'un système d'apprentissage par renforcement, peut adapter son comportement une fois déployé. Les changements peuvent être basés sur une fonction de récompense ou viser à s'adapter à un nouvel environnement opérationnel, mais les détails spécifiques de ces changements ne peuvent être prédits à l'avance. Par exemple, une plateforme de commerce électronique pourrait utiliser une IA adaptative pour recommander des produits aux utilisateurs en fonction de leur comportement passé et de l'évolution de leurs préférences.

De nombreux systèmes d'IA générative, tels que les chatbots basés sur des modèles de langage (LLM), sont déployés sous forme de modèles figés au moment de l'exécution, mais sont mis à jour périodiquement, ce qui les place à mi-chemin entre les systèmes entièrement figés et les systèmes entièrement adaptatifs. Dans la pratique, les systèmes basés sur l'IA couvrent un spectre allant des systèmes entièrement déterministes et figés, qui produisent le même résultat pour une entrée donnée, aux systèmes délibérément non déterministes et auto-apprenants, qui adaptent et font évoluer leur comportement.

Les systèmes basés sur une IA « figée » sont bien plus faciles à tester que ceux basés sur une IA adaptative, car ils sont en grande partie déterministes et les résultats attendus ne varient donc pas. Un système basé sur l'IA figée qui a été mis à jour est généralement considéré comme un nouveau système, et une nouvelle série de tests est nécessaire.

Notez toutefois que, bien que les systèmes basés sur l'IA figée soient généralement considérés comme déterministes, dans la pratique, les grands réseaux neuronaux peuvent présenter un comportement non déterministe en raison de facteurs tels que les limites de précision en virgule flottante et les variations dans l'exécution matérielle, en particulier lorsqu'on utilise le calcul parallèle ou les GPU. Un système adaptatif basé sur l'IA peut être rigoureusement testé avant son déploiement (par exemple, en simulant des changements dans les conditions environnementales, en testant le mécanisme d'apprentissage lui-même ou en testant sa capacité à s'adapter de manière appropriée dans des scénarios contrôlés). Cependant, ces tests sont plus complexes que pour un système figé basé sur l'IA, car un système adaptatif peut modifier son comportement pendant les tests ou à la suite de ceux-ci.

Comme les nouveaux comportements d'un système adaptatif basé sur l'IA ne peuvent pas toujours être prédits, ces comportements imprévisibles ne peuvent être testés à l'avance, et il n'est pas possible de préparer des cas de test pour les couvrir. Une suite de tests automatisés, axée sur les fonctionnalités principales du système, peut être mise en place et exécutée chaque fois que le système subit des modifications importantes, afin de vérifier que les adaptations sont sûres.

Des tests peuvent également être utilisés pour vérifier que les performances du système ne se sont pas dégradées au-delà d'un certain seuil. Ces tests peuvent être effectués en réponse à des modifications importantes du système ou dans le cadre du suivi continu.

4.1.2 Justification d'une approche statistique pour tester les systèmes basés sur l'IA

Les tests des systèmes basés sur l'IA posent des défis particuliers en raison de leur nature probabiliste et pilotée par les données. Voici quelques-unes des raisons pour lesquelles une approche statistique est nécessaire pour tester ces systèmes :

- Non-déterminisme - Les systèmes basés sur l'IA sont fondamentalement probabilistes et présentent donc souvent un comportement non déterministe, ce qui signifie que des entrées identiques ne produisent pas toujours la même sortie. Cela peut être dû à des éléments stochastiques présents dans leur architecture ou à l'implémentation de cartographies probabilistes apprises à partir des données d'entraînement. Par conséquent, un test isolé, tel qu'un cas où un modèle classe à tort un chat comme un chien, ne peut refléter avec précision l'exactitude fonctionnelle globale de l'IA du modèle. Pour confirmer que les résultats des tests surmontent de manière fiable cette incertitude, la suite de tests doit être suffisamment vaste pour fournir des résultats statistiquement significatifs.
- Évaluation des performances distributionnelles - Les modèles d'IA sont entraînés sur des distributions de données spécifiques qui ne correspondent pas exactement à leur environnement opérationnel. Pour évaluer les performances d'un modèle dans des conditions réelles, un échantillon statistiquement significatif de scénarios issus des distributions de données opérationnelles pertinentes doit être testé. Cela permet de confirmer que les tests reflètent la variabilité opérationnelle des données et les comportements subséquents du modèle.
- Gestion de l'incertitude et des biais - Les systèmes basés sur l'IA sont sensibles aux biais des données et peuvent produire des prédictions sûres mais incorrectes. Les tests statistiques permettent aux praticiens de quantifier et d'analyser la précision, l'équité et la robustesse des

modèles d'IA à l'aide de métriques de performance telles que les intervalles de confiance, les tests d'hypothèse et l'analyse des erreurs.

- Contexte réglementaire et de sûreté - Dans les secteurs réglementés (par exemple, la santé, les transports), il est souvent nécessaire de démontrer avec un haut degré de confiance qu'un système basé sur l'IA respecte les seuils de sûreté ou d'équité. Les méthodes statistiques étayent les affirmations concernant la fiabilité d'un modèle dans un large éventail de scénarios, plutôt que pour des exemples spécifiques.

Une approche statistique pour tester les MLS probabilistes est présentée au paragraphe 6.1.3.

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

Tester des systèmes basés sur l'IA peut s'avérer difficile, notamment lorsqu'il s'agit de déterminer les résultats attendus (le problème de l'oracle de test). Ces difficultés découlent de divers facteurs inhérents à l'IA:

- Nature probabiliste et non déterministe : les résultats de l'IA peuvent varier même avec des entrées identiques. Alors que de nombreux systèmes (par exemple, dans l'apprentissage supervisé) ont une seule valeur cible correcte, les résultats du modèle sont probabilistes et la définition d'un oracle strict de type « réussi/échec » nécessite souvent de fixer des seuils ou des plages de tolérance.
- Développement exploratoire et spécifications incomplètes : le développement de l'IA est souvent exploratoire. Les exigences du système peuvent évoluer, être incomplètes ou tout simplement faire défaut, ce qui empêche de disposer des spécifications détaillées nécessaires pour générer des résultats attendus précis.
- Complexité des tâches : les systèmes basés sur l'IA s'attaquent souvent à des tâches trop complexes pour une vérification humaine directe, rendant les contrôles manuels des résultats attendus peu pratiques.
- Subjectivité du comportement : la justesse du comportement d'un système basé sur l'IA peut être subjective. Par exemple, les attentes des utilisateurs vis-à-vis des assistants virtuels peuvent varier considérablement, ce qui complique l'établissement de résultats attendus faisant l'unanimité.
- Systèmes d'auto-apprentissage : ces systèmes basés sur l'IA mettent continuellement à jour leurs modèles internes en fonction des nouvelles données rencontrées après leur déploiement. Cela entraîne une évolution du comportement « correct » du système au fil du temps, de sorte que les réponses du système restent efficaces et appropriées même si la définition de ce qui est « correct » évolue. En conséquence, un ensemble initial de résultats attendus peut rapidement devenir invalide.

Le problème de l'oracle de test dans les systèmes basés sur l'IA peut être résolu de différentes manières:

- Définition des limites de sortie : les testeurs peuvent s'appuyer sur des plages acceptables convenues, des distributions, des limites spécifiées et des tolérances, comme le fait qu'une voiture autonome s'arrête dans un rayon maximal.
- Définition des limites environnementales : les testeurs doivent spécifier les valeurs des conditions de l'environnement de test (par exemple, les niveaux d'éclairage, la température, la latence du réseau) afin de garantir que les résultats soient prévisibles et reproductibles.

- Consultation d'experts : les experts du domaine peuvent aider à définir les résultats attendus, bien que leurs opinions puissent diverger ou être sujettes à erreur.
- Tests spécialisés : les tests A/B, les tests dos-à-dos et les tests métamorphiques, entre autres, permettent d'évaluer les systèmes basés sur l'IA en comparant les comportements ou en vérifiant les propriétés, souvent sans nécessiter de résultats attendus explicites pour chaque cas.
- Oracle proxy : utilisez des systèmes ou des modèles secondaires (y compris d'autres systèmes d'IA) pour évaluer ou valider les résultats lorsque les résultats attendus directs ne sont pas disponibles, par exemple en entraînant un modèle proxy sur des données étiquetées afin de prédire les résultats pour des tests non étiquetés.

4.2 Test de l'IA générative et des grands modèles de langage (LLMs)

Cette section présente des méthodes pratiques pour valider les systèmes d'IA générative, notamment l'évaluation en boîte noire, les exercices de « red teaming » et des techniques pratiques. Elle met en évidence les défis liés à la diversité des entrées, aux paramètres réglables et aux fenêtres contextuelles, et aborde les mesures de qualité tant fonctionnelles que non fonctionnelles à l'aide de benchmarks et d'exercices ciblés. Pour plus de détails sur l'utilisation des systèmes d'IA générative à des fins de test, consultez le syllabus CT-GenAI [CT-GenAI].

4.2.1 Test de l'IA générative

Tester l'IA générative implique d'évaluer l'exactitude, la cohérence et la créativité de ses résultats, y compris l'originalité et la nouveauté des résultats générés, ainsi que de vérifier que les exigences spécifiées, tant fonctionnelles que non fonctionnelles, sont respectées. Étant donné que les systèmes d'IA générative peuvent produire du texte, des images, des vidéos et du son, les stratégies de test doivent être adaptées pour évaluer les caractéristiques de qualité de ces contenus.

Une approche courante est le test boîte noire, dans lequel les testeurs introduisent diverses données d'entrée (prompts, images, données partielles) dans le système et évaluent les résultats des tests (voir « red teaming », 4.2.2). Des facteurs tels que la clarté, l'originalité et le respect des règles spécifiques au domaine sont pris en compte. Cette approche, qu'elle soit manuelle ou automatisée, est particulièrement utile pour les applications destinées aux utilisateurs finaux, telles que les chatbots ou les outils de conception, où la satisfaction des utilisateurs dépend de l'utilité et de la plausibilité du contenu généré.

L'un des principaux défis liés au test d'un modèle d'IA générative réside dans le « problème de l'explosion des entrées », les données d'entrée pouvant être extrêmement variées et difficiles à contrôler. Par exemple, il existe des prompts système facultatifs et un prompt utilisateur, qui peuvent contenir d'énormes quantités de données disparates, et l'IA générative est également accessible via une API. Il existe également de nombreux paramètres à prendre en compte, tels que la température et le nombre maximal de tokens, qui auront également une incidence sur la sortie. De plus, la fenêtre de contexte, qui conserve les parties précédentes d'une conversation, affecte également la sortie générée.

Dans de nombreux cas, l'évaluation de la sortie peut nécessiter un examen manuel ; cependant, la détermination de la réussite ou de l'échec d'un test dépend de critères d'évaluation qualitatifs définis dans les exigences. Il est également souvent possible d'utiliser un deuxième système GenAI pour déterminer automatiquement le résultat du test. Par exemple, l'exactitude d'une image générée par un système GenAI peut être vérifiée par un système de reconnaissance d'images. De telles approches doivent être utilisées avec prudence, car elles peuvent reproduire des biais ou des erreurs similaires.

Les tests non fonctionnels peuvent s'avérer tout aussi indispensables que les tests fonctionnels pour les systèmes d'IA générative. Cela implique notamment d'évaluer l'utilisation des ressources tant lors de l'inférence que de l'entraînement, afin de vérifier l'efficacité opérationnelle et la rentabilité.

Les mesures comprennent l'utilisation du CPU et du GPU, la consommation de mémoire, la bande passante réseau et les temps de réponse.

Les benchmarks fournissent des frameworks d'évaluation standardisés pour évaluer les capacités de la GenAI. Ces jeux de données sélectionnés et les tâches associées permettent une comparaison cohérente entre différents modèles, en mesurant divers aspects, de la compréhension du langage aux capacités de raisonnement et de codage. En évaluant la GenAI par rapport à ces benchmarks, les domaines à améliorer peuvent être identifiés de manière systématique et reproductible.

4.2.2 Red Teaming

Le Red Teaming (RT) est une forme systématique d'attaque de faute, souvent de type « boîte noire », qui consiste à tester un système basé sur l'IA afin d'identifier ses capacités nuisibles, en particulier au niveau de ses résultats. S'inspirant de pratiques historiques telles que les jeux de guerre militaires et les « tiger teams » de la NASA, l'AI RT consiste à « attaquer » un système basé sur l'IA, dans le but de le pousser délibérément à produire des résultats nuisibles ou indésirables, tels que des violations de la vie privée, l'expression d'opinions racistes ou la fourniture de conseils sur la manière de mener des attaques chimiques, biologiques, radiologiques ou nucléaires (CBRN). Une fois ces capacités identifiées, elles sont utilisées pour mettre à jour et renforcer le système, le rendant ainsi moins susceptible de produire de tels résultats à l'avenir. Si le RT peut s'appliquer à n'importe quel système basé sur l'IA, il revêt une importance particulière pour l'IA générative (GenAI) en raison de la vaste gamme d'entrées et de sorties possibles, qui créent un « espace d'attaque » étendu. Le RT couvre généralement l'ensemble du système basé sur l'IA de bout en bout, mais peut être appliqué uniquement au modèle d'IA.

De nombreuses organisations concentrent leurs efforts en matière de RT sur les vulnérabilités de sécurité et de sûreté ; cependant, cette approche peut également servir à détecter des capacités préjudiciables dans d'autres domaines, tels que la fiabilité, la confidentialité, l'équité, les biais et la diffusion de fausses informations. La RT s'impose de plus en plus comme une exigence réglementaire pour certains types de systèmes basés sur l'IA, comme en témoignent des frameworks réglementaires tels que la loi européenne sur l'IA [EU AI Act]. En tant qu'approche d'évaluation adaptative et dynamique dans laquelle les prompts peuvent être immédiatement mis à jour en fonction des résultats, le RT complète les approches statiques telles que l'analyse comparative en testant les systèmes dans des conditions extrêmes et imprévues.

