Certified Tester AI Testing Syllabus

Versão 2.0

International Software Testing Qualifications Board

Fornecido por

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

BSTQB®
CT-AI 2.0

Aviso de Direitos Autorais

Histórico de revisões

Versão	Data	Observações
1.0	01/10/2021	Lançamento para GA
2.0 Alpha	15/08/2025	Revisão da versão Alpha
2.0 Beta	07/01/2026	Revisão da versão Beta
2.0	17/04/2026	Lançamento para o público geral

Índice

Agradecimentos

A Assembleia Geral do ISTQB® publicou formalmente este documento em 17 de abril de 2026.

Força-tarefa de AI do ISTQB® (v2.0): Klaudia Dussa-Zieger (presidente), Stuart Reid, Vipul Kocher, Qin Liu, Jaroslaw Hryszko, Kyle Alexander Siemens e Werner Henschelchen

As seguintes pessoas participaram da revisão e dos comentários sobre o syllabus do curso: 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, Mario Winter

Força-tarefa de AI do ISTQB® (v1.0): Klaudia Dussa-Zieger (presidente), Vipul Kocher, Qin Liu, Stuart Reid, Adam Leon Smith, Kyle Alexander Siemens e Werner Henschelchen

A equipe agradece aos autores dos três syllabi que contribuíram:

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

A equipe agradece aos Grupos de Trabalho de Exames, Glossário e Marketing pelo apoio durante todo o desenvolvimento do syllabus, e aos Conselhos de Membros por suas sugestões e contribuições.

O BSTQB® agradece aos voluntários de seu grupo de trabalho de traduções (WG Traduções) pelo empenho e esforço na tradução deste material: George Fialkovitz, Irene Nagase, Osmar Higashi, Paula de Oliveira, Stênio Viveiros, Thiago Cesar Andrade.

O BSTQB® também agradece aos ISTQB® Accredited Training Providers no Brasil que foram convidados a contribuir na revisão da tradução. No momento deste trabalho os seguintes provedores estavam credenciados: Conecteseaqui (conecteseaqui.com.br), Exseed (www.exseed.com.br), Iterasys (www.iterasys.com.br).

O BSTQB® também agradece às empresas brasileiras credenciadas no ISTQB® Partner Program no Brasil que foram convidados a contribuir na revisão da tradução. No momento deste trabalho as seguintes empresas estavam credenciadas: Cast Group (www.castgroup.com.br), Deltapoint (www.deltapoint.com.br), Keeggo (www.keeggo.com), Venturus (www.venturus.org.br).

0 Introdução

0.1 Objetivo deste Syllabus

Este syllabus constitui a base para a Certificação ISTQB® de Testador de AI. O ISTQB® disponibiliza este syllabus da seguinte forma:

- Aos conselhos membros, para tradução para o idioma local e para credenciar provedores de treinamento. Os conselhos membros podem adaptar o syllabus às suas necessidades específicas de idioma e modificar as referências para alinhá-las com suas publicações locais.
- Aos organismos de certificação, para elaborar questões de exame em seu idioma local, adaptadas aos objetivos de aprendizagem deste syllabus.
- Aos provedores de treinamento, para produzir material didático e determinar métodos de ensino adequados.
- Aos candidatos à certificação, para se prepararem para o exame de certificação (seja como parte de um curso de treinamento ou de forma independente).
- À comunidade internacional de engenharia de software e sistemas, para promover a profissão de testes de software e sistemas, e como base para livros e artigos.

0.2 O Teste de AI para Testadores Certificados

O Certified Tester AI Testing (CT-AI) foi projetado para indivíduos envolvidos no teste de sistemas baseados em AI. Isso inclui profissionais em diversas funções, como testadores, analistas de testes, analistas de dados, engenheiros de testes, consultores de testes, gerentes de testes, testadores de aceite de usuário e desenvolvedores de software. Esta certificação também é adequada para profissionais que buscam uma compreensão fundamental sobre o teste de sistemas baseados em AI incluindo gerentes de projeto, gerentes da qualidade, gerentes de desenvolvimento de software, analistas de negócios, membros de equipes de operações, diretores de TI e consultores de gestão.

0.3 Trajetória profissional para testadores

O esquema ISTQB® oferece suporte a profissionais de testes em todas as fases de suas carreiras. Indivíduos que obtiverem a certificação ISTQB® Certified Tester Foundation Level também podem se interessar pelos Níveis Avançados (Test Analyst, Technical Test Analyst, and Test Management) e, posteriormente, pelo Nível Especialista (Test Management or Improving the Test Process). A vertente de Especialista oferece um aprofundamento em abordagens e atividades de teste específicas, por exemplo, Agile Testing, Test Automation, AI Testing, Testing with Generative AI, or Mobile App Testing, ou em conhecimentos de testes em grupo para determinados setores, como Automotive ou Gaming. Visite www.istqb.org para obter as informações mais recentes sobre o Esquema Certified Tester do ISTQB®.

0.4 Resultados de Negócios

Esta seção lista os Resultados de Negócios esperados de um candidato que tenha obtido a certificação CT-AI.

Um Testador Certificado em Testes de AI pode:

BO1	Compreender o estado atual da AI incluindo a AI generativa.
BO2	Experimentar a implementação e o teste de modelos de machine learning.
BO3	Compreender o funcionamento e o teste de redes neurais simples.

BO4	Compreender as características específicas de qualidade da AI definidas pela norma ISO/IEC 25059.
BO5	Calcular e interpretar métricas de performance funcional do ML para modelos de machine learning.
BO6	Reconhecer o escopo e a importância dos dois níveis de teste específicos para o teste de sistemas de machine learning.
BO7	Contribuir para o desenvolvimento de uma estratégia de teste eficaz para um sistema de machine learning.
BO8	Projetar e executar casos de teste para sistemas de machine learning.

0.5 Objetivos de Aprendizagem, Objetivos Práticos e Nível Cognitivo de Conhecimento

Os Objetivos de Aprendizagem (LO) apoiam os resultados de negócios e são usados para criar os exames CT-AI. Os níveis específicos dos objetivos de aprendizagem são apresentados no início de cada capítulo, e classificados da seguinte forma:

- K1: Lembrar
- K2: Compreender
- K3: Aplicar
- K4: Analisar

Mais detalhes e exemplos de objetivos de aprendizagem são fornecidos no Apêndice A.

Para todos os termos listados como palavras-chave logo abaixo dos títulos dos capítulos, o nome e a definição corretos do glossário do ISTQB® ou do Capítulo 9 devem ser memorizados (K1), mesmo que não sejam explicitamente mencionados no objetivo de aprendizagem.

Os Objetivos Práticos (HO) concentram-se na aplicação prática dos Objetivos de Aprendizagem e são apresentados no início de cada capítulo. O nível de um HO é classificado da seguinte forma:

- H0: Pode incluir uma demonstração ao vivo de um exercício ou um vídeo gravado. Como o aluno não realiza isso, não se trata estritamente de um exercício.
- H1: Exercício guiado. Os alunos seguem uma sequência de etapas realizadas pelo instrutor.
- H2: Exercício com dicas. O aluno recebe um exercício com dicas relevantes para permitir que o exercício seja resolvido dentro do prazo determinado.

0.6 O exame Certified Tester AI Testing

O exame CT-AI será baseado nos Objetivos de Aprendizagem descritos neste syllabus. Todas as seções do syllabus são objeto de avaliação, exceto a Introdução, os Objetivos Práticos, as Referências e os Anexos. A resposta às questões do exame pode exigir o uso de material proveniente de mais de uma seção deste syllabus. Normas e livros são incluídos como referências, mas seu conteúdo não será avaliado além do que está resumido no próprio syllabus.

Consulte o documento Exam Structures and Rules para o CT-AI v2.0 para obter mais detalhes.

O principal critério de admissão para qualquer pessoa interessada em fazer o exame CT-AI é possuir a certificação ISTQB® Certified Tester Foundation Level [CTFL].

Recomenda-se fortemente que os candidatos também:

- Tenham uma formação mínima em desenvolvimento de software ou em testes de software, como seis meses de experiência como testador de aceite de sistemas ou de usuários, cientista de dados ou desenvolvedor de software.
- Faça um curso credenciado pelas normas do ISTQB® (por um dos conselhos membros reconhecidos pelo ISTQB®).

0.7 Credenciamento

Um Conselho Membro do ISTQB® pode credenciar provedores de treinamento cujo material didático siga este syllabus. Os provedores de treinamento devem obter as diretrizes de credenciamento junto ao Conselho Membro ou órgão responsável pela credenciação. Um curso credenciado é reconhecido como estando em conformidade com este syllabus e permite que um exame do ISTQB® seja incluído no curso.

As diretrizes de credenciamento para este syllabus seguem as Diretrizes Gerais de Credenciamento publicadas pelo Processes Management and Compliance Working Group.

0.8 Tratamento de Normas

Organizações internacionais de padronização, como IEEE e ISO, emitiram normas associadas a características de qualidade e testes de software. O objetivo dessas referências é fornecer uma estrutura (como nas referências às normas ISO/IEC 25059 e ISO/IEC 25010 relativas a um modelo de qualidade para sistemas baseados em AI) ou fornecer uma fonte de informações adicionais, caso o leitor deseje. Observe que os syllabi utilizam os documentos normativos como referência. Os documentos de normas não se destinam a ser objeto de avaliação.

0.9 Nível de detalhe

O nível de detalhamento deste syllabus permite a realização de cursos e exames com padrões internacionais consistentes. Para atingir esse objetivo, o syllabus consiste em:

- Objetivos instrucionais gerais que descrevem a intenção do syllabus do CT-AI
- Uma lista de palavras-chave que os alunos devem ser capazes de recordar
- Objetivos de aprendizagem para cada área de conhecimento, descrevendo o resultado cognitivo a ser alcançado
- Uma descrição dos conceitos-chave, incluindo referências a fontes como literatura ou normas reconhecidas

O conteúdo do syllabus não descreve toda a área de conhecimento de testes de software; ele reflete o nível de detalhe a ser abordado nos cursos de treinamento CT-AI. Ele se concentra na introdução dos conceitos básicos de AI e, em particular, de machine learning (ML), e em como os sistemas baseados nessas tecnologias podem ser testados.

O syllabus utiliza a terminologia (ou seja, o nome e o significado) dos termos usados em testes de software e garantia da qualidade de acordo com o Glossário do ISTQB®.

0.10 Como este syllabus está organizado

Existem sete capítulos com conteúdo passível de avaliação. O título de nível superior de cada capítulo especifica a duração do capítulo; o tempo não é fornecido no nível do capítulo. *Para cursos de treinamento credenciados, o syllabus exige um mínimo de* 19,5 horas de instrução, distribuídas pelos sete capítulos da seguinte forma:

- Capítulo 1: 120 minutos - Introdução à Inteligência Artificial (AI)

  - Compreender as principais diferenças entre sistemas baseados em AI e sistemas convencionais, e explorar o espectro de capacidades da AI, que vai da AI estreita à super AI.
  - Adquirir uma compreensão básica das tecnologias de AI incluindo AI generativa (GenAI), e das estruturas de hardware, hospedagem e desenvolvimento utilizadas para implementar sistemas de machine learning (MLS).
  - Aprenda como regulamentações e normas influenciam o desenvolvimento e o teste de soluções baseadas em AI.
- Capítulo 2: 45 minutos - Características de qualidade para sistemas baseados em AI
  - Conheça as características de qualidade específicas dos sistemas baseados em AI incluindo aquelas definidas na ISO/IEC 25059, e compreenda as considerações relacionadas à segurança ao usar AI em sistemas críticos.
  - Explore como definir critérios de aceite adequados, adaptados ao comportamento e à performance exclusivos das soluções baseadas em AI.
- Capítulo 3: 375 minutos - Machine learning
  - Compreenda os tipos de ML, as etapas principais no fluxo de trabalho de desenvolvimento de ML e como modelos pré-treinados, ajuste fino e geração aumentada por recuperação contribuem para os sistemas modernos baseados em AI.
  - Aprenda sobre a preparação de dados, as funções dos conjuntos de dados de treinamento, validação e teste, e como estes influenciam o desenvolvimento e a performance dos modelos ML.
  - Explore redes neurais, incluindo sua estrutura e medidas de cobertura, e ganhe experiência prática com métricas de performance e o uso de uma matriz de confusão.
- Capítulo 4: 195 minutos - Teste de sistemas baseados em AI
  - Compreenda os desafios únicos de teste dos sistemas baseados em AI incluindo diferenças na testabilidade entre sistemas fixos e adaptativos, a necessidade de testes estatísticos e as dificuldades em definir oráculos de teste.
  - Aprenda a testar a AI de geração (GenAI) e os Large Language Model (LLM), utilizando técnicas como o red teaming e os testes exploratórios para a AI realizar tarefas de teste.
  - Explore estratégias de teste para MLS, abrangendo vários níveis de teste e a aplicação de testes baseados em riscos.
- Capítulo 5: 180 minutos - Teste de dados de entrada para sistemas de machine learning
  - Aprenda a testar e validar dados de entrada para MLS, incluindo técnicas para detectar viés, verificar a correção de rótulos, avaliar a representatividade dos dados e testar o pipeline de dados.
- Capítulo 6: 225 minutos - Testes de modelos para sistemas de machine learning
  - Descubra abordagens de teste para mitigar riscos em modelos ML, incluindo revisões de documentação, testes de performance funcional do ML e detecção de overfitting, underfitting e desvio.
  - Aprenda abordagens avançadas de teste, incluindo testes contraditórios, testes A/B, testes back-to-back e o uso de ataques para revelar fraquezas do modelo.
  - Adquira compreensão prática de testes metamórficos, incluindo como derivar e aplicar casos de teste onde os oráculos de teste tradicionais são insuficientes.

- Capítulo 7: 30 minutos - Teste de desenvolvimento de machine learning
    - Aprenda abordagens de teste e estratégias de teste para mitigar riscos durante o desenvolvimento e a implantação de MLS, a fim de garantir um comportamento robusto do sistema em ambientes de produção.

1 Introdução à Inteligência Artificial – 120 minutos

Palavras-chave

Nenhuma

Palavras-chave específicas de AI

Sistema baseado em AI, inteligência artificial, AI geral, machine learning, estrutura de desenvolvimento de machine learning, AI restrita, super AI

Objetivos de aprendizagem para o Capítulo 1:

1.1 Introdução à AI

AI-1.1.1 (K2) Distinguir entre sistemas baseados em AI e sistemas convencionais

AI-1.1.2 (K2) Distinguir entre AI restrita, AI geral e super AI

AI-1.1.3 (K2) Explicar os diferentes tipos de tecnologias de AI

AI-1.1.4 (K2) Explicar o que é AI generativa

AI-1.1.5 (K2) Comparar as opções de hardware disponíveis para implementar sistemas de machine learning

AI-1.1.6 (K2) Comparar as opções para o desenvolvimento e hospedagem de modelos de AI

AI-1.1.7 (K2) Resumir as funcionalidades oferecidas pelas estruturas de desenvolvimento de ML

AI-1.1.8 (K2) Explicar como regulamentações e normas afetam o desenvolvimento e o teste de sistemas baseados em AI

1.1 Introdução à AI

Este capítulo apresenta as principais diferenças entre os sistemas convencionais e os baseados em IA, com foco em suas abordagens de projeto, adaptabilidade e explicabilidade. Ele descreve o espectro de capacidades da IA, que vai da IA restrita à IA geral e à super-IA, e destaca tecnologias fundamentais como o ML, o deep learning e a IA gerativa (GenAI). O capítulo descreve o hardware e os ambientes de desenvolvimento comuns para sistemas baseados em IA, bem como as estruturas de ML mais populares. Por fim, ele aborda normas regulatórias e técnicas essenciais que orientam o desenvolvimento e a implantação responsáveis da IA em diversos domínios.

1.1.1 Sistemas baseados em AI e convencionais

Os sistemas de computação convencionais são normalmente programados usando linguagens imperativas, nas quais desenvolvedores humanos definem explicitamente instruções passo a passo, incluindo construções como instruções if-then-else e loops. Essa abordagem determinística ajuda a tornar o comportamento do sistema previsível e com transparência, facilitando para os humanos compreenderem como as entradas produzem diferentes saídas. Em contraste, a maioria dos sistemas baseados em AI — especialmente aqueles que utilizam ML — não segue regras predefinidas. Em vez disso, eles analisam padrões nos dados para determinar como responder a novas entradas. Por exemplo, um sistema de reconhecimento de imagens baseado em AI treinado para identificar gatos não depende de regras explicitamente codificadas. Em vez disso, ele aprende a partir de um conjunto de dados de imagens de gatos, extraindo padrões que posteriormente aplica a imagens não vistas para classificá-las com precisão.

Uma diferença fundamental entre os sistemas convencionais e os baseados em AI é a forma como abordam a resolução de problemas. Muitos sistemas baseados em AI dependem de raciocínio probabilístico, inferência estatística e reconhecimento de padrões para gerar resultados. Isso permite que os modelos de AI lidem com formas complexas de incerteza e ambiguidade de maneira mais efetiva, resultando em saídas que nem sempre são previsíveis.

Um dos principais desafios dos sistemas baseados em AI é a explicabilidade. Muitos modelos de AI, particularmente arquiteturas de deep learning, podem conter bilhões de parâmetros, tornando seu funcionamento interno difícil de ser interpretado por seres humanos. Essa natureza de “caixa-preta” suscita preocupações em domínios críticos, como saúde, finanças, defesa e transporte, onde é crucial compreender por que um modelo de AI tomou uma determinada decisão. Alcançar transparência (ver 2.1.1) e explicabilidade na tomada de decisões orientada por AI tornou-se um foco crítico na regulamentação da AI.

Outra distinção significativa é a adaptabilidade. Os sistemas convencionais são estáticos e normalmente requerem atualizações manuais para incorporar novos conhecimentos ou responder a mudanças ambientais. Os sistemas baseados em AI, em contrapartida, podem ser auto aprendizes, melhorando continuamente sua performance à medida que encontram novos dados. Essa adaptabilidade torna a AI particularmente poderosa em ambientes dinâmicos. A adaptabilidade também requer monitoramento contínuo para manter o alinhamento constante com os requisitos essenciais.

1.1.2 AI restrita, AI geral e super AI

A AI restrita, também conhecida como AI fraca, é projetada para realizar tarefas específicas e representa todos os sistemas de AI implantados em uso atualmente. Os sistemas baseados em AI restrita operam dentro de um domínio limitado e podem ser altamente eficientes na resolução de problemas especializados, como reconhecimento de imagens, processamento de fala e tradução de idiomas. No entanto, eles carecem da capacidade de generalizar além das funções que aprenderam. Por exemplo, um sistema baseado em AI que se destaca no reconhecimento de rostos não pode realizar tradução de idiomas, a menos que seja explicitamente retreinado para isso. A AI de Fronteira, um subconjunto da AI restrita, representa a forma mais avançada desses sistemas, ampliando os limites das capacidades

atuais com modelos de AI Geral. A AI de Fronteira inclui sistemas de grande escala com capacidades de tomada de decisão altamente autônomas; no entanto, eles permanecem específicos para determinadas tarefas e ainda não alcançaram a versatilidade da AI geral.

A AI geral, também conhecida como AI forte, refere-se a um sistema baseado em AI que possui a capacidade de realizar a maioria das tarefas intelectuais que um ser humano é capaz de realizar. A AI geral seria capaz de compreender, aprender e aplicar conhecimento em uma ampla gama de tarefas sem a necessidade de ser retreinada para cada nova tarefa. Esse tipo de AI exibiria raciocínio e adaptabilidade semelhantes aos humanos, capaz de resolver problemas desconhecidos em vários domínios, da mesma forma que os humanos fazem. Apesar do progresso significativo na AI, nenhum sistema baseado em AI hoje possui inteligência geral.

A super AI (ou superinteligência artificial) é a forma de AI na qual um sistema baseado em AI se aprimora continuamente sem a necessidade de intervenção ou controle humano. Para que a super AI seja possível, o acesso à Internet não é estritamente requerido, mas tal acesso poderia expandir significativamente suas capacidades e influência. Ela superaria a inteligência humana e a AI geral, o que muitos acreditam que representaria um risco existencial para a humanidade. O ponto em que os sistemas baseados em AI fazem a transição da AI geral para a super AI, caso isso ocorra, é comumente conhecido como singularidade tecnológica.

1.1.3 Diferentes tipos de tecnologias de AI

A Inteligência Artificial abrange uma gama de tecnologias, cada uma adequada a tarefas e desafios específicos. Um dos principais ramos da AI é o ML, que permite que os sistemas aprendam com dados e construam modelos sem programação explícita. Enquanto alguns modelos de ML podem se adaptar e melhorar continuamente com novos dados ao longo de sua vida útil, outros operam com base no conhecimento que aprenderam inicialmente e requerem retreinamento explícito para atualizar suas capacidades. O ML inclui várias abordagens:

- A aprendizagem supervisionada utiliza dados rotulados e algoritmos, como regressão linear e árvores de decisão, para tarefas como previsão e classificação.
- A aprendizagem não supervisionada descobre padrões em dados não rotulados por meio de técnicas como o agrupamento (usando algoritmos de agrupamento).
- A aprendizagem por reforço permite que agentes inteligentes aprendam comportamentos ótimos por meio de interações de tentativa e erro com seu ambiente.

As principais tecnologias de ML incluem redes neurais, modelos bayesianos, máquinas de vetores de suporte (SVM) e florestas aleatórias. Consulte o Capítulo 3 para obter mais informações sobre ML.

O Deep Learning (DL), um subconjunto dos métodos de ML, utiliza redes neurais profundas para resolver problemas complexos. Por exemplo:

- As redes neurais convolucionais (CNN) são altamente efetivas para reconhecimento de imagens e detecção de objetos.
- As redes neurais recorrentes (RNN) são especializadas no processamento de dados sequenciais, como texto ou séries temporais.
- Os transformadores lidam com dependências de longo alcance em sequências, alimentando modelos para processamento de linguagem natural e transformadores de visão para imagens.

A GenAI se baseia nessas tecnologias para criar conteúdos, incluindo texto, imagens e áudio (ver 1.1.4). Isso é impulsionado por modelos como o LLM, que combinam redes neurais profundas (DNN) com processamento de linguagem natural (NLP) para analisar e gerar linguagem semelhante à humana.

Outras tecnologias especializadas de AI incluem:

- NLP para análise de linguagem, incluindo tarefas como análise de sentimentos e tradução automática.

- Visão computacional para análise de dados visuais, dando suporte a aplicações como reconhecimento facial e robótica.
- Lógica difusa para raciocínio em condições de incerteza.
- Algoritmos de busca para resolver problemas de otimização, como navegação e tomada de decisões estratégicas.
- Sistemas de raciocínio baseados em regras, ou sistemas especialistas, para apoio à decisão estruturada.

Embora a integração completa dessas tecnologias de AI ainda seja limitada, novos desenvolvimentos, como o LLM, demonstram o potencial de combinar diferentes tecnologias de AI em sistemas unificados e inteligentes. A AI agentiva amplia essas tecnologias por meio de agentes autônomos que planejam, raciocinam e agem de forma independente para atingir objetivos em ambientes dinâmicos.

1.1.4 AI Gerativa

A AI generativa (GenAI) refere-se a sistemas baseados em AI especializados na criação de novos conteúdos, como texto, imagens, vídeo, música ou dados complexos, enquanto muitos também oferecem suporte à classificação e previsão. Esses sistemas aprendem com vastas quantidades de dados para produzir resultados que se assemelham aos seus dados de treinamento, possibilitando uma ampla gama de aplicações criativas e práticas.

As principais tecnologias por trás da GenAI incluem redes adversariais generativas (GANs), modelos de difusão e transformadores. As GANs utilizam duas redes neurais em competição para criar dados sintéticos altamente realistas. Os modelos de difusão geram conteúdo adicionando e removendo gradualmente ruído dos dados, resultando em resultados de alta qualidade. Os modelos transformadores, que sustentam o LLM, utilizam mecanismos de autoatenção para gerar texto coerente e contextualmente relevante, e estão sendo cada vez mais adaptados para tarefas multimodais.

Além de seus fundamentos técnicos, os sistemas de GenAI levantam preocupações sociais e éticas significativas. O uso indevido é uma grande preocupação: esses modelos podem ser explorados para criar deepfakes, espalhar desinformação ou gerar conteúdo fraudulento convincente, minando assim a confiança na mídia digital e no discurso público. A facilidade de produzir conteúdo sintético amplifica os riscos relacionados à privacidade, segurança e manipulação, mas pode trazer benefícios notáveis e abre oportunidades para inovação em entretenimento, marketing, educação e pesquisa.

O impacto no emprego também é notável, particularmente entre as profissões de colarinho branco. À medida que a GenAI automatiza tarefas como redação, design, programação e até mesmo documentação jurídica ou médica, cresce o debate sobre a potencial substituição de postos de trabalho. Embora essas tecnologias possam impulsionar a produtividade e fomentar novas oportunidades criativas, elas também podem levar a perturbações na força de trabalho e exigir uma requalificação generalizada.

A sustentabilidade é outra questão urgente. O treinamento e a execução de grandes modelos de GenAI envolvem recursos computacionais substanciais, resultando em alto consumo de energia e uma pegada de carbono significativa. Esse impacto ambiental tem gerado apelos por projetos de modelos mais eficientes e infraestrutura mais ecológica.

A maioria das ferramentas práticas de GenAI hoje se baseia em modelos de base, que são então ajustados para aplicações específicas. O campo também está avançando em direção a modelos multimodais capazes de processar e gerar conteúdo em texto, imagens e áudio, possibilitando sistemas baseados em AI mais ricos e com maior flexibilidade. Estruturas regulatórias, como a Lei de AI da UE [EU AI Act], estão surgindo para orientar o desenvolvimento e o uso responsáveis dessas tecnologias.

1.1.5 Hardware para sistemas de machine learning

Uma variedade de hardware é utilizada para MLS. Diferentes tipos de hardware podem ser usados para treinamento e inferência. Por exemplo, um modelo de reconhecimento de fala pode ser executado em um smartphone de baixo custo, embora possa ser necessário acessar o poder da computação em nuvem para treiná-lo.

O ML normalmente se beneficia de hardware que ofereça suporte ao seguinte:

- A capacidade de trabalhar com grandes estruturas de dados.
- Processamento massivamente paralelo (concorrência), por exemplo, para suportar multiplicação de matrizes.
- Aritmética de baixa precisão (quantização): essa abordagem usa menos bits para o cálculo (por exemplo, 4 bits em vez de 32 bits), resultando em processamento mais rápido, menor consumo de energia, chips menores e mais econômicos e requisitos de largura de banda reduzidos.

Unidades de processamento central (CPU) de uso geral oferecem suporte para operações complexas com alta precisão que normalmente não são requisitos para aplicações de ML, mas geralmente fornecem apenas alguns núcleos. Como resultado, sua arquitetura é menos eficiente para treinar e executar os modelos ML em comparação com as unidades de processamento gráfico (GPU), que possuem milhares de núcleos e são projetadas para realizar processamento gráfico massivamente paralelo, mas relativamente simples. Consequentemente, as GPUs normalmente superam as CPUs em aplicações de ML, mesmo que as CPUs geralmente operem em velocidades de clock mais altas. Para trabalhos de ML em pequena escala, as GPUs geralmente oferecem a melhor opção.

Alguns hardwares são projetados especificamente para AI, como os circuitos integrados de aplicação específica (ASIC) e os dispositivos System-on-a-Chip. Essas soluções específicas para AI apresentam múltiplos núcleos, gerenciamento de dados especializado e a capacidade de realizar processamento na memória. Elas são mais adequadas para computação de ponta, enquanto o treinamento do modelo ML é realizado na nuvem usando hardware especializado.

Arquiteturas de hardware específicas para AI continuam sendo desenvolvidas. Isso inclui processadores neuromórficos, que não utilizam a arquitetura tradicional de von Neumann, mas sim projetos inspirados no cérebro que imitam estruturas neuronais.

1.1.6 Desenvolvimento e hospedagem de modelos de AI

Os sistemas baseados em AI podem ser adquiridos de fornecedores terceirizados ou desenvolvidos internamente por uma organização. Esses sistemas geralmente se baseiam em modelos pré-treinados e podem ser implantados no local ou na nuvem, com os próprios modelos hospedados no local ou na nuvem, sendo que as opções baseadas na nuvem costumam ser acessadas como um serviço (AIaaS). Sistemas baseados em AI de terceiros geralmente vêm como modelos pré-treinados ou como AI como Serviço (AIaaS,), permitindo uma implantação mais rápida e um tempo de comercialização mais curto. Alternativamente, sistemas baseados em AI privados (instalações locais ou configurações personalizadas na nuvem) podem ser mais bem adaptados a requisitos específicos, mas seu desenvolvimento provavelmente exigirá habilidades especializadas, seja por meio de especialistas internos ou equipes terceirizadas.

O desenvolvimento local (codificação e treinamento localmente) permite controle direto e privacidade. Modelos pequenos, como árvores de decisão ou redes neurais compactas, podem ser desenvolvidos em computadores pessoais, enquanto modelos de médio porte podem exigir GPUs dedicadas. Para modelos de grande escala, clusters de servidores locais de alta performance tornam-se necessários, com seus custos associados de energia, refrigeração e hardware.

O desenvolvimento em nuvem oferece flexibilidade significativa. As nuvens públicas, em particular, fornecem ambientes pré-configurados com preços pré-pagos, o que limita o investimento inicial em hardware e permite fácil escalabilidade. Em contraste, as nuvens privadas podem oferecer segurança e

privacidade aprimoradas para aplicativos que exigem isso, mas esse controle requer um maior investimento inicial em infraestrutura.

Muitas organizações adotam abordagens híbridas, incluindo o desenvolvimento de protótipos localmente antes de escalar para a infraestrutura em nuvem, mantendo componentes confidenciais no local, como a preparação de dados privados, e aproveitando os recursos da nuvem para tarefas que exigem grande capacidade de computação.

Modelos de AI podem ser hospedados em vários ambientes, variando de configurações locais a plataformas baseadas em nuvem. A hospedagem local envolve a execução de modelos menores em computadores pessoais ou smartphones, oferecendo privacidade e eliminando custos de licenciamento de nuvem, embora isso ofereça recursos de hardware limitados. Para modelos de AI maiores, as organizações podem estabelecer servidores dedicados, o que requer um investimento inicial significativo, mas oferece maior controle.

A hospedagem em nuvem de modelos de AI pode ocorrer em nuvens públicas ou privadas. Os serviços de nuvem pública oferecem acesso escalável a uma infraestrutura de robustez e poder, eliminando preocupações com manutenção e tornando-os ideais para cargas de trabalho flutuantes. As nuvens privadas oferecem benefícios semelhantes, com segurança aprimorada e opções de personalização, e são gerenciadas internamente ou por meio de provedores dedicados, embora a um custo mais alto.

Abordagens híbridas combinam esses métodos, permitindo que as organizações executem algumas operações localmente enquanto aproveitam a elasticidade da nuvem para tarefas intensivas.

As soluções ideais de desenvolvimento e hospedagem, que normalmente são decididas separadamente, dependem de fatores como tamanho do modelo, complexidade, requisitos de performance, restrições orçamentárias, considerações de segurança e privacidade do dado, necessidades de implantação e requisitos regulatórios. Algumas organizações adotam estratégias em várias camadas para equilibrar eficiência e controle.

1.1.7 Estruturas de desenvolvimento de machine learning

As framework de desenvolvimento de ML oferecem um conjunto de ferramentas para a criação e o treinamento de modelos ML. As funcionalidades típicas oferecidas por essas estruturas incluem:

- Manipulação de dados: auxiliam no carregamento, pré-processamento e gerenciamento dos dados utilizados para treinar e testar o modelo. Isso pode envolver a limpeza, formatação e transformação dos dados em um formato adequado para o modelo escolhido.
- Construção de modelos: Essas estruturas oferecem bibliotecas de algoritmos de ML e ferramentas para projetar a arquitetura do modelo ML construído. Isso inclui especificar o tipo de modelo (por exemplo, rede neural, árvore de decisão), o número de camadas e conexões, e as operações matemáticas realizadas dentro do modelo.
- Treinamento e otimização: As estruturas fornecem algoritmos que ajustam iterativamente os parâmetros internos do modelo com base nos dados de treinamento e no resultado desejado. O objetivo é otimizar a performance do modelo na realização da tarefa desejada (por exemplo, classificação, regressão de ML). Algumas estruturas podem oferecer suporte a treinamento distribuído e aprimorar ou ajustar modelos pré-treinados.
- Avaliação: Elas oferecem ferramentas para avaliar a performance do modelo treinado em dados não vistos. Isso pode envolver a medição de exatidão, precisão e recall para tarefas de classificação, ou taxas de erro para tarefas de regressão de ML (ver 3.1.1).
- Implantação: Algumas estruturas oferecem recursos para implantar o modelo treinado para uso no mundo real. Isso pode envolver a conversão do modelo para um formato adequado para integração com aplicativos web, dispositivos móveis, dispositivos de borda ou sistemas embarcados.

Essas estruturas podem operar em diferentes níveis de abstração. Algumas oferecem uma Application Programming Interface (API) de nível mais baixo, proporcionando aos desenvolvedores maior controle sobre a construção do modelo, mas exigindo mais conhecimento em programação. Outras oferecem uma API de nível mais alto, simplificando a criação do modelo, mas oferecendo menos opções de personalização.

Diferentes frameworks podem se concentrar em diferentes domínios de aplicação. Alguns são de uso geral e oferecem suporte a uma ampla gama de áreas de aplicação. Em contrapartida, outros são mais especializados, concentrando-se em áreas específicas, como reconhecimento de imagem, reconhecimento de fala e tradução de idiomas.

A seleção da estrutura mais adequada pode depender de vários fatores, tais como:

- a área de aplicação;
- a necessidade de uma interface do usuário amigável para prototipagem rápida;
- a configurabilidade para modelos complexos;
- a experiência dos usuários;
- considerações de implantação, já que algumas estruturas são mais adequadas para ambientes com recursos limitados;
- nível de suporte (da comunidade);
- maturidade do ecossistema.

1.1.8 Regulamentações e normas para AI

As regulamentações e normas sobre AI são cruciais para o desenvolvimento, a implantação e o uso responsáveis da AI. O objetivo principal é fomentar a confiança na AI e ajudar a promover a concretização de seus benefícios, ao mesmo tempo em que se mitigam os possíveis danos. Idealmente, a conformidade com essas regulamentações e normas deve garantir que os sistemas baseados em AI sejam seguros, justos, com transparência, sustentáveis, responsáveis, éticos e utilizados de forma responsável.

Internacionalmente, os Princípios da OCDE sobre AI [OECD AI] e o relatório da ONU Governing AI for Humanity [UN Gov AI] servem como instrumentos de soft law influentes que promovem um entendimento comum sobre a gestão responsável da AI. Eles atuam como uma bússola para governos nacionais e organizações na formulação de suas próprias estratégias de AI. Esses princípios enfatizam a AI centrada no ser humano, considerações éticas e a importância da cooperação internacional.

A Lei de AI da UE representa um marco regulatório, demonstrando uma abordagem baseada no risco para a regulamentação da AI. Ao categorizar os sistemas baseados em AI por risco, de mínimo a inaceitável, as regulamentações são adaptadas de acordo com isso. Sistemas de alto risco, particularmente aqueles que afetam direitos fundamentais ou a segurança, enfrentam requisitos rigorosos que abrangem testes rigorosos, governança de dados e supervisão humana. As penalidades financeiras substanciais por não conformidade, baseadas em uma porcentagem do faturamento global, ressaltam o compromisso da UE com a fiscalização. Em contraste, muitos países fora da UE estão adotando uma abordagem mais permissiva, favorecendo regulamentações menos restritivas para incentivar a inovação.

Normas técnicas, desenvolvidas por organizações como a ISO e o IEEE, são cruciais para traduzir aspirações de alto nível em implementações práticas. Elas fornecem especificações técnicas concretas e melhores práticas, preenchendo a lacuna entre política e prática. Por exemplo, a ISO/IEC TR 29119-11 fornece orientações detalhadas sobre o teste de sistemas baseados em AI, um elemento crítico para demonstrar conformidade regulatória. Enquanto isso, a série ISO/IEC 42119 está sendo desenvolvida para abranger vários aspectos dos testes de sistemas baseados em AI. Além disso, regulamentações

específicas para cada setor estão surgindo em áreas como saúde e finanças, reconhecendo os riscos únicos que a AI representa nesses domínios.

Para navegar efetivamente por esse cenário em evolução da governança da AI, o diálogo e a colaboração contínuos são fundamentais. Governos, indústria, academia e sociedade civil devem se engajar ativamente para alcançar uma abordagem harmonizada e de efetividade para a governança da AI em todo o mundo. Além disso, dada a natureza dinâmica da AI, as regulamentações e normas devem ser submetidas a revisões e atualizações regulares para permanecerem relevantes e eficazes na orientação do desenvolvimento, implantação e uso responsáveis da AI.

2 Características de Qualidade para Sistemas Baseados em AI – 45 minutos

Palavras-chave

Adaptabilidade funcional, correção funcional da AI, capacidade de intervenção, robustez da AI, segurança, mitigação de riscos sociais e éticos, transparência, controlabilidade do usuário

Palavras-chave específicas da AI

Nenhuma

Objetivos de aprendizagem para o Capítulo 2:

2.1 Características de Qualidade para Sistemas Baseados em AI

AI-2.1.1 (K2) Classificar comportamentos de sistemas baseados em AI de acordo com as características de qualidade definidas na norma ISO/IEC 25059

AI-2.1.2 (K2) Explicar as considerações especiais que surgem quando a AI é utilizada em sistemas relacionados à segurança

2.2 Critérios de Aceite para Sistemas Baseados em AI

AI-2.2.1 (K2) Apresentar exemplos de critérios de aceite para sistemas baseados em AI

2.1 Características de Qualidade para Sistemas Baseados em AI

Esta seção aborda as características de qualidade específicas da AI descritas na norma ISO/IEC 25059, que amplia os modelos tradicionais de qualidade de software para abordar aspectos únicos dos sistemas baseados em AI. Ela apresenta características novas e adaptadas, incluindo correção funcional da AI, adaptabilidade funcional, controlabilidade do usuário, transparência, robustez da AI e capacidade de intervenção, juntamente com a mitigação de riscos sociais e éticos. Os principais desafios de segurança da AI, como especificações vagas, não determinismo, autoaprendizagem, explicabilidade limitada e padrões em evolução, também são abordados, com ênfase em seu impacto nos testes e na regulamentação.

2.1.1 Características de qualidade específicas da AI

A ISO/IEC 25059 amplia o modelo de qualidade da ISO/IEC 25010 para abordar considerações específicas da AI. Essa ampliação avalia os sistemas baseados em AI a partir de duas perspectivas: qualidade do produto e qualidade em uso. Do ponto de vista dos testes, essas características de qualidade influenciam diretamente como os objetivos do teste são definidos, como os critérios de aceite são formulados e como os resultados dos testes são interpretados para sistemas baseados em AI. As características novas e modificadas, em comparação com a ISO/IEC 25010, incluem:

- Correção funcional da AI (qualidade do produto): os sistemas baseados em AI, especialmente aqueles que utilizam machine learning probabilístico, não podem garantir precisão perfeita. Como é esperada uma certa taxa de erro nos resultados da AI, o conceito de correção funcional foi ajustado de acordo. A norma ISO/IEC 25059 avalia a correção funcional considerando tanto os resultados corretos quanto os incorretos e definindo limites aceitáveis para resultados incorretos, refletindo a variabilidade inerente aos resultados dos sistemas baseados em AI (ver 3.3).
- Adaptabilidade funcional (qualidade do produto): uma nova subcaracterística da adequação funcional. A capacidade do sistema baseado em AI de se adaptar autonomamente a mudanças em seu ambiente operacional após sua implantação.
- Controlabilidade do usuário (qualidade do produto): uma nova subcaracterística da capacidade de interação (note-se que a capacidade de interação é, por si só, um novo termo que substitui a usabilidade na versão de 2023 da ISO/IEC 25010). Uma propriedade de um sistema baseado em AI que permite que um ser humano ou outro agente externo possa intervir em seu funcionamento de maneira oportuna.
- Transparência (qualidade do produto e qualidade de uso): uma nova subcaracterística da capacidade de interação e uma nova subcaracterística da satisfação. Refere-se ao grau em que informações apropriadas sobre o sistema baseado em AI são comunicadas às partes interessadas (ver 6.1.2).
- Robustez da AI (qualidade do produto): uma nova subcaracterística da confiabilidade. Descreve a capacidade de um sistema baseado em AI de manter seu nível de correção funcional da AI independentemente das circunstâncias, tais como a presença de entradas de dados tendenciosas, adversas ou inválidas, interferência externa, condições ambientais adversas e uso indevido por parte do operador.
- Intervenibilidade (qualidade do produto): uma nova subcaracterística da segurança. O grau em que um operador pode intervir no funcionamento de um sistema baseado em AI de maneira oportuna para prevenir danos ou riscos.
- Mitigação de riscos sociais e éticos (qualidade de uso): uma nova subcaracterística de “Ausência de risco”. Considera diversas áreas para mitigar riscos sociais e éticos, incluindo contabilização, equidade e não discriminação, responsabilidade profissional, promoção de valores humanos,

privacidade, segurança e proteção, controle humano da tecnologia, envolvimento e desenvolvimento comunitário, modelagem centrada em humano, respeito pelo Estado de Direito, respeito pelas normas internacionais de comportamento, sustentabilidade ambiental e práticas trabalhistas.

2.1.2 AI e Segurança

Os sistemas relacionados à segurança têm o potencial de causar ferimentos ou danos a pessoas, bens ou ao meio ambiente. Desenvolver e testar sistemas relacionados à segurança que não utilizam AI pode exigir muito esforço, mas é viável; no entanto, para sistemas baseados em AI, há vários desafios adicionais:

- Especificações: Em sistemas tradicionais relacionados à segurança, os requisitos são definidos para o sistema completo e refinados até que o desenvolvedor possa transformá-los em código. Os requisitos para muitos sistemas baseados em AI geralmente começam com objetivos vagos e são então fornecidos implicitamente por meio dos dados de treinamento que codificam padrões, regras e objetivos, sem formalizar totalmente todos os detalhes antecipadamente. Isso pode significar que a rastreabilidade necessária, desde os requisitos até a implementação, é inadequada para sistemas baseados em AI.
- Não determinismo: Essa característica de muitos sistemas baseados em AI torna inerentemente desafiador garantir o comportamento preciso desses sistemas. Mesmo modelos rigorosamente testados podem apresentar comportamentos inesperados devido a fatores como geração de números aleatórios ou pequenas variações nos valores de entrada.
- Autoaprendizagem: Testes rigorosos são utilizados para demonstrar a integridade de segurança de um sistema antes da implantação. Para sistemas baseados em AI com autoaprendizagem, isso é prejudicado, pois o comportamento do sistema se afasta progressivamente do comportamento originalmente testado. Gerenciar como o modelo aprende e os dados que ele utiliza pode, às vezes, ajudar a evitar o surgimento de novos comportamentos problemáticos. Alternativamente, proteções de segurança podem ser implementadas para ajudar a impedir que o modelo aprenda ou tome decisões que possam comprometer a segurança (por exemplo, um componente de moderação de conteúdo para filtrar prompts).
- Explicabilidade e Transparência: Para sistemas relacionados à segurança, é essencial compreender como e por que o sistema toma decisões. No entanto, os processos de tomada de decisão de sistemas baseados em AI muitas vezes não são transparentes. Técnicas de AI explicável, como LIME (explicações locais interpretáveis e independentes de modelo), podem fornecer insights sobre o raciocínio do sistema baseado em AI; no entanto, elas não estão amplamente disponíveis e podem comprometer a performance do sistema.
- Regulamentações em evolução: O panorama regulatório para sistemas baseados em AI relacionados à segurança está em constante evolução. Atualmente, o uso da AI não está incluído em normas internacionais maduras de segurança funcional, e algumas dessas normas chegam a proibir seu uso em tais sistemas. A Lei de AI da UE [EU AI Act] (ver 1.1.8) classifica os sistemas de AI utilizados como componentes de segurança (como na aviação, em dispositivos médicos ou no setor automotivo) como de alto risco e impõe requisitos rigorosos para seu desenvolvimento e testes.

2.2 Critérios de Aceite para Sistemas Baseados em AI

Esta seção descreve os critérios de aceite relacionados às características de qualidade especificadas na norma ISO/IEC 25059, bem como à segurança. Para sistemas baseados em AI, os critérios de aceite frequentemente precisam ser estatísticos, probabilísticos ou baseados em limites, em vez de binários, o que introduz desafios adicionais aos testes.

2.2.1 Critérios de aceite para sistemas baseados em AI

Ao avaliar a qualidade de um sistema baseado em AI, é essencial considerar tanto as características de qualidade funcionais quanto as não funcionais. Isso ajuda a confirmar que o sistema baseado em AI funciona conforme o esperado e satisfaz requisitos de qualidade mais amplos. As normas ISO/IEC 25010 e ISO/IEC 25059 fornecem uma estrutura abrangente para definir a qualidade do software. Nesta seção, o foco está nos critérios de aceite associados às características de qualidade específicas da AI (ou seja, aquelas definidas na ISO/IEC 25059) e da segurança (ver 2.1.2).

A tabela a seguir lista exemplos de critérios de aceite para segurança e cada uma das características de qualidade definidas na norma ISO/IEC 25059.

Característica	Exemplos de critérios de aceite
Correção funcional da AI (ver 3.3)	• Precisão de 95% para um sistema de reconhecimento de imagens. • Recuperação de 90% para um sistema de previsão de defeitos.
Adaptabilidade funcional	• Um máximo de 20 segundos para que o sistema de gerenciamento do motor se adapte ao ultrapassar um limite de altitude especificado. • Um serviço de streaming de vídeo deve ajustar sua página inicial para recomendar pelo menos 40% de documentários após um usuário assistir a três documentários completos em uma única sessão.
Controlabilidade do usuário	• Um supervisor pode assumir o controle de um drone autônomo em até 0,5 segundo quando ele envia um sinal de socorro devido à perda de sua localização GPS. • O sistema de controle agrícola notifica o agricultor quando a performance visual do sensor se degrada em mais de 30%, permitindo a intervenção manual imediata; ele se desativa totalmente se a degradação exceder 50% sem resposta do usuário.
Transparência	• São fornecidas informações suficientes sobre o modelo ML de terceiros e a proveniência de seus dados de treinamento para atender aos requisitos da norma corporativa pertinente. • O painel de controle e a API do sistema devem fornecer um endpoint que retorne o identificador de versão exclusivo do modelo de previsão atualmente implantado e um link para a documentação correspondente.
Robustez da AI	• O tempo de resposta das previsões do sistema de alerta de penetração de segurança baseado em AI permanece abaixo de 1 segundo quando o acesso ao banco de dados central de vulnerabilidades é interrompido por 30 segundos. • O dispositivo de AI de ponta deve transitar automaticamente para um modo de inferência de menor fidelidade e consumo reduzido de energia (em vez de travar) quando sua temperatura operacional interna exceder 85 °C por um período contínuo de 10 segundos.
Intervenibilidade	• Se um robô violar sua zona de segurança, a linha de produção é capaz de ser desligada em até 0,5 segundos após o início do desligamento. • Para evitar possíveis apagões, o sistema de gerenciamento da rede elétrica deve fornecer uma janela de confirmação de 30 segundos, durante a qual um

	engenheiro pode vetar qualquer ação proposta pela AI classificada como “crítica” antes que ela seja executada automaticamente.
Mitigação de riscos sociais e éticos	• O sistema automatizado de determinação de penas não discrimina entre grupos raciais com base na métrica de equidade especificada. • O chatbot deve passar por uma avaliação interna de “red teaming” com uma pontuação de 95% ou mais, demonstrando sua recusa em gerar conteúdo que promova violência, automutilação ou discurso de ódio.
Segurança	• Os componentes não baseados em AI do sistema de controle de direção baseado em AI estão em conformidade com a norma ISO 26262-6 no nível ASIL (nível de integridade da segurança automotiva) C. • 100% das relações entre entradas e saídas do modelo ML no sistema de controle da usina nuclear podem ser mapeadas com uma precisão média não inferior a 99,9% por uma ferramenta de explicabilidade. • Sinais de controle que excedam os limites de segurança especificados em mais de 10% são analisados e regulados em até 0,15 segundos após serem detectados pelo subsistema de monitoramento de segurança.

3 Machine Learning – 375 minutos

Palavras-chave

Cobertura de neurônios com k seções, critérios de desempenho funcional ML, métricas de performance funcional ML, modelo ML, cobertura dos limites dos neurônios, cobertura de neurônios, perceptron

Palavras-chave específicas de AI

Associação, classificação, agrupamento, preparação de dados, machine learning, algoritmo de machine learning, estrutura de desenvolvimento de machine learning, fluxo de trabalho de machine learning, modelo pré-treinado, regressão de machine learning, aprendizado por reforço, aprendizado supervisionado, aprendizado não supervisionado

Objetivos de aprendizagem para o Capítulo 3:

3.1 Introdução ao Machine Learning

AI-3.1.1 (K2) Distinguir entre as diferentes formas de ML

AI-3.1.2 (K2) Resumir o fluxo de trabalho utilizado para criar um sistema de ML

HO-3.1.3 (H2) Criar um modelo ML

AI-3.1.4 (K2) Resumir o uso de modelos pré-treinados, ajuste fino e geração aumentada por recuperação

3.2 Dados para Machine Learning

AI-3.2.1 (K2) Explicar as atividades relacionadas à preparação de dados

HO-3.2.2 (H2) Realizar a preparação de dados para apoiar a criação de um modelo ML

AI-3.2.3 (K2) Comparar o uso de conjuntos de dados de treinamento, validação e teste no desenvolvimento de um modelo ML

3.3 Métricas de Performance Funcional do ML para Classificação

AI-3.3.1 (K3) Calcular métricas de performance funcional ML a partir de um determinado conjunto de dados da matriz de confusão

HO-3.3.2 (H2) Avaliar um modelo ML utilizando métricas de performance funcional ML selecionadas

HO-3.3.3 (H2) Mostrar o impacto de diferentes combinações de modelos ML e conjuntos de dados no treinamento e no comportamento dos modelos

3.4 Redes neurais

AI-3.4.1 (K2) Explicar a estrutura e o funcionamento de uma rede neural profunda

HO-3.4.2 (H1) Experimentar a implementação de um perceptron

AI-3.4.3 (K2) Descrever as diferentes medições de cobertura para redes neurais

3.1 Introdução ao Machine Learning

Esta seção apresenta as principais categorias de algoritmos de ML: aprendizado supervisionado, não supervisionado e por reforço, distinguindo seus respectivos tipos de problemas e aplicações típicas. Ela descreve o fluxo de trabalho padrão para o desenvolvimento de modelos ML, desde a definição de objetivos e preparação de dados até o treinamento, avaliação e implantação de modelos, com ênfase na iteração e integração de sistemas. Aspectos práticos, como a criação prática de modelos, o uso de modelos pré-treinados, o ajuste fino e a geração aumentada por recuperação também são abordados, destacando abordagens para adaptar e aprimorar com eficiência modelos de AI para novas tarefas, ao mesmo tempo em que se gerenciam limitações herdadas.

Compreender esse fluxo de trabalho é essencial para os testadores, pois diferentes atividades de teste se aplicam em diferentes estágios, e as falhas frequentemente se originam em etapas anteriores, como a preparação dos dados de teste ou a seleção de modelos.

3.1.1 Diferentes formas de machine learning

Os algoritmos de ML são categorizados em aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço.

Na aprendizagem supervisionada, os algoritmos treinam modelos usando dados rotulados, em que cada conjunto de entradas tem um rótulo de saída correspondente (por exemplo, imagens rotuladas como “cachorro” ou “gato”). O modelo aprende a mapear entradas para saídas identificando padrões nos dados de treinamento. A aprendizagem supervisionada é tipicamente dividida em:

- Classificação: Isso envolve atribuir entradas a classes predefinidas, como classificar e-mails como spam ou não, ou o reconhecimento de objetos em imagens.
- Regressão de ML: Isso envolve a previsão de valores numéricos contínuos, como estimar a idade de uma pessoa com base em dados de estilo de vida ou prever preços de ações.

  Observe que o termo regressão de ML, quando usado no contexto de ML, difere de seu uso em outros syllabi do ISTQB®, onde regressão descreve o problema de modificações de software que causam defeitos relacionados a alterações.

Na aprendizagem não supervisionada, o algoritmo treina modelos usando dados não rotulados, inferindo padrões ou estruturas sem rótulos de saída explícitos. O modelo agrupa entradas semelhantes com base em características compartilhadas. A aprendizagem não supervisionada é tipicamente categorizada em:

- Agrupamento: Isso envolve agrupar pontos de dados com base em semelhanças, como segmentar clientes em diferentes grupos para marketing direcionado.
- Associação: Isso envolve identificar relações ou dependências entre atributos de dados, como descobrir padrões no comportamento de compra dos clientes para recomendar produtos.

Na aprendizagem por reforço, o sistema baseado em AI (um “agente inteligente”) aprende ao interagir com seu ambiente. O agente recebe feedback positivo (recompensas) ou negativo (penalidades) com base no resultado de suas ações, o que lhe permite aprender com a experiência, em vez de com um conjunto de dados. Os desafios da aprendizagem por reforço incluem a configuração do ambiente, o projeto da função de recompensa e a seleção da melhor estratégia para atingir o objetivo desejado. As aplicações incluem robótica, veículos autônomos e sistemas adaptativos, como chatbots.

Cada abordagem de ML aborda diferentes tipos de problemas, sendo que a escolha depende da disponibilidade dos dados e da tarefa específica em questão.

3.1.2 Fluxo de trabalho de machine learning

As atividades no fluxo de trabalho de ML, mostradas na Figura 1, são:

Compreender os objetivos

O propósito do modelo ML é compreendido e acordado pelas partes interessadas para verificar o alinhamento com as prioridades de negócios. Critérios de aceite (incluindo métricas de performance funcional ML – consulte3.3) são definidos para o modelo desenvolvido.

Selecionar uma Estrutura

Um framework de desenvolvimento de ML adequada (consulte 1.1.7 para obter detalhes sobre as funcionalidades fornecidas) é selecionada com base nos objetivos, nos critérios de aceite (consulte 2.2) e nas prioridades de negócios.

Selecionar e construir o algoritmo

Um algoritmo de ML é selecionado com base em vários fatores, incluindo os objetivos, os critérios de aceite e a disponibilidade de dados (consulte 3.2). O algoritmo pode ser codificado manualmente, mas geralmente é obtido de uma biblioteca de software. O algoritmo é então compilado, se necessário.

Preparar e testar os dados de teste

A preparação de dados (consulte 3.2) compreende a aquisição de dados, o pré-processamento de dados e a engenharia de características. A análise exploratória de dados (EDA) pode ser realizada paralelamente a essas atividades.

Os dados utilizados pelo algoritmo e pelo modelo baseiam-se nos objetivos e são utilizados por todas as atividades na caixa “geração e teste do modelo” mostrada na Figura 1.

Os dados utilizados para treinar, avaliar, ajustar e testar o modelo devem ser representativos dos dados que serão utilizados pelo modelo operacionalmente.

É realizado o teste dos dados e de quaisquer etapas automatizadas de preparação de dados (consulte o Capítulo5).

Treinar o modelo

O algoritmo de ML selecionado utiliza dados de treinamento para treinar o modelo ML.

Os parâmetros que definem a estrutura do modelo (por exemplo, o número de camadas de uma rede neural ou a profundidade de uma árvore de decisão) foram passados para o algoritmo. Esses parâmetros são conhecidos como hiper parâmetros do modelo.

Os parâmetros que controlam o treinamento (por exemplo, o número de iterações a serem usadas ao treinar uma rede neural) também são passados para o algoritmo. Esses parâmetros são conhecidos como hiperparâmetros do algoritmo.

Avaliar o modelo

O modelo é avaliado em relação às métricas de performance funcional do ML acordadas (ver 3.3), utilizando o conjunto de dados de validação, e os resultados são usados para melhorar o modelo na atividade “ajustar o modelo”. Na prática, vários modelos são normalmente criados e treinados utilizando diferentes algoritmos (por exemplo, florestas aleatórias, SVM e redes neurais) e vários conjuntos de dados de treinamento, e a melhor combinação é escolhida com base nos resultados da avaliação.

Ajustar o modelo

Os resultados da avaliação são usados para ajustar os hiper parâmetros do modelo e os hiper parâmetros do algoritmo. O modelo é então retreinado com essas configurações ajustadas para melhorar sua performance funcional do ML.

As três atividades de treinamento, avaliação e ajuste compõem a “geração do modelo”, conforme mostrado na Figura 1.

Teste o modelo

Uma vez que um modelo aceitável tenha sido gerado pelas atividades de “geração de modelo”, ele é testado utilizando um conjunto de dados de teste independente para verificar se os critérios de desempenho funcional ML acordados foram atendidos. Os resultados do teste também são comparados com os da avaliação. Se o desempenho do modelo com os dados de teste independentes for significativamente inferior ao observado durante a avaliação, pode ser necessário retornar às atividades de “geração de modelo” ou mesmo à atividade de “preparação e teste de dados” para treinar um novo modelo.

Além dos testes de performance funcional do ML, também podem ser realizados testes não funcionais, como aqueles que avaliam o tempo necessário para gerar uma previsão. Normalmente, os testes nessa atividade são realizados por engenheiros ou cientistas de dados; no entanto, testadores com conhecimento suficiente da área e acesso aos recursos relevantes também podem realizar esses testes.

Implantar o modelo

Assim que a “geração e teste do modelo” estiver concluída, o modelo ajustado é normalmente reestruturado para implantação, juntamente com seu pipeline de dados. Isso geralmente é feito por meio da estrutura de desenvolvimento de ML. As plataformas de destino podem incluir sistemas embarcados e a nuvem, onde o modelo pode ser acessado por meio de uma API web. O modelo reestruturado e implantado é testado para verificar se ainda atende aos critérios de aceite.

Utilização do modelo

Uma vez implantado, o modelo é normalmente integrado a um sistema operacional maior baseado em AI. Os modelos podem realizar previsões em lote programadas em intervalos de tempo definidos ou ser executados em tempo real mediante solicitação.

Monitorar e ajustar o modelo

Enquanto o modelo está sendo usado, sua situação pode evoluir, e o modelo pode se desviar do desempenho pretendido (consulte 6.1.7). Para verificar se qualquer desvio é identificado e gerenciado, o modelo operacional é avaliado regularmente em relação aos seus critérios de aceite.

Pode ser considerado necessário criar um modelo por meio de retreinamento com novos dados, retreinamento com novos hiperparâmetros ou ambos. O modelo mais recente pode então ser comparado ao modelo existente usando uma forma de teste A/B (consulte 6.1.9).

O fluxo de trabalho de ML mostrado na Figura 1 é uma sequência lógica; no entanto, na prática, o fluxo de trabalho é aplicado de forma iterativa, com etapas repetidas.

As etapas mostradas na Figura 1 não incluem a integração do modelo ML com as partes não relacionadas a ML do sistema geral. Normalmente, os modelos ML não podem ser implantados isoladamente e devem ser integrados a componentes não relacionados a ML (por exemplo, em aplicações de visão, um pipeline de dados é usado para limpar e modificar os dados antes de enviá-los ao modelo ML). Quando o modelo faz parte de um sistema de grande porte, ele deve ser integrado a esse sistema antes da implantação. Nesse caso, podem ser realizados testes de integração, de sistema e de aceite.

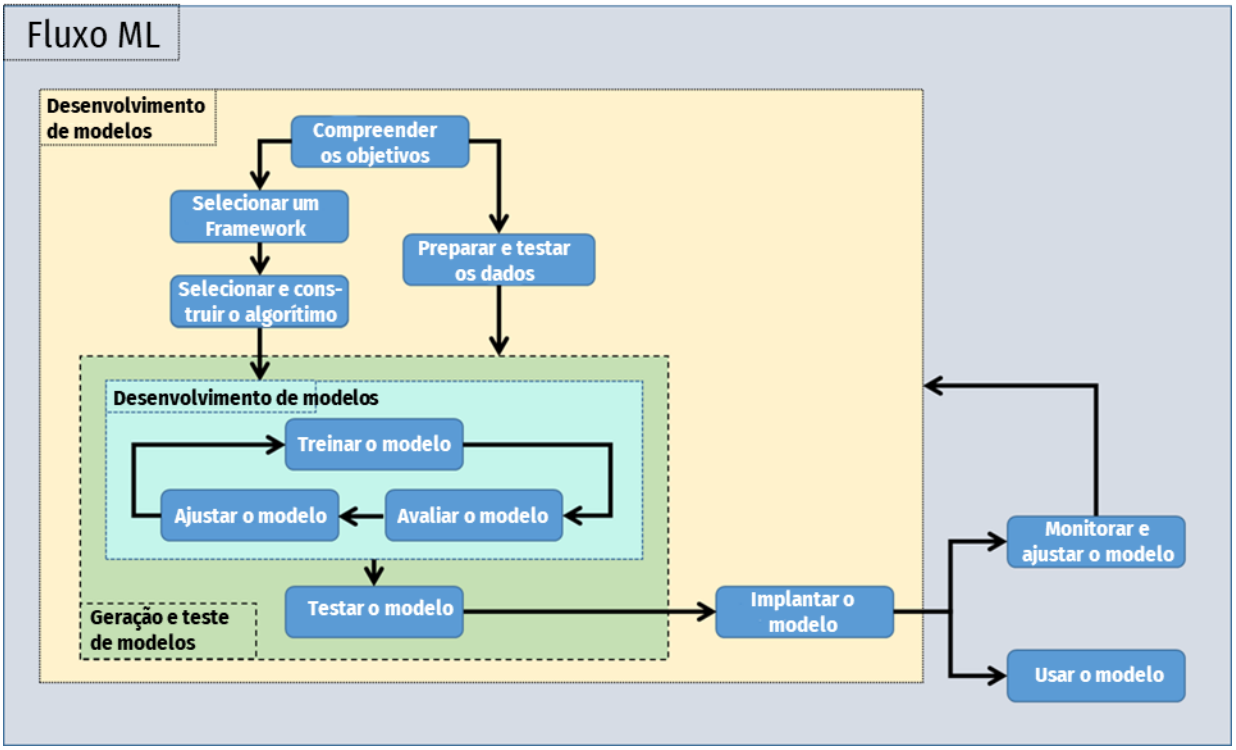

Figura 1: Fluxo de trabalho de ML

3.1.3 Exercício prático: Crie um modelo de machine learning

Selecione, treine e teste um modelo de classificação usando aprendizado supervisionado.

Explique a diferença entre avaliação/ajuste e teste, comparando a precisão alcançada com conjuntos de dados de validação e de teste.

3.1.4 Modelos pré-treinados, ajuste fino e geração aumentada por recuperação

Treinar um novo modelo de AI do zero é caro e demorado. Para resolver isso, uma solução comum é o ajuste fino, que consiste em pegar uma rede neural pré-treinada e adaptá-la para realizar uma tarefa nova e diferente. Uma das principais vantagens é que isso requer muito menos dados de treinamento (e esforço de treinamento) em comparação com a construção de um modelo do zero.

O modelo pré-treinado é ajustado por meio de um treinamento adicional com dados específicos para a nova tarefa. O ajuste pode ser aplicado a toda a rede neural, apenas a camadas específicas (normalmente próximas à saída da rede neural) ou a camadas adicionais. Após o treinamento, a performance funcional do ML do modelo é avaliada e, com base nesses resultados, um ajuste adicional pode ser realizado até que o modelo atenda aos critérios de aceite necessários.

O sucesso do ajuste fino depende da semelhança entre as tarefas original e nova. Pequenas diferenças podem levar a um ajuste fino altamente eficaz. Por exemplo, adaptar um classificador de imagens de raças de gatos para identificar raças de cães provavelmente funcionará bem. No entanto, adaptá-lo para sotaques falados é menos efetivo devido à diferença maior. Da mesma forma, o ajuste fino de um LLM para a análise de valor limite definida pelo ISTQB requer uma pequena alteração no LLM e é facilmente alcançável com bons dados de treinamento.

Uma alternativa ao ajuste fino é a Geração Aumentada por Recuperação (RAG), que envolve fornecer ao LLM fontes de dados específicas para a tarefa exigida. Essas fontes de dados são transformadas em um

formato pesquisável que permite que sejam comparadas ao assunto do prompt. Uma vez identificados os documentos relevantes, estes são incorporados a um prompt aprimorado, que passou para o LLM. Como agora são fornecidas informações mais pertinentes ao LLM, é provável que sua resposta correspondente seja mais precisa. Com o RAG, nenhuma alteração é feita no modelo pré-treinado.

Um modelo pré-treinado pode usar o RAG, o ajuste fino ou ambos juntos para melhorar a performance.

Normalmente, quaisquer vieses ou vulnerabilidades presentes no modelo pré-treinado serão transferidos para o novo modelo; portanto, é necessário realizar testes para confirmar se ele tem uma performance confiável e imparcial na nova tarefa.

3.2 Dados para Machine Learning

A preparação de dados é reconhecida como uma das atividades mais cruciais e intensivas em recursos no fluxo de trabalho de ML. Se os dados operacionais diferirem significativamente dos dados de treinamento, a performance funcional do ML e as premissas de segurança podem não se manter. A preparação de dados normalmente consome uma proporção significativamente maior do esforço total em comparação com outras etapas, como a seleção e a construção de modelos ML. A preparação de dados está intrinsecamente ligada ao pipeline de dados, que processa dados brutos e os transforma em um formato com alta usabilidade tanto para treinamento quanto para previsão por modelos ML.

3.2.1 Atividades na preparação de dados

A preparação de dados contribui para garantir a qualidade e a adequação dos dados para o treinamento do modelo. Ela envolve várias atividades-chave:

- Aquisição de dados:
    - Identificação de tipos de dados relevantes (por exemplo, numéricos, categóricos, imagens, texto).
    - Coleta de dados de diversas fontes, como bancos de dados, APIs ou sensores em tempo real.
    - Rotulagem de dados para tarefas de aprendizado supervisionado, verificando a precisão e a consistência.

Os dados adquiridos podem assumir várias formas (por exemplo, numéricos, categóricos, imagens, tabulares, texto, séries temporais, sensores, geoespaciais, vídeo e áudio).

- Pré-processamento de dados:
    - Limpeza de dados, incluindo:
        - remoção de defeitos, duplicatas e outliers para verificar a precisão e a consistência dos dados;
        - imputação de valores ausentes usando técnicas como média, mediana ou moda para manter a integridade dos dados;
        - anonimização ou remoção de informações pessoais para proteger a privacidade e cumprir as regulamentações.
    - Transformação de formatos de dados, escalonamento e normalização para alcançar a consistência dos dados.
    - Aumentar os dados para ampliar o tamanho da amostra, incorporando exemplos contraditórios para melhorar a robustez contra-ataques adversários e gerando dados sintéticos.
    - Amostragem de subconjuntos para reduzir tempos de treinamento e custos computacionais.

- Engenharia de características:
    - Seleção de características relevantes com base em sua contribuição para a performance do modelo ML.
    - Extrair um subconjunto de características informativas e não redundantes das características existentes para reduzir os tempos de treinamento e os custos computacionais.

Paralelamente a essas atividades de preparação de dados, a análise exploratória de dados (EDA) também é normalmente realizada para fornecer insights sobre os dados. Isso inclui:

- descobrir tendências, padrões e anomalias nos dados;
- visualização dos dados usando gráficos e tabelas para melhor compreensão.

A preparação dos dados de treinamento é, normalmente, um processo iterativo, muitas vezes realizado manualmente, e as atividades individuais de preparação de dados podem ser reordenadas ou omitidas com base nos requisitos específicos do projeto. Os dados operacionais devem corresponder às características dos dados de treinamento (por exemplo, distribuições de dados e intervalos de características) para que o modelo tenha a performance esperada em produção. No entanto, as próprias etapas de preparação podem ser ajustadas para garantir a eficiência e a escalabilidade da produção.

3.2.2 Exercício prático: Preparação de dados para apoiar a criação de um modelo de machine learning

Para um determinado conjunto de dados, execute as etapas de preparação de dados aplicáveis, conforme descrito na Seção 3.2.1, para produzir um conjunto de dados que será usado para criar um modelo de classificação usando aprendizado supervisionado.

Esta atividade serve como o primeiro passo na criação de um modelo ML que será utilizado em exercícios futuros.

Para realizar a performance, os alunos receberão materiais apropriados (e específicos para a linguagem), incluindo:

- bibliotecas;
- estrutura de desenvolvimento de ML;
- ferramentas.

3.2.3 Conjuntos de dados de treinamento, validação e teste

Logicamente, são necessários três conjuntos de dados equivalentes (por exemplo, selecionados aleatoriamente a partir de um único conjunto de dados representativo) como requisitos para desenvolver um modelo ML:

- Um conjunto de dados de treinamento é usado para treinar o modelo.
- Um conjunto de dados de validação é usado para avaliar e, posteriormente, ajustar o modelo.
- Um conjunto de dados de teste, também conhecido como conjunto de dados de retenção, é usado para testar o modelo ajustado.

Se houver uma abundância de dados adequados disponíveis, a quantidade de dados utilizados no fluxo de trabalho de ML para treinamento, avaliação e teste depende normalmente dos seguintes fatores:

- A complexidade esperada do modelo.

- O algoritmo usado para treinar o modelo.
- A disponibilidade de recursos, como RAM, espaço em disco, poder de computação, largura de banda de rede e o tempo disponível.
- O nível de confiança desejado no modelo resultante.

Quando os dados são limitados, uma divisão em três partes (treinamento, validação e teste) pode resultar em dados insuficientes para um treinamento eficaz do modelo, aumentando assim o risco de underfitting. Para resolver isso, uma estratégia comum é reservar um pequeno conjunto de teste final, se possível. Os dados restantes (o conjunto combinado de treinamento e validação) são então usados para técnicas como a validação cruzada k-fold (onde k é um número inteiro especificado pelo usuário, geralmente 5 ou 10).

Na validação cruzada k-fold, esses dados são divididos em k “dobras”. Para cada dobra, o modelo é treinado em k-1 dobras e validado na dobra de teste reservada. Esse processo é repetido k vezes, com cada dobra servindo como conjunto de validação uma vez. Isso permite um ajuste robusto dos hiperparâmetros e uma estimativa da performance funcional do ML. Os dados são normalmente atribuídos aleatoriamente às dobras, frequentemente usando amostragem estratificada para tornar cada dobra representativa, especialmente com dados desequilibrados ou conjuntos de dados pequenos.

As métricas de desempenho (por exemplo, precisão, F1-score – ver 3.3.1) da validação de cada dobra são então calculadas como média para fornecer uma estimativa mais confiável da capacidade de generalização do modelo. Após identificar os hiperparâmetros ótimos por meio da validação cruzada, um modelo final é normalmente treinado em todo o conjunto de treinamento e validação (todos os dados, exceto o conjunto de teste de retenção) utilizando esses hiperparâmetros. Esse modelo final é então avaliado uma vez no conjunto de teste de retenção para uma avaliação final e imparcial do desempenho. Se um conjunto de teste de retenção não for viável devido à extrema escassez de dados, o desempenho médio da validação cruzada é otimisticamente tendencioso e não pode servir como uma estimativa imparcial.

Outros métodos de reamostragem para dados limitados incluem a validação cruzada leave-one-out (um caso especial de k-fold, em que k é igual ao número de amostras) e técnicas de bootstrap.

3.3 Métricas de Performance Funcional do ML para Classificação

Em tarefas de classificação (ver 3.1.1), uma matriz de confusão pode ser usada para avaliar as previsões de um modelo, categorizando-as como verdadeiros positivos, verdadeiros negativos, falsos positivos ou falsos negativos. Métricas-chave, como precisão, recall e F1-score, são derivadas disso. Essas métricas medem a qualidade da classificação, destacando os pontos fortes e as fraquezas do modelo ML. Esta seção explora o cálculo e a interpretação dessas métricas para avaliar a performance funcional do ML.

3.3.1 Cálculo de métricas de performance de machine learning

Em um problema de classificação, um modelo raramente prevê os resultados corretamente o tempo todo, em parte devido à natureza probabilística dos modelos ML e ao ruído dos dados. Para qualquer problema desse tipo, uma matriz de confusão pode ser criada com as seguintes possibilidades:

		Real	
		Positivo	Negativo
Previsto	Positivo	Verdadeiro positivo (TP)	Falso positivo (FP)
	Negativo	Falso negativo (FN)	Negativo verdadeiro (TN)

Figura 2: Matriz de confusão

Observe que a matriz de confusão mostrada na Figura 2 pode ser apresentada de maneira diferente (por exemplo, com os valores previstos e reais trocados), mas sempre fornecerá valores para as quatro situações possíveis: verdadeiro positivo (TP), verdadeiro negativo (TN), falso positivo (FP) e falso negativo (FN).

Com base na matriz de confusão, definem-se as seguintes métricas:

- Precisão = (TP + TN) / (TP + TN + FP + FN) * 100%

  A medição da precisão mede a porcentagem de todas as classificações corretas.

- Precisão = TP / (TP + FP) * 100%

  A medição da precisão mede a proporção de resultados positivos que foram corretamente previstos. É uma medida do grau de certeza que se pode ter em relação às previsões positivas.

- Recall = TP / (TP + FN) * 100%

  Recall (também conhecido como sensibilidade) mede a proporção de resultados positivos reais que foram corretamente previstos. É uma medição do grau de confiança de que nenhum resultado positivo será perdido.

- Pontuação F1 = 2 * (Precisão * Recall) / (Precisão + Recall)

  O F1-score é calculado como a média harmônica da precisão e do recall, com valores que variam de 0 a 100. Uma pontuação próxima a 100 significa que o modelo alcança alta precisão e alto recall, indicando que os erros de classificação (falsos positivos e falsos negativos) têm impacto mínimo. Por outro lado, um F1-score baixo indica que o modelo tem dificuldade em identificar os positivos com precisão, seja perdendo casos verdadeiros ou gerando muitos alarmes falsos.

3.3.2 Exercício prático: Avalie um modelo de machine learning usando métricas de performance funcional ML selecionadas

Usando o modelo de classificação treinado no exercício anterior, calcule e exiba os valores de exatidão, precisão, recall e pontuação F1. Quando aplicável, use as funções da biblioteca fornecidas pela sua estrutura de desenvolvimento de ML para realizar os cálculos.

3.3.3 Exercício prático: Mostre o impacto de diferentes combinações de modelos de machine learning e conjuntos de dados

Seguindo o exercício anterior, use diferentes modelos ML e combinações de conjuntos de dados para observar seu efeito no treinamento do modelo ML, bem como no comportamento final do modelo. Observe os tempos de treinamento e as métricas de performance funcional do ML.

3.4 Redes Neurais

As redes neurais artificiais foram inicialmente projetadas para imitar o funcionamento do cérebro humano, que pode ser considerado como uma rede de neurônios biológicos interconectados.

O perceptron de camada única é um dos primeiros exemplos de implementação de uma rede neural artificial, composto por apenas uma camada. Ele pode ser usado para o aprendizado supervisionado de classificadores binários para problemas linearmente separáveis, que determinam se uma entrada pertence a uma classe específica ou não. Por exemplo, um perceptron pode distinguir entre e-mails que são spam e aqueles que não são spam, aprendendo a separar as características das duas categorias com uma linha reta no espaço de entrada.

A maioria das redes neurais atuais é considerada redes neurais profundas, pois compreende várias camadas. Redes totalmente conectadas podem ser consideradas perceptrons multicamadas (ver Figura 3).

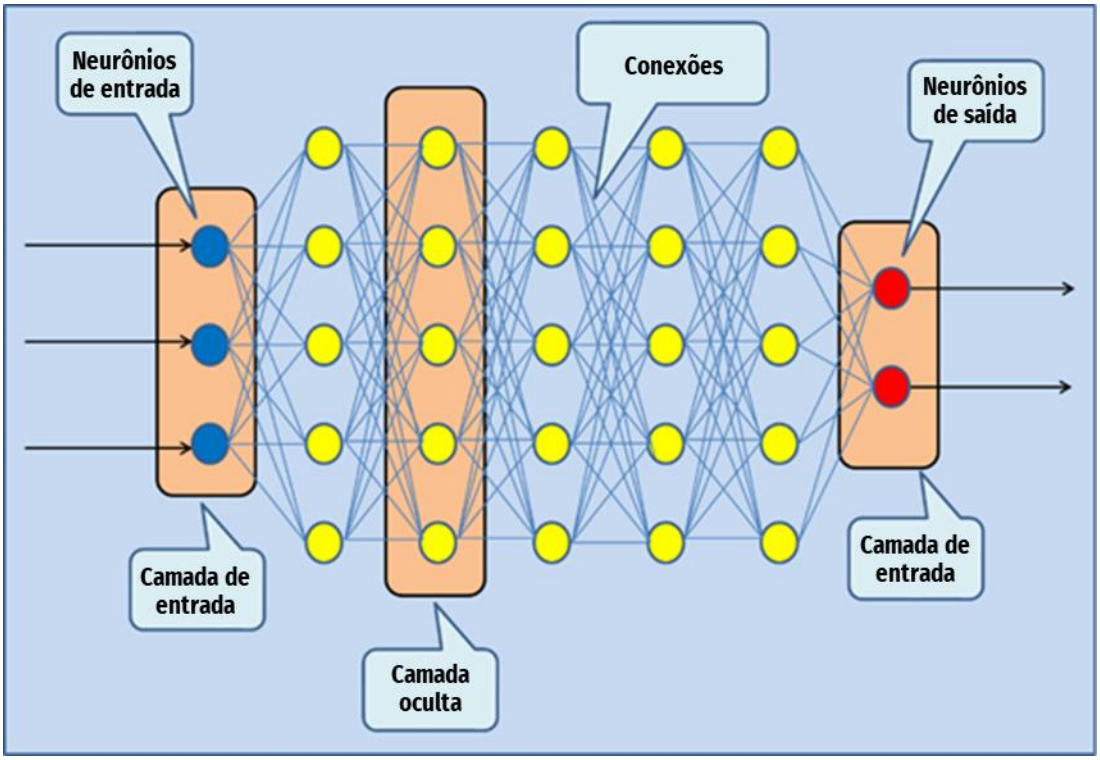

Figura 3: Estrutura de uma rede neural profunda

3.4.1 Estrutura e funcionamento de uma rede neural profunda

Uma rede neural profunda é normalmente descrita como composta por três tipos principais de camadas. A camada de entrada recebe entradas, por exemplo, valores de pixels de uma câmera. A camada de saída fornece resultados ao mundo exterior. Isso pode ser, por exemplo, um valor indicando a probabilidade de que a imagem de entrada seja de um gato. Entre as camadas de entrada e saída, há camadas ocultas compostas por neurônios artificiais, também conhecidos como nós. Em muitas arquiteturas comuns, como redes totalmente conectadas, os neurônios de uma camada estão conectados a cada um dos neurônios da camada seguinte, e pode haver um número diferente de neurônios em cada camada sucessiva.

Os neurônios realizam cálculos e transmitem informações pela rede, dos neurônios de entrada aos de saída, transformando gradualmente os dados de entrada em representações cada vez mais abstratas até chegar à saída.

O cálculo realizado por cada neurônio (após aqueles na camada de entrada) gera o que é conhecido como valor de ativação. Esse valor é calculado primeiro pela soma ponderada dos valores de ativação de todos os neurônios conectados na camada anterior, onde cada conexão tem seu próprio peso independente, e pela adição do viés individual do neurônio. Essa soma é então passada por uma fórmula não linear chamada função de ativação. Observe que esse viés não está relacionado ao viés considerado em 5.1.2 . O uso de diferentes funções de ativação produz diferentes valores de ativação.

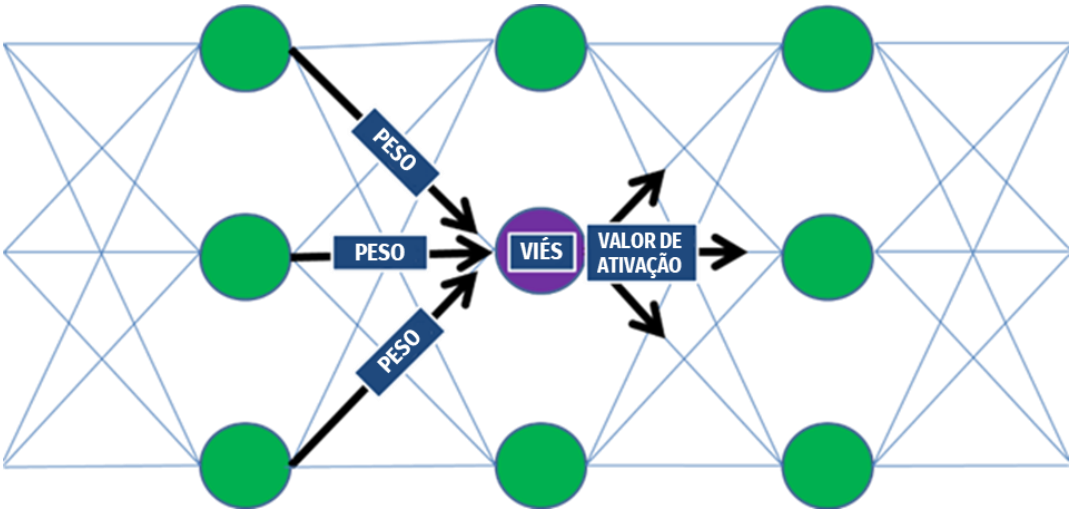

Figura 4: Cálculo realizado por cada neurônio

Os pesos que conectam os neurônios e o valor do viés de cada neurônio são normalmente inicializados com pequenos valores aleatórios (os viéses, às vezes, com zero) no início do treinamento. Os dados de treinamento são passados pela rede, com cada neurônio executando a função de ativação, para gerar uma saída final. A saída gerada é então comparada com o resultado correto conhecido. O erro resultante (ou perda), que quantifica essa diferença, é então realimentado pela rede para ajustar os valores dos pesos e dos vieses, minimizando assim essa diferença. À medida que mais dados de treinamento são alimentados pela rede (cada passagem pelo conjunto de dados de treinamento é chamada de época), os pesos e os valores de bias são gradualmente ajustados à medida que a rede aprende. Após algum tempo, idealmente, a saída produzida é considerada boa o suficiente para encerrar o treinamento.

3.4.2 Exercício prático: Experimente a implementação de um perceptron

Os alunos serão orientados em um exercício que demonstra um perceptron aprendendo uma função simples, como uma função AND.

O exercício deve abordar como um perceptron aprende modificando seus pesos e valores de viés ao longo de várias épocas até que o erro seja minimizado a zero. Vários mecanismos (por exemplo, planilha, simulação) podem ser usados para esta atividade.

3.4.3 Medidas de cobertura para redes neurais

Métricas de cobertura estrutural de redes neurais surgiram para avaliar até que ponto as entradas de teste exercitam os mecanismos internos de um modelo. Como as redes neurais não seguem caminhos explicitamente codificados, mas seu comportamento é ditado por pesos, valores de polarização e ativações aprendidos, isso cria a necessidade de medições de cobertura especializadas para avaliar até que ponto diferentes partes da rede foram ativadas sob condições de teste.
Abordagens típicas incluem [COV_REF]:

- Cobertura de Neurônios: Realiza a medição da proporção de neurônios na rede cuja saída excede um limite especificado durante o teste.
- Cobertura de Neurônios com k seções (kMNC): O intervalo de saída possível de cada neurônio é dividido em k seções. kMNC é a proporção dessas seções ativadas durante o teste.
- Cobertura dos Limites dos Neurônios (NBC): Realiza a medição da proporção de neurônios na rede cuja saída excede o máximo alcançado no treinamento ou é menor que o mínimo alcançado no treinamento durante o teste.

Essas métricas de cobertura podem ser úteis em testes de AI, principalmente por revelarem áreas do modelo que permanecem não testadas, o que pode potencialmente ocultar defeitos ou comportamentos não aprendidos. Por exemplo, se certos neurônios ou camadas nunca forem ativados durante o teste, o desempenho do modelo em limites de decisão relacionados pode ser questionável. Essas percepções ajudam os testadores a projetar entradas de teste adicionais ou condições de teste para exercitar partes pouco exploradas da rede. Atualmente, as ferramentas comerciais que suportam essas medições específicas de cobertura são limitadas.

No entanto, a cobertura estrutural por si só não garante que uma rede neural generalize bem ou lide com variações do mundo real. As redes neurais podem aprender correlações espúrias, levando a ativações corretas por motivos incorretos. Consequentemente, os testadores também devem usar outras técnicas de teste, como testes contraditórios e testes metamórficos, conforme descrito nos capítulos 5, 6 e 7.

4 Teste de Sistemas Baseados em AI – 195 minutos

Palavras-chave

Ataque, testes exploratórios, testes baseados em riscos, oráculo de teste

Palavras-chave específicas de AI

Sistema adaptativo baseado em AI, sistema baseado em AI, AI generativa, modelo de linguagem de grande porte, sistema baseado em AI bloqueado

Objetivos de aprendizagem do Capítulo 4:

4.1 Introdução aos Testes de Sistemas Baseados em AI

AI-4.1.1 (K2) Comparar a testabilidade de sistemas baseados em AI fixos e adaptativos

AI-4.1.2 (K2) Explicar por que uma abordagem estatística é frequentemente necessária ao testar sistemas baseados em AI

AI-4.1.3 (K2) Explicar os desafios e as soluções relacionados a oráculos de teste para sistemas baseados em AI

4.2 Testeando AI Generativa e LLM

AI-4.2.1 (K2) Explicar como a AI generativa pode ser testada

AI-4.2.2 (K3) Implementar red teaming para sistemas de AI generativa

HO-4.2.3 (H2) Aplicar testes exploratórios a um LLM realizando análise de valor limite

4.3 Níveis de Teste e Sistemas de Machine Learning

AI-4.3.1 (K2) Resumir os níveis de teste utilizados para desenvolver sistemas de machine learning

AI-4.3.2 (K2) Explicar como os testes baseados em riscos são aplicados a sistemas de machine learning

4.1 Introdução ao Teste de Sistemas Baseados em AI

O teste de sistemas baseados em AI apresenta desafios únicos em comparação com o teste de software convencional. Em primeiro lugar, os sistemas baseados em AI podem ser amplamente categorizados como fixos ou adaptativos. Os sistemas baseados em AI fixos, com comportamento fixo após a implantação, são mais fáceis de testar devido à sua natureza predominantemente determinística, enquanto os sistemas adaptativos, que evoluem e aprendem, introduzem complexidade, pois seu comportamento pode mudar de forma imprevisível. Além disso, a natureza probabilística de muitos modelos de AI frequentemente requer testes estatísticos, já que métodos determinísticos podem ser insuficientes para avaliar resultados influenciados por dados e probabilidades. Isso exige a avaliação de distribuições de desempenho e níveis de confiança em diversos cenários.

Um desafio central nos testes, conhecido como o “problema do oráculo de teste”, surge ao determinar se a saída produzida pelo sistema em teste é correta para uma determinada entrada. Em softwares tradicionais, especificações bem definidas tornam relativamente simples especificar os resultados esperados e verificar a correção. No entanto, com sistemas baseados em AI, especialmente aqueles que lidam com tarefas complexas ou subjetivas, definir claramente os resultados esperados pode ser difícil ou até mesmo impossível. Essa incerteza é agravada para tarefas que excedem as capacidades humanas, envolvem imprecisão ou carecem de uma “verdade fundamental” objetiva, dificultando a implementação de oráculos de teste automatizados e, às vezes, exigindo julgamento de especialistas ou raciocínio estatístico.

Requisitos de sistema vagos ou incompletos agravam ainda mais o problema dos oráculos de teste nos testes de AI. As soluções podem incluir o uso de avaliações estatísticas, a consulta a especialistas na área ou a definição de uma “verdade fundamental” contra a qual avaliar os resultados. Essas abordagens visam estabelecer expectativas confiáveis e avaliar efetivamente os sistemas baseados em AI.

4.1.1 Sistemas baseados em AI fixos e adaptativos

Muitos sistemas baseados em AI em uso atualmente são sistemas baseados em AI bloqueados, que não alteram seu comportamento após serem implantados. Um exemplo de sistema baseado em AI bloqueado é um modelo ML implantado baseado em uma DNN, em que os pesos e vieses da DNN são fixados após o desenvolvimento e só podem ser modificados se a DNN for retreinada. A tecnologia de carros autônomos relacionada à segurança frequentemente depende de sistemas baseados em AI bloqueados para tarefas específicas, como detecção de faixa ou reconhecimento de sinais de trânsito.

Em contrapartida, um sistema adaptativo baseado em AI, como um sistema de aprendizado por reforço, pode adaptar seu comportamento após a implantação. As mudanças podem ser baseadas em uma função de recompensa ou na adaptação a um novo ambiente operacional, mas os detalhes específicos dessas mudanças não podem ser previstos com antecedência. Por exemplo, uma plataforma de comércio eletrônico poderia usar AI adaptativa para recomendar produtos aos usuários com base em seu comportamento passado e em suas preferências em evolução.

Muitos sistemas de AI de última geração, como chatbots baseados em LLM, são implantados como modelos fixos em tempo de execução, mas são atualizados periodicamente, situando-os entre sistemas totalmente fixos e totalmente adaptativos.

Na prática, os sistemas baseados em AI abrangem um continuum que vai desde sistemas totalmente determinísticos e bloqueados, que produzem a mesma saída para uma determinada entrada, até sistemas deliberadamente não determinísticos e de autoaprendizagem, que adaptam e evoluem seu comportamento.

Sistemas baseados em AI bloqueados são muito mais fáceis de testar do que sistemas baseados em AI adaptativa, pois são amplamente determinísticos e, portanto, os resultados esperados não mudam. Um sistema baseado em AI bloqueado e atualizado é normalmente considerado um novo sistema, sendo necessária uma nova rodada de testes. Observe, no entanto, que embora os sistemas baseados em AI

bloqueados sejam geralmente considerados determinísticos, na prática, grandes redes neurais podem exibir comportamento não determinístico devido a fatores como limites de precisão de ponto flutuante e variações na execução de hardware, particularmente ao usar computação paralela ou GPUs.

Um sistema adaptativo baseado em AI pode ser rigorosamente testado antes da implantação (por exemplo, por meio de simulações de mudanças nas condições ambientais, testando o próprio mecanismo de aprendizagem ou testando sua capacidade de se adaptar adequadamente em cenários controlados). No entanto, esses testes são mais complexos do que para um sistema baseado em AI bloqueado, pois um sistema adaptativo pode alterar seu comportamento durante os testes ou como resultado deles.

Como novos comportamentos de um sistema adaptativo baseado em AI nem sempre podem ser previstos, tais comportamentos imprevisíveis não podem ser testados antecipadamente, nem casos de teste podem ser preparados para eles. Uma suíte de testes automatizados, focada na funcionalidade central do sistema, pode ser construída e executada sempre que o sistema passar por mudanças significativas para verificar se as adaptações são seguras.

Os testes também podem ser utilizados para verificar se a performance do sistema não se deteriorou além de um determinado limite. Esses testes podem ser realizados em resposta a alterações significativas no sistema ou como parte do monitoramento contínuo.

4.1.2 Fundamentação para uma abordagem estatística nos testes de sistemas baseados em AI

O teste de sistemas baseados em AI apresenta desafios únicos devido à sua natureza orientada por dados e probabilística.

As razões pelas quais é necessária uma abordagem estatística para testar sistemas baseados em AI incluem:

- Não determinismo — os sistemas baseados em AI são fundamentalmente probabilísticos e, portanto, frequentemente exibem comportamento não determinístico, o que significa que as mesmas entradas nem sempre produzem a mesma saída. Isso pode ser devido a elementos estocásticos em sua arquitetura ou à implementação de mapeamentos probabilísticos aprendidos a partir de dados de treinamento. Consequentemente, um único teste, como um caso em que um modelo classifica incorretamente um gato como um cão, não pode refletir com precisão a correção funcional geral da AI do modelo. Para confirmar que os resultados dos testes superam essa incerteza de forma confiável, a suíte de testes deve ser suficientemente grande para fornecer resultados estatisticamente significativos.
- Avaliação de desempenho distributivo - Os modelos de AI são treinados em distribuições de dados específicas que não correspondem exatamente ao seu ambiente operacional. Para avaliar como um modelo se comportará em condições reais, deve-se testar uma amostra estatisticamente significativa de cenários a partir de distribuições de dados operacionais relevantes. Isso confirma que o teste captura a variabilidade operacional nos dados e nos comportamentos subsequentes do modelo.
- Lidando com incerteza e viés - Sistemas baseados em AI são suscetíveis a viéses de dados e podem produzir previsões confiantes, mas incorretas. Testes estatísticos permitem que os profissionais quantifiquem e analisem a precisão, a imparcialidade e a robustez da AI por meio de métricas de desempenho, como intervalos de confiança, testes de hipóteses e análise de erros.
- Contexto regulatório e de segurança — Em setores regulamentados (por exemplo, saúde, transporte), muitas vezes é necessário demonstrar que um sistema baseado em AI atende a limites de segurança ou imparcialidade com alta confiança. Métodos estatísticos respaldam afirmações sobre a confiabilidade de um modelo em uma ampla gama de cenários, e não apenas para exemplos específicos.

Uma abordagem estatística para o teste de MLS probabilístico é apresentada na seção 6.1.3.

4.1.3 Oráculos de teste para sistemas baseados em AI

Testar sistemas baseados em IA pode ser um desafio, especialmente ao determinar os resultados esperados (o problema do oráculo de teste). Essas dificuldades decorrem de vários fatores inerentes à IA:

- Natureza probabilística e não determinística: os resultados da AI podem variar mesmo com entradas idênticas. Embora muitos sistemas (por exemplo, na aprendizagem supervisionada) tenham um único valor-alvo correto, os resultados do modelo são probabilísticos, e definir um oráculo estrito de aprovação/reprovação geralmente requer o estabelecimento de limites ou intervalos de tolerância.
- Desenvolvimento exploratório e especificações incompletas: O desenvolvimento de AI é frequentemente exploratório. Os requisitos do sistema podem evoluir, estar incompletos ou simplesmente faltar, sem as especificações detalhadas necessárias para gerar resultados esperados precisos.
- Complexidade das tarefas: os sistemas baseados em AI frequentemente lidam com tarefas que são complexas demais para uma verificação humana direta, tornando as verificações manuais dos resultados esperados impraticáveis.
- Subjetividade do comportamento: A correção do comportamento de um sistema baseado em AI pode ser subjetiva. Por exemplo, as expectativas dos usuários em relação a assistentes virtuais podem variar amplamente, complicando o estabelecimento de resultados esperados universalmente aceitos.
- Sistemas de autoaprendizagem: Esses sistemas baseados em AI atualizam continuamente seus modelos internos com base em novos dados encontrados após a implantação. Isso faz com que o comportamento “correto” do sistema baseado em AI mude ao longo do tempo, de modo que as respostas do sistema permaneçam eficazes e apropriadas, mesmo que a definição de “correto” evolua com o tempo. Como resultado, um conjunto inicial de resultados esperados pode rapidamente se tornar inválido.

O problema do oráculo de teste com sistemas baseados em AI pode ser abordado de várias maneiras:

- Definição de limites de saída: Os testadores podem usar intervalos, distribuições, limites especificados e tolerâncias acordados como aceitáveis, como um carro autônomo parando dentro de uma distância máxima.
- Definição de limites ambientais: Os testadores devem especificar valores para as condições do ambiente de teste (por exemplo, níveis de iluminação, temperatura, latência de rede) para garantir que as saídas sejam previsíveis e repetíveis.
- Consulta a especialistas: especialistas na área podem ajudar a definir os resultados esperados, embora suas opiniões possam diferir ou ser falíveis.
- Testes especializados: testes A/B, testes back-to-back e testes metamórficos, entre outros, podem avaliar sistemas baseados em AI comparando comportamentos ou verificando propriedades, muitas vezes sem exigir saídas esperadas explícitas para cada caso.
- Oráculos proxy: use sistemas ou modelos secundários (incluindo outros sistemas de AI) para avaliar ou validar resultados quando os resultados esperados diretos não estiverem disponíveis, como treinar um modelo proxy em dados rotulados para fazer previsões em testes não rotulados.

4.2 Testando AI Generativa e Large Language Model

Esta seção descreve métodos práticos para validar sistemas de AI gerativa, abrangendo avaliação de caixa-preta, red teaming e técnicas práticas. Ela destaca desafios decorrentes de entradas diversas, parâmetros ajustáveis e janelas de contexto, e aborda medidas de qualidade tanto funcionais quanto não funcionais usando benchmarks e exercícios direcionados. Consulte o syllabus do CT-GenAI [CT-GenAI] para obter mais detalhes sobre o uso de sistemas de AI gerativa para testes.

4.2.1 Testando a AI Generativa

O teste de AI Generativa envolve a avaliação da correção, coerência e criatividade de seus resultados, incluindo a originalidade e a novidade dos resultados gerados, e a verificação do cumprimento dos requisitos especificados, tanto funcionais quanto não funcionais. Como os sistemas de AI Gen podem produzir texto, imagens, vídeo e áudio, as estratégias de teste devem ser adaptadas para avaliar as características de qualidade desses conteúdos.

Uma abordagem comum é o teste caixa-preta, em que os testadores inserem várias entradas (prompts, imagens, dados parciais) no sistema e avaliam os resultados do teste (ver red teaming, 4.2.2). Fatores como clareza, originalidade e adesão a regras específicas do domínio são considerados. Essa abordagem, seja manual ou automatizada, é particularmente útil para aplicações de usuário final, como chatbots ou ferramentas de design, nas quais a satisfação do usuário depende da utilidade e da plausibilidade do conteúdo gerado.

Um desafio significativo no teste de um modelo de GenAI é o “problema da explosão de entradas”, no qual as entradas podem ser altamente diversas e complexas de controlar. Por exemplo, há prompts opcionais do sistema e um prompt do usuário, que podem incluir enormes quantidades de dados díspares, e a GenAI também pode ser acessada por meio de uma API. Há também vários parâmetros a serem considerados, como temperatura e tokens máximos, que também afetarão a saída. Além disso, a janela de contexto, que retém partes anteriores de uma conversa, também afeta a saída gerada.

Em muitas situações, a avaliação da saída pode envolver revisão manual; no entanto, determinar se um teste passou ou falhou depende de critérios de avaliação qualitativos definidos nos requisitos. Alternativamente, um segundo sistema GenAI pode frequentemente ser usado para determinar automaticamente o resultado do teste. Por exemplo, a correção de uma imagem gerada por um sistema GenAI pode ser verificada por um sistema de reconhecimento de imagem. Tais abordagens devem ser usadas com cuidado, pois podem reproduzir vieses ou erros semelhantes.

Os testes não funcionais podem ser tão necessários quanto os testes funcionais para sistemas GenAI. Isso inclui avaliar a utilização de recursos tanto durante a inferência quanto durante o treinamento para verificar a eficiência da operação e a relação custo-benefício. As medições incluem uso de CPU e GPU, consumo de memória, largura de banda da rede e tempos de resposta.

Conjuntos de benchmarks fornecem estruturas de avaliação padronizadas para avaliar as capacidades da GenAI. Esses conjuntos de dados selecionados e tarefas associadas permitem uma comparação consistente entre diferentes modelos, medindo vários aspectos, desde a compreensão da linguagem até habilidades de raciocínio e codificação. Ao avaliar a GenAI em relação a esses benchmarks, áreas para melhoria podem ser identificadas de maneira sistemática e reproduzível.

4.2.2 Red Teaming

O red teaming (RT) é uma forma sistemática, frequentemente do tipo caixa preta, de ataque de falha que examina um sistema baseado em AI para identificar capacidades prejudiciais, particularmente em seus resultados. Inspirando-se em práticas históricas como jogos de guerra militares e as equipes-tigre da NASA, o AI RT envolve “ataque” a um sistema baseado em AI, com o objetivo de fazer com que o sistema produza deliberadamente resultados prejudiciais ou indesejáveis, tais como violações de privacidade, a expressão de visões racistas ou o fornecimento de orientações sobre a realização de ataques químicos, biológicos, radiológicos ou nucleares (CBRN). Uma vez identificadas essas

capacidades, elas são usadas para atualizar e fortalecer o sistema, tornando-o menos propenso a tais resultados no futuro. Embora o RT possa ser aplicado a qualquer sistema baseado em AI, ele é especialmente crítico para a GenAI devido à vasta gama de entradas e saídas possíveis, que criam um extenso “espaço de ataque”. O RT normalmente abrange todo o sistema baseado em AI de ponta a ponta, mas pode ser aplicado apenas ao modelo de AI.

Muitas organizações concentram a RT em vulnerabilidades de segurança e proteção; no entanto, ela também pode ser usada para detectar capacidades prejudiciais em outras áreas, como confiabilidade, privacidade, equidade, preconceito e a geração de desinformação. A RT está se tornando cada vez mais uma expectativa regulatória para alguns tipos de sistemas baseados em IA, como se observa em marcos normativos como a Lei de IA da UE [EU AI Act]. Como uma abordagem de avaliação adaptativa e dinâmica, na qual os prompts podem ser atualizados imediatamente em resposta aos resultados, a RT complementa abordagens estáticas, como o benchmarking, ao testar sistemas em condições extremas e inesperadas.

Quando o RT é utilizado para avaliações de segurança, o sistema é testado para identificar vulnerabilidades a ataques externos, incluindo tanto fatores de segurança não relacionados à IA quanto específicos da IA, tais como ataques de injeção indireta de prompts e a ocultação de conteúdo malicioso em documentos utilizados pelo RAG. Enquanto o RT de segurança tende a se concentrar em entradas maliciosas, nas avaliações de segurança e outras, o objetivo geralmente é identificar como o sistema pode gerar resultados prejudiciais em condições normais de uso, como a geração de orientações médicas inseguras, sem qualquer intenção adversária por parte do usuário.

A RT é mais efetiva quando realizada antes da implantação de um sistema, mas após a conclusão das avaliações internas iniciais de qualidade. A atividade principal consiste frequentemente em perguntas interativas, nas quais os membros da equipe vermelha participam de diálogos com várias trocas de mensagens (por exemplo, 15 a 20 trocas) para identificar comportamentos com defeitos ou que violem as políticas. A abordagem normalmente envolve:

1) Formar uma equipe diversificada de testadores para abranger uma ampla gama de perspectivas e vetores de ataque.
2) Conceder acesso ao sistema baseado em AI em um ambiente de teste seguro.
3) Solicitar ao sistema que identifique vulnerabilidades, por meio de exploração aberta ou usando listas de verificação.
4) Analisar as falhas identificadas para compreender as ameaças.
5) Criaar conjuntos de dados a partir dessas ameaças para apoiar medidas de mitigação e melhorias no sistema.

Para alcançar uma cobertura abrangente por meio de testes de resistência (RT), as organizações empregam várias estratégias além de pequenas equipes especializadas. Entre elas estão abordagens manuais, como a geração colaborativa de prompts, e abordagens automatizadas, nas quais um modelo LLM é utilizado para gerar inúmeros prompts de ataque, cujos resultados são posteriormente verificados por outro LLM. As abordagens híbridas combinam a criatividade dos testadores humanos com a escalabilidade da automação.

O RT, que é proativo e focado na fase pré-implantação, complementa o Blue Teaming, que envolve monitoramento defensivo contínuo em tempo real e filtragem de entradas em um sistema operacional baseado em IA para protegê-lo contra ataques. As informações obtidas pelo RT podem ser utilizadas para aprimorar o monitoramento e os filtros empregados no Blue Teaming.

4.2.3 Exercício prático: Teste exploratório de um modelo de uma LLM

Os alunos realizarão testes exploratórios de um LLM. Será fornecida aos alunos uma ficha de sessão de testes exploratórios voltada para testar a capacidade do LLM de gerar casos de teste utilizando a análise

de valor limite de 2 e 3 valores. Como parte da análise da sessão, eles verificarão a exatidão e a integridade dos resultados.

4.3 Níveis de Teste e Sistemas de Machine Learning

A MLS exige níveis de teste especializados para lidar com os riscos específicos do machine learning. Trata-se do teste de dados de entrada e do teste do modelo ML. Além disso, os níveis de teste convencionais continuam sendo relevantes, incluindo testes de componentes, de integração de componentes, de sistemas e de teste de aceite.

4.3.1 Níveis de teste para sistemas de machine learning

Os componentes não relacionados à AI de um MLS podem ser testados usando níveis de teste convencionais. Além disso, os modelos ML e o MLS exigem testes adicionais para lidar com os riscos específicos associados ao ML.

Dois níveis de teste especializados são usados para lidar com riscos específicos do ML:

- Teste de dados de entrada (ver Capítulo 5): Dedicado aos dados de treinamento utilizados para treinar um modelo ML e aos dados de produção utilizados por um MLS para gerar uma previsão no ambiente operacional.
- Teste do modelo ML (ver Capítulo 6): Dedicado ao teste dos modelos ML, que são o resultado do fluxo de trabalho de ML (ver 3.1.2).

Nem todos os riscos associados ao ML podem ser abordados usando esses dois níveis de teste específicos para ML. Os seguintes níveis de teste também são normalmente necessários, dependendo dos riscos percebidos:

- Teste de Componentes: Aplicável a quaisquer componentes não relacionados à AI, como interface do usuário, pipeline de dados e componentes de comunicação.
- Teste de integração de componentes: Inclui o teste para verificar se as entradas do pipeline de dados são recebidas conforme o esperado pelo modelo e se quaisquer previsões geradas pelo modelo são trocadas com os componentes relevantes do sistema (por exemplo, a interface do usuário) e utilizadas corretamente. Quando a AI é fornecida como um serviço (ver1.1.6), o Teste de API do serviço fornecido é realizado como parte do teste de integração de componentes.
- Teste de sistema: Incluem testes de confirmação de que a performance funcional do ML, obtida no teste do modelo ML inicial, não é afetada negativamente quando o modelo é incorporado a um sistema completo. Esses testes são especialmente importantes quando o modelo ML foi alterado deliberadamente (por exemplo, pela compactação de uma DNN para reduzir seu tamanho). Testes não funcionais também são abrangidos, por exemplo, testes de eficiência de performance do tempo necessário para fornecer uma previsão a partir de um sistema completo baseado em AI.
- Teste de integração do sistema: Concentra-se na verificação das interfaces e trocas de dados entre o sistema baseado em AI e sistemas ou serviços externos, utilizando um ambiente representativo das condições operacionais.
- Testes de aceite: Quando a AI é utilizada como um serviço, podem ser necessários testes de aceite para determinar a adequação do serviço ao sistema pretendido e se, por exemplo, os critérios de desempenho funcional do ML foram alcançados.

4.3.2 Testes baseados em riscos de sistemas de machine learning

Testes baseados em riscos devem ser aplicados a todos os sistemas, independentemente de conterem ou não componentes de AI. A maioria das estruturas regulatórias, sejam elas publicadas ou em desenvolvimento, exige uma abordagem baseada em risco para o desenvolvimento e a gestão de sistemas baseados em AI. Como os sistemas baseados em AI apresentam riscos únicos, testá-los difere do que se faz com sistemas não baseados em AI.

Não existe uma forma padronizada de categorizar os riscos de ML; no entanto, os riscos específicos de ML estão associados tanto ao desenvolvimento do MLS (riscos de projeto) quanto ao próprio MLS (riscos de produto). Uma maneira de categorizar esses riscos é utilizar o fluxo de trabalho de ML e dividi-lo em três áreas principais:

- Desenvolvimento — diz respeito ao algoritmo de ML, ao desenvolvimento do modelo ML e à estrutura de desenvolvimento de ML. Exemplos de riscos de projeto incluem seleção de algoritmo subótima, má escolha da abordagem de avaliação e vulnerabilidades de segurança da estrutura. Consulte o Capítulo 7 para obter mais detalhes.
- Dados de entrada — diz respeito ao fornecimento de dados de treinamento para apoiar o ML e ao fornecimento de dados de produção usados pelo modelo em seu ambiente operacional. Exemplos de riscos de produto incluem dados de treinamento tendenciosos, defeitos no pipeline de dados e dados de treinamento não representativos. Consulte o Capítulo 5 para obter mais detalhes.
- Modelo — diz respeito ao modelo ML gerado. Exemplos de riscos de produto incluem a falha em atingir os requisitos de performance funcional do ML, um modelo overfitting e a suscetibilidade a exemplos contraditórios. Consulte o Capítulo 6 para obter mais detalhes.

Outra forma de categorizar os riscos associados à AI é de acordo com as características de qualidade definidas na ISO/IEC 25059.

Várias formas de teste estão disponíveis para abordar esses riscos e são projetadas especificamente para o teste de MLS. Por exemplo, teste de pipeline de dados, teste contraditório e revisão da adequação do algoritmo/modelo. Este syllabus abrange vários desses tópicos nos capítulos 5 ,6 e7 .

5 Teste de Dados de Entrada para Sistemas de Machine Learning – 180 minutos

Palavras-chave

Teste de pipeline de dados, teste de representatividade dos dados, teste de restrições de conjuntos de dados, teste de dados de entrada, label correctness testing, revisão, teste de viés

Palavras-chave específicas de AI

Análise de impacto desigual, anotação múltipla

Objetivos de aprendizagem para o Capítulo 5:

5.1 Teste de Dados de Entrada para Sistemas de Machine Learning

AI-5.1.1 (K2) Apresentar exemplos de abordagens de teste utilizadas para a mitigação de risco dos dados de entrada de um sistema de machine learning

AI-5.1.2 (K2) Explicar como testar a presença de viés

AI-5.1.3 (K2) Resumir as várias formas de teste de pipeline de dados

AI-5.1.4 (K2) Explicar como realizar o teste de representatividade dos dados

AI-5.1.5 (K3) Aplicar testes de restrições de conjuntos de dados

AI-5.1.6 (K2) Explicar o label correctness testing

HO-5.1.7 (H2) Realizar testes de dados de entrada para conjuntos de dados de ML

5.1 Teste de Dados de Entrada para Sistemas de Machine Learning

O objetivo do teste de dados de entrada é confirmar que os dados utilizados pelo MLS para treinamento, teste e previsão sejam de qualidade suficiente (consulte3.2). Isso inclui revisões, técnicas estatísticas (por exemplo, teste de dados para verificar se há viés), EDA dos dados de treinamento e testes estáticos e dinâmicos do pipeline de dados.

5.1.1 Riscos e medidas de mitigação dos dados de entrada

A tabela a seguir lista exemplos de riscos de dados de entrada e os testes correspondentes que poderiam ser usados para mitigação de risco:

Riscos potenciais	**Possíveis medidas de mitigação de risco**
Defeitos nos dados de treinamento que levam a viés Problemas no algoritmo, modelo ou estrutura de desenvolvimento de ML que introduzem injustiça sistêmica	Teste de viés – consulte 5.1.2
Dados de treinamento obtidos de fontes não confiáveis Dados mal gerenciados	Teste de proveniência de dados
Dados de treinamento contaminados	Teste A/B – consulte 6.1.9 Teste de proveniência de dados EDA – consulte 3.2.1 Ataques realizados no âmbito de exercícios de simulação de ataques (red teaming) – ver 4.2.2
Conjunto de dados internamente inconsistente Dados fora do intervalo Tipos de dados incorretos	Teste de restrições de conjuntos de dados – consulte 5.1.5
Seleção de características subótima	Teste de características
Conjunto de dados desequilibrado devido à cobertura insuficiente de todas as classes-alvo Conjunto de dados distorcido por aumento de dados Dados ausentes Dados de treinamento focados em um subconjunto de todos os casos de uso Gama completa de valores não abrangida no conjunto de dados	Teste de representatividade dos dados – consulte 5.1.4
Diretrizes de rotulagem inadequadas Dados ambíguos Anotação inadequada levando a rótulos imprecisos ou inconsistentes	Label correctness testing – consulte 5.1.6
Projeto ou integração inadequados, levando a falhas no pipeline de dados Defeitos na qualidade dos dados comprometendo os resultados do pipeline de dados	Teste de pipeline de dados – consulte 5.1.3

Riscos potenciais	Possíveis medidas de mitigação de risco
Deterioração da performance na utilização operacional do pipeline de dados Violações de segurança ou alterações descontroladas que afetam o pipeline de dados	

5.1.2 Teste de viés

O viés em um MLS refere-se a diferenças não aleatórias e injustas no tratamento com base em atributos sensíveis, como gênero, idade ou raça, o que muitas vezes é ilegal e torna o sistema discriminatório.

O teste de viés em um MLS envolve a compreensão de suas fontes potenciais, que incluem principalmente:

- Defeitos nos dados de treinamento, como falta de representatividade, distorções históricas ou adulteração deliberada (viés de dados).
- Defeitos no algoritmo, modelo ou estrutura de desenvolvimento que introduzem injustiça sistêmica, como um algoritmo que utiliza um limiar de decisão para pontuações de crédito em um sistema de aprovação de empréstimos (viés algorítmico).

As abordagens de teste para detectar viés incluem o seguinte:

- Realizar uma revisão de o es do fluxo de trabalho geral de machine learning, especialmente a preparação de dados, para identificar o risco de introdução de viés.
- Revisões da documentação do conjunto de dados para identificar e mitigar possíveis fontes de injustiça, examinando como os dados foram coletados, anotados e quais populações estão representadas.
- Análise estática tanto dos programas de preparação de dados quanto do código de implementação do modelo, a fim de identificar antipadrões ou o tratamento inadequado de atributos confidenciais.
- Análise exploratória (EDA) dos dados de treinamento, utilizando métodos de visualização e agrupamento para revelar dados desequilibrados, distribuições distorcidas ou agrupamentos anômalos entre diferentes atributos sensíveis.
- Teste dinâmico para ajudar a detectar viés em um modelo ML, alimentando o sistema com um conjunto de dados conhecido, imparcial e representativo, e analisando suas previsões quanto a diferenças estatisticamente significativas nos resultados entre vários grupos sensíveis. Isso identifica viés nas saídas do modelo, independentemente de o viés ter sido introduzido durante o treinamento de dados ou durante o desenvolvimento do modelo.
- Label correctness testing (ver 5.1.6) para identificar rótulos incorretos que causam o aprendizado de associações incorretas entre atributos e resultados para atributos sensíveis específicos.
- Análise de impacto desigual :
    1. Identifique atributos sensíveis para o modelo.
    2. Gerar contrafactuais para os atributos sensíveis. (por exemplo, cenários em que o gênero é alterado de masculino para feminino em um pedido de empréstimo).
    3. Gerar a saída do modelo apresentando-lhe os contrafactuais.
    4. Analise os resultados de vários testes para obter um resultado estatisticamente significativo, o que determina se a alteração de um atributo sensível faz com que os resultados do modelo mudem, indicando a presença de viés.

Observação: a análise de impacto desigual também pode ser aplicada a combinações de atributos sensíveis, permitindo a identificação de viés oculto associado a essas combinações. No entanto, certifique-se de que os exemplos contrafactuais não sejam irrealistas, pois o modelo pode então responder a eles em vez de a qualquer viés subjacente.

5.1.3 Teste de pipeline de dados

O teste eficaz do pipeline de dados é essencial não apenas para confirmar a confiabilidade e o desempenho de sistemas orientados a dados, mas também para manter alta qualidade dos dados em todo o fluxo de trabalho de ML (ver 3.1.2).

É empregada uma abordagem em camadas, começando com revisões do projeto do pipeline de dados durante a fase de projeto.

O teste de componentes inclui o teste de componentes de ingestão de dados, scripts de transformação e interfaces de sensores. Esse teste utiliza revisões de código, análise estática e testes específicos de hardware para verificar a confiabilidade da captura de dados. Os testes de componentes validam a lógica de transformação de dados, testam a implementação de regras de validação de dados, confirmam o tratamento robusto de erros e testam vulnerabilidades que poderiam ser exploradas para introduzir malware ou dados corrompidos.

O teste de integração de componentes verifica o fluxo contínuo de dados entre interfaces internas e a interpretação correta dos dados à medida que se movem pelo pipeline. Ele detecta defeitos decorrentes de interfaces incompatíveis ou suposições incorretas entre componentes.

Os testes de sistema avaliam o pipeline totalmente montado, começando com testes de fumaça para confirmar a funcionalidade básica do pipeline. Os testes funcionais verificam a conformidade do pipeline com os requisitos especificados, incluindo transformações e roteamento de dados. Os testes não funcionais avaliam o desempenho sob carga, a escalabilidade para lidar com volumes crescentes de dados e as medidas de segurança para proteger a integridade dos dados. Os testes de injeção de falhas medem a robustez do pipeline simulando entradas de dados defeituosas, avaliando a capacidade do sistema de manter a integridade dos dados quando confrontado com dados inesperados ou corrompidos. Os testes back-to-back (ver 6.1.10) comparam o pipeline operacional com o pipeline de treinamento para verificar a funcionalidade consistente.

O teste de integração do sistema verifica a interação correta entre o pipeline de dados e sistemas ou serviços externos, incluindo fontes de dados, plataformas de armazenamento, ferramentas de monitoramento e consumidores a jusante, como modelos ML.

Os testes em produção são realizados no sistema operacional. O teste back-to-back verifica se o desempenho é consistente ou melhorado em comparação com versões anteriores. O teste A/B (ver 6.1.9) permite comparar novas iterações do pipeline com as linhas de base, validando melhorias e confirmando que não há degradação no fluxo de dados em tempo real. Ferramentas podem ser integradas para monitorar e observar continuamente o comportamento do modelo, o desempenho e possíveis defeitos em tempo real.

As revisões de gerenciamento de configuração ajudam a verificar se as versões corretas do código do pipeline, as configurações e os conjuntos de dados são usados no treinamento, nos testes e na produção.

Por fim, as estratégias de teste devem estar alinhadas com o objetivo do pipeline. Pipelines de treinamento, frequentemente protótipos exploratórios, têm prioridades diferentes das de pipelines operacionais robustos. O teste de pipelines de treinamento pode se concentrar na integridade dos dados, enquanto o teste de pipelines operacionais prioriza a confiabilidade, a performance e a manutenibilidade.

5.1.4 Teste de representatividade dos dados

O teste de representatividade dos dados determina em que medida as características dos conjuntos de dados usados para treinamento, validação e teste de modelos ML correspondem aos dados do mundo

real que o modelo ML encontrará em operação. Esse teste aborda várias formas de deturpação, incluindo conjuntos de dados distorcidos, dados ausentes, distribuições desproporcionais de características, cobertura inadequada de cenários operacionais e representação desequilibrada de classes (consulte5.1.1).

Esse teste normalmente inclui as seguintes etapas:

1. Definir a população-alvo:
    - Compreender os casos de uso pretendidos e o contexto operacional do MLS
    - Analisar as características dos usuários finais e dos ambientes operacionais
    - Identificar as distribuições de dados operacionais esperadas e os casos de borda críticos por meio de:
        - consultar especialistas na área para compreender os padrões de dados do mundo real
        - analisar dados de sistemas existentes ou de aplicativos semelhantes
        - utilizar conjuntos de dados de referência de fontes confiáveis (por exemplo, NIST, bancos de dados do setor)
    - Aplicar amostragem estratificada aos dados de referência representativos para criar uma linha de base que abranja todos os subgrupos relevantes no domínio-alvo
2. Analise as características dos dados:
    - Aplique a EDA (consulte3.2.1) aos:
        - conjuntos de dados de treinamento/teste que estão sendo avaliados quanto à representatividade
        - conjunto de dados de referência que representa os dados operacionais esperados
        - visualizar distribuições por meio de histogramas, gráficos de dispersão e outras técnicas gráficas
    - Examine as relações entre características, e as correlações em particular, para identificar padrões que devem ser preservados
    - Identificar possíveis anomalias, lacunas ou concentrações incomuns nos dados
3. Aplicar técnicas de avaliação estatística:
    - Utilizar testes estatísticos formais, como o qui-quadrado e o Kolmogorov-Smirnov, para comparar distribuições [STATS]
    - Verifique se há desequilíbrios nos dados, especialmente em problemas de classificação
    - Verifique a cobertura adequada tanto de cenários típicos quanto de casos de borda/limítrofes

O teste de representatividade dos dados deve ser realizado antes do treinamento do modelo para evitar a construção de modelos com base em dados não representativos. Após a implantação, as propriedades dos dados de entrada operacionais devem ser monitoradas continuamente para detectar mudanças que possam indicar desvio dos dados em relação às distribuições dos dados de treinamento originais (consulte6.1.7 sobre testes de desvio).

5.1.5 Teste de restrições de conjuntos de dados

O teste de restrições de conjuntos de dados verifica se os dados em um conjunto de dados seguem regras ou restrições predefinidas. O objetivo é confirmar a integridade e a consistência dos dados usados em ML. Por exemplo, um teste de consistência dos valores em um conjunto de dados poderia ser realizado.

Restrições sobre os dados são normalmente encontradas em esquemas de banco de dados. Um esquema de banco de dados define a estrutura, os tipos e as relações dos dados. Da mesma forma, para conjuntos de dados de ML, pode-se definir um conjunto de restrições que atuam como um modelo lógico dos dados no conjunto de dados, o qual deve ser satisfeito se os dados estiverem corretos. Existem várias maneiras de categorizar as restrições de conjuntos de dados. Uma restrição pode ser aplicada a um único valor de um atributo, encontrado em uma única instância (restrição de valor único), por exemplo:

- Ausência – testes para valores e atributos ausentes.
- Intervalo – realiza um teste para verificar se um valor está dentro de um determinado intervalo.
- Tipo – teste que verifica se um valor de atributo corresponde ao tipo especificado (por exemplo, se um atributo for especificado como um inteiro, o valor fornecido não é uma string ou um número real).

Alternativamente, uma restrição pode ser aplicada a vários valores, normalmente considerando os valores de um único atributo em várias instâncias (uma restrição de múltiplos valores), por exemplo:

- Soma – teste que verifica se a soma de todos os valores é igual, superior ou não superior a um valor especificado (por exemplo, o valor total de todos os pontos atribuídos numa corrida de Fórmula 1 não pode exceder 102 e deve ser superior a 50,5 pontos).
- Contagem – teste que verifica se a contagem de todos os valores não nulos para atributos ou instâncias é igual, superior ou não superior a um valor especificado.
- Duplicate – realiza testes para verificar se há valores de atributos ou instâncias idênticos ou quase idênticos em um conjunto de dados e impõe um limite ao número permitido (geralmente zero).
- Útil – o teste verifica se um atributo contém alguns valores repetidos, como se cada entrada fosse única (como um ID ou carimbo de data/hora); normalmente, isso não fornece padrões úteis a serem aprendidos por um modelo ML.
- Outlier – identifica quaisquer valores que possam ser considerados outliers estatísticos.

Uma forma especial de restrição multivalor pode comparar valores diferentes (uma restrição de comparação), por exemplo:

- Maior que – teste que verifica se um valor de um atributo é maior que um valor de um segundo atributo (por exemplo, a contagem de linhas de código de um programa excede a contagem de linhas de código com defeitos).
- Correlacionar – realiza-se um teste para verificar se os valores de um atributo se correlacionam com os valores de um segundo atributo (por exemplo, todos os alunos com um valor de atributo de notas pelo menos 1,33 desvios padrão acima da média também têm um valor de ‘A’ para seu atributo de nota).

O teste dos dados em relação às restrições definidas pode ser realizado manualmente. No entanto, a escala da atividade e o tamanho tipicamente grande do conjunto de dados exigiriam que isso fosse automatizado como parte do pipeline de dados. Quando implementada como parte do pipeline, a ferramenta que realiza o teste de restrições de conjuntos de dados pode fornecer relatórios aos cientistas de dados (para anomalias nos dados de treinamento) ou à equipe de operações (para problemas nos dados operacionais).

5.1.6 Label correctness testing

O teste de correção de rótulos de dados é essencial no aprendizado supervisionado. Rótulos imprecisos ou inconsistentes prejudicam diretamente a performance e a generalização dos modelos ML.

Abordagens comuns para o label correctness testing incluem:

- Revisão por especialistas: Especialistas na área ou anotadores treinados realizam a revisão manualmente de uma amostra de dados rotulados. Eles avaliam a precisão dos rótulos usando seu conhecimento da área e diretrizes.
- Anotação múltipla: os pontos de dados são rotulados de forma independente por vários anotadores e comparados por meio de um teste back-to-back (ver 6.1.10). As divergências destacam defeitos que precisam ser investigados e resolvidos. A concordância entre anotadores (IAA) pode ser mediada utilizando métricas como o Kappa de Cohen ou a simples porcentagem de concordância [STATS]. Pontuações baixas de IAA podem indicar defeitos nas diretrizes de rotulagem, dados ambíguos ou anotação de baixa qualidade.
- Priorização baseada no risco para revisão e anotação: Tanto as revisões de especialistas quanto as anotações múltiplas podem ser direcionadas por meio de uma abordagem baseada no risco, a fim de identificar amostras para revisão e anotações múltiplas. A priorização pode basear-se na probabilidade de rotulagem incorreta, como no caso de pontos de dados ambíguos (por exemplo, quando não está claro a qual categoria pertencem ou quando estão próximos de um limite entre classes), e nos dados que têm maior probabilidade de afetar o sucesso ou a segurança da aplicação.
- Análise de distribuição de dados: Quando existem conjuntos de dados comparáveis, comparar a distribuição de rótulos do conjunto de dados em teste com conjuntos de dados semelhantes pode revelar anomalias e rótulos potencialmente incorretos.
- Testes automatizados baseados em regras: Testes automatizados podem ser implementados com base em regras ou restrições de rótulos predefinidas para determinadas tarefas. Por exemplo, verificar se as caixas delimitadoras (retângulos usados para localizar objetos em imagens) não se sobrepõem ou se estendem além dos limites da imagem.
- Análise de perda do modelo: Pontos de dados que apresentam alta perda durante o treinamento do modelo — o que significa que as previsões do modelo para esses pontos se desviam substancialmente de seus rótulos verdadeiros — podem indicar rotulagem incorreta. Alta perda reflete um erro significativo, indicando que o modelo tem dificuldade para aprender o rótulo atribuído e apontando para um possível defeito.
- Análise da pontuação de confiança do modelo: Pontos de dados com baixa confiança de previsão de um modelo treinado pode estar mal rotulados, ser ambíguos ou estar fora da distribuição dos dados de treinamento do modelo.

O uso de uma combinação de abordagens costuma ser mais eficaz. Por exemplo, revisões de especialistas são valiosas para estabelecer diretrizes claras de rotulagem desde o início. Pontuações de IAA de múltiplas anotações podem então refinar o processo de rotulagem. Posteriormente, abordagens baseadas em modelos podem identificar ainda mais possíveis defeitos de rotulagem à medida que o modelo continua a se desenvolver.

5.1.7 Exercício prático: Teste de dados de entrada

Para um determinado conjunto de dados (por exemplo, dados estruturados e tabulares), realize testes de dados de entrada para verificar itens como dados ausentes, dados duplicados e outliers.

6 Teste de Modelos para Sistemas de Machine Learning – 225 minutos

Palavras-chave

Teste A/B, teste contraditório, teste back-to-back, desvio de conceito, desvio de dados, teste de desvio, testes metamórficos, performance funcional do ML, teste do modelo ML, revisão

Palavras-chave específicas de AI

overfitting, underfitting

Objetivos de aprendizagem do Capítulo 6:

6.1 Testes de Modelos para Sistemas de Machine Learning

AI-6.1.1 (K2) Dar exemplos de abordagens de teste utilizadas para a mitigação de risco de modelos ML

AI-6.1.2 (K2) Explicar o objetivo e o foco da revisão da documentação de modelos ML

AI-6.1.3 (K2) Explicar como são realizados os testes de performance funcional do ML para sistemas de machine learning probabilístico

AI-6.1.4 (K2) Resumir os testes contraditórios de sistemas de machine learning

AI-6.1.5 (K3) Utilizar testes metamórficos para derivar casos de teste para um determinado cenário

HO-6.1.6 (H2) Aplicar testes metamórficos

AI-6.1.7 (K2) Explicar como o teste de desvio é utilizado em sistemas operacionais de machine learning

AI-6.1.8 (K2) Explicar como o overfitting e o underfitting são detectados por meio de testes

AI-6.1.9 (K2) Explicar como o teste A/B é utilizado no contexto de sistemas de machine learning

AI-6.1.10 (K2) Explicar como o teste back-to-back é utilizado no contexto de sistemas de machine learning

6.1 Testes de Modelos para Sistemas de Machine Learning

O teste do modelo ML envolve lidar com riscos específicos. Isso inclui riscos funcionais, como viés, overfitting e vulnerabilidades adversárias, bem como riscos não funcionais, como falta de robustez da AI e eficiência de performance, e riscos de implantação.

6.1.1 Riscos e medidas de mitigação no de modelos de machine learning

A tabela a seguir apresenta exemplos de riscos associados a modelos ML e os testes correspondentes que poderiam ser utilizados para a mitigação de risco:

Risco potencial	**Possível Mitigação de Risco**
Modelo ML tendencioso ou injusto	Teste de viés – consulte 5.1.2
Modelo antiético	Teste de sistemas éticos
Exemplos contraditórios	Teste contraditório – consulte 6.1.4
Modelo overfitting	Teste de overfitting – ver 6.1.8
Modelo underfitting	Teste de underfitting – consulte 6.1.8
Operação de dados inaceitável Desvio de conceito inaceitável	Teste de desvio – consulte 6.1.7
O modelo causa efeitos colaterais	Teste de efeitos colaterais
O modelo apresenta hacking de recompensa	Teste de hacking de recompensa
Defeito na API do modelo	Teste de API – consulte 7.1.2
Falha em atingir as medições de performance exigidas para o modelo ML (por exemplo, falta de precisão, recall)	Teste de performance funcional do ML – consulte 6.1.3
Inexatidão funcional Defeitos não funcionais	Testes metamórficos – consulte 6.1.5
Problema de oráculo de teste	Testes metamórficos – consulte 6.1.5 Teste back-to-back – consulte 6.1.10 Teste A/B – consulte 6.1.9
Requisitos de sistema inadequados	Revisão de requisitos Red teaming – consulte 4.2.2 Teste exploratório
Falta de robustez do modelo de AI devido a entradas inesperadas	Teste contraditório – consulte 6.1.4 Teste de fuzz
Eficiência inadequada da performance do modelo ML	Teste de performance
Documentação do modelo deficiente (por exemplo, função, precisão, interface)	Revisão da documentação do modelo – consulte 6.1.2
Atualizações do modelo introduzem defeitos	Teste back-to-back – consulte 6.1.10

Risco potencial	Possível Mitigação de Risco
Atualizações do modelo reduzem a performance funcional do modelo ML	Teste A/B – consulte 6.1.9
A implantação do modelo atualizado causa falha imediata	Teste de fumaça
A implantação do modelo atualizado causa regressão	Teste de regressão
Vulnerabilidades de segurança Vulnerabilidades de segurança Violações de privacidade Resultados prejudiciais ou indesejáveis (por exemplo, opiniões racistas, orientações perigosas)	Red teaming – consulte 4.2.2

6.1.2 Documentação e revisão do modelo de machine learning

A documentação abrangente para modelos ML não é apenas uma formalidade; é um recurso essencial. Ao contrário de sistemas em que o código-fonte pode ser submetido à inspeção direta, os MLS apresentam desafios únicos devido à compreensibilidade limitada do código gerado por máquina, à natureza inerente de caixa preta dos modelos e à sua dependência de dados. Os modelos também são atualizados com frequência, de modo que a documentação se torna a principal ferramenta para desenvolvedores, testadores e reguladores compreenderem, avaliarem e confiarem nos sistemas baseados em AI. A transparência desempenha um papel central para possibilitar essa compreensão, permitindo que os stakeholders rastreiem o comportamento do modelo, a lógica de decisão e a linhagem dos dados ao longo do ciclo de vida da AI.

A documentação padronizada melhora a comunicação, apoia a tomada de decisões informadas e verifica a qualidade e a manutenibilidade dos modelos ML. Ela é cada vez mais importante para a conformidade regulatória, como evidenciado por estruturas como a Lei de AI da UE, que enfatiza obrigações de transparência que exigem documentação clara das decisões do modelo, limitações e medidas de interpretabilidade. Para sistemas de alto risco, passar por uma auditoria de documentação é frequentemente um pré-requisito para a implantação, enquanto, para todos os sistemas, uma revisão completa ajuda a verificar a qualidade.

Existem várias estruturas de documentação, incluindo Model Cards, que fornecem visões gerais concisas dos usos pretendidos de um modelo, resultados de avaliação e considerações éticas [MODEL_DOC], e Datasheets para conjuntos de dados de aprendizagem automática, que oferecem formatos padronizados para descrever conjuntos de dados, incluindo sua motivação, composição, processo de coleta e usos [DATA_DOC].

A lista a seguir descreve o conteúdo típico esperado para uma documentação abrangente de modelos ML, estruturada de forma a poder ser usada como uma lista de verificação prática tanto por desenvolvedores quanto por testadores para ajudar a alcançar completude, clareza e testabilidade:

- Geral: Identificadores, descrição, desenvolvedor, versão, data, contato, licença, requisitos de hardware
- Projeto: Suposições, decisões técnicas, algoritmo de ML
- Uso: Finalidade pretendida, usos primários/secundários, usuários, abordagem de autoaprendizagem, viés, ética, segurança, transparência, limites, plataforma, operação de dados, desvio de conceito
- Conjuntos de dados: Características, fonte, coleta, disponibilidade, pré-processamento, uso, conteúdo, rótulos, tamanho, privacidade, segurança, viés/equidade, restrições

- Testes: Detalhes do conjunto de dados de teste, independência do teste, resultados dos testes, atividades de teste (por exemplo, funcionais, adversariais)
- Funcional: Medições, conjunto de dados de validação, limites, performance real.
- Não-funcional: Escalabilidade, confiabilidade, disponibilidade, eficiência de performance (por exemplo, latência, utilização de recurso), manutenibilidade, robustez da AI.
- Operacional: Plano de implantação, ambiente de implantação, recursos computacionais, métricas/alertas de monitoramento, estratégia de retreinamento, plano de atualização/reversão do modelo, plano de descontinuação, segurança (riscos adversários), métodos de explicabilidade

A revisão da documentação em relação a essas listas de verificação é uma atividade central de teste, com o objetivo de:

- Identificar informações ausentes, imprecisões e inconsistências.
- Melhorar a clareza e a legibilidade.
- Aumentar a manutenibilidade, identificando pontos de, onde a documentação precisa de melhorias.
- Fornecer informações suficientes para as atividades de teste e implantação.
- Verificar se todos os requisitos regulatórios relevantes foram atendidos.

6.1.3 Teste de performance funcional do ML em sistemas de machine learning probabilísticos

O teste de performance funcional do ML avalia a performance do modelo ML nas funções pretendidas, realizando medições de métricas como precisão, recall, precisão e F1-score (ver 3.3) e comparando os resultados com critérios de aceite definidos.

Este teste para MLS probabilístico vai além de um simples status de aprovado/reprovado para realizar uma medição estatística do desempenho de um modelo em relação aos seus critérios de aceite [STATS]. Essa abordagem é necessária para lidar com o não determinismo inerente ao MLS, avaliando o comportamento em um conjunto de dados grande e representativo (ver 4.1.2).

Uma pré-condição para esse teste é ter critérios de aceite definidos em termos estatísticos. Em vez de uma meta simples, um requisito pode especificar uma métrica de desempenho, uma margem de erro (MoE) e um nível de confiança (CL). Tal requisito pode ser usado para determinar o número mínimo de testes necessários. À medida que o número de testes aumenta, há duas opções:

- Fixar o CL e a MoE diminuirá. Isso significa que o resultado do teste se torna mais preciso.
- Se fixarmos o MoE, o IC aumentará. Isso significa que o resultado do teste se torna mais confiável.

Por exemplo, um critério poderia ser “98% de precisão com um MoE de ±4% a um IC de 95%”. Para confirmar que a precisão medida tem um MoE máximo de ±4% no CL de 95%, é necessária uma amostra de 601 casos de teste. Isso é calculado usando a fórmula de tamanho de amostra para estimar uma proporção populacional. Esse tamanho de amostra se baseia na suposição conservadora de que a precisão real pode estar em qualquer ponto entre 0% e 100%, o que produz a maior incerteza possível, mas ainda garante o MoE de ±4% em todas as condições. Para atingir a precisão alvo de 98%, pelo menos 589 dos 601 casos de teste devem ser aprovados. Se, após a execução de todos os 601 casos de teste, a precisão observada for de 98%, então o MoE medido no IC de 95% será menor que ±4% (aproximadamente ±1,1%), pois a variância é menor em níveis elevados de precisão. O MoE é calculado usando a fórmula da margem de erro para uma proporção amostral. Isso significa que, durante os testes, nem sempre é necessário executar todos os 601 casos de teste. Os testes sequenciais fornecem uma estrutura estatística formal para essa interrupção antecipada. Em vez de sempre executar a amostra fixa

completa, essas abordagens analisam os resultados à medida que se acumulam e interrompem antecipadamente quando evidências suficientes apoiam a meta de precisão (ou a rejeitam). Se a precisão observada permanecer consistentemente alta (por exemplo, não inferior a 98%), a incerteza estatística diminui à medida que mais testes são executados. Nessa situação, o MoE exigido de ±4% no IC de 95% é atingido após cerca de 170 casos de teste, permitindo que os testes sejam concluídos mais cedo.

NOTA: As fórmulas e cálculos numéricos neste exemplo (para tamanho da amostra, margem de erro e intervalos de confiança) são fornecidos apenas para ilustração e compreensão; os candidatos não serão obrigados a derivar ou calcular esses valores usando fórmulas estatísticas na prova.

Para sistemas críticos de segurança, pode ser utilizado um requisito de confiabilidade mais rigoroso, como “99% de confiabilidade com 95% de confiança”, o que exigiria 299 casos de teste, todos os quais devem ser aprovados para atender aos critérios.

Para validar esses critérios, os testes exigem um grande conjunto de dados de teste que seja completamente independente dos conjuntos de dados de treinamento e validação. Esse conjunto de dados de teste deve ser uma amostra representativa do domínio de entrada operacional (ver 5.1.4) para verificar se a avaliação reflete as condições do mundo real. Os casos de teste são então executados utilizando o modelo ML dentro de uma estrutura de desenvolvimento de ML capaz de realizar análises estatísticas.

Por fim, os resultados agregados são interpretados e relatados com confiança estatística, não como uma simples proporção de aprovação/reprovação. Um relatório de teste final indicaria, por exemplo: “O modelo atingiu uma precisão de 94% ±4% no IC de 95%.” Isso permite que as partes interessadas compreendam a faixa de performance operacional esperada e tomem uma decisão informada sobre se a performance funcional do ML do modelo é aceitável para implantação.

6.1.4 Teste contraditório de sistemas de machine learning

O teste contraditório envolve fornecer deliberadamente ao modelo perturbações nos dados de entrada que muitas vezes são imperceptíveis aos seres humanos. Essas entradas são projetadas para fazer com que o modelo faça previsões incorretas e, se bem-sucedidas, são chamadas de exemplos contraditórios. Assim, as entradas de teste para o teste contraditório, e, portanto, os potenciais exemplos contraditórios, geralmente consistem em versões ligeiramente modificadas de entradas legítimas que fazem com que o modelo as classifique incorretamente.

A identificação de vulnerabilidades por meio de testes contraditórios permite que os desenvolvedores incorporem salvaguardas e tornem o modelo mais robusto contra exemplos contraditórios. Estes podem ser exemplos contraditórios acidentais que o modelo encontra, ou podem ser ataques contraditórios que envolvem o uso malicioso de exemplos contraditórios . Gerar entradas de teste eficazes para testes contraditórios é tecnicamente complexo, e manter-se atualizado com as técnicas de ataque em evolução é um desafio contínuo.

Os testes contraditórios podem ser realizados usando testes caixa-preta, com foco no comportamento de entrada e saída do modelo, sem exigir conhecimento de seu funcionamento interno. Isso pode ser alcançado criando-se um modelo equivalente cujos detalhes internos sejam conhecidos, a partir do qual entradas de teste contraditórias possam ser criadas por meio do conhecimento de seu funcionamento interno. Então, com base na suposição de que modelos equivalentes compartilham limites de classificação (ou seja, transferibilidade), os testes contraditórios podem ser aplicados ao modelo original. Em contrapartida, uma abordagem de força bruta utilizando muitos testes podem ser empregadas na esperança de que alguns testes aleatórios coincidam com um exemplo contraditório.

Os testes contraditórios também podem ser realizados utilizando testes caixa-branca. Compreender o funcionamento interno de um modelo ML (arquitetura, parâmetros, processo de treinamento) normalmente facilita a criação de exemplos contraditórios.

Os testes contraditórios podem ser realizados manualmente, criando exemplos contraditórios específicos, ou por meio de algoritmos automatizados que geram muitas variações para descobrir entradas contraditórias efetivas.

6.1.5 Testes metamórficos

O teste metamórfico (MT) é uma técnica de teste na qual novos casos de teste (derivados) são obtidos a partir de um caso de teste de origem que já passou . Um ou mais casos de teste de acompanhamento são gerados através da alteração (metamorfose) do caso de teste de origem, utilizando uma relação metamórfica (MR). A MR baseia-se em uma propriedade de uma função exigida do objeto de teste e descreve como uma alteração nas entradas de um caso de teste se reflete nos resultados esperados do mesmo caso de teste.

A MT pode ser usada para a maioria dos objetos de teste e pode ser aplicada tanto ao teste funcional quanto ao teste não funcional . Os testadores verificam objetivos do teste, tais como consistência (as saídas se alinham entre entradas relacionadas), monotonicidade (as saídas mudam direcionalmente com a entrada) e invariância (as saídas permanecem estáveis sob perturbações). É particularmente útil quando a geração de resultados esperados é problemática devido à indisponibilidade de um oráculo de teste acessível . Esse é o caso de alguns MLS que utilizam big data, ou de sistemas em que os testadores não têm clareza sobre como o modelo ML deriva suas previsões, o que é comum. Na área de AI e ML, os testes metamórficos têm sido utilizados para testar reconhecimento de imagens, mecanismos de busca, otimização de rotas e reconhecimento de voz.

O MT é normalmente preferido aos testes tradicionais baseados em oráculos quando:

- não existem resultados esperados com alta confiabilidade devido à opacidade do modelo ou à escala dos dados;
- o sistema é uma caixa preta; ou
- propriedades relacionais (não valores absolutos) são suficientes para garantir a confiança.

O MT geralmente se baseia em um caso de teste de origem que passou e pode ser útil quando não é possível gerar um resultado esperado. Por exemplo, quando o programa implementa uma função que é complexa demais para um testador humano replicar e usar como oráculo de teste, como em alguns MLS complexos. Nessa situação, o MT pode ser usado para gerar casos de teste que, quando executados, criarão um conjunto de resultados em que as relações entre os resultados (em vez de seus valores reais) são verificadas quanto à validade. Com essa forma de MT, se as relações entre as saídas de teste forem válidas, isso proporciona maior confiança no programa. Por exemplo, um MLS de avaliação de risco que prevê a idade na morte, em que o aumento do número de cigarros fumados deve diminuir a previsão (monotonicidade).

Os testadores derivam as MRs a partir do conhecimento do domínio, dos requisitos ou das propriedades do domínio (por exemplo, leis da física). As MRs podem ser validadas por revisão de especialistas, executando-as em modelos de referência e verificando a cobertura de casos de borda.

MRs incorretas (por exemplo, ignorando interações complexas entre variáveis) ou conjuntos incompletos de MRs podem levar a uma falsa confiança. O MT detecta falhas relacionais, mas não todos os erros absolutos; portanto, deve ser utilizado em combinação com outras técnicas de teste.

6.1.6 Exercício prático: Aplique os testes metamórficos

Neste exercício, os alunos adquirirão experiência prática no seguinte:

- Derivar várias MRs para um determinado MLS. Essas MRs devem incluir algumas em que os resultados esperados dos casos de teste de origem e de acompanhamento sejam os mesmos, e outras em que sejam diferentes.

- Gerar casos de teste de origem para o MLS. Não é necessário garantir que eles passem, mas os alunos devem ser lembrados das limitações do MT quando os casos de teste de origem que passaram não estiverem disponíveis.
- Usar os MRs derivados e os casos de teste de origem gerados para derivar casos de teste de acompanhamento.
- Executar os casos de teste de acompanhamento.

6.1.7 Teste de Desvio

O teste de desvio pode ser usado para identificar duas formas de desvio no MLS operacional:

- Desvio de dados - ocorre quando as propriedades estatísticas dos dados de entrada operacionais mudam ao longo do tempo. Por exemplo, os dados de entrada agora são significativamente diferentes dos dados com os quais o modelo foi treinado, devido a fatores como mudanças no comportamento do usuário ou sazonalidade. Por exemplo, um filtro de spam encontra novos tipos de ataques de phishing que não existiam durante seu treinamento.
- Desvio de conceito - ocorre quando a relação entre os dados de entrada e a saída correta muda ao longo do tempo. Isso significa que os padrões ou regras originalmente aprendidas pelo modelo não refletem mais a realidade atual. Por exemplo, devido a novas regulamentações financeiras, um tipo de transação anteriormente considerado de “baixo risco” pode agora ser classificado como de “alto risco”. O significado dos dados mudou, fazendo com que os limites de decisão aprendidos pelo modelo se tornassem desatualizados, levando a uma queda em sua precisão preditiva.

O teste dinâmico depende da disponibilidade de feedback dos usuários, que fornece a verdade de referência atual. Essa verdade de referência atual é comparada com a saída do modelo, e a diferença entre as duas é determinada e comparada com um valor limite. O feedback do usuário pode ser direto ou indireto. Para um sistema de recomendação de filmes, um exemplo de feedback direto é quando o usuário avalia uma recomendação. Um exemplo de feedback indireto para o mesmo sistema seria extraído dos dados sobre os filmes que o usuário assistiu.

O teste de desvio estático não depende dos dados reais atuais, mas compara as propriedades estatísticas das distribuições dos dados de entrada e dos dados de saída previstos, utilizando um teste como o de Kolmogorov-Smirnov [STATS]. Uma diferença significativa em qualquer uma dessas distribuições é um indicador de que ocorreu um desvio.

6.1.8 Teste de overfitting e underfitting

O overfitting e o underfitting são dois dos três resultados possíveis encontrados em modelos ML, sendo o terceiro um modelo com “ajuste adequado”. Os testes de overfitting e underfitting devem ocorrer durante o treinamento, a avaliação e o ajuste.

O overfitting ocorre quando um modelo aprende os dados de treinamento muito bem, a ponto de capturar ruído nos dados em vez do padrão subjacente. Isso resulta em uma generalização deficiente de dados novos e não vistos.

Para testar o overfitting, a performance funcional do ML do modelo é avaliada em um conjunto de dados de teste separado que não foi usado durante o treinamento. Esse conjunto de dados de teste deve incluir alguns exemplos menos comuns, improváveis de terem sido usados durante o treinamento. O modelo pode estar sofrendo de overfitting se tiver uma performance significativamente pior no conjunto de dados de teste do que no conjunto de dados de validação.

O underfitting ocorre quando um modelo é simples demais para capturar a estrutura subjacente dos dados ou quando os dados de treinamento não contêm características que reflitam uma relação

importante entre entradas e saídas, resultando em baixa performance funcional do ML tanto no conjunto de dados de treinamento quanto no de validação.

Durante o teste, o underfitting pode ser detectado avaliando-se as métricas de performance funcional ML do modelo, como exatidão, precisão, recall ou pontuação F1. Se essas métricas forem consistentemente baixas tanto no conjunto de treinamento quanto no de validação, isso sugere que o modelo está underfitting.

A inspeção visual das curvas de aprendizagem do modelo também pode ajudar a detectar o underfitting. Se os erros de treinamento e validação permanecerem elevados e relativamente próximos entre si, sem melhora significativa à medida que o treinamento avança, isso indica underfitting.

Em resumo, detectar overfitting e underfitting durante os testes envolve avaliar a performance funcional do ML do modelo nos dados de validação, analisar métricas de performance funcional ML e examinar as curvas de aprendizagem.

6.1.9 Teste A/B

O teste A/B é uma abordagem em que a resposta de duas variantes do programa (A e B) às mesmas entradas é comparada com o objetivo de determinar qual das duas variantes é melhor. Trata-se de uma abordagem de teste estatístico que normalmente requer a comparação dos resultados de várias rodadas de teste para determinar as diferenças entre os programas.

Um exemplo simples dessa abordagem é quando duas ofertas promocionais são enviadas por e-mail para uma lista de marketing dividida em dois conjuntos. Metade da lista recebe a oferta A e a outra metade, a oferta B; o sucesso de cada oferta ajuda a decidir qual usar no futuro. Muitas empresas de comércio eletrônico e baseadas na web utilizam o teste A/B em produção, direcionando diferentes consumidores para diferentes funcionalidades, a fim de ajudar a identificar as preferências dos consumidores.

O teste A/B é uma abordagem para lidar com o problema do oráculo de teste, normalmente utilizando o sistema existente como um oráculo de teste parcial. O teste A/B não gera casos de teste e não fornece orientação sobre como os testes devem ser projetados, embora entradas operacionais sejam frequentemente incorporadas aos testes.

O teste A/B pode ser usado para testar atualizações em um sistema baseado em AI, desde que haja critérios de aceite acordados, como métricas de performance funcional ML, conforme descrito em 3.3. Sempre que o sistema é atualizado, o teste A/B é usado para determinar se a variante atualizada tem performance igual ou superior à da variante anterior.

Essa abordagem de teste pode ser usada para um classificador simples, mas também para testar sistemas muito mais complexos. Por exemplo, uma atualização para melhorar a efetividade de um sistema de roteamento de transporte de cidade inteligente pode ser testada usando o teste A/B. Por exemplo, comparando os tempos médios de deslocamento para duas variantes do sistema em semanas consecutivas.

O teste A/B também pode ser usado para testar sistemas de autoaprendizagem. Quando o sistema faz uma alteração, testes automatizados são executados, e as características resultantes do sistema são comparadas com as anteriores à alteração. Se o sistema for aprimorado, a alteração é aceita; caso contrário, o sistema reverte ao seu estado anterior.

As técnicas estatísticas mais populares para testes A/B são o teste t, o teste z, o teste qui-quadrado e o teste U de Mann-Whitney [STATS].

6.1.10 Teste back-to-back

O teste back-to-back oferece uma solução prática para o problema do oráculo de teste.

O teste back-to-back envolve o uso de uma versão alternativa do sistema como ponto de referência (um pseudo-oráculo) e a comparação de seus resultados com os do sistema em teste quando apresentados

com as mesmas entradas. Esse pseudo-oráculo pode ser um sistema existente ou um desenvolvido especificamente para testes, mas isso tem um custo.

Idealmente, o pseudo-oráculo e o sistema em teste não devem compartilhar componentes de software comuns. Caso contrário, ambos os sistemas podem conter o mesmo defeito, fazendo com que suas saídas coincidam mesmo quando ambas estiverem erradas. Isso pode ser particularmente problemático, dada a ampla utilização de componentes de AI reutilizáveis e de código aberto no desenvolvimento de MLS. Por esse motivo, o pseudo-oráculo é frequentemente desenvolvido por uma equipe diferente e, idealmente, independente, talvez utilizando diferentes estruturas de desenvolvimento de ML, algoritmos ou configurações de modelo. Às vezes, um software convencional pode servir como pseudo-oráculo se resolver o mesmo problema.

Ao realizar testes back-to-back, o pseudo-oráculo precisa apenas corresponder ao comportamento funcional. Ele não precisa atender aos mesmos requisitos não funcionais do sistema que está sendo testado, o que pode tornar sua construção mais econômica.

Essa abordagem de teste requer apenas a geração de entradas de teste, e não de resultados esperados, uma vez que o pseudo-oráculo fornece o ponto de comparação. Essas entradas podem vir de casos de teste existentes, como suítes de testes de regressão, ou podem ser geradas automaticamente a partir de dados de treinamento, permitindo que muitos testes sejam executados se a execução automatizada de testes for suportada.

O teste back-to-back oferece um valor significativo ao migrar um MLS para um novo ambiente, como a transição do desenvolvimento para a produção, e ao comparar resultados de teste entre ambientes. Além disso, essa abordagem de teste pode revelar defeitos sutis no comportamento do modelo que podem não ser aparentes por meio de outras abordagens de teste, especialmente ao comparar respostas em uma ampla gama de casos de borda ou entradas incomuns.

Uma diferença significativa entre o teste A/B (ver 6.1.9) e o teste back-to-back é o uso do teste A/B para comparar duas variantes do mesmo MLS utilizando métricas de performance funcional do ML e técnicas estatísticas, em contraposição ao uso do teste back-to-back para detectar defeitos.

7 Teste de Desenvolvimento de Machine Learning – 30 minutos

Palavras-chave

Testes de desenvolvimento de ML, performance funcional do ML, shadow testing

Palavras-chave específicas de AI

Nenhuma

Objetivos de aprendizagem do Capítulo 7:

7.1 Teste de Desenvolvimento de Machine Learning

AI-7.1.1 (K2) Dê exemplos de abordagens de teste utilizadas para a mitigação de risco no desenvolvimento de ML

AI-7.1.2 (K2) Explicar as várias formas de testes de implantação de sistemas de ML

7.1 Teste de Desenvolvimento de Machine Learning

Este capítulo enfoca especificamente os riscos introduzidos pelas ferramentas de desenvolvimento de ML, opções de configuração e mecanismos de implantação, em vez do próprio modelo ML. Ele abrange abordagens de teste e tipos de teste, como Teste de API, performance funcional do ML, Teste A/B, teste back-to-back e revisões, para verificar a robustez e a eficiência da AI.

7.1.1 Riscos e medidas de mitigação no desenvolvimento de machine learning

A tabela a seguir lista exemplos de riscos no desenvolvimento de ML e os testes correspondentes que poderiam ser usados para mitigação de risco:

Risco potencial	Possível Mitigação de Risco
Uso incorreto ou não intencional de APIs de bibliotecas ou frameworks (por exemplo, TensorFlow, PyTorch)	Teste de API – consulte 7.1.2
Seleção de framework abaixo do ideal	Revisão da adequação da estrutura
Problemas no algoritmo, modelo ou framework de desenvolvimento que introduzem injustiça sistêmica	Teste de viés – consulte 5.1.2
Instalação ou compilação com defeito da estrutura	Teste de fumaça
Implementação com defeito da avaliação pela estrutura	Revisões do código de avaliação da estrutura Verificação cruzada dos resultados da avaliação da estrutura (por exemplo, em comparação com benchmarks manuais)
Baixa eficiência de performance (por exemplo, o framework demora a responder)	Teste de performance
Baixa usabilidade da estrutura	Teste de usabilidade
Defeito em uma biblioteca usada pela estrutura (por exemplo, defeito no PyTorch) Implementação com defeito do algoritmo	Teste de performance funcional do ML – consulte 6.1.3 Teste back-to-back – consulte 6.1.10
Vulnerabilidades de segurança na estrutura	Teste de segurança
Documentação do usuário inadequada para a estrutura	Revisão da documentação da estrutura
Seleção de algoritmos abaixo do ideal	Revisão da adequação dos algoritmos Teste A/B – consulte 6.1.9
Seleção subótima de hiperparâmetros (por exemplo, estrutura da rede, taxa de aprendizagem)	Teste de performance funcional do ML – consulte 6.1.3 Teste A/B – ver 6.1.9

Risco potencial	Possível Mitigação de Risco
Alocação inadequada de dados para conjuntos de dados de treinamento, validação e teste	Revisão da alocação de dados
Seleção inadequada da abordagem de avaliação (por exemplo, validação cruzada k-fold)	Teste de performance funcional do ML – consulte 6.1.3
Interpretação incorreta dos resultados dos testes devido à natureza estocástica do processo de aprendizado	Teste de performance funcional do ML – consulte 6.1.3
Defeito de implantação (por exemplo, devido à geração de uma versão modificada para uma plataforma de destino)	Teste de fumaça Teste de performance funcional do ML – consulte 6.1.3 Teste A/B – consulte 6.1.9
O modelo implantado é incompatível com o ambiente operacional	Teste de fumaça Teste de implantação do MLS – consulte 7.1.2
O modelo implantado não representa nenhuma melhoria em relação ao modelo atual	Shadow testing – consulte 7.1.2

7.1.2 Teste de implantação de sistemas de machine learning

A implantação do MLS envolve várias atividades de teste essenciais, focadas em verificar se o sistema baseado em AI funciona de forma correta e com alta confiabilidade em seu ambiente de destino (por exemplo, nuvem, dispositivo de borda, dispositivos móveis). Cada um dos tipos de teste a seguir aborda riscos específicos durante a implantação:

- Teste de instalabilidade - verifica se o MLS pode ser instalado, configurado e, posteriormente, desinstalado com sucesso em todos os ambientes suportados. Isso inclui testar dependências do sistema (por exemplo, drivers de GPU), compatibilidade com frameworks e a execução bem-sucedida de scripts de instalação.
- Teste de reversão - verifica a capacidade do sistema de reverter com sucesso para um estado anteriormente estável e operacional após uma implantação degradada ou que falhou. Isso pode abranger apenas o modelo ou o sistema completo (por exemplo, incluindo o pipeline de dados). Observe que o teste de reversão deve ser realizado antes da implantação para confirmar a prontidão para a reversão.
- Teste Canary – valida novas implantações ao lançar um modelo atualizado para um pequeno subconjunto do tráfego de produção (por exemplo, 5% dos usuários). Métricas em tempo real, como latência, precisão e taxas de erro, são monitoradas para detectar regressões antes de uma implementação completa.
- Shadow testing – executa um novo modelo em paralelo com o modelo de produção atual em tempo real, encaminhando as mesmas solicitações para ambos os sistemas sem afetar as respostas em tempo real. Ele permite comparar modelos novos e antigos usando dados em tempo real em um ambiente controlado e de baixo risco e pode revelar defeitos, como regressões de desempenho e desvio de dados, antes da implantação completa.

- Testes de conversão de modelos - verifica se um modelo ML mantém precisão preditiva aceitável, comportamento consistente e eficiência operacional (por exemplo, velocidade de inferência, uso de memória) após ser convertido de seu formato de treinamento original para um formato de implantação adequado ao ambiente de produção de destino.
- Teste entre dispositivos - verifica se o MLS funciona corretamente em toda a gama de destinos de implantação previstos, desde dispositivos móveis e dispositivos de borda até servidores em nuvem.
- Teste de API - verifica se o MLS expõe interfaces bem definidas e em conformidade com os padrões. Ele valida o tratamento correto de entradas e saídas, mensagens de erro e fluxos de trabalho de integração com sistemas externos, como feeds de dados, clientes e o pipeline.

8 Lista de Abreviações

Abreviação	Descrição
AI	inteligência artificial
AIaaS	AI como serviço
API	Application Programming Interface
CL	nível de confiança
CNN	rede neural convolucional
CPU	unidade central de processamento
DL	deep learning
DNN	rede neural profunda
EDA	análise exploratória de dados
FN	falso negativo
FP	falso positivo
GAN	rede adversária generativa
GenAI	AI generativa
GPU	unidade de processamento gráfico
HO	objetivo prático
IAA	concordância entre anotadores
kMNC	cobertura de neurônios com k seções
LIME	Explicações locais interpretáveis e independentes do modelo
LLM	Large Language Model
LO	objetivo de aprendizagem
ML	machine learning
MLS	sistema(s) de machine learning
MoE	margem de erro
NBC	cobertura dos limites dos neurônios
NLP	processamento de linguagem natural
RAG	geração aumentada por recuperação
RNN	rede neural recorrente
RT	red teaming
SVM	máquina de vetores de suporte
TN	negativo verdadeiro
TP	verdadeiro positivo

9 Termos Específicos de AI

Nome do termo	Definição
precisão	Métricas de performance funcional do ML usadas para avaliar um classificador, que mede a proporção de previsões que estavam corretas. (De acordo com a ISO/IEC TR 29119-11)
função de ativação	A fórmula associada a um neurônio em uma rede neural que determina a saída do neurônio a partir das entradas recebidas por ele.
valor de ativação	A saída de uma função de ativação de um neurônio em uma rede neural.
sistema adaptativo baseado em AI	Um sistema baseado em AI que ajusta seu comportamento em resposta a mudanças em seu ambiente operacional.
ataque adversário	O uso deliberado de exemplos contraditórios para fazer com que um modelo ML falhe.
AI como Serviço	Um modelo de licenciamento e entrega de software no qual a AI e os serviços de desenvolvimento de AI são hospedados centralmente.
componente de AI	Um componente que fornece funcionalidade de AI.
Modelo de AI	Um programa de computador que implementa AI
Sistema baseado em AI	Um sistema que incorpora um ou mais componentes de AI.
viés algorítmico	Um tipo de viés causado pelo algoritmo de ML.
anotação	A atividade de identificar objetos em imagens com caixas delimitadoras para fornecer dados rotulados para classificação.
inteligência artificial	A capacidade de um sistema projetado de adquirir, processar, criar e aplicar conhecimentos e habilidades. (ISO/IEC TR 29119-11)
associação	Uma técnica de machine learning não supervisionada que identifica relações e dependências entre amostras.
aumentação	A atividade de criar pontos de dados com base em um conjunto de dados existente.
Modelo bayesiano	Um modelo estatístico que utiliza a probabilidade para representar a incerteza tanto das entradas quanto das saídas do modelo.
viés	A diferença sistemática no tratamento de certos objetos, pessoas ou grupos em comparação com outros. (Conforme ISO/IEC TR 24027)
big data	Conjuntos de dados extensos cujas características em termos de volume, variedade, velocidade e/ou variabilidade exigem tecnologias e técnicas especializadas para serem processados.
técnica de bootstrap	Técnica de reamostragem que extrai repetidamente amostras com reposição de um conjunto de dados de treinamento para estimar os critérios de desempenho funcional ML de um modelo ML.
chatbot	Um aplicativo usado para conduzir uma conversa por meio de texto ou conversão de texto em fala.
classificação	Uma função de ML que prevê a classe de saída para uma determinada entrada. (De acordo com a ISO/IEC TR 29119-11)

Nome do termo	Definição
classificador	Um modelo ML usado para classificação. Sinônimo: modelo de classificação
agrupamento	Uma função de ML que agrupa pontos de dados semelhantes.
algoritmo de agrupamento	Um tipo de algoritmo de ML usado para agrupar objetos semelhantes em clusters.
desvio de conceito	Uma mudança na performance funcional do modelo ML ao longo do tempo, causada por mudanças nas expectativas dos usuários, no comportamento e no ambiente operacional.
matriz de confusão	Uma técnica para resumir a performance funcional do ML de um algoritmo de classificação.
rede neural convolucional	Um tipo de modelo de deep learning projetado para processar dados em forma de grade, como imagens, permitindo-lhe reconhecer padrões e características espaciais por meio de operações em camadas.
ataque de injeção entre prompts	Um ataque no qual instruções maliciosas em um prompt ou segmento de contexto perturbam o comportamento de um modelo de AI em um prompt, turno ou segmento de contexto diferente.
aquisição de dados	A atividade de adquirir dados relevantes para o problema de negócios a ser resolvido por um modelo ML.
viés de dados	Um erro sistemático causado por dados de treinamento imprecisos, incompletos ou não representativos que leva a resultados injustos em modelos ML.
operação de dados	Uma mudança na distribuição dos dados de entrada ao longo do tempo, que pode impactar negativamente a performance funcional do modelo ML operacional.
pipeline de dados	A implementação de atividades de preparação de dados para fornecer dados de entrada que apoiem o treinamento por um algoritmo de ML ou a previsão por um modelo ML.
ponto de dados	Conjunto de uma ou mais medições que compõem uma única observação usada como parte de um conjunto de dados.
preparação de dados	As atividades de aquisição de dados, pré-processamento de dados e engenharia de características no fluxo de trabalho de machine learning.
pré-processamento de dados	As atividades de limpeza de dados, transformação de dados, aumento de dados e amostragem de dados no fluxo de trabalho de ML.
conjunto de dados	Uma coleção de dados usada para treinamento, avaliação, teste e previsão em ML.
árvore de decisão	Um modelo ML em forma de árvore cujos nós representam decisões e cujos ramos representam resultados possíveis.
deep learning	ML que utiliza redes neurais profundas para aprender automaticamente características e representações complexas a partir de grandes conjuntos de dados.
rede neural profunda	Uma rede neural composta por várias camadas de neurônios. Sinônimo: perceptron multicamadas
deepfake	Mídia sintética, como vídeo, áudio ou imagens, criada ou editada usando AI para imitar ou personificar de forma convincente pessoas ou eventos reais.

Nome do termo	Definição
previsão de defeitos	Técnica para prever as áreas dentro do objeto de teste nas quais ocorrerão defeitos ou a quantidade de defeitos presentes
determinístico	Produz o mesmo conjunto de resultados e estado final a partir de um determinado conjunto de entradas e estado inicial.
análise de impacto desigual	Técnica para detectar viés comparando decisões em cenários originais com suas versões contrafactuais, nas quais atributos sensíveis são trocados.
AI de ponta	A implantação de modelos de AI em dispositivos locais de borda, permitindo o processamento em tempo real próximo à fonte dos dados, sem depender de sistemas baseados em nuvem.
computação de ponta	A parte de uma arquitetura distribuída na qual o processamento de informações é realizado próximo ao local onde essas informações são utilizadas.
época	Uma iteração do treinamento de ML em todo o conjunto de dados de treinamento.
sistema especialista	Um sistema baseado em AI para resolver problemas em um domínio ou área de aplicação específica, tirando conclusões a partir de uma base de conhecimento desenvolvida a partir da expertise humana.
AI explicável	O campo de estudo relacionado à compreensão dos fatores que influenciam os resultados dos sistemas de AI.
análise exploratória de dados	Um processo interativo, visual e orientado por hipóteses para resumir, explorar e compreender as principais características e padrões dos dados.
F1-Score	Uma métrica de performance funcional do ML usada para avaliar um classificador, que proporciona um equilíbrio entre recall e precisão.
falso negativo	Uma previsão de um modelo ML em que o modelo prevê erroneamente a classe negativa.
falso positivo	Uma previsão de um modelo ML em que o modelo prevê erroneamente a classe positiva.
característica	Um atributo mensurável individual dos dados de entrada usado para treinamento por um algoritmo de ML e para previsão por um modelo ML.
engenharia de características	A atividade na qual os atributos dos dados brutos que melhor representam as relações subjacentes que devem aparecer no modelo ML são identificados para uso nos dados de treinamento. (ISO/IEC TR 29119-11)
teste de características	Um tipo de teste para determinar se um conjunto de dados de treinamento de um modelo de AI contém um conjunto adequado de características.
modelo de base	Um modelo ML em grande escala treinado em grandes conjuntos de dados usando aprendizado auto-supervisionado, projetado como uma base versátil que pode ser ajustada ou adaptada a uma ampla gama de tarefas em diferentes domínios.
AI de ponta	Um sistema de AI de uso geral que supera as capacidades dos sistemas de AI mais avançados da atualidade.
lógica difusa	Um tipo de lógica baseada no conceito de verdade parcial, representada por fatores de certeza entre 0 e 1.
AI geral	Um tipo de AI capaz de igualar as habilidades cognitivas humanas na maioria das tarefas intelectuais.

Nome do termo	Definição
AI generativa	Um tipo de AI que cria conteúdos ao aprender padrões a partir de dados existentes.
unidade de processamento gráfico	Um circuito integrado específico para aplicações, projetado para manipular e alterar a memória a fim de acelerar a criação de imagens em um buffer de quadros destinado à saída para um dispositivo de exibição.
verdade fundamental	A informação fornecida por observação e medição diretas, que se sabe ser real ou verdadeira.
hiperparâmetro	Parâmetros usados para controlar o treinamento de um modelo ML ou para definir a configuração de um modelo ML.
ajuste de hiperparâmetros	A atividade de determinar os hiperparâmetros ideais com base em objetivos específicos.
agente inteligente	Um programa autônomo que direciona sua atividade para alcançar objetivos por meio de observações e ações.
concordância entre anotadores	O grau de consenso ou similaridade entre as anotações feitas por diferentes anotadores sobre os mesmos dados. (ISO/IEC TS 12791)
Large Language Model	Um sistema de AI generativa texto-para-texto treinado em coleções muito grandes de dados de linguagem.
regressão linear	Técnica estatística que modela a relação entre variáveis ajustando uma equação linear aos dados observados quando a variável-alvo é numérica.
sistema baseado em AI bloqueado	Um sistema determinístico baseado em AI com um modelo fixo e imutável, que não altera seu comportamento após ser implantado.
Algoritmo de ML	Um algoritmo usado para criar um modelo ML a partir de um conjunto de dados de treinamento.
Estrutura de desenvolvimento de ML	Uma plataforma de software que fornece ferramentas e bibliotecas para construir, treinar e implantar modelos ML.
Função de ML	Funcionalidade implementada por um modelo ML, como classificação, regressão de ML ou agrupamento.
Regressão de ML	Um tipo de função de ML que resulta em um valor de saída numérico ou contínuo para uma determinada entrada. (De acordo com a ISO/IEC TR 29119-11)
MLS	Um sistema que integra um ou mais modelos ML.
Fluxo de trabalho de ML	Conjunto de atividades utilizadas para desenvolver, implantar e operar um modelo ML.
Análise da pontuação de confiança do modelo	Técnica que identifica pontos de dados com baixas pontuações de confiança no treinamento, que são indicadores de pontos de dados rotulados incorretamente.
Análise de perda do modelo	Técnica que identifica pontos de dados com altos valores de perda durante o treinamento, que são indicadores de pontos de dados rotulados incorretamente.
modelo multimodal	Um modelo ML projetado para lidar com vários tipos de modalidades de dados, como texto, imagens, áudio e vídeo.
anotação múltipla	Uma abordagem na qual pontos de dados rotulados por vários anotadores são comparados.

Nome do termo	Definição
AI estreita	AI focada em uma única tarefa bem definida para resolver um problema específico. Sinônimo: AI fraca (ISO/IEC TR 29119-11)
processamento de linguagem natural	Um campo da computação que oferece a capacidade de ler, compreender e extrair significado de linguagens naturais.
rede neural	Uma rede de elementos de processamento primitivos conectados por ligações ponderadas com pesos ajustáveis, na qual cada elemento produz um valor aplicando uma função não linear aos seus valores de entrada e o transmite a outros elementos ou o apresenta como um valor de saída. Sinônimo: rede neural artificial (ISO/IEC 2382)
processador neuromórfico	Circuito integrado projetado para imitar os neurônios biológicos do cérebro humano.
neurônio	Um nó em uma rede neural, que geralmente recebe múltiplos valores de entrada e gera um valor de ativação.
ruído	Uma distorção ou corrupção nos dados.
não determinismo	Propriedade de um sistema ou processo em que um resultado não é determinado exclusivamente por suas condições iniciais.
outlier	Uma observação que se encontra fora do padrão geral da distribuição dos dados.
overfitting	A geração de um modelo ML que se ajusta excessivamente ao conjunto de dados de treinamento, resultando em um modelo que tem dificuldade em generalizar para novos dados. (Conforme ISO/IEC TR 29119-11)
perceptron	Uma rede neural com apenas uma camada e um neurônio
precisão	Uma métrica de performance funcional ML usada para avaliar um classificador, que mede a proporção de resultados positivos previstos que estavam corretos. (De acordo com a ISO/IEC TR 29119-11)
modelo pré-treinado	Um modelo ML que já foi treinado em um grande conjunto de dados de uso geral e pode ser reutilizado ou ajustado para tarefas específicas.
probabilístico	Comportamento descrito em termos de probabilidades, em que os resultados são incertos e descritos por probabilidades, em vez de certezas.
floresta aleatória	Tecnologia de machine learning de conjunto para classificação, regressão de machine learning e outras tarefas que operam construindo e executando muitas árvores de decisão e, em seguida, gerando o modo da classe ou a previsão média das árvores individuais.
recall	Métricas de performance funcional ML usadas para avaliar um classificador, que mede a proporção de casos positivos reais que foram previstos corretamente. Sinônimo: sensibilidade (De acordo com a norma ISO/IEC TR 29119-11)

Nome do termo	Definição
rede neural recorrente	Um tipo de modelo de deep learning projetado para processar dados sequenciais, permitindo-lhe reconhecer padrões e dependências ao longo do tempo.
aprendizado por reforço	Uma abordagem na qual um modelo ML, conhecido como agente, aprende por meio de um ciclo de feedback de tentativa e recompensa com seu ambiente para atingir objetivos específicos.
geração aumentada por recuperação	Uma técnica de ML em que um sistema GenAI aprimora dinamicamente sua saída ao recuperar informações externas relevantes para complementar seu conhecimento interno.
função de recompensa	Uma função que define o sucesso da aprendizagem por reforço.
hackeamento de recompensa	A atividade realizada por um agente inteligente para maximizar sua função de recompensa em detrimento do cumprimento do objetivo original. (Conforme ISO/IEC TR 29119-11)
algoritmo de busca	Um algoritmo que percorre sistematicamente um subconjunto de todos os estados ou estruturas possíveis até que o estado ou estrutura alvo seja alcançado. (Segundo a ISO/IEC TR 29119-11)
sistema de autoaprendizagem	Um sistema adaptativo que altera seu comportamento com base na aprendizagem por tentativa e erro. (Baseado na ISO/IEC TR 29119-11)
atributos sensíveis	Características que definem grupos e indivíduos protegidos legal ou eticamente e que devem ser controladas para evitar a discriminação.
análise de sentimentos	O uso de processamento de linguagem natural e machine learning para identificar e classificar o tom emocional expresso em um texto.
amostragem estratificada	Técnica para confirmar se uma amostra representa proporcionalmente diferentes subpopulações dentro da população geral.
AI super	Um tipo de AI que excede em muito as capacidades humanas.
aprendizado supervisionado	Uma abordagem para treinar um modelo ML usando um conjunto de dados rotulados.
máquina de vetores de suporte	Um algoritmo de ML supervisionado que descobre o hiperplano ideal entre pontos de dados para tarefas de classificação ou regressão de ML.
singularidade tecnológica	Um momento no futuro em que os avanços tecnológicos não serão mais controláveis pelas pessoas. (De acordo com a ISO/IEC TR 29119-11)
temperatura	Uma configuração que controla a aleatoriedade das saídas da GenAI, com valores mais baixos gerando resultados mais previsíveis e valores mais altos gerando resultados mais criativos.
problema do oráculo de teste	O desafio de determinar se um teste passou ou falhou para um determinado conjunto de entradas de teste e estado.
conjunto de dados de treinamento	Um conjunto de dados usado para treinar um modelo ML.
transformador	Uma arquitetura de rede neural que processa dados sequenciais para capturar dependências de longo alcance, impulsionando tarefas de PLN, visão computacional e aplicações multimodais.
negativo verdadeiro	Uma previsão em que o modelo prevê corretamente a classe negativa.
verdadeiro positivo	Uma previsão em que o modelo prevê corretamente a classe positiva.

Nome do termo	Definição
underfitting	A geração de um modelo ML que não reflete a tendência subjacente do conjunto de dados de treinamento, resultando em um modelo que faz previsões imprecisas. (segundo a ISO/IEC TR 29119-11)
aprendizado não supervisionado	Uma abordagem para treinar um modelo ML usando um conjunto de dados não rotulado.
conjunto de dados de validação	Um conjunto de dados utilizado para avaliar um modelo ML treinado com o objetivo de ajustar o modelo.
arquitetura von Neumann	Uma arquitetura de computador que consiste em cinco componentes principais: memória, unidade central de processamento, unidade de controle, entrada e saída.
peso	Uma variável interna de uma conexão entre neurônios em uma rede neural que afeta a forma como ela calcula suas saídas e que muda à medida que a rede neural é treinada.

10 Referências

10.1 Normas

- **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 Documentos 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 (accessed 29.07.2025)

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

10.3 Glossário Referências

Referência para a terminologia utilizada neste syllabus:

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

10.4 Livros, artigos e páginas da Web

[COV_REF] An Overview of Structural Coverage Metrics for Testing Neural Networks, Usman et al, Aug 2022, arXiv:2208.03407 (accessed 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 (accessed 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 (accessed 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 (accessed 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 (accessed 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 (accessed 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 Marcas registradas

ISTQB® é uma marca registrada do International Software Testing Qualifications Board

BSTQB® é uma marca registrada do Brazilian Software Testing Qualifications Board

12 Apêndice A – Objetivos de Aprendizagem/Nível Cognitivo de Conhecimento

Os objetivos de aprendizagem específicos aplicáveis a este syllabus são apresentados no início de cada capítulo. Cada tópico do syllabus será examinado de acordo com o respectivo objetivo de aprendizagem.

Os objetivos de aprendizagem começam com um verbo de ação correspondente ao seu nível cognitivo de conhecimento, conforme listado abaixo.

Nível 1: Lembrar (K1)

O candidato irá lembrar, reconhecer e recordar um termo ou conceito.

Verbos de ação: Recordar, reconhecer.

Exemplos
Recordar os conceitos da pirâmide de teste.
Reconhecer os objetivos típicos dos testes.

Nível 2: Compreender (K2)

O candidato é capaz de selecionar as razões ou explicações para afirmações relacionadas ao tema e pode resumir, comparar, classificar e dar exemplos para o conceito de teste.

Verbos de ação: Classificar, comparar, diferenciar, distinguir, explicar, dar exemplos, interpretar, resumir

Exemplos	Notas
Classifique as ferramentas de teste de acordo com sua finalidade e as atividades de teste que elas suportam.	
Comparar os diferentes níveis de teste.	Pode ser usado para procurar semelhanças, diferenças ou ambos.
Diferenciar testes de depuração.	Procure diferenças entre conceitos.
Distinguir entre riscos de projeto e riscos de produto.	Permite que dois (ou mais) conceitos sejam classificados separadamente.
Explique o impacto do contexto no processo de teste.	
Dê exemplos de por que o teste é necessário.	
Inferir a causa-raiz dos defeitos a partir de um determinado perfil de falhas.	
Resumir as atividades do processo de revisão do produto de trabalho.	

Nível 3: Aplicar (K3)

O candidato é capaz de executar um procedimento quando confrontado com uma tarefa familiar, ou selecionar o procedimento correto e aplicá-lo a um determinado contexto.

Verbos de ação: Aplicar, implementar, preparar, usar

Exemplos	Notas
Aplicar a análise de valor limite para derivar casos de teste a partir de requisitos dados.	Deve consultar um procedimento/técnica/processo etc.
Implementar métodos de coleta de métricas para atender aos requisitos técnicos e de gestão.	
Prepare testes de instalabilidade para aplicativos móveis.	

Utilizar a rastreabilidade para monitorar o progresso dos testes quanto à integridade e consistência com os objetivos do teste, a estratégia de teste e o plano de teste.	Pode ser usado em um LO que deseja que o candidato seja capaz de utilizar uma técnica ou procedimento. Semelhante a “aplicar”.

Nível 4: Analisar (K4)

O candidato é capaz de separar as informações relacionadas a um procedimento ou técnica em suas partes constituintes para melhor compreensão e distinguir entre fatos e inferências. A aplicação típica consiste em analisar um documento, software ou situação de projeto e propor ações adequadas para resolver um problema ou tarefa.

Verbos de ação: analisar, desconstruir, esboçar, priorizar, selecionar.

Exemplos	Notas
Analisar uma determinada situação de projeto para determinar quais técnicas de teste caixa-preta ou baseadas na experiência devem ser aplicadas para atingir objetivos específicos.	Avaliável apenas em combinação com um objetivo de medição da análise. Deve ter a forma “Analisar xxxx para xxxx” (ou similar).
Priorizar casos de teste em uma determinada suíte de teste para execução com base nos riscos de produto.	
Selecionar os níveis de teste e tipos de teste apropriados para verificar um determinado conjunto de requisitos.	Necessário quando a seleção requer análise como requisito.

Referência

(Para os níveis cognitivos dos objetivos de aprendizagem)

- Anderson, L. W. e Krathwohl, D. R. (eds) (2001) Uma taxonomia para aprendizagem, ensino e
- Avaliação: Uma Revisão da Taxonomia de Objetivos Educacionais de Bloom, Allyn & Bacon

13 Apêndice B – Matriz de Rastreabilidade dos Resultados de Negócios com Objetivos de Aprendizagem

Esta seção lista a rastreabilidade entre os Resultados de Negócios e os Objetivos de Aprendizagem do Teste de AI para Testador Certificado.

A primeira parte da tabela mostra o número de Objetivos de Aprendizagem por Resultado de Negócios, enquanto a segunda parte mostra quais Objetivos de Aprendizagem estão associados a cada Resultado de Negócios.

Resultados de Negócios: Teste de AI			**BO1**	**BO2**	**RO3**	**RO4**	**BO5**	**BO6**	**BO7**	**BO8**
BO1	Compreender o estado atual da AI incluindo a AI generativa.		8							
BO2	Experimentar a implementação e o teste de modelos de machine learning.			10						
BO3	Compreenda o funcionamento e o teste de redes neurais simples.				2					
BO4	Compreender as características específicas de qualidade da AI definidas pela norma ISO/IEC 25059.					3				
BO5	Calcular e interpretar métricas de performance funcional ML para modelos de machine learning.						1			
BO6	Reconhecer o escopo e a importância dos dois níveis de teste específicos para a avaliação de sistemas de machine learning.							3		
BO7	Contribuir para o desenvolvimento de uma estratégia de teste eficaz para um sistema de machine learning.								5	
BO8	Projetar e executar casos de teste para sistemas de machine learning.									14

LO exclusivo	**Objetivo de Aprendizagem**	**Nível K**	**BO1**	**BO2**	**RO3**	**RO4**	**BO5**	**BO6**	**BO7**	**BO8**
1	Introdução à Inteligência Artificial									
1.1	Introdução à AI									
AI-1.1.1	Diferenciar entre sistemas baseados em AI e sistemas convencionais	K2	X							

LO exclusivo	Objetivo de Aprendizagem	Nível K	BO1	BO2	RO3	RO4	BO5	BO6	BO7	BO8
AI-1.1.2	Distinguir entre AI restrita, AI geral e super AI	K2	X							
AI-1.1.3	Explicar os diferentes tipos de tecnologias de AI	K2	X							
AI-1.1.4	Explicar a AI generativa	K2	X							
AI-1.1.5	Comparar as opções de hardware disponíveis para implementar sistemas de machine learning	K2	X							
AI-1.1.6	Comparar as opções para o desenvolvimento e hospedagem de modelos de AI	K2	X							
AI-1.1.7	Resumir as funcionalidades oferecidas pelas estruturas de desenvolvimento de ML	K2	X							
AI-1.1.8	Explicar como as regulamentações e normas afetam o desenvolvimento e o teste de sistemas baseados em AI	K2	X							
2	Características de qualidade para sistemas baseados em AI									
2.1	Características de qualidade para sistemas baseados em AI									
AI-2.1.1	Classificar comportamentos de sistemas baseados em AI de acordo com as características de qualidade definidas na ISO/IEC 25059	K2				X				
AI-2.1.2	Explicar as considerações especiais que surgem quando a AI é utilizada em sistemas relacionados à segurança	K2				X				
2.2	Critérios de aceite para sistemas baseados em AI									
AI-2.2.1	Dê exemplos de critérios de aceite para sistemas baseados em AI	K2				X				
3	Machine Learning									
3.1	Introdução ao Machine Learning									
AI-3.1.1	Distinguir entre as diferentes formas de ML	K2		X						
AI-3.1.2	Resumir o fluxo de trabalho utilizado para criar um sistema de ML	K2		X						
AI-3.1.4	Resumir o uso de modelos pré-treinados, ajuste fino e geração aumentada por recuperação	K2		X						
3.2	Dados para Machine Learning									

LO exclusivo	Objetivo de Aprendizagem	Nível K	BO1	BO2	RO3	RO4	BO5	BO6	BO7	BO8
AI-3.2.1	Explique as atividades relacionadas à preparação de dados	K2		X						
AI-3.2.3	Compare o uso de conjuntos de dados de treinamento, validação e teste no desenvolvimento de um modelo ML	K2		X						
3.3	Métricas de performance funcional ML para classificação									
AI-3.3.1	Calcular métricas de performance funcional do ML comuns a partir de um determinado conjunto de dados da matriz de confusão	K3					X			
3.4	Redes Neurais									
AI-3.4.1	Explique a estrutura e o funcionamento de uma rede neural profunda	K2			X					
AI-3.4.3	Descrever as diferentes medições de cobertura para redes neurais	K2			X					
4	Teste de sistemas baseados em AI									
4.1	Introdução ao teste de sistemas baseados em AI									
AI-4.1.1	Comparar a testabilidade de sistemas baseados em AI fixos e adaptativos	K2		X						
AI-4.1.2	Explicar por que uma abordagem estatística é frequentemente necessária ao testar sistemas baseados em AI	K2		X						
AI-4.1.3	Explicar os desafios e as soluções relacionados a oráculos de teste para sistemas baseados em AI	K2		X						
4.2	Teste de AI generativa e LLM									
AI-4.2.1	Explique como a AI generativa pode ser testada	K2		X						
AI-4.2.2	Implementar red teaming para sistemas de AI generativa	K3		X						
4.3	Níveis de teste e sistemas de machine learning									
AI-4.3.1	Resumir os níveis de teste utilizados para desenvolver sistemas de machine learning	K2							X	
AI-4.3.2	Explicar como os testes baseados em riscos são aplicados aos sistemas de machine learning	K2							X	

LO exclusivo	Objetivo de Aprendizagem	Nível K	BO1	BO2	RO3	RO4	BO5	BO6	BO7	BO8
5	Teste de dados de entrada para sistemas de machine learning									
5.1	Teste de dados de entrada para sistemas de machine learning									
AI-5.1.1	Dê exemplos de abordagens de teste utilizadas para a mitigação de risco dos dados de entrada de um sistema de machine learning	K2						X	X	
AI-5.1.2	Explique como testar a presença de viés	K2								X
AI-5.1.3	Resumir as várias formas de teste de pipeline de dados	K2								X
AI-5.1.4	Explicar como testar a representatividade dos dados	K2								X
AI-5.1.5	Aplicar testes de restrições de conjuntos de dados	K3								X
AI-5.1.6	Explicar o label correctness testing	K2								X
6	Teste de modelos para sistemas de machine learning									
6.1	Testes de modelos para sistemas de machine learning									
AI-6.1.1	Dê exemplos de abordagens de teste utilizadas para a mitigação de riscos de modelos ML	K2						X	X	
AI-6.1.2	Explique o objetivo e o foco da revisão da documentação do modelo ML	K2								X
AI-6.1.3	Explicar como são realizados os testes de performance funcional do ML para sistemas de machine learning probabilístico	K2								X
AI-6.1.4	Resumir os testes contraditórios de sistemas de machine learning	K2								X
AI-6.1.5	Utilizar testes metamórficos para derivar casos de teste para um determinado cenário	K3								X
AI-6.1.7	Explicar como o teste de desvio é utilizado em sistemas operacionais de machine learning	K2								X
AI-6.1.8	Explique como o overfitting e o underfitting são detectados por meio de testes	K2								X

LO exclusivo	Objetivo de Aprendizagem	Nível K	BO1	BO2	RO3	RO4	BO5	BO6	BO7	BO8
AI-6.1.9	Explique como o teste A/B é utilizado no contexto de sistemas de machine learning	K2								X
AI-6.1.10	Explique como o teste back-to-back é utilizado no contexto dos sistemas de machine learning	K2								X
7	Teste de desenvolvimento de machine learning									
7.1	Teste de desenvolvimento de machine learning									
AI-7.1.1	Dê exemplos de abordagens de teste utilizadas para a mitigação de risco no desenvolvimento de ML	K2						X	X	
AI-7.1.2	Explique as várias formas de testes de implantação de sistemas de ML	K2								X

14 Apêndice C – Notas de Lançamento

O ISTQB CT-AI v2.0 é uma grande atualização e reescrita da v1.0. Devido à rápida evolução da tecnologia de AI, uma grande atualização era necessária. O foco da v2.0 está claramente nos testes de sistemas baseados em AI. Devido ao lançamento do ISTQB CT Testing with Generative AI, o capítulo sobre testes com AI foi completamente removido.

Esta versão importante trouxe as seguintes alterações:

- Introdução resumida à AI em geral
- Inclusão da AI generativa e dos testes de AI generativa
- Consolidação das características de qualidade da AI e seus desafios
- Menor ênfase nas métricas de performance de ML
- Níveis de teste refinados: teste de dados de entrada e teste do modelo ML
- Tipos de teste refinados
- Remoção de testes com AI
- Exclusão de ambientes de teste para sistemas baseados em AI
- Tempo mínimo de treinamento exigido reduzido de 4 para 3 dias

15 Índice

Todos os termos relacionados a testes estão definidos no Glossário do ISTQB® (https://glossary.istqb.org/).