Certified Tester

AI Testing Syllabus

v1.0

International Software Testing Qualifications Board

Fornecido por:

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

Direitos autorais

Histórico da Revisão

Versão	Data	Observações
1.0	01/10/2021	Lançamento no GA

Histórico da versão BSTQB

Versão	Data	Observações
1	14/06/2023	Padronização do layout com o ISTQB

Índice

Agradecimentos

Este documento foi formalmente divulgado pela Assembleia Geral do ISTQB® em 1 de outubro de 2021.

Foi produzido pela equipe do International Software Testing Qualifications Board: Klaudia Dussa-Zieger (presidente), Werner Henschelchen, Vipul Kocher, Qin Liu, Stuart Reid, Kyle Siemens, and Adam Leon Smith.

A equipe agradece aos três autores a contribuição sobre este syllabus:

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

A equipe agradece aos Grupos de Trabalho de Exame, Glossário e Marketing por seu apoio durante todo o desenvolvimento do syllabus, Graham Bath por sua edição técnica e os Conselhos de Membros por suas sugestões e contribuições.

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

O BSTQB agradece aos voluntários de seu grupo de trabalho de traduções (WG Traduções) pelo empenho e esforço na tradução deste material: O BSTQB agradece aos voluntário do GT Traduções pelo empenho e esforço na tradução e revisão deste material: Eduardo Medeiros, George Fialkovitz, Irene Nagase, Osmar Higashi, Paula de Oliveira, Rogério Athaide, Stênio Viveiros, Yuri Fialkovitz.

0 Introdução

0.1 Objetivo deste syllabus

Este syllabus forma a base para o ISTQB® Certified Tester AI Testing. O ISTQB® fornece este syllabus da seguinte forma:

1. Aos Conselhos Membros, para traduzir para seu idioma local e para credenciar os provedores de treinamento. Os Conselhos Membros podem adaptar o syllabus às suas necessidades linguísticas particulares e modificar as referências para se adaptarem às suas publicações locais.
2. Aos Provedores de Exame, para derivar perguntas de exame em sua língua local adaptadas aos objetivos de aprendizagem deste syllabus.
3. Aos Provedores de Treinamento, para produzir material didático e determinar métodos de ensino apropriados.
4. Aos Candidatos à certificação, para preparar-se para o exame de certificação (seja como parte de um curso de treinamento ou independentemente).
5. À Comunidade Internacional de engenharia de software e sistemas, para promover a profissão de teste de software e sistemas, e como base para livros e artigos.

0.2 O Certified Tester AI Testing

O *Certified Tester AI Testing* é destinado a qualquer pessoa envolvida no teste de sistemas baseados em AI ou testes de AI. Isto inclui pessoas em funções como testadores, analistas de teste, analistas de dados, engenheiros de teste, consultores de teste, gerentes de teste, testadores de aceite de usuários e desenvolvedores de software. Esta certificação também é apropriada para qualquer pessoa que queira um entendimento básico dos testes de sistemas baseados em AI ou testes de AI, tais como gerentes de projeto, gerentes de qualidade, gerentes de desenvolvimento de software, analistas de negócios, membros da equipe de operações, diretores de TI e consultores de gerenciamento.

A Visão Geral do *Certified Tester AI Testing* [I03] é um documento separado que inclui as seguintes informações:

- Resultados comerciais para o syllabus
- Matriz de resultados de negócios e conexão com objetivos de aprendizado
- Resumo do syllabus
- Relacionamentos entre os syllabi

0.3 Objetivos de aprendizagem e nível cognitivo de conhecimento

Os objetivos de aprendizagem apoiam os resultados comerciais e são usados para criar os exames do *Certified Tester AI Testing*.

Os candidatos podem ser solicitados a reconhecer, lembrar ou relembrar uma palavra-chave ou conceito mencionado em qualquer um dos onze capítulos. Os níveis específicos dos objetivos de aprendizagem são mostrados no início de cada capítulo, e classificados da seguinte forma:

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

Todos os termos listados como palavras-chave logo abaixo dos títulos dos capítulos devem ser lembrados (K1), mesmo que não explicitamente mencionados nos objetivos de aprendizagem.

0.4 Níveis práticos de competência

O *Certified Tester AI Testing* inclui objetivos práticos que se concentram em habilidades e competências.

Os níveis a seguir se aplicam aos objetivos práticos (como mostrado):

- **H0**: Demonstração ao vivo ou em vídeo gravado.
- **H1**: Exercício prático. O estudante segue uma sequência de passos executados pelo instrutor.
- **H2**: Exercício com dicas. O estudante recebe um exercício com dicas relevantes para que o exercício possa ser resolvido dentro do prazo determinado, ou os estudantes participam de uma discussão.