Lorsque le RT est utilisée pour des évaluations de sécurité, le système est testé afin d'identifier les vulnérabilités face à des attaques externes, y compris des facteurs de sécurité non spécifiques à l'IA et spécifiques à l'IA, tels que les attaques par injection indirecte de prompts et la dissimulation de contenu malveillant dans les documents utilisés par le RAG. Alors que le RT de sécurité tend à se concentrer sur les entrées malveillantes, pour les évaluations de sécurité et autres, l'objectif est souvent d'identifier comment le système pourrait générer des résultats nuisibles dans le cadre d'une utilisation normale, comme la production de conseils médicaux dangereux, sans aucune intention hostile de la part de l'utilisateur.

Le Red Teaming est particulièrement efficace lorsqu'il est mené avant le déploiement d'un système, mais après la réalisation des premiers audits internes de la qualité. L'activité principale consiste souvent en des interactions guidées, au cours desquelles les membres de cette « équipe rouge » engagent des dialogues en plusieurs échanges (par exemple, 15 à 20 échanges) afin d'élucider des défauts ou des comportements contraires aux politiques. Cette approche implique généralement :

1) Constituer une équipe de testeurs diversifiée afin de couvrir un large éventail de points de vue et de vecteurs d'attaque.
2) Fournir un accès au système basé sur l'IA dans un environnement de test sécurisé.
3) Inciter le système à identifier les vulnérabilités, soit par une exploration libre, soit à l'aide de checklists.
4) Analyser les défaillances identifiées afin de comprendre les menaces.
5) Créer des jeux de données à partir de ces menaces pour faciliter la mise en place de mesures d'atténuation et l'amélioration du système.

Pour assurer une couverture exhaustive par le RT, les organisations ont recours à plusieurs stratégies qui vont au-delà des petites équipes d'experts. Il s'agit notamment d'approches manuelles, telles que la génération de prompts par crowdsourcing, et d'approches automatisées, par exemple lorsqu'un LLM est utilisé pour générer de nombreux prompts d'attaque, dont les résultats sont ensuite vérifiés par un autre LLM. Les approches hybrides combinent la créativité des testeurs humains et l'évolutivité de l'automatisation.

Le RT, qui est proactif et axé sur le pré-déploiement, complète le Blue Teaming, qui implique un suivi défensif continu en temps réel et le filtrage des entrées d'un système opérationnel basé sur l'IA afin de le protéger contre les attaques. Les informations issues du RT peuvent être utilisées pour améliorer le suivi et les filtres utilisés dans le Blue Teaming.

4.2.3 Exercice pratique: Test exploratoire d'un LLM

Les étudiants effectueront des tests exploratoires sur un modèle LLM. Ils recevront une fiche de session de tests exploratoires visant à évaluer la capacité du modèle LLM à générer des cas de test à l'aide d'une analyse des valeurs limites à deux et trois valeurs. Dans le cadre du débriefing de la session, ils vérifieront l'exactitude et la complétude des résultats.

4.3 Niveaux de test et systèmes de Machine Learning

Les systèmes de ML (ou MLS) nécessitent des niveaux de test spécialisés pour faire face aux risques spécifiques liés au ML. Il s'agit du test des données d'entrée et du test du modèle de ML. Par ailleurs, les niveaux de test classiques restent d'actualité, notamment les tests unitaires, les tests d'intégration, les tests système et les tests d'acceptation.

4.3.1 Niveaux de test pour systèmes de Machine Learning

Les composants non liés à l'IA d'un système de ML (MLS) peuvent être testés à l'aide de niveaux de test classiques. En outre, les modèles ML et les MLS nécessitent des tests supplémentaires afin de traiter les risques spécifiques associés à la ML. Deux niveaux de test spécialisés sont utilisés pour traiter ces risques spécifiques au ML:

- Test des données d'entrée (voir chapitre 5) : ils portent sur les données d'entraînement utilisées pour former un modèle de ML et sur les données de production utilisées par un système d'apprentissage automatique pour générer une prédiction dans l'environnement d'exploitation.

- Test de modèle de ML (voir chapitre 6) : ils portent sur les tests des modèles ML, qui constituent le résultat final du flux de travail d'apprentissage automatique (voir 3.1.2).

Ces deux niveaux de test spécifiques au ML ne permettent pas de couvrir tous les risques associés à cette technologie. En fonction des risques identifiés, les niveaux de test suivants sont généralement également requis:

- Test de composants : s'appliquent à tous les composants non liés à l'IA, tels que l'interface utilisateur, le pipeline de données et les composants de communication.
- Test d'intégration des composants : comprennent les tests visant à vérifier que les entrées provenant du pipeline de données sont reçues comme prévu par le modèle et que toutes les prédictions générées par le modèle sont échangées avec les composants système concernés (par exemple, l'interface utilisateur) et utilisées correctement. Lorsque l'IA est fournie en tant que service (voir 1.1.6), les tests de l'API du service fourni sont effectués dans le cadre du test d'intégration des composants.
- Test système : comprennent des tests de confirmation visant à vérifier que la performance fonctionnelle de ML issue du test du modèle de ML initial n'est pas affectée négativement lorsque le modèle est intégré dans un système complet. Ces tests sont particulièrement importants lorsque le modèle de ML a été délibérément modifié (par exemple, en compressant un DNN pour réduire sa taille). Les tests non fonctionnels sont également couverts, par exemple les tests d'efficacité de performance concernant le temps nécessaire pour fournir une prédiction à partir d'un système complet basé sur l'IA.
- Test d'intégration des systèmes : se concentrent sur la vérification des interfaces et des échanges de données entre le système basé sur l'IA et les systèmes ou services externes, en utilisant un environnement représentatif des conditions opérationnelles.
- Test d'acceptation : lorsque l'IA est utilisée en tant que service, des tests d'acceptation peuvent être nécessaires pour déterminer l'adéquation du service au système prévu et vérifier, par exemple, si les critères de performance fonctionnelle de ML ont été atteints.

4.3.2 Test basé sur les risques pour les systèmes de Machine Learning

Le test basé sur les risques doit être appliqué à tous les systèmes, qu'ils comportent ou non des composants d'IA. La plupart des frameworks réglementaires, qu'ils soient déjà publiés ou en cours d'élaboration, exigent une approche basée sur les risques pour le développement et le management des systèmes basés sur l'IA. Les systèmes basés sur l'IA présentant des risques spécifiques, leur test diffère de celui des systèmes non basés sur l'IA. Il n'existe pas de méthode standard pour classer les risques liés au ML ; toutefois, les risques spécifiques au ML sont associés à la fois au développement du système de ML (risques projet) et au système de ML lui-même (risques produit). Une façon de classer ces risques consiste à utiliser le workflow du ML et à le diviser en trois domaines principaux:

- Développement - concerne l'algorithme ML, le développement du modèle et le framework de développement ML. Parmi les risques projet, on peut citer le choix d'un algorithme sous-optimal, une mauvaise sélection de la méthode d'évaluation et des vulnérabilités de sécurité du framework. Voir le chapitre 7 pour plus de détails.

- Données d'entrée - concerne la fourniture de données d'entraînement pour soutenir le ML et la mise à disposition des données de production utilisées par le modèle dans son environnement opérationnel. Parmi les exemples de risques produits, on peut citer des données d'entraînement biaisées, des défauts dans le pipeline de données et des données d'entraînement non représentatives. Voir le chapitre 5 pour plus de détails.
- Modèle - concerne le modèle de ML généré. Parmi les exemples de risques produits, on peut citer la défaillance à atteindre les exigences en matière de performance fonctionnelle de ML, un surajustement et la vulnérabilité aux exemples adverses. Voir le chapitre 6 pour plus de détails.

Une autre façon de classer les risques liés à l'IA consiste à se référer aux caractéristiques de qualité définies dans la norme ISO/IEC 25059.

Il existe plusieurs types de tests permettant de traiter ces risques et spécialement conçus pour l'évaluation des systèmes à plusieurs niveaux de sécurité (MLS). Citons, par exemple, les tests des pipelines de données, les tests adverses et l'évaluation de l'adéquation des algorithmes et des modèles. Ce syllabus aborde plusieurs de ces aspects dans les chapitres 5, 6 et 7.

5 Test des données d'entrée pour les systèmes de Machine Learning – 180 minutes

Mots-clés
Tests du pipeline de données, tests de représentativité des données, tests des contraintes des jeux de données, test des données d'entrée, tests de l'exactitude des étiquettes, revue, tests de détection des biais

Mots-clés spécifiques à l'IA

Analyse d'impact discriminatoire, annotations multiples

Objectifs d'apprentissage du chapitre 5:

5.1 Test des données d'entrée pour les systèmes de machine learning

AI-5.1.1 (K2) Donner des exemples d'approches de test utilisées pour l'atténuation des risques liés aux données d'entrée d'un système de machine learning

AI-5.1.2 (K2) Expliquer comment tester la présence de biais

AI-5.1.3 (K2) Résumer les différentes formes de tests du pipeline de données

AI-5.1.4 (K2) Expliquer comment tester la représentativité des données

AI-5.1.5 (K3) Appliquer des tests de contraintes sur les jeux de données

AI-5.1.6 (K2) Expliquer les tests de vérification de l'exactitude des étiquettes

HO-5.1.7 (H2) Effectuer des tests des données d'entrée pour les jeux de données ML

5.1 Test des données d'entrée pour les systèmes de Machine Learning

L'objectif du test des données d'entrée est de vérifier que les données utilisées par le MLS pour l'entraînement, les tests et la prédiction sont d'une qualité suffisante (voir 3.2). Cela comprend des revues, des techniques statistiques (par exemple, la détection de biais dans les données), l'analyse exploratoire des données d'entraînement, ainsi que des tests statiques et dynamiques du pipeline de données.

5.1.1 Risques liés aux données d'entrée et mesures d'atténuation des risques

Le tableau suivant présente des exemples de risques liés aux données d'entrée ainsi que les tests correspondants qui pourraient être utilisés pour l'atténuation des risques:

Risques potentiels	**Mesures possibles d'atténuation des risques**
Des défauts dans les données d'entraînement entraînant un biais Des problèmes au sein de l'algorithme, du modèle ou du framework de développement du ML qui introduisent une discrimination systémique	Tester la présence de biais – voir 5.1.2
Données d'entraînement provenant de sources non fiables Données mal gérées	Tester la provenance des données
Données d'entraînement corrompues (NDT data poisoning)	Tests A/B – voir 6.1.9 Tests de traçabilité des données EDA – voir 3.2.1 Attaques dans le cadre d'exercices de simulation d'attaques (red teaming) – voir 4.2.2
Jeu de données présentant des incohérences internes Données hors limites Types de données incorrects	Tester les contraintes sur les jeux de données – voir 5.1.5
Sélection de caractéristiques sous-optimale	Test des caractéristiques
Jeu de données déséquilibré en raison d'une couverture insuffisante de toutes les classes cibles	Test de la représentativité des données – voir 5.1.4

Jeu de données biaisé en raison de l'augmentation des données Données manquantes Données d'entraînement centrées sur un sous-ensemble de tous les cas d'utilisation L'ensemble complet des valeurs n'est pas couvert par le jeu de données	
Directives d'étiquetage inadéquates Données ambiguës Annotations insuffisantes entraînant un étiquetage inexact ou incohérent	Test de l'exactitude des étiquettes – voir 5.1.6
Une conception ou une intégration défaillantes entraînant des défaillances du pipeline de données Des défauts de qualité des données compromettant les résultats du pipeline Une baisse des performances lors de l'exploitation du pipeline de données Des failles de sécurité ou des modifications incontrôlées affectant le pipeline de données	Test du pipeline de données – voir 5.1.3

5.1.2 Test de détection des biais

Dans un système de MLS, on entend par « biais » des différences de traitement non aléatoires et inéquitables fondées sur des caractéristiques sensibles telles que le sexe, l'âge ou l'origine ethnique, ce qui est souvent illégal et rend le système discriminatoire. Pour tester les biais dans un système de MLS, il faut en comprendre les sources potentielles, qui comprennent principalement:

- Des défauts dans les données d'entraînement, tels que le manque de représentativité, des biais historiques ou un empoisonnement délibéré (biais des données).
- Des défauts au sein de l'algorithme, du modèle ou du framework de développement qui introduisent une discrimination systémique, comme un algorithme utilisant un seuil de décision pour les cotes de crédit dans un système d'approbation de prêts (biais algorithmique).

Les approches de test permettant de détecter les biais comprennent notamment:

- Revue de l'ensemble du workflow de ML, et en particulier de la préparation des données, afin d'identifier les risques de biais.
- Revue de la documentation relative aux jeux de données afin d'identifier et d'atténuer les sources potentielles d'injustice en analysant la manière dont les données ont été collectées et annotées, ainsi que les populations représentées.
- Analyse statique des programmes de préparation des données et du code d'implémentation du modèle de ML afin d'identifier les anti-modèles ou les erreurs de traitement des attributs sensibles.
- Analyse exploratoire des données d'entraînement (EDA) à l'aide de méthodes de visualisation et de regroupement afin de mettre en évidence les déséquilibres dans les données, les distributions asymétriques ou les regroupements anormaux entre différents attributs sensibles.
- Tests dynamiques visant à détecter les biais dans un modèle de ML en introduisant dans le système un jeu de données connu pour être non biaisé et représentatif, puis en analysant ses prédictions afin de détecter des différences statistiquement significatives dans les résultats entre divers groupes sensibles. Cela permet d'identifier les biais dans les résultats du modèle, que ceux-ci aient été introduits lors de l'entraînement des données ou lors du développement du modèle.
- Tests de l'exactitude des étiquettes (voir 5.1.6) pour identifier les erreurs d'étiquetage qui entraînent l'apprentissage d'associations incorrectes entre les attributs et les résultats pour des attributs sensibles spécifiques.
- Analyse d'impact :
    1. Identifier les attributs sensibles pour le modèle.
    2. Générer des contrefactuels pour les attributs sensibles. (Par exemple, des scénarios où le sexe passe de masculin à féminin dans une demande de prêt).
    3. Générer les résultats du modèle en lui soumettant les contrefactuels.
    4. Analyser les résultats de plusieurs tests afin d'obtenir un résultat statistiquement significatif, qui détermine si la modification d'un attribut sensible entraîne une modification des résultats du modèle, signalant ainsi la présence d'un biais.

  Remarque : l'analyse d'impact disparate peut également s'appliquer à des combinaisons d'attributs sensibles, ce qui permet d'identifier les préjugés cachés associés à ces combinaisons. Il convient toutefois de veiller à ce que les exemples contrefactuels ne soient pas irréalistes, car le modèle pourrait alors réagir à ces derniers plutôt qu'à un éventuel préjugé sous-jacent.

5.1.3 Test du pipeline de données

Un test efficace du pipeline de données est essentiel non seulement pour confirmer la fiabilité et les performances des systèmes pilotés par les données, mais aussi pour garantir une qualité élevée des données tout au long du workflow ML (voir 3.1.2).

Une approche par couches est employée, commençant par des revues de conception du pipeline de données pendant la phase de conception.

Les tests de composants comprennent les composants d'ingestion de données, les scripts de transformation et les interfaces de capteurs. Ces tests utilisent des revues de code, des analyses

statiques et des tests spécifiques au matériel pour vérifier la fiabilité de la capture des données. Les tests de composants valident la logique de transformation des données, vérifient l'implémentation des règles de validation des données, confirment la robustesse de la gestion des erreurs et recherchent les vulnérabilités qui pourraient être exploitées pour introduire des logiciels malveillants ou des données corrompues.

Les tests d'intégration des composants vérifient la fluidité du flux de données entre les interfaces internes et l'interprétation correcte des données tout au long du pipeline. Ils permettent de détecter les défauts résultant d'une incompatibilité entre les interfaces ou d'hypothèses erronées entre les composants.

Les tests système évaluent le pipeline entièrement assemblé, en commençant par des tests de vérification de base (smoke test) afin de confirmer son fonctionnement de base. Les tests fonctionnels vérifient la conformité du pipeline aux exigences spécifiées, y compris les transformations et l'acheminement des données. Les tests non fonctionnels évaluent les performances sous charge, l'évolutivité pour gérer des volumes de données croissants et les mesures de sécurité visant à protéger l'intégrité des données. Les tests d'injection de défauts mesurent la robustesse du pipeline en simulant des entrées de données défectueuses, évaluant ainsi la capacité du système à maintenir l'intégrité des données face à des données inattendues ou corrompues. Les tests dos-à-dos (voir 6.1.10) comparent le pipeline opérationnel au pipeline d'entraînement afin de vérifier la cohérence des fonctionnalités.

Les tests d'intégration des systèmes permettent de vérifier la bonne interaction entre le pipeline de données et les systèmes ou services externes, notamment les sources de données, les plateformes de stockage, les outils de monitoring et les consommateurs en aval tels que les modèles ML.

Les tests en production sont effectués sur le système opérationnel. Les tests dos-à-dos vérifient que les performances sont constantes ou améliorées par rapport aux versions précédentes. Les tests A/B (voir 6.1.9) permettent de comparer les nouvelles itérations du pipeline aux baselines, validant ainsi les améliorations tout en confirmant l'absence de dégradation du flux de données en direct. Des outils peuvent être intégrés pour effectuer le suivi et observer en continu, en temps réel, le comportement des modèles, leurs performances et les défauts potentiels.

Les revues de gestion de la configuration permettent de vérifier que les versions de code, les configurations et les jeux de données corrects sont utilisés dans les environnements d'entraînement, de test et de production.

Enfin, les stratégies de test doivent être adaptées à l'objectif du pipeline. Les pipelines d'entraînement, qui sont souvent des prototypes exploratoires, ont des priorités différentes de celles des pipelines opérationnels robustes. Les tests des pipelines d'entraînement peuvent se concentrer sur l'intégrité des données, tandis que les tests des pipelines opérationnels privilégient la fiabilité, les performances et la maintenabilité.

5.1.4 Test de la représentativité des données

Les tests de représentativité des données permettent de déterminer dans quelle mesure les caractéristiques des jeux de données utilisés pour l'entraînement, la validation et l'évaluation des modèles ML correspondent aux données réelles auxquelles le modèle sera confronté en conditions réelles d'exploitation. Ces tests portent sur diverses formes de non-représentativité, notamment les jeux de données asymétriques, les données manquantes, les distributions disproportionnées des caractéristiques, la couverture insuffisante des scénarios opérationnels et la représentation déséquilibrée des classes (voir 5.1.1).

Ces tests comprennent généralement les étapes suivantes:

1. Définir la population cible :

- Comprendre les cas d'utilisation prévus et le contexte opérationnel du MLS
- Analyser les caractéristiques des utilisateurs finaux et des environnements opérationnels
- Identifier les distributions de données opérationnelles attendues et les cas limites critiques en :
  - consultant des experts du domaine pour comprendre les schémas de données réels
  - analysant les données provenant de systèmes existants ou d'applications similaires
  - utilisant des jeux de données de benchmark provenant de sources fiables (par exemple, le NIST, des bases de données sectorielles)
- Appliquer un échantillonnage stratifié aux données de référence représentatives afin de créer une baseline couvrant tous les sous-groupes pertinents du domaine ciblé

2. Analyser les caractéristiques des données :

- Appliquer l'analyse exploratoire des données (EDA) (voir 3.2.1):
  - Aux jeux de données d'entraînement/de test dont la représentativité est évaluée
  - Aux jeux de données de référence représentant les données opérationnelles attendues
  - Pour visualiser les distributions à l'aide d'histogrammes, de nuages de points et d'autres techniques graphiques
- Examiner les relations entre les caractéristiques, et en particulier les corrélations, afin d'identifier les canevas à préserver
- Identifier les anomalies potentielles, les lacunes ou les concentrations inhabituelles dans les données

3. Appliquer des techniques d'évaluation statistique :

- Utiliser des tests statistiques formels tels que le test du chi carré et le test de Kolmogorov-Smirnov pour comparer les distributions [STATS]
- Vérifier l'absence de déséquilibre dans les données, en particulier dans les problèmes de classification
- S'assurer que les scénarios typiques ainsi que les cas limites sont correctement couverts

Il convient de vérifier la représentativité des données avant l'entraînement du modèle afin d'éviter de construire des modèles à partir de données non représentatives. Après le déploiement, les caractéristiques des données d'entrée opérationnelles doivent faire l'objet d'un suivi continu afin de

détecter tout changement susceptible d'indiquer une dérive par rapport aux distributions des données d'entraînement d'origine (voir la section 6.1.7 sur les tests de dérive).

5.1.5 Test des contraintes des jeux de données

Le test des contraintes d'un jeu de données permet de vérifier si les données contenues dans ce jeu respectent des règles ou des contraintes prédéfinies. L'objectif est de confirmer l'intégrité et la cohérence des données utilisées en ML. Par exemple, on peut effectuer un test de cohérence des valeurs d'un jeu de données. Les contraintes sur les données se trouvent généralement dans les schémas de base de données. Un schéma de base de données définit la structure, les types et les relations entre les données. De même, pour les jeux de données ML, il est possible de définir un ensemble de contraintes qui servent de modèle logique aux données du jeu de données et qui doivent être satisfaites pour que les données soient correctes. Il existe plusieurs façons de classer les contraintes d'un jeu de données. Une contrainte peut s'appliquer à une seule valeur d'un attribut, qui est trouvée dans une seule instance (contrainte à valeur unique), par exemple:

- Valeurs manquantes – test pour détecter les valeurs ou les attributs manquants.
- Intervalle – test pour vérifier que la valeur se situe dans un intervalle donné.
- Type – test pour vérifier que la valeur d'attribut correspond au type spécifié (par exemple, si un attribut est défini comme un entier, la valeur fournie ne doit pas être une chaîne de caractères ni un nombre réel).

Une contrainte peut également s'appliquer à plusieurs valeurs, en prenant généralement en compte les valeurs d'un même attribut sur plusieurs instances (contrainte à valeurs multiples), par exemple:

- Somme – teste si la somme de toutes les valeurs est égale à, supérieure à ou inférieure ou égale à une valeur spécifiée (par exemple, le total des points attribués pour une course de Formule 1 ne peut pas dépasser 102 et doit être supérieur à 50,5 points).
- Nombre – teste si le nombre de toutes les valeurs non nulles pour des attributs ou des instances est égal à, supérieur à ou inférieur ou égal à une valeur spécifiée.
- Doublon : teste la présence de valeurs d'attributs ou d'instances identiques ou quasi identiques dans un jeu de données et impose une limite sur le nombre autorisé (souvent zéro).
- Utile : teste qu'un attribut contient des valeurs répétées, comme si chaque entrée était unique (par exemple un identifiant ou un horodatage) ; ne fournit généralement aucun modèle utile à apprendre pour un modèle de ML.
- Valeur aberrante : identifie toute valeur pouvant être considérée comme une valeur aberrante statistique.

Une forme particulière de contrainte à valeurs multiples permet de comparer différentes valeurs (une contrainte de comparaison), par exemple:

- Supérieur à – teste qu'une valeur d'un attribut est supérieure à une valeur d'un deuxième attribut (par exemple, que le nombre de lignes de code d'un programme dépasse le nombre de lignes de code comportant des défauts).
- Corrélation – teste si les valeurs d'un attribut sont corrélées avec celles d'un deuxième attribut (par exemple, tous les candidats dont la valeur de l'attribut « notes » est supérieure d'au moins 1,33 écart-type à la moyenne ont également la valeur « A » pour leur attribut « note »).

La vérification de la conformité des données aux contraintes définies peut être effectuée manuellement. Toutefois, l'ampleur de cette tâche et la taille généralement importante de l'ensemble de données nécessiteraient de l'automatiser dans le cadre du pipeline de données. Une fois intégré au pipeline, l'outil chargé de vérifier la conformité de l'ensemble de données peut fournir des rapports aux data scientists (pour les anomalies dans les données d'entraînement) ou au personnel d'exploitation (pour les problèmes liés aux données opérationnelles).

5.1.6 Test de l'exactitude des étiquettes

L'exactitude des étiquettes de données est essentielle dans l'apprentissage supervisé. Des étiquettes inexactes ou incohérentes nuisent directement aux performances et à la capacité de généralisation des modèles ML. Parmi les approches courantes pour tester l'exactitude des étiquettes de données, on peut citer:

- Revue par des experts : des experts du domaine ou des annotateurs formés examinent manuellement un échantillon de données annotées. Ils évaluent la précision des annotations en utilisant leurs connaissances du domaine et des directives.
- Annotation multiple : les points de données sont annotés de manière indépendante par plusieurs annotateurs, puis comparés à l'aide d'une méthode de tests dos-à-dos (voir 6.1.10). Les divergences mettent en évidence les défauts qui doivent faire l'objet d'une analyse et d'une résolution. La concordance inter-annotateurs (IAA) peut être mesurée en utilisant des métriques telles que le coefficient Kappa de Cohen ou le simple pourcentage de concordance [STATS]. De faibles scores IAA peuvent indiquer des défauts dans les directives d'étiquetage, des données ambiguës ou une annotation de mauvaise qualité.
- Priorisation basée sur les risques pour la revue et l'annotation : tant les revues par des experts que les annotations multiples peuvent être ciblées en utilisant une approche basée sur les risques afin d'identifier les échantillons à revoir et à annoter. La hiérarchisation peut être fondée sur la probabilité d'erreurs d'étiquetage, par exemple en cas de points de données ambigus (par exemple, lorsqu'il est difficile de déterminer à quelle catégorie ils appartiennent ou lorsqu'ils se situent à la limite entre deux classes), et sur les données les plus susceptibles d'influencer le succès ou la sûreté de l'application.
- Analyse de la distribution des données : lorsqu'il existe des jeux de données comparables, la comparaison de la distribution des étiquettes de l'ensemble de données testé avec celle d'ensembles de données similaires peut révéler des anomalies et des étiquettes potentiellement incorrectes.
- Tests automatisés basés sur des règles : des tests automatisés peuvent être mis en œuvre sur la base de règles ou de contraintes d'étiquetage prédéfinies pour certaines tâches. Par exemple, vérifier que les cadres de sélection (rectangles utilisés pour localiser des objets dans les images) ne se chevauchent pas ou ne dépassent pas les limites de l'image.
- Analyse de la perte du modèle : les points de données présentant une perte élevée pendant l'entraînement du modèle – ce qui signifie que les prédictions du modèle pour ces points s'écartent considérablement de leurs étiquettes réelles – peuvent indiquer une erreur d'étiquetage. Une perte élevée reflète une erreur significative, indiquant que le modèle a du mal à apprendre l'étiquette attribuée et signalant un défaut potentiel.

- Analyse du score de confiance du modèle : les points de données présentant une faible confiance de prédiction provenant d'un modèle entraîné peuvent être mal étiquetés, ambigus ou se situer en dehors de la distribution des données d'entraînement du modèle.

Il est souvent plus efficace d'utiliser plusieurs approches. Par exemple, les revues d'experts sont précieuses pour établir dès le début des directives claires en matière d'étiquetage. Les scores IAA issus de multiples annotations permettent ensuite de réaliser un remaniement du processus d'étiquetage. Par la suite, les approches basées sur des modèles peuvent permettre d'identifier davantage d'éventuels défauts d'étiquetage à mesure que le modèle continue de se développer.

5.1.7 Exercice pratique: Test des données d’entrée

Pour un jeu de données donné (par exemple, des données structurées sous forme de tableaux), effectuez un test des données d'entrée afin de détecter les données manquantes, les doublons et les valeurs aberrantes.

6 Test des modèles pour les systèmes de Machine Learning – 225 minutes

Mots-clés

Tests A/B, tests adverses, tests dos-à-dos, dérive conceptuelle, dérive des données, tests de dérive, tests métamorphiques, performance fonctionnelle du ML, test de modèle de ML, revue

Mots-clés spécifiques à l'IA

Surajustement, sous-ajustement

Objectifs d'apprentissage du chapitre 6:

6.1 Test de modèles pour les systèmes de machine learning

AI-6.1.1 (K2) Donner des exemples d'approches de test utilisées pour l'atténuation des risques liés aux modèles ML

AI-6.1.2 (K2) Expliquer l'objectif et l'intérêt de la revue de la documentation des modèles ML

AI-6.1.3 (K2) Expliquer comment sont réalisés les tests de performance de ML pour les systèmes d'apprentissage automatique probabilistes

AI-6.1.4 (K2) Résumer les tests adverses des systèmes de machine learning

AI-6.1.5 (K3) Utiliser les tests métamorphiques pour dériver des cas de test pour un scénario donné

HO-6.1.6 (H2) Appliquer les tests métamorphiques

AI-6.1.7 (K2) Expliquer comment les tests de dérive sont utilisés sur les systèmes de machine learning opérationnels

AI-6.1.8 (K2) Expliquer comment le surajustement et le sous-ajustement sont détectés par les tests

AI-6.1.9 (K2) Expliquer comment les tests A/B sont utilisés dans le contexte des systèmes de machine learning

AI-6.1.10 (K2) Expliquer comment les tests dos-à-dos sont utilisés dans le contexte des systèmes de machine learning

6.1 Test de modèles pour les systèmes de Machine Learning

Le test de modèle de ML consiste à traiter des risques spécifiques. Il s'agit notamment de risques fonctionnels, tels que les biais, le surajustement et les vulnérabilités face aux attaques adverses, ainsi que de risques non fonctionnels, tels que le manque de robustesse de l'IA et d'efficacité de performance, et de risques liés au déploiement.

6.1.1 Risques liés aux modèles de Machine Learning et mesures d'atténuation des risques

Le tableau suivant présente des exemples de risques liés aux modèles ML ainsi que les tests correspondants qui pourraient être utilisés pour l'atténuation de ceux-ci:

Risque potentiel	**Mesures possibles d'atténuation des risques**
Modèle de ML biaisé ou discriminatoire	Tester la présence de biais – voir 5.1.2
Modèle contraire à l'éthique	Tests systèmes basés sur l'éthique
Exemples adverses	Tests adverses – voir 6.1.4
Modèle surajusté	Tester le surajustement – voir 6.1.8
Modèle sous-ajusté	Tester le sous-ajustement – voir 6.1.8
Déviation inacceptable des données Déviation inacceptable du concept	Test de dérive – voir 6.1.7
Le modèle entraîne des effets secondaires	Tester les effets secondaires
Le modèle illustre le piratage de récompense	Tests de piratage de récompense
Défaut de l'API du modèle	Tests de l'API – voir 7.1.2
Défaillance concernant les mesures de la performance requise pour le modèle de ML (par exemple, manque de précision, de rappel)	Tests de performance fonctionnelle de ML – voir 6.1.3
Anomalie fonctionnelle Défauts non fonctionnels	Test métamorphique – voir 6.1.5
Problème avec l'oracle de test	Test métamorphique – voir 6.1.5 Tests dos-à-dos – voir 6.1.10 Tests A/B – voir 6.1.9

Exigences système médiocres	Revue des exigences Exercice de simulation d'attaque (Red Teaming) – voir 4.2.2 Test exploratoire
Manque de robustesse du modèle d'IA dû à des entrées inattendues	Tests adverses – voir 6.1.4 Test à données aléatoires
Efficacité de performance insuffisante du modèle de ML	Test de performance
Documentation médiocre du modèle (par exemple, fonctionnalités, précision, interface)	Revue de la documentation du modèle – voir 6.1.2
Les mises à jour du modèle introduisent des défauts	Tests dos-à-dos – voir 6.1.10
Les mises à jour du modèle de ML réduisent la performance fonctionnelle de ML	Tests A/B – voir 6.1.9
Le déploiement du modèle mis à jour entraîne une défaillance immédiate	Smoke test
Le déploiement du modèle mis à jour entraîne une régression	Test de régression
Vulnérabilités de sécurité Vulnérabilités de sûreté Atteintes à la vie privée Résultats préjudiciables ou indésirables (par exemple, propos racistes, conseils dangereux)	Red teaming – voir 4.2.2

6.1.2 Documentation et révision des modèles de Machine Learning

Une documentation complète des modèles de ML n'est pas une simple formalité ; c'est une ressource essentielle. Contrairement aux systèmes dont le code source peut être inspecté directement, les modèles de ML posent des défis particuliers en raison de la facilité de compréhension limitée du code généré par les machines, de la nature intrinsèquement opaque des modèles et de leur dépendance vis-à-vis des données. Les modèles étant en outre fréquemment mis à jour, la documentation devient l'outil principal dont disposent les développeurs, les testeurs et les régulateurs pour comprendre, évaluer et faire confiance aux systèmes basés sur l'IA. La transparence joue un rôle central pour permettre cette compréhension, en permettant aux parties prenantes de retracer le comportement du modèle, la logique de décision et la traçabilité des données tout au long du cycle de vie de l'IA.

Une documentation standardisée améliore la communication, soutient une prise de décision éclairée et permet de vérifier la qualité et la maintenabilité des modèles de ML. Elle revêt une importance croissante

pour la conformité réglementaire, comme en témoignent des frameworks tels que la loi européenne sur l'IA, qui met l'accent sur les obligations de transparence exigeant une documentation claire des décisions des modèles, de leurs limites et des mesures d'interprétabilité. Pour les systèmes à haut risque, la réussite d'un audit de documentation est souvent une condition préalable au déploiement, tandis que pour tous les systèmes, un examen approfondi permet de vérifier la qualité.

Il existe plusieurs cadres documentaires, notamment les « Model Cards », qui fournissent des aperçus concis des utilisations prévues d'un modèle, des résultats d'évaluation et des considérations éthiques [MODEL_DOC], ainsi que les « Datasheets for Datasets », qui proposent des formats normalisés pour décrire les jeux de données, y compris leur justification, leur composition, leur processus de collecte et leurs utilisations [DATA_DOC].

La liste suivante présente les éléments typiques attendus dans une documentation complète du modèle de ML, structurée de manière à pouvoir servir de checklist pratique tant pour les développeurs que pour les testeurs, afin de garantir la complétude, la clarté et la testabilité:

- Généralités : identifiants, description, développeur, version, date, coordonnées, licence, configuration matérielle requise
- Conception : hypothèses, décisions techniques, algorithme de ML
- Utilisation : usage prévu, utilisations principales/secondaires, utilisateurs, approche d'auto-apprentissage, biais, éthique, sûreté, transparence, seuils, plateforme, dérive des données, dérive conceptuelle
- Jeux de données : caractéristiques, source, collecte, disponibilité, prétraitement, utilisation, contenu, étiquettes, taille, confidentialité, sécurité, biais/équité, restrictions
- Tests : détails de l'ensemble de données de test, indépendance des tests, résultat du test, activités de test (par exemple, fonctionnels, adverses)
- Fonctionnel : mesures, jeu de données de validation, seuils, performances réelles
- Non fonctionnels : évolutivité, fiabilité, disponibilité, efficacité de performance (par exemple, latence, utilisation des ressources), maintenabilité, robustesse de l'IA
- Opérationnels : plan de déploiement, environnement de déploiement, ressources informatiques, indicateurs de suivi/alertes, stratégie de réentraînement, plan de mise à jour/rétrogradation du modèle, plan de dépréciation, sécurité (risques adversaires), méthodes d'explicabilité

La vérification de la documentation à l'aide de ces checklists constitue une activité de test principale, visant à:

- Identifier les informations manquantes, les inexactitudes et les incohérences
- Améliorer la clarté et la lisibilité
- Faciliter la maintenabilité en identifiant les points de la documentation qui doivent être améliorés
- Fournir des informations suffisantes pour les activités de test et de déploiement
- Vérifier que toutes les exigences réglementaires applicables ont été respectées

6.1.3 Test de performance fonctionnelle de ML pour les systèmes de ML probabilistes

Les tests de performance fonctionnelle de ML évaluent dans quelle mesure un modèle de ML remplit les fonctions qui lui sont assignées, en mesurant des métriques telles que la précision, le rappel, l'exactitude et le score F1 (voir 3.3) et en comparant les résultats à des critères d'acceptation prédéfinis.

Ces tests pour le MLS probabiliste vont au-delà d'un simple statut « réussi/échec » pour mesurer statistiquement les performances d'un modèle par rapport à ses critères d'acceptation [STATS]. Cette approche est nécessaire pour tenir compte du non-déterminisme inhérent au MLS en évaluant le comportement sur un ensemble de jeux de données volumineux et représentatifs (voir 4.1.2). Une précondition pour ces tests est de disposer de critères d'acceptation définis en termes statistiques. Au lieu d'une simple cible, une exigence peut spécifier une métrique de performance, une marge d'erreur (MoE) et un niveau de confiance (CL). Une telle exigence peut être utilisée pour déterminer le nombre minimum de tests requis. À mesure que le nombre de tests augmente, deux options s'offrent:

- Si l'on fixe le niveau de confiance (CL), la marge d'erreur (MoE) diminuera. Cela signifie que le résultat du test sera plus précis.
- Si l'on fixe la marge d'erreur (MoE), le niveau de confiance (CL) augmentera. Cela signifie que le résultat du test sera plus certain.

Par exemple, un critère pourrait être « une précision de 98 % avec une marge d'erreur de ±4 % à un niveau de confiance de 95 % ». Pour confirmer que la précision mesurée présente une marge d'erreur maximale de ±4 % avec un niveau de confiance de 95 %, un échantillon de 601 cas de test est nécessaire. Ce chiffre est calculé à l'aide de la formule de calcul de la taille d'échantillon permettant d'estimer une proportion dans une population. Cette taille d'échantillon repose sur l'hypothèse prudente selon laquelle la précision réelle pourrait se situer entre 0 % et 100 %, ce qui génère l'incertitude la plus grande possible tout en garantissant une marge d'erreur de ±4 % dans toutes les conditions. Pour cibler la précision de 98 %, au moins 589 des 601 cas de test doivent être passés. Si, après avoir exécuté les 601 cas de test, la précision observée est de 98 %, alors la MoE mesurée avec un CL à 95 % sera inférieure à ±4 % (environ ±1,1 %), car la variance est plus faible à des niveaux de précision élevés. La marge d'erreur est calculée à l'aide de la formule de la marge d'erreur pour une proportion d'échantillon. Cela signifie que lors des tests, il n'est pas toujours nécessaire d'exécuter les 601 cas de test. Les tests séquentiels fournissent un framework statistique formel pour cet arrêt précoce. Au lieu d'exécuter systématiquement l'échantillon fixe complet, ces approches analysent les résultats au fur et à mesure de leur accumulation et s'arrêtent prématurément lorsque des preuves suffisantes soutiennent (ou infirment) l'objectif de précision. Si la précision observée reste constamment élevée (par exemple, pas moins de 98 %), l'incertitude statistique diminue à mesure que davantage de tests sont effectués. Dans ce cas, l'exigence en matière de marge d'erreur de ±4 % avec un niveau de confiance de 95 % est atteinte après environ 170 cas de test, ce qui permet de conclure les tests plus tôt.

> REMARQUE : Les formules et les calculs numériques présentés dans cet exemple (concernant la taille de l'échantillon, la marge d'erreur et les intervalles de confiance) sont fournis à titre d'illustration et pour faciliter la compréhension uniquement ; les candidats ne seront pas tenus de déduire ou de calculer ces valeurs en utilisant des formules statistiques lors de l'examen.

Pour les systèmes critiques en termes de sécurité, on peut recourir à une exigence de fiabilité plus stricte, telle que « une fiabilité de 99 % avec un niveau de confiance de 95 % », ce qui nécessiterait 299 cas de test, qui doivent tous être réussis pour satisfaire aux critères. Pour valider ces critères, les tests nécessitent un vaste jeu de données de test totalement indépendant des ensembles de données d'entraînement et de validation. Cet ensemble de données de test doit constituer un échantillon représentatif du domaine d'entrée opérationnel (voir 5.1.4) afin de vérifier que l'évaluation reflète les

conditions réelles. Les cas de test sont ensuite exécutés à l'aide du modèle de ML au sein d'un framework de développement de ML capable d'effectuer des analyses statistiques.

Enfin, les résultats agrégés sont interprétés et présentés avec un niveau de confiance statistique, et non sous la forme d'un simple ratio réussite/échec. Un rapport de test final indiquerait, par exemple : « Le modèle a atteint une précision de 94 % ± 4 % avec un intervalle de confiance de 95 %. » Cela permet aux parties prenantes de comprendre la fourchette des performances opérationnelles attendues et de prendre une décision en connaissance de cause sur la performance fonctionnelle de ML du modèle.

6.1.4 Test adverse des systèmes de Machine Learning

Les tests adverses consistent à soumettre délibérément le modèle à des perturbations des données d'entrée qui sont souvent imperceptibles pour l'œil humain. Ces entrées sont conçues pour amener le modèle à produire des prédictions erronées et, en cas de succès, sont appelées « exemples adverses » . Ainsi, les entrées utilisées pour les tests adverses, et donc les exemples adverses potentiels, consistent souvent en des versions légèrement modifiées d'entrées légitimes qui conduisent le modèle à les classer de manière erronée.

L'identification des vulnérabilités par le biais des tests adverses permet aux développeurs d'intégrer des mesures de protection et de rendre le modèle plus robuste face aux exemples adverses. Il peut s'agir d'exemples adverses accidentels rencontrés par le modèle, ou d'attaques adverses impliquant l'utilisation malveillante d'exemples adverses. La génération d'entrées de test efficaces pour les tests adverses est techniquement complexe, et se tenir à jour face à l'évolution des techniques d'attaque constitue un défi permanent.

Les tests adverses peuvent être réalisés en utilisant des tests boîte noire, qui se concentrent sur le comportement du modèle en entrée et en sortie sans nécessiter de connaissance de son fonctionnement interne. Pour ce faire, on peut créer un modèle équivalent dont le fonctionnement interne est connu, à partir duquel des entrées de test adverses peuvent être générées grâce à cette connaissance. Ensuite, en partant du principe que les modèles équivalents partagent les mêmes limites de classification (c'est-à-dire la transférabilité), les tests adverses peuvent être appliqués au modèle d'origine. À l'inverse, une approche par force brute utilisant un grand nombre de tests peut être employée dans l'espoir que certains tests aléatoires coïncident avec un exemple adverse.

Le test adverse peut également être réalisé en utilisant des tests en boîte blanche. La compréhension des composants internes d'un modèle de ML (architecture, paramètres, processus d'entraînement) facilite généralement la création d'exemples adverses.

Il est possible de réaliser des tests adverses manuellement en créant des exemples adverses spécifiques ou à l'aide d'algorithmes automatisés qui génèrent un grand nombre de variations afin de trouver des entrées adverses efficaces.

6.1.5 Test métamorphique

Le test métamorphique (MT) constitue une technique de test dans laquelle de nouveaux cas de test (dérivés) sont générés à partir d'un cas de test source ayant déjà été passé. Un ou plusieurs cas de test de vérification sont générés en modifiant (métamorphisant) le cas de test source à l'aide d'une relation métamorphique (MR). La MR repose sur une propriété d'une fonction requise de l'objet de test et décrit comment une modification des entrées d'un cas de test se répercute sur les résultats attendus de ce même cas de test.

Le MT peut être utilisé pour la plupart des objets de test et s'applique aussi bien aux tests fonctionnels que non fonctionnels. Les testeurs vérifient des objectifs de test tels que la cohérence (les sorties s'alignent sur les entrées associées), la monotonie (les sorties changent de manière directionnelle avec l'entrée) et l'invariance (les sorties restent stables face à des perturbations). Il est particulièrement utile lorsque la génération des résultats attendus pose problème en raison de l'indisponibilité d'un oracle de test abordable. C'est le cas de certains systèmes de Machine Learning (ML) qui utilisent le big data, ou de systèmes où les testeurs ne savent pas exactement comment le modèle de ML dérive ses prédictions, ce qui est souvent le cas. Dans le domaine de l'IA et du ML, les tests métamorphiques ont été utilisés pour tester la reconnaissance d'images, les moteurs de recherche, l'optimisation d'itinéraires et la reconnaissance vocale.

Le MT est généralement préféré aux tests traditionnels basés sur un oracle lorsque:

- Il n'existe aucun résultat fiable attendu en raison de l'opacité du modèle ou de l'échelle des données ;
- Le système est une boîte noire ; ou
- Les propriétés relationnelles (et non les valeurs absolues) suffisent pour établir la confiance.

Le MT repose souvent sur un cas de test source qui a passé, et il peut également s'avérer utile lorsqu'il n'est pas possible de générer un résultat attendu. Par exemple, lorsque le programme implémente une fonction trop complexe pour qu'un testeur humain puisse la reproduire et l'utiliser comme oracle de test, comme c'est le cas avec certains MLS complexes. Dans cette situation, le MT peut être utilisé pour générer des cas de test qui, une fois exécutés, produiront un ensemble de résultats dont la validité sera vérifiée en fonction des relations entre ces résultats (plutôt que de leurs valeurs réelles). Avec cette forme de MT, si les relations entre les sorties de test sont vérifiées, cela renforce la confiance dans le programme. Par exemple, un MLS d'évaluation des risques prédisant l'âge au décès, où l'augmentation du nombre de cigarettes fumées devrait diminuer la prédiction (monotonicité).

Les testeurs dérivent les relations de test (MR) à partir de la connaissance du domaine, des exigences ou des propriétés du domaine (par exemple, les lois de la physique). Les MR peuvent être validées par une revue d'experts, en les exécutant sur des modèles de référence et en vérifiant la couverture des cas limites.

Des MR incorrectes (par exemple, négligeant des interactions complexes entre les variables) ou des ensembles incomplets de MR peuvent conduire à une fausse confiance. Le MT détecte les défauts relationnels mais pas toutes les erreurs absolues ; elle doit donc être utilisée en combinaison avec d'autres techniques de test.

6.1.6 Exercice pratique: Appliquer le test métamorphique

Dans cet exercice, les candidats acquerront une expérience pratique des éléments suivants:

- Dériver plusieurs MR pour un MLS donné. Ces MR doivent inclure certains cas de test où les résultats attendus des cas de test source et de cas de test supplémentaires sont identiques, et d'autres où ils diffèrent.
- Générer des cas de test sources pour le MLS. Il n'est pas nécessaire de garantir leur réussite, mais il convient de rappeler aux étudiants les limites du MT lorsque les cas de test sources ayant passé les tests ne sont pas disponibles.

- Utiliser les MR dérivées et les cas de test sources générés pour dériver des cas de test supplémentaires.
- Exécuter les cas de test supplémentaires.

6.1.7 Test de dérive

Les tests de dérive permettent d'identifier deux types de dérive dans les MLS en exploitation:

- Dérive des données - se produit lorsque les propriétés statistiques des données d'entrée opérationnelles évoluent au fil du temps. Par exemple, les données d'entrée diffèrent désormais considérablement de celles sur lesquelles le modèle a été entraîné, en raison de facteurs tels que des changements dans le comportement des utilisateurs ou la saisonnalité. Ainsi, un filtre anti-spam peut être confronté à de nouveaux types d'attaques d'hameçonnage qui n'existaient pas lors de son entraînement.
- Dérive conceptuelle : se produit lorsque la relation entre les données d'entrée et la sortie correcte évolue au fil du temps. Cela signifie que les modèles ou règles initialement appris par le modèle ne reflètent plus la réalité actuelle. Par exemple, en raison de nouvelles réglementations financières, un type de transaction auparavant considéré comme « à faible risque » pourrait désormais être classé comme « à haut risque ». La signification des données a changé, rendant obsolètes les limites de décision apprises par le modèle, ce qui entraîne une baisse de sa précision prédictive.

Le test de dérive dynamique reposent sur la disponibilité des retours d'information des utilisateurs, qui fournissent la référence actuelle. Cette référence actuelle est comparée au résultat du modèle, et la différence entre les deux est déterminée puis comparée à une valeur seuil. Les retours d'information des utilisateurs peuvent être directs ou indirects. Dans le cas d'un système de recommandation de films, un exemple de retour direct est le fait pour l'utilisateur de noter une recommandation. Un exemple de retour indirect pour ce même système serait tiré des données relatives aux films que l'utilisateur a visionnés.

Le test de dérive statique ne dépend pas de la vérité terrain actuelle, mais compare plutôt les propriétés statistiques des distributions des données d'entrée et des données de sortie prédites à l'aide d'un test tel que celui de Kolmogorov-Smirnov [STATS]. Une différence significative entre l'une ou l'autre de ces distributions indique qu'une dérive s'est produite.

6.1.8 Test de surajustement et de sous-ajustement

Le surajustement et le sous-ajustement sont deux des trois résultats possibles rencontrés dans les modèles ML, le troisième étant un modèle « bien ajusté ». La détection du surajustement et du sous-ajustement doit avoir lieu pendant l'entraînement, l'évaluation et l'ajustement (NDT: tuning).

Le surajustement se produit lorsqu'un modèle apprend trop bien les données d'entraînement, au point de capturer le bruit présent dans les données plutôt que la structure sous-jacente. Cela se traduit par une mauvaise généralisation aux nouvelles données non vues.

Pour détecter le surajustement, la performance fonctionnelle de ML du modèle est évaluée sur un jeu de données de test distinct qui n'a pas été utilisé pendant l'entraînement. Ce jeu de données de test doit inclure des exemples moins courants, peu susceptibles d'avoir été utilisés pendant l'entraînement. Le modèle peut être en situation de surajustement s'il obtient des résultats nettement moins bons sur le jeu de données de test que sur le jeu de données de validation.

On parle de sous-ajustement lorsqu'un modèle est trop simple pour saisir la structure sous-jacente des données ou lorsque les données d'entraînement ne contiennent pas de caractéristiques reflétant une relation importante entre les entrées et les sorties, ce qui se traduit par une mauvaise performance fonctionnelle de ML tant sur l'ensemble d'entraînement que sur l'ensemble de validation.

Lors des tests, le sous-ajustement peut être détecté en évaluant les mesures de performance fonctionnelle ML, telles que l'exactitude, la précision, le rappel ou le score F1. Si ces mesures sont systématiquement faibles tant sur l'ensemble d'entraînement que sur l'ensemble de validation, cela suggère que le modèle est en situation de sous-ajustement.

Une inspection visuelle des courbes d'apprentissage du modèle peut également aider à détecter le sous-ajustement. Si les erreurs d'apprentissage et de validation restent élevées et relativement proches l'une de l'autre, sans amélioration significative au fur et à mesure que l'entraînement progresse, cela indique un sous-ajustement.

En résumé, la détection du surajustement et du sous-ajustement lors des tests implique d'évaluer les performances fonctionnelles du modèle sur les données de validation, d'analyser les mesures de performance fonctionnelle ML et d'examiner les courbes d'apprentissage.

6.1.9 Test A/B

Le test A/B est une méthode qui consiste à comparer la réaction de deux variantes d'un programme (A et B) face aux mêmes données d'entrée, dans le but de déterminer laquelle des deux variantes est la plus performante. Il s'agit d'une approche statistique qui nécessite généralement de comparer les résultats de plusieurs exécutions de test afin de mettre en évidence les différences entre les programmes.

Un exemple simple de cette approche consiste à envoyer deux offres promotionnelles par e-mail à une liste de marketing divisée en deux groupes. La moitié de la liste reçoit l'offre A, et l'autre moitié l'offre B ; le succès de chaque offre aide à décider laquelle utiliser à l'avenir. De nombreuses entreprises de commerce électronique et en ligne utilisent le test A/B en production, en redirigeant différents consommateurs vers différentes fonctionnalités, afin d'identifier leurs préférences.

Le test A/B constitue une approche pour résoudre le problème de l'oracle de test, en utilisant généralement le système existant comme oracle de test 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 des données opérationnelles soient souvent intégrées aux tests. Le test A/B peut être utilisé pour tester les mises à jour d'un système basé sur l'IA, à condition qu'il existe des critères d'acceptation convenus, tels que des mesures de performance fonctionnelle de ML, comme décrit au point 3.3. Chaque fois que le système est mis à jour, le test A/B est utilisé pour déterminer si la variante mise à jour fonctionne aussi bien, voire mieux, que la variante précédente.

Une telle approche de test peut s'appliquer à un simple classificateur, mais aussi à des systèmes bien plus complexes. Par exemple, une mise à jour visant à améliorer l'efficacité d'un système de calcul d'itinéraires de transport dans une ville intelligente peut être testée à l'aide de tests A/B. On peut notamment comparer les temps de trajet moyens pour deux variantes du système sur des semaines consécutives.

Le test A/B peut également servir à tester des systèmes d'auto-apprentissage. Lorsque le système effectue une modification, des tests automatisés sont exécutés, et les caractéristiques du système qui en

résultent sont comparées à celles d'avant la modification. Si le système est amélioré, la modification est acceptée ; sinon, le système revient à son état précédent.

Les techniques statistiques les plus courantes pour les tests A/B sont le test t (NDT : comparaison de moyennes), le test z (NDT : similaire au test t mais lorsque l'échantillon de données est grand), le test du chi carré et le test U de Mann-Whitney [STATS].

6.1.10 Test dos-à-dos

Le test dos-à-dos constitue une solution pratique au problème de l'oracle de test.

Le tests dos-à-dos consiste à utiliser une version alternative du système comme point de référence (un pseudo-oracle) et à comparer ses résultats avec ceux du système sous test lorsqu'ils reçoivent les mêmes entrées. Ce pseudo-oracle peut être un système existant ou un système développé spécifiquement pour les tests, mais cela a un coût.

Idéalement, le pseudo-oracle et le système sous test ne devraient pas partager de composants logiciels communs. Sinon, les deux systèmes pourraient contenir le même défaut, ce qui ferait que leurs résultats correspondraient même lorsque les deux sont incorrects. Cela peut être particulièrement problématique compte tenu de l'utilisation généralisée de composants d'IA réutilisables et open source dans le développement du MLS. Pour cette raison, le pseudo-oracle est souvent développé par une équipe différente et, idéalement, indépendante, utilisant peut-être des frameworks de développement de ML, des algorithmes ou des paramètres de modèle différents. Parfois, un logiciel conventionnel peut servir de pseudo-oracle s'il résout le même problème.

Lors de la réalisation de test fonctionnel dos-à-dos, le pseudo-oracle doit uniquement reproduire le comportement fonctionnel. Il n'a pas besoin de satisfaire aux mêmes exigences non fonctionnelles que le système testé, ce qui peut rendre sa mise en place moins coûteuse.

Cette approche de test nécessite uniquement la génération d'entrées de test, et non de résultats attendus, puisque le pseudo-oracle fournit le point de comparaison. Ces entrées peuvent provenir de cas de test existants, tels que des suites de tests de régression, ou être générées automatiquement à partir de données d'entraînement, ce qui permet d'exécuter un grand nombre de tests si le soutien à l'exécution automatisée des tests est offert.

Le test dos-à-dos apporte une valeur ajoutée significative lors de la migration d'un MLS vers un nouvel environnement, par exemple lors du passage du développement à la production, et lors de la comparaison des résultats de test entre différents environnements. De plus, cette approche de test peut révéler des défauts subtils dans le comportement du modèle qui pourraient ne pas être apparents avec d'autres approches de test, en particulier lors de la comparaison des réponses sur un large éventail de cas limites ou d'entrées inhabituelles.

Une différence significative entre les tests A/B (voir 6.1.9) et les tests dos-à-dos réside dans l'utilisation des tests A/B pour comparer deux variantes d'un même MLS à l'aide de mesures de performance fonctionnelle de ML et de techniques statistiques, par opposition à l'utilisation des tests dos-à-dos pour détecter des défauts.

7 Test du développement de Machine Learning – 30 minutes

Mots-clés

Tests de développement de ML, performance fonctionnelle de ML, tests en mode fantôme (NDT : shadow testing)

Mots-clés spécifiques à l'IA

Aucun

Objectifs d'apprentissage du chapitre 7:

7.1 Test du développement de Machine Learning

AI-7.1.1 (K2) Donner des exemples d'approches de test utilisées pour l'atténuation des risques liés au développement de ML

AI-7.1.2 (K2) Expliquer les différentes formes de tests de déploiement des systèmes de ML

7.1 Test du développement de Machine Learning

Ce chapitre se concentre spécifiquement sur les risques liés aux outils de développement de ML, aux choix de configuration et aux mécanismes de déploiement, plutôt que sur le modèle de ML lui-même. Il aborde les approches et les types de tests, tels que les tests de l'API, les tests de performance fonctionnelle du ML, les tests A/B, les tests dos-à-dos et les revues, afin de vérifier la robustesse et l'efficacité de l'IA.

7.1.1 Risques liés au développement de machine learning et mesures d'atténuation des risques

Le tableau suivant présente des exemples de risques liés au développement de ML et les tests correspondants qui pourraient être utilisés pour l'atténuation de ceux-ci:

Risque potentiel	**Mesures possibles d'atténuation des risques**
Utilisation incorrecte ou non prévue des API de bibliothèques ou de frameworks (par exemple, TensorFlow, PyTorch)	Tests de l'API – voir 7.1.2
Choix d'un framework sous-optimal	Revue de l'adéquation du framework
Problèmes au sein de l'algorithme, du modèle ou du framework de développement qui entraînent une discrimination systémique	Tester la présence de biais – voir 5.1.2
Installation ou build incorrects du framework	Smoke testing
Implémentation déficiente de l'évaluation par le framework	Revue du code d'évaluation du framework Vérification croisée des résultats de l'évaluation du framework (par exemple, par rapport à des benchmarks manuels)
Faible efficacité de performance (par exemple, le framework est lent à répondre)	Test de performance
Mauvaise utilisabilité du framework	Tests d'utilisabilité
Défaut dans une bibliothèque utilisée par le framework (par exemple, défaut dans PyTorch) Implémentation défectueuse d'un algorithme	Tests fonctionnels de la performance de ML – voir 6.1.3 Tests dos-à-dos – voir 6.1.10

Vulnérabilités de sécurité dans le framework	Tests de sécurité
Documentation utilisateur insuffisante pour le framework	Revue de la documentation du framework
Choix d'un algorithme non optimal	Évaluation de l'adéquation des algorithmes par test A/B – voir 6.1.9
Sélection sous-optimale des hyperparamètres (par exemple, structure du réseau, taux d'apprentissage)	Tests fonctionnels de ML – voir 6.1.3 Test A/B – voir 6.1.9
Répartition inadéquate des données entre les jeux de données d'entraînement, de validation et de test	Revue de l'allocation des données
Mauvais choix de la méthode d'évaluation (par exemple, la validation croisée en k-fold)	Tests de performance fonctionnelle de ML – voir 6.1.3
Interprétation erronée du résultat du test due au caractère aléatoire du processus d'entraînement	Test de performance fonctionnelle de ML – voir 6.1.3
Défaut lié au déploiement (par exemple, résultant de la génération d'une version modifiée pour une plateforme cible)	Smoke tests Tests fonctionnels de la performance de ML – voir 6.1.3 Test A/B – voir 6.1.9
Le modèle déployé n'est pas compatible avec l'environnement opérationnel	Smoke tests Test de déploiement du MLS – voir 7.1.2
Le modèle mis en œuvre n'apporte aucune amélioration par rapport au modèle actuel	Test en mode fantôme – voir 7.1.2

7.1.2 Test de déploiement des systèmes de Machine Learning

Le déploiement de MLS implique plusieurs activités de test essentielles visant à vérifier que le système basé sur l'IA fonctionne correctement et de manière fiable dans son environnement cible (par exemple, le cloud, les terminaux périphériques, les appareils mobiles). Chacun des types de test suivants permet de traiter des risques spécifiques liés au déploiement :

- Test de facilité d'installation : vérifie que le MLS peut être installé, configuré puis désinstallé sans problème dans tous les environnements pris en charge. Cela inclut le test des dépendances

système (par exemple, les pilotes GPU), la compatibilité avec les frameworks et l'exécution réussie des scripts d'installation.

- Test de restauration (rollback): vérifie la capacité du système à revenir sans problème à un état précédemment stable et opérationnel à la suite d'un déploiement dégradé ou ayant connu un échec. Ces tests peuvent porter uniquement sur le modèle ou sur l'ensemble du système (par exemple, y compris le pipeline de données). Notez que les tests de restauration doivent être effectués avant le déploiement afin de confirmer que la restauration est prête.
- Test canari – valident les nouveaux déploiements en déployant un modèle mis à jour sur un petit sous-ensemble du trafic de production (par exemple, 5 % des utilisateurs). Les métriques en temps réel, telles que la latence, la précision et les taux d'erreur, sont suivies afin de détecter les régressions avant un déploiement complet.
- Test en mode fantôme : exécutent un nouveau modèle en parallèle avec le modèle de production actuel en temps réel, en acheminant les mêmes requêtes vers les deux systèmes sans affecter les réponses en production. Ils permettent de comparer les anciens et les nouveaux modèles à l'aide de données en temps réel dans un environnement contrôlé et à faible risque, et peuvent mettre en évidence des défauts, tels que des régressions de performances et des dérives de données, avant le déploiement complet.
- Test de conversion de modèle : vérifie qu'un modèle de ML conserve une précision prédictive acceptable, un comportement cohérent et une efficacité opérationnelle (par exemple, vitesse d'inférence, utilisation de la mémoire) après avoir été converti de son format d'entraînement d'origine vers un format de déploiement adapté à l'environnement de production cible.
- Test multi-appareils : vérifie que le MLS fonctionne correctement sur l'ensemble de ses cibles de déploiement prévues, des appareils mobiles et périphériques aux serveurs cloud.
- Test de l'API : vérifient que le MLS expose des interfaces bien définies et conformes aux normes. Ils valident le traitement correct des entrées et des sorties, des messages d'erreur et des flux de travail d'intégration avec des systèmes externes, tels que les flux de données, les clients et le pipeline.

8 Liste des abréviations

Note : Les abréviations ont été conservées en anglais dans le texte et sont généralement suivies de leur traduction entre parenthèses et précédées de NDT (note de traduction).
Certains mots ont volontairement été conservés dans leur langue d'origine en raison de l'usage courant.

Abréviation	Description	Traduction française
AI	artificial intelligence	intelligence artificielle
AIaaS	AI as a service	IA en tant que service
API	application programming interface	interface de programmation d'application (API)
CL	confidence level	niveau de confiance
CNN	convolutional neural network	réseau neuronal convolutif
CPU	central processing unit	unité centrale de traitement
DL	deep learning	deep learning
DNN	deep neural network	réseau neuronal profond
EDA	exploratory data analysis	analyse exploratoire des données
FN	false negative	faux négatif
FP	false positive	faux positif
GAN	generative adversarial network	réseau adverse / antagoniste génératif
GenAI	generative AI	IA générative
GPU	graphics processing unit	unité de traitement graphique
HO	hands-on objective	objectif pratique
IAA	inter-annotator agreement	concordance entre annotateurs
kMNC	k-multisection neuron coverage	couverture des neurones k-multisection
LIME	Local interpretable model-agnostic explanations	explications locales interprétables indépendantes du modèle
LLM	large language model(s)	Grand(s) modèle(s) de langage

LO	learning objective	objectif d'apprentissage
ML	machine learning	machine learning
MLS	machine learning system(s)	système(s) de machine learning
MoE	margin of error	marge d'erreur
NBC	neuron boundary coverage	couverture des limites neuronales
NLP	natural language processing	traitement du langage naturel
RAG	retrieval-augmented generation	retrieval-augmented generation (RAG)
RNN	recurrent neural network	réseau neuronal récurrent
RT	red teaming	red teaming
SVM	support vector machine	machine à vecteurs de support
TN	true negative	vrai négatif
TP	true positive	vrai positif

9 Termes spécifiques à l'IA

Terme	Définition
précision	Mesure de performance fonctionnelle de ML utilisée pour évaluer un classificateur, qui évalue la proportion de prédictions correctes. (D'après la norme 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 qui lui sont fournies.
valeur d'activation	La sortie de la fonction d'activation d'un neurone dans un réseau neuronal.
système adaptatif basé sur l'IA	Un système basé sur l'intelligence artificielle qui adapte son comportement en fonction des changements survenant dans son environnement opérationnel.
attaque adverse	L'utilisation délibérée d'exemples adverses pour provoquer l'échec d'un modèle de ML.
IA en tant que service	Un modèle de licence et de fourniture de logiciels dans lequel l'IA et les services de développement d'IA sont hébergés de manière centralisée.
composant d'IA	Un composant offrant des fonctionnalités d'intelligence artificielle.
modèle d'IA	Un programme informatique qui implémente l'intelligence artificielle
système basé sur l'IA	Un système intégrant un ou plusieurs composants d'intelligence artificielle.
biais algorithmique	Un type de biais causé par l'algorithme de ML.
annotation	L'activité consistant à identifier des objets dans des images à l'aide de cadres de sélection afin de fournir des données étiquetées pour la classification.
intelligence artificielle	La capacité d'un système technique à acquérir, traiter, créer et mettre en œuvre des connaissances et des compétences. (ISO/IEC TR 29119-11)
association	Une technique de ML non supervisée qui identifie les relations et les dépendances entre les échantillons.
augmentation	Le processus consistant à créer de nouveaux points de données à partir d'un jeu de données existant.

modèle bayésien	Un modèle statistique qui utilise la probabilité pour représenter l'incertitude tant au niveau des données d'entrée que des résultats du modèle.
biais	La différence systématique de traitement entre certains objets, personnes ou groupes et d'autres. (D'après la norme ISO/IEC TR 24027)
big data	Jeux de données volumineux dont les caractéristiques en termes de volume, de diversité, de vitesse et/ou de variabilité exigent des technologies et des techniques spécialisées pour être traités.
technique du bootstrap	Une technique de rééchantillonnage qui consiste à prélever de manière répétée des échantillons avec remplacement à partir d'un jeu de données d'entraînement afin d'estimer les critères de performance fonctionnelle de ML pour un modèle de ML.
chatbot	Une application permettant de converser par messages texte ou grâce à la synthèse vocale.
classification	Une fonction de ML qui prédit la classe de sortie pour une entrée donnée. (D'après la norme ISO/IEC TR 29119-11)
classificateur	Modèle de ML utilisé pour la classification. Synonyme : modèle de classification
clustering (NDT regroupement)	Une fonction de 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 conceptuelle	Une évolution de la perception de la performance fonctionnelle d'un modèle de ML au fil du temps, due à des changements dans les attentes des utilisateurs, leur comportement et l'environnement opérationnel.
matrice de confusion	Une technique permettant de résumer la performance fonctionnelle de ML d'un algorithme de classification en apprentissage automatique.
réseau neuronal convolutif	Un type de modèle de deep learning conçu pour traiter des données de type matriciel, telles que des images, ce qui lui permet de reconnaître des motifs et des caractéristiques spatiales grâce à l'exploitation par couches.
attaque par injection de messages croisés	Une attaque dans laquelle des instructions malveillantes contenues dans une invite ou un segment de contexte perturbent le comportement d'un modèle d'IA au niveau d'une autre invite,

	d'un autre tour de conversation ou d'un autre segment de contexte.
acquisition de données	Le processus consistant à collecter des données pertinentes pour le problème métier que doit résoudre un modèle de ML.
biais des données	Une erreur systématique due à des données d'entraînement inexactes, incomplètes ou non représentatives, qui entraîne des résultats biaisés dans les modèles ML.
dérive des données	Une évolution de la distribution des données d'entrée au fil du temps, susceptible d'avoir un impact négatif sur la performance fonctionnelle d'un modèle de ML en production.
pipeline de traitement des données	L'implémentation d'activités de préparation des données visant à fournir des données d'entrée destinées à fournir un soutien à l'entraînement d'un algorithme de ML ou à la prédiction d'un modèle de ML.
donnée (data point)	Ensemble d'une ou plusieurs mesures comprenant une seule observation, utilisé dans le cadre d'un jeu de données.
préparation des données	Les étapes d'acquisition des données, de prétraitement des données et d'ingénierie des caractéristiques dans le workflow ML.
prétraitement des données	Les opérations de nettoyage, de transformation, d'enrichissement et d'échantillonnage des données dans le workflow ML.
jeu de données	Un ensemble de données utilisées pour l'entraînement, l'évaluation, les tests et la prédiction en ML.
arbre de décision	Un modèle de ML en forme d'arbre dont les nœuds représentent des décisions et dont les branches représentent les résultats possibles.
deep learning	ML utilisant des réseaux neuronaux profonds pour extraire automatiquement des caractéristiques et des représentations complexes à partir de vastes jeux de données.
réseau neuronal profond	Réseau neuronal composé de plusieurs couches de neurones. Synonyme : perceptron multicouche
deepfake	Contenus synthétiques, tels que des vidéos, des fichiers audio ou des images, créés ou modifiés en utilisant l'IA afin d'imiter ou de se faire passer de manière convaincante pour des personnes ou des événements réels.
prédiction des défauts	Une technique permettant de prédire les zones de l'objet de test dans lesquelles des défauts vont apparaître ou le nombre de défauts présents

déterministe	Produire le même ensemble de résultats et le même état final à partir d'un ensemble donné d'entrées et d'un état initial.
analyse d'impact discriminatoire	Une technique permettant de détecter les biais en comparant les décisions prises dans des scénarios originaux avec celles prises dans leurs versions contrefactuelles, où les attributs sensibles sont inversés.
IA en périphérie (NDT : edge AI)	Le déploiement de modèles d'IA sur des terminaux locaux, permettant un traitement en temps réel à proximité de la source des données sans avoir recours à des systèmes basés sur le cloud.
edge computing	La partie d'une architecture distribuée dans laquelle le traitement de l'information s'effectue à proximité du lieu où cette information est utilisée.
époque	Une itération de l'entraînement ML sur l'ensemble des jeux de données d'entraînement.
système expert	Un système basé sur l'intelligence artificielle permettant de résoudre des problèmes dans un domaine ou un secteur d'application particulier en tirant des conclusions à partir d'une base de connaissances développée à partir de l'expertise humaine.
IA explicable	Le domaine de recherche consacré à la compréhension des facteurs qui influencent les résultats des systèmes d'IA.
analyse exploratoire des données	Un processus interactif, visuel et fondé sur des hypothèses permettant de synthétiser, d'explorer et de comprendre les principales caractéristiques et canevas des données.
F1-Score	Une mesure de performance fonctionnelle ML utilisée pour évaluer un classificateur, qui offre un équilibre entre le rappel et la précision.
faux négatif	Une prédiction d'un modèle de ML dans laquelle le modèle prédit par erreur la classe négative.
faux positif	Une prédiction d'un modèle de ML dans laquelle le modèle classe à tort un échantillon dans la classe positive.
Caractéristique (NDT: feature)	Une caractéristique individuelle mesurable des données d'entrée utilisées pour l'entraînement par un algorithme ML et pour la prédiction par un modèle de ML.
ingénierie des caractéristiques	L'activité consistant à identifier, parmi les données brutes, les attributs qui reflètent le mieux les relations sous-jacentes devant

	apparaître dans le modèle de ML, en vue de leur utilisation dans les données d'entraînement. (ISO/IEC TR 29119-11)
tests de caractéristiques	Un type de test permettant de déterminer si l'ensemble de données utilisé pour l'entraînement d'un modèle d'IA contient un ensemble approprié de caractéristiques.
modèle fondateur	Un modèle de ML à grande échelle, formé en utilisant un vaste jeu de données grâce à l'apprentissage auto-supervisé, avec une conception polyvalente pouvant être affinée ou adaptée à un large éventail de tâches dans différents domaines.
IA de pointe	Un système d'IA polyvalent qui surpasse les capacités des systèmes d'IA les plus avancés à l'heure actuelle.
logique floue	Un type de logique fondé sur le concept de vérité partielle, représentée par des facteurs de certitude compris entre 0 et 1.
IA générale	Un type d'IA capable d'égaler les capacités cognitives humaines dans la plupart des tâches intellectuelles.
IA générative	Un type d'IA qui crée du nouveau contenu en apprenant des canevas à partir de données existantes.
processeur graphique (GPU)	Circuit intégré spécifique à une application, doté d'une conception permettant de manipuler et de modifier la mémoire afin d'accélérer la création d'images dans un tampon de trame destiné à être transmis à un dispositif d'affichage.
réalité sur le terrain	Les informations issues de l'observation directe et de la mesure, dont on sait qu'elles sont réelles ou vraies.
hyperparamètre	Paramètre utilisé soit pour contrôler l'entraînement d'un modèle de ML, soit pour définir la configuration d'un modèle de ML.
ajustement des hyperparamètres	Le processus consistant à déterminer les hyperparamètres optimaux en fonction d'objectifs spécifiques.
agent intelligent	Un programme autonome qui oriente son activité vers la réalisation d'objectifs en utilisant des observations et des actions.
concordance entre les annotateurs	Le degré de consensus ou de similitude entre les annotations réalisées par différents annotateurs sur les mêmes données. (ISO/IEC TS 12791)
grand modèle de langage (Large Language Model – LLM)	Un système d'IA générative de texte à texte formé à partir d'énormes collections de données linguistiques.

régression linéaire	Une technique statistique qui modélise la relation entre des variables en ajustant une équation linéaire aux données observées lorsque la variable cible est numérique.
système figé basé sur l'IA	Un système déterministe basé sur l'IA, doté d'un modèle fixe et immuable, qui ne modifie pas son comportement une fois déployé.
algorithme ML	Un algorithme utilisé pour créer un modèle de ML à partir d'un jeu de données d'entraînement.
framework de développement ML	Une plateforme logicielle qui fournit des outils et des bibliothèques permettant de créer, d'entraîner et de déployer des modèles ML.
fonction ML	Fonctionnalité implémentée par un modèle de ML, telle que la classification, la régression ou le regroupement.
régression ML	Type de fonction de ML qui produit une valeur de sortie numérique ou continue pour une entrée donnée. (D'après la norme ISO/IEC TR 29119-11)
MLS	Un système qui intègre un ou plusieurs modèles ML.
workflow ML	L'ensemble des activités nécessaires au développement, au déploiement et à l'exploitation d'un modèle de ML.
analyse des scores de confiance du modèle	Une technique qui identifie, à partir de la phase d'entraînement, les points de données présentant de faibles scores de confiance, lesquels sont des indicateurs de points de données mal étiquetés.
analyse des pertes du modèle	Une technique qui identifie, pendant l'entraînement, les points de données présentant des valeurs de perte élevées, ces derniers étant des indicateurs de points de données mal étiquetés.
modèle multimodal	Un modèle de ML conçu pour traiter plusieurs types de données, tels que le texte, les images, l'audio et la vidéo.
annotation multiple	Une approche consistant à comparer les données étiquetées provenant de plusieurs annotateurs.
IA étroite	IA axée sur une seule tâche bien définie afin de résoudre un problème spécifique. Synonyme : IA faible (ISO/IEC TR 29119-11)
traitement du langage naturel	Un domaine de l'informatique qui permet de lire, de comprendre et d'extraire du sens à partir des langues naturelles.
réseau neuronal	Réseau d'éléments de traitement primitifs reliés par des liaisons pondérées 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. Synonyme : réseau neuronal artificiel (ISO/IEC 2382)
processeur neuromorphique	Une conception de circuit intégré pour imiter les neurones biologiques du cerveau humain.
neurone	Un nœud dans un réseau neuronal, qui reçoit généralement plusieurs valeurs d'entrée et génère une valeur d'activation.
bruit	Une altération ou une corruption des données.
non-déterminisme	Propriété d'un système ou d'un processus dans lequel le résultat n'est pas déterminé de manière univoque par ses conditions initiales.
valeur aberrante	Une observation qui s'écarte du canevas général de la distribution des données.
surajustement	La création d'un modèle de ML qui correspond trop étroitement au jeu de données d'entraînement, ce qui donne un modèle qui s'adapte difficilement à de nouvelles données. (D'après la norme ISO/IEC TR 29119-11)
perceptron	Un réseau neuronal composé d'une seule couche et d'un seul neurone
précision	Mesure de performance fonctionnelle de ML utilisée pour évaluer un classificateur, qui évalue la proportion de résultats positifs prédits qui étaient corrects. (D'après la norme ISO/IEC TR 29119-11)
modèle pré-entraîné	Un modèle de ML qui a déjà été entraîné sur un vaste jeu de données à usage général et qui peut être réutilisé ou ajusté pour des tâches spécifiques.
probabiliste	Comportement décrit en termes de probabilités, où les résultats sont incertains et caractérisés par des degrés de vraisemblance plutôt que par une certitude.
forêt aléatoire	Technologie ML par ensembles pour la classification, la régression et d'autres tâches qui consistent à construire et à exécuter de nombreux arbres de décision, puis à fournir soit la classe la plus fréquente, soit la prédiction moyenne des différents arbres.

rappel	Mesure de performance fonctionnelle de ML utilisée pour évaluer un classificateur, qui évalue la proportion de cas positifs réels qui ont été correctement prédits. Synonyme : sensibilité (d'après la norme ISO/IEC TR 29119-11)
réseau neuronal récurrent	Un type de modèle de deep learning conçu pour traiter des données séquentielles, ce qui lui permet de reconnaître des schémas et des relations de dépendance au fil du temps.
apprentissage par renforcement	Une approche dans laquelle un modèle de ML, appelé « agent », apprend en interagissant avec son environnement via une boucle de rétroaction « essai-récompense » afin d'atteindre des objectifs spécifiques.
retrieval-augmented generation	Une technique de ML dans laquelle un système d'IA générative améliore dynamiquement ses résultats en récupérant des informations externes pertinentes afin de compléter ses connaissances internes.
fonction de récompense	Une fonction qui définit le succès de l'apprentissage par renforcement.
piratage de récompense	Action menée par un agent intelligent visant à maximiser sa fonction de récompense au détriment de la réalisation de l'objectif initial. (D'après la norme ISO/IEC TR 29119-11)
algorithme de recherche	Algorithme qui explore systématiquement un sous-ensemble de tous les états ou structures possibles jusqu'à ce que l'état ou la structure cible soit atteint(e). (D'après ISO/IEC TR 29119-11)
système d'auto-apprentissage	Système adaptatif qui modifie son comportement en fonction de l'apprentissage par essais et erreurs. (D'après la norme ISO/IEC TR 29119-11)
attributs sensibles	Caractéristiques qui définissent les groupes et les personnes bénéficiant d'une protection juridique ou éthique et qui doivent être prises en compte afin d'éviter toute discrimination.
analyse des sentiments	Utilisation d'un modèle de traitement du langage naturel et de machine learning pour identifier et classer la tonalité émotionnelle exprimée dans un texte.

échantillonnage stratifié	Une technique permettant de vérifier si un échantillon est représentatif, au niveau proportionnel, des différentes sous-populations composant la population globale.
super IA	Un type d'IA qui dépasse de loin les capacités humaines.
apprentissage supervisé	Une approche pour l'entraînement d'un modèle de ML en utilisant un jeu de données étiquetées.
machine à vecteurs de support	Un algorithme de ML supervisé qui trouve l'hyperplan optimal entre les points de données pour des tâches de classification ou de régression.
singularité technologique	Un moment dans le futur où les progrès technologiques ne pourront plus être maîtrisés par l'homme. (D'après la norme ISO/IEC TR 29119-11)
température	Un paramètre permettant de contrôler le caractère aléatoire des résultats générés par GenAI : les valeurs faibles produisent des résultats plus prévisibles, tandis que les valeurs élevées donnent lieu à des résultats plus créatifs.
problème avec l'oracle de test	Le défi consistant à déterminer si un test a passé ou échoué pour un ensemble donné d'entrées de test et d'états.
jeu de données d'entraînement	Un jeu de données utilisé pour entraîner un modèle de ML.
“transformer”	Une architecture de réseau neuronal qui traite des données séquentielles afin de détecter les dépendances à long terme, et qui alimente les tâches de traitement du langage naturel, la vision par ordinateur et les applications multimodales.
vrai négatif	Une prédiction dans laquelle le modèle identifie correctement la classe négative.
vrai positif	Une prédiction dans laquelle le modèle identifie correctement la classe positive.
sous-ajustement	La création d'un modèle de ML qui ne reflète pas la tendance sous-jacente du jeu de données d'entraînement, ce qui donne lieu à un modèle produisant des prédictions inexactes. (d'après la norme ISO/IEC TR 29119-11)
apprentissage non supervisé	Une approche pour l'entraînement d'un modèle de ML en utilisant un jeu de données non étiqueté.
jeu de données de validation	Un jeu de données utilisé pour évaluer un modèle de ML déjà formé, dans le but de l'ajuster.

architecture von Neumann	Une architecture informatique composée de cinq composants principaux : la mémoire, l'unité centrale de traitement, l'unité de commande, les entrées et les sorties.
poids	Variable interne d'une connexion entre neurones dans un réseau neuronal, qui influence la manière dont celui-ci calcule ses sorties et qui évolue au fur et à mesure que le réseau neuronal est entraîné.

10 Références

10.1 Normes

- **ISO/IEC/IEEE 12207** (2017), Systems and software engineering — Software life cycle processes
- **ISO/IEC 2382** (2015), Information technology — Vocabulary
- **ISO/IEC 22989** (2022), Information technology — Artificial intelligence — Artificial intelligence concepts and terminology
- **ISO/IEC TR 24027** (2021) Information technology — Artificial intelligence (AI) — Bias in AI systems and AI aided decision making
- **ISO/IEC 25010** (2023), Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) - Product quality model
- **ISO/IEC 25059** (2023), Software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality model for AI systems
- **ISO 26262-6** (2018), Road vehicles — Functional safety — Part 6: Product development at the software level
- **ISO/IEC TR 29119-11** (2020), Software and systems engineering — Software testing — Part 11: Guidelines on the testing of AI-based systems
- **ISO/IEC TS 42119-2** (2025) Artificial intelligence — Testing of AI – Part 2: Overview of testing AI systems
- **ISO/IEC 42119 series** (in development) Artificial intelligence — Testing of AI

10.2 Documents ISTQB®

[CTFL] ISTQB® Certified Tester Foundation Level v4.0.1, https://istqb.org/wp-content/uploads/2024/11/ISTQB_CTFL_Syllabus_v4.0.1.pdf (dernier accès le 29.07.2025)

CTFL-FR ISTQB® Testeur certifié niveaux Fondation v4.0, https://cftl.fr/wp-content/uploads/2023/11/ISTQB-Syllabus-Niveau-Fondation-v4.0-Francais.pdf

[CT-GenAI] ISTQB® Certified Tester – Testing with Generative AI (CT-GenAI), https://istqb.org/certifications/gen-ai/ (dernier accès le 19.12.2025)

CT-GenAI FR ISTQB® Testeur certifié – Tester avec l'IA Générative (CT-GenAI), https://cftl.fr/wp-content/uploads/2025/08/CT-GenAI-Syllabus-v1.0-FR.pdf

10.3 Références du glossaire

Référence concernant la terminologie utilisée dans ce syllabus:

- Glossaire ISTQB® https://glossary.istqb.org/

10.4 Livres, articles et pages web

[COV_REF] An Overview of Structural Coverage Metrics for Testing Neural Networks, Usman et al, Aug 2022, arXiv:2208.03407 (dernier accès le 29.07.2025)

[DATA_DOC] Gebru, T., Morgenstern, J., Vecchione, B., et al. (2021). Datasheets for Datasets. Communications of the ACM, 64(12), 86-92, https://dl.acm.org/doi/pdf/10.1145/3502158 (dernier accès le 07.10.2025)

[EU AI Act] EU Artificial Intelligence Act, July 2024, https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=OJ:L_202401689 (dernier accès le 30.07.2025)

[MODEL_DOC] Mitchell, M., Wu, S., Varma, R., et al. (2019). Model Cards for Model Reporting. Proceedings of the Conference on Fairness, Accountability, and Transparency, https://dl.acm.org/doi/10.1145/3287560.3287596 (dernier accès le 07.10.2025)

[OECD AI] OECD Recommendation of the Council on Artificial Intelligence, May 2024, Organisation for Economic Co-operation and Development (OECD), https://legalinstruments.oecd.org/en/instruments/oecd-legal-0449 (dernier accès le 07.10.2025)

[UN Gov AI] Governing AI for Humanity: Final Report, September 2024, United Nations, https://www.un.org/sites/un2.un.org/files/governing_ai_for_humanity_final_report_en.pdf (dernier accès le 29.07.2025)

[STATS] An Introduction to Statistical Learning: With Applications in R (2nd ed.) by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, Springer.

11 Marques déposées

ISTQB® est une marque déposée de International Software Testing Qualifications Board

12 Appendice A – Objectifs d'apprentissage / Niveau cognitif des connaissances

Les objectifs d'apprentissage spécifiques au syllabus sont indiqués au début de chaque chapitre. Chaque thème du syllabus sera abordé en fonction de l'objectif d'apprentissage qui lui est associé. Les objectifs d'apprentissage commencent par un verbe d'action correspondant au niveau cognitif de la compétence, comme indiqué ci-dessous.

Niveau 1 : Se souvenir (K1)

Le candidat sera capable de mémoriser, de reconnaître et de rappeler un terme ou un concept.

Verbes d'action : se souvenir, reconnaître.

Exemples
Se souvenir des concepts de la pyramide des tests.
Reconnaître les objectifs typiques des tests.

Niveau 2: Comprendre (K2)

Le candidat est capable de sélectionner les raisons ou les explications correspondant aux affirmations liées au sujet, et peut résumer, comparer, classer et donner des exemples illustrant le concept évalué.

Verbes d'action: classer, comparer, différencier, distinguer, expliquer, donner des exemples, interpréter, résumer

Exemples	**Notes**
Classer les outils de test en fonction de leur objectif et des activités de test qu'ils prennent en charge.	
Comparer les différents niveaux de test.	Peut être utilisé pour rechercher des similitudes, des différences ou les deux.
Différencier le test et le débogage.	Recherche les différences entre les concepts.
Distinguer les risques projet et les risques produit.	Permet de classer séparément deux concepts (ou plus).
Expliquer l'influence du contexte sur le processus de test.	

Exemples	Notes
Donner des exemples illustrant pourquoi il est nécessaire de tester.	
Déterminer la cause racine des défauts à partir d'un profil donné de défaillances.	
Résumer les étapes du processus de revue des produits d'activités.	

Niveau 3: Appliquer (K3)

Le candidat est capable de mettre en œuvre une procédure lorsqu'il est confronté à une tâche qu'il connaît bien, ou de choisir la procédure correcte et de l'appliquer à un contexte donné.

Verbes d'action: Appliquer, implémenter, préparer, utiliser

Exemples	Notes
Appliquer l'analyse des valeurs limites pour définir des cas de test à partir des exigences fournies.	Doit faire référence à une procédure, une technique, un processus, etc.
Implémenter des méthodes de collecte de métriques afin de répondre aux exigences techniques et de management.	
Préparer des tests de facilité d'installation pour les applications mobiles.	
Utiliser la traçabilité pour suivre l'avancement des tests et s'assurer de leur complétude et de leur cohérence avec les objectifs, la stratégie et le plan de test.	Ce terme peut être utilisé dans un objectif d'apprentissage (LO) visant à ce que le candidat soit capable d'utiliser une technique ou une procédure. Similaire à « appliquer ».

Niveau 4: Analyser (K4)

Le candidat est capable de décomposer les informations relatives à une procédure ou à une technique en leurs éléments constitutifs afin d'en faciliter la compréhension, et sait faire la distinction entre les faits et les déductions. Cela s'applique généralement à l'analyse d'un document, d'un logiciel ou d'un projet, ainsi qu'à la proposition de mesures appropriées pour résoudre un problème ou mener à bien une tâche.

Verbes d'action: Analyser, décomposer, structurer, prioriser, sélectionner.

Exemples	Notes
Analyser la situation d'un projet donné afin de déterminer quelles techniques de test boîte noire ou	Ne peut être évalué qu'en association avec un objectif mesurable de l'analyse.

Exemples	Notes
basées sur l'expérience doivent être appliquées pour atteindre des objectifs spécifiques.	
Prioriser les cas de test d'une suite de tests donnée en vue de leur exécution, en fonction des risques produit.	Doit se présenter sous la forme « Analyser xxxx pour xxxx » (ou similaire).
Sélectionner les niveaux et les types de test appropriés pour vérifier un ensemble donné d'exigences.	

Références

(Pour les niveaux cognitifs des objectifs d'apprentissage)

Anderson, L. W. and Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon

13 Appendice B – Matrice de traçabilité des objectifs métier et des objectifs d'apprentissage

Cette section présente la traçabilité entre les objectifs métier et les objectifs d'apprentissage de la certification « Testeur certifié - Test d'IA ». La première partie du tableau indique le nombre d'objectifs d'apprentissage par objectif métier, tandis que la seconde partie précise quels objectifs d'apprentissage sont associés à chaque objectif métier.

Objectifs métier : tests d'IA			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8
BO1	Comprendre l'état actuel de l'IA, y compris l'IA générative.		8							
BO2	Expérimenter la mise en œuvre et les tests de modèles de machine learning.			10						
BO3	Comprendre le fonctionnement et le test de réseaux neuronaux simples.				2					
BO4	Comprendre les caractéristiques de qualité spécifiques à l'IA définies par la norme ISO/IEC 25059.					3				
BO5	Calculer et interpréter les mesures de performance fonctionnelle ML pour les modèles de ML.						1			

Objectifs métier : tests d'IA			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8
BO6	Reconnaître la portée et l'importance des deux niveaux de test spécifiques à l'évaluation des systèmes de machine learning.							3		
BO7	Contribuer à l'élaboration d'une stratégie de test efficace pour un système de machine learning.								5	
BO8	Concevoir et exécuter des cas de test pour les systèmes de machine learning.									14
LO Unique	Objectif d'apprentissage	Niveau K								
1	Introduction à l'intelligence artificielle									
1.1	Introduction à l'IA									
AI-1.1.1	Distinguer les systèmes basés sur l'IA des systèmes conventionnels	K2	X							
AI-1.1.2	Distinguer l'IA étroite, l'IA générale et la super-IA	K2	X							
AI-1.1.3	Expliquer les différents types de technologies d'IA	K2	X							

Objectifs métier : tests d'IA			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8
AI-1.1.4	Expliquer l'IA générative	K2	X							
AI-1.1.5	Comparer les choix disponibles en matière de matériel pour implémenter des systèmes de machine learning	K2	X							
AI-1.1.6	Comparer les options pour le développement et l'hébergement de modèles d'IA	K2	X							
AI-1.1.7	Résumer les fonctionnalités offertes par les frameworks de développement de machine learning	K2	X							
AI-1.1.8	Expliquer comment les réglementations et les normes influencent le développement et les tests des systèmes basés sur l'IA	K2	X							
2	Caractéristiques de qualité pour les systèmes basés sur l'IA									
2.1	Caractéristiques de qualité des systèmes basés sur l'IA									
AI-2.1.1	Classifier les comportements des systèmes basés sur l'IA selon les caractéristiques de qualité définies dans la norme ISO/IEC 25059	K2				X				

Objectifs métier : tests d'IA			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8
AI-2.1.2	Expliquer les considérations particulières qui se posent lorsque l'IA est utilisée dans des systèmes liés à la sûreté	K2				X				
2.2	Critères d'acceptation pour les systèmes basés sur l'IA									
AI-2.2.1	Donner des exemples de critères d'acceptation pour les systèmes basés sur l'IA	K2				X				
3	Machine learning									

Objectifs métier : tests d'IA			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8
3.1	Introduction au machine learning									
AI-3.1.1	Distinguer les différentes formes de machine learning	K2		X						
AI-3.1.2	Résumer le workflow utilisé pour créer un système de machine learning	K2		X						
AI-3.1.4	Résumer l'utilisation des modèles pré-entraînés, du réglage fin et de la retrieval-augmented generation (RAG)	K2		X						
3.2	Données pour le machine learning									
AI-3.2.1	Expliquer les activités liées à la préparation des données	K2		X						
AI-3.2.3	Comparer l'utilisation des ensembles de données d'entraînement, de validation et de test dans le développement d'un modèle de ML	K2		X						
3.3	Mesures de performance fonctionnelle de ML pour la classification									

Objectifs métier : tests d'IA			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8
AI-3.3.1	Calculer des mesures de performance fonctionnelle de ML à partir d'un ensemble donné de données de matrice de confusion	K3					X			
3.4	Réseaux neuronaux									
AI-3.4.1	Expliquer la structure et le fonctionnement d'un réseau neuronal profond	K2			X					
AI-3.4.3	Décrire les différentes mesures de couverture pour les réseaux neuronaux	K2			X					
4	Test des systèmes basés sur l'IA									
4.1	Introduction au test des systèmes basés sur l'IA									
AI-4.1.1	Comparer la testabilité des systèmes basés sur l'IA de type « figé » et « adaptatif »	K2		X						
AI-4.1.2	Expliquer pourquoi une approche statistique est souvent nécessaire lors des tests de systèmes basés sur l'IA	K2		X						
AI-4.1.3	Expliquer les défis et les solutions liés aux oracles de test pour les systèmes basés sur l'IA	K2		X						

Objectifs métier : tests d'IA			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8
4.2	Tester l'IA générative et les LLMs									
AI-4.2.1	Expliquer comment tester l'IA générative	K2		X						
AI-4.2.2	Implémenter des tests de « red teaming » pour les systèmes d'IA générative	K3		X						

Objectifs métier : tests d'IA			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8
4.3	Niveaux de test et systèmes de ML									
AI-4.3.1	Résumer les niveaux de test utilisés pour développer des systèmes de ML	K2							X	
AI-4.3.2	Expliquer comment le test basé sur les risques est appliqué aux systèmes de ML	K2							X	
5	Test des données d'entrée pour les systèmes de ML									
5.1	Test des données d'entrée pour les systèmes de ML									
AI-5.1.1	Donner des exemples d'approches de test utilisées pour l'atténuation des risques liés aux données d'entrée d'un système de ML	K2						X	X	
AI-5.1.2	Expliquer comment tester la présence de biais	K2								X
AI-5.1.3	Résumer les différentes formes de tests du pipeline de données	K2								X
AI-5.1.4	Expliquer comment tester la représentativité des données	K2								X

Objectifs métier : tests d'IA			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8
AI-5.1.5	Appliquer les tests de contraintes sur les jeux de données	K3								X
AI-5.1.6	Expliquer les tests de vérification de l'exactitude des étiquettes	K2								X
6	Test des modèles pour les systèmes de Machine Learning									
6.1	Test des modèles pour les systèmes de Machine Learning									
AI-6.1.1	Donner des exemples d'approches de test utilisées pour l'atténuation des risques liés aux modèles ML	K2						X	X	
AI-6.1.2	Expliquer l'objectif et l'intérêt de la revue de la documentation des modèles ML	K2								X
AI-6.1.3	Expliquer comment sont réalisés les tests de performance de ML pour les systèmes d'apprentissage automatique probabilistes	K2								X
AI-6.1.4	Résumer les tests adverses des systèmes ML	K2								X
AI-6.1.5	Utiliser les tests métamorphiques pour dériver des cas de test pour un scénario donné	K3								X

Objectifs métier : tests d'IA			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8
AI-6.1.7	Expliquer comment les tests de dérive sont utilisés dans les systèmes de machine learning en exploitation	K2								X
AI-6.1.8	Expliquer comment le surajustement et le sous-ajustement sont détectés par des tests	K2								X
AI-6.1.9	Expliquer comment les tests A/B sont utilisés dans le contexte des systèmes de machine learning	K2								X
AI-6.1.10	Expliquer comment les tests dos-à-dos sont utilisés dans le contexte des systèmes de machine learning	K2								X
7	Test du développement de machine learning									
7.1	Test du développement de machine learning									
AI-7.1.1	Donner des exemples d'approches de test utilisées pour l'atténuation des risques liés au développement de machine learning	K2						X	X	
AI-7.1.2	Expliquer les différentes formes de tests de déploiement des systèmes ML	K2								X

14 Appendice C – Notes de release

La version 2.0 de l'ISTQB CT-AI constitue une mise à jour majeure et une refonte de la version 1.0. Compte tenu de l'évolution rapide des technologies d'IA, une mise à jour majeure s'imposait. La version 2.0 met clairement l'accent sur les tests des systèmes basés sur l'IA. Suite à la publication du syllabus ISTQB CT «Tester avec l'IA générative », le chapitre consacré aux tests avec l'IA a été entièrement supprimé. Cette version majeure a apporté les modifications suivantes:

- Introduction plus concise à l'IA en général
- Intégration de l'IA générative et des tests liés à l'IA générative
- Synthèse des caractéristiques de qualité de l'IA et des défis qui y sont associés
- Réduction de l'accent mis sur les indicateurs de performance du ML
- Affinement des niveaux de test : test des données d'entrée et test des modèles ML
- Affinement des types de test
- Suppression des tests avec l'IA
- Exclusion des environnements de test pour les systèmes basés sur l'IA Durée minimale de formation requise réduite de 4 à 3 jours
-

15 Index

Tous les termes relatifs aux tests sont définis dans le glossaire de l'ISTQB® (https://glossary.istqb.org/).