As competências são alcançadas através da realização de exercícios práticos, como mostrado na lista a seguir:

- (H0) Demonstrar *underfitting* e *overfitting*.
- (H2) Preparar os dados em apoio à criação de um modelo ML.
- (H2) Identificar conjuntos de dados de treinamento e testes e criar um modelo ML.
- (H2) Avaliar o modelo ML criado usando métricas de performance funcional ML selecionadas.
- (H1) Experiência da implementação de um *perceptron*.
- (H2) Usar uma ferramenta para mostrar como a explicabilidade pode ser usada pelos testadores.
- (H2) Aplicar teste em pares para derivar e executar casos de teste para um sistema baseado em AI.
- (H2) Aplicar testes metamórficos para derivar e executar casos de teste para um determinado cenário.
- (H2) Aplicar testes exploratórios a um sistema baseado em AI.
- (H2) Discutir, usando exemplos, aquelas atividades de testes onde é menos provável que a AI seja utilizada.
- (H2) Implementar um sistema simples de previsão de defeitos baseado em AI.

0.5 O exame de certificação

O exame do *Certified Tester AI Testing* será baseado neste syllabus. As respostas às perguntas do exame podem exigir o uso de material baseado em mais de um capítulo. Todos os capítulos são examináveis, exceto a Introdução e os Anexos. As normas e livros são incluídos como referências, mas seu conteúdo não é examinável, além do que está resumido neste documento a partir de tais normas e livros.

Consulte o documento "Overview" do *Certified Tester AI Testing* para obter mais detalhes no capítulo "*Exam structure*".

Nota sobre os requisitos de entrada: possuir o certificado ISTQB® Foundation Level (CTFL) é pré-requisito para o exame *Certified Tester AI Testing*.

0.6 Credenciamento

Um Conselho Membro do ISTQB® pode credenciar provedores de treinamento cujo material didático siga este syllabus. Os provedores de treinamento devem obter diretrizes de credenciamento do Conselho Membro ou do órgão que realiza o credenciamento. Um curso credenciado é reconhecido como estando em conformidade com este syllabus e é permitido ter um exame ISTQB® como parte do curso.

As diretrizes de credenciamento seguem as *General Accreditation Guidelines* publicadas pelo *Processes Management and Compliance Working Group*.

0.7 Nível de detalhe

O nível de detalhes deste syllabus permite cursos e exames internacionalmente consistentes. A fim de atingir este objetivo, o syllabus consiste em:

- Objetivos instrucionais gerais descrevendo a intenção do Testador Certificado em AI.
- Uma lista de termos que os aprendizes devem ser capazes de lembrar.
- Aprendizagem e objetivos práticos para cada área de conhecimento, descrevendo os objetivos de aprendizagem a serem alcançados.
- Uma descrição das palavras-chave, incluindo referências a fontes tais como literatura ou normas aceitas.

O conteúdo do syllabus não é uma descrição de toda a área de conhecimento para os testes de sistemas baseados em AI; ele reflete o nível de detalhe a ser coberto nos cursos de treinamento de especialista em *Certified Tester AI Testing*. Ele se concentra na introdução dos conceitos básicos de inteligência artificial (AI) e aprendizagem de máquinas (ML) em particular, e como os sistemas baseados nestas tecnologias podem ser testados.

0.8 Como este syllabus está organizado

Há onze capítulos com conteúdo examinável. O cabeçalho de nível superior para cada capítulo especifica o tempo para o capítulo; o tempo de cada nível não é fornecido. Para cursos de treinamento credenciados, o syllabus requer um mínimo de 25,1 horas de instrução, distribuídas pelos onze capítulos da seguinte forma:

Capítulo	Tempo (min)	Conteúdo
1	105	Introdução à AI
2	105	Características de qualidade para sistemas baseados em AI
3	145	Machine Learning (ML), Visão geral
4	230	Machine Learning (ML), Dados
5	120	Machine Learning (ML), Métricas de desempenho funcional
6	65	Redes neurais e testes
7	115	Visão geral dos sistemas baseados em AI
8	150	Teste das características de qualidade específicas da AI
9	245	Métodos e técnicas para os testes de sistemas baseados em AI
10	30	Ambientes de teste para sistemas baseados em AI
11	195	Usando AI para testes

1 Introdução a AI (105 min)

Palavras-chave

Nenhuma

Palavras-chave específicas de AI

AI como serviço (AIaaS), AI-estreita, AI-geral, AI-super, deep learning (DL), efeito AI, estrutura de desenvolvimento de AI, General Data Protection Regulation (GDPR), inteligência artificial (AI), machine learning (ML), modelo pré-treinado, rede neural profunda, rede neural, singularidade tecnológica, sistema baseado em AI, transferência de aprendizagem.

Objetivos de aprendizagem

1.1 Definição de AI e Efeito AI

AI-1.1.1 K2: Descrever o efeito da AI e como ela influencia a definição de AI.

1.2 AI-estreita, geral e AI-super

AI-1.2.1 K2: Distinguir entre AI-estreita, AI-geral, e AI-super.

1.3 Sistemas convencionais e sistemas baseados em AI

AI-1.3.1 K2: Distinguir entre sistemas baseados em AI e sistemas convencionais.

1.4 Tecnologias de AI

AI-1.4.1 K1: Reconhecer as diferentes tecnologias usadas para implementar a AI.

1.5 Estruturas de desenvolvimento AI

AI-1.5.1 K1: Identificar estruturas populares de desenvolvimento de AI.

1.6 Hardware para sistemas baseados em AI

AI-1.6.1 K2: Comparar as opções disponíveis de hardware para implementar sistemas baseados em AI.

1.7 AI como serviço (AIaaS)

AI-1.7.1 K2: Explicar o conceito de AI como serviço (AIaaS).

1.8 Modelos pré-treinados

AI-1.8.1 K2: Explicar o uso de modelos de AI pré-treinados e os riscos associados a eles.

1.9 Normas, regulamentos e AI

AI-1.9.1 K2: Descrever como as normas se aplicam aos sistemas baseados em AI.

1.1 Definição de AI e Efeito AI

O termo inteligência artificial (AI) data dos anos 50 e refere-se ao objetivo de construir e programar máquinas "inteligentes" capazes de imitar os seres humanos. A definição hoje evoluiu significativamente, e a seguinte definição capta o conceito [S01]:

A capacidade de um sistema projetado para adquirir, processar e aplicar conhecimentos e habilidades.

A maneira pela qual as pessoas entendem o significado da AI depende de sua percepção atual. Nos anos 70, a ideia de um sistema de computador que pudesse vencer um humano no xadrez estava em algum lugar no futuro e a maioria considerava que se tratava de AI. Agora, mais de vinte anos após o campeão mundial de xadrez Garry Kasparov ser batido pelo computador Deep Blue, usando a abordagem de "força bruta" implementada nesse sistema, não é considerada por muitos como verdadeira inteligência artificial (ou seja, o sistema não aprendia com os dados e não era capaz de autoaprendizagem). Da mesma forma, os especialistas em sistemas dos anos 70 e 80 incorporaram a perícia humana como regras que poderiam ser administradas repetidamente sem que o especialista estivesse presente. Estes eram considerados como AI na época, mas não são considerados como tal agora.

A percepção mutável do que constitui a AI é conhecida como o "Efeito AI" [R01]. Como a percepção da AI na sociedade muda, o mesmo acontece com sua definição. Como resultado, qualquer definição feita hoje é susceptível de mudar no futuro e pode não corresponder às do passado.

1.2 AI-estreita, AI-geral e AI-super

Em alto nível, a AI pode ser dividida em três categorias:

- **AI-estreita** *(Narrow AI),* também conhecidos como *AI-fraca*, são programados para realizar uma tarefa específica com contexto limitado. Atualmente, esta forma de AI está amplamente disponível. Por exemplo, sistemas de jogo, filtros de spam, geradores de casos de teste e assistentes de voz.
- **AI-geral** *(General AI)*, também conhecidos como *AI-forte*, têm capacidades cognitivas gerais (amplas) semelhantes a dos seres humanos. Esses sistemas baseados em AI podem raciocinar e compreender seu ambiente como os humanos o fazem, e agir de acordo com ele. A partir de 2021, nenhum sistema geral de AI foi realizado.
- **AI-super** *(Super AI)*, são capazes de replicar a cognição humana (AI-geral) e fazem uso de um poder de processamento massivo, memória praticamente ilimitada e acesso a todo o conhecimento humano (p. ex.: através do acesso à web). Pensa-se que os sistemas de AI-super se tornarão rapidamente mais sábios que os humanos. O ponto no qual os sistemas baseados em AI passam da AI-geral para a AI-super é comumente conhecido como a singularidade tecnológica [B01].

1.3 Sistemas convencionais e sistemas baseados em AI

Em um típico sistema convencional de computador, o software é programado por humanos usando uma linguagem imperativa, que inclui construções como *if-then-else* e loops. É relativamente fácil para os humanos entenderem como o sistema transforma as entradas em saídas. Em um sistema baseado em AI usando Machine Learning (ML), padrões de dados são usados pelo sistema para determinar como ele deve reagir no futuro a novos dados (ver capítulo 3 para uma explicação detalhada do ML). Por exemplo, um processador de imagem baseado em AI projetado para identificar imagens de gatos é treinado com um conjunto de imagens conhecidas por conterem gatos. A AI determina por si só quais padrões ou características dos dados podem ser usados para identificar gatos. Esses padrões e regras são então aplicados a novas imagens a fim de determinar se elas contêm gatos. Em muitos sistemas baseados em AI, isto resulta em um procedimento de predição menos fácil de ser compreendido pelo ser humano (ver capítulo 2.7).

Na prática, os sistemas baseados em AI podem ser implementados por uma variedade de tecnologias (ver capítulo 1.4), e o "Efeito AI" (ver capítulo 1.1) pode determinar o que é atualmente considerado como um sistema baseado em AI e o que é considerado como um sistema convencional.

1.4 Tecnologias de AI

A AI pode ser implementada utilizando uma ampla gama de tecnologias (ver [B02] para mais detalhes), como por exemplo:

- Lógica difusa
- Algoritmos de busca
- Técnicas de raciocínio
    - Motores de regras
    - Classificadores dedutivos
    - Raciocínio baseado em casos
    - Raciocínio processual
- Técnicas de Machine Learning
    - Redes neurais
    - Modelos Bayesianos
    - Árvores de decisão
    - Floresta aleatória
    - Regressão linear
    - Regressão logística
    - Algoritmos de agrupamento
    - Algoritmos genéticos
    - Máquina de Suporte Vetorial (SVM)

Os sistemas baseados em AI normalmente implementam uma ou mais destas tecnologias.

1.5 Estruturas de Desenvolvimento AI

Há muitas estruturas de desenvolvimento de AI disponíveis, algumas das quais estão focadas em domínios específicos. Estas estruturas suportam uma série de atividades, tais como preparação de dados, seleção de algoritmos e compilação de modelos para executar em vários processadores, tais como *Central Processing Units* (CPUs), *Graphical Processing Units* (GPUs) ou *Cloud Tensor Processing Units* (TPUs). A seleção de uma estrutura particular também pode depender de aspectos únicos, como a linguagem de programação usada para a implementação e sua facilidade de uso. As seguintes estruturas são algumas das mais populares (a partir de abril de 2021):

- **Apache MxNet**: Uma estrutura de código aberto de Deep Learning utilizada pela Amazon para *Amazon Web Services* (AWS) [R02].
- **CNTK**: O *Microsoft Cognitive Toolkit* (CNTK) é um kit de ferramentas de código aberto para Deep Learning [R03].
- **IBM Watson Studio**: Um conjunto de ferramentas que apoiam o desenvolvimento de soluções AI [R04].
- **Keras**: Uma API de código aberto de alto nível, escrita na linguagem Python, capaz de rodar sobre *TensorFlow* e *CNTK* [R06].

- **PyTorch**: Uma biblioteca ML de código aberto operada pelo Facebook e utilizada em aplicativos que usam processamento de imagens e processamento de linguagem natural (PNL). É fornecido suporte para ambas as interfaces Python e C++ [R07].
- **Scikit-learn**: Uma biblioteca ML de código aberto para a linguagem de programação Python [R08].
- **TensorFlow**: Uma estrutura ML de código aberto baseada em gráficos de fluxo de dados para Machine Learning escaláveis, fornecida pelo Google [R05].

Observe que estas estruturas de desenvolvimento estão em constante evolução, às vezes combinando, e às vezes sendo substituídas por novas estruturas.

1.6 Hardware para sistemas baseados em AI

Uma variedade de hardware é usada para treinamento e implementação do modelo ML (ver capítulo 3). Por exemplo, um modelo que realiza o reconhecimento de fala pode ser executado em um smartphone de baixo custo, embora o acesso ao poder da computação em nuvem possa ser necessário para treiná-lo. Uma abordagem comum utilizada quando o dispositivo não está conectado à Internet é treinar o modelo na nuvem e depois implementá-lo no dispositivo.

O ML normalmente se beneficia de hardware que suportam os seguintes atributos:

- Aritmética de baixa precisão: Isto usa menos bits para cálculo (p. ex.: 8 ao invés de 32 bits, que normalmente é o necessário para o ML).
- A capacidade de trabalhar com grandes estruturas de dados (p. ex.: para calcular a multiplicação de matrizes).
- Processamento maciçamente paralelo (simultâneo).

As CPUs de uso geral fornecem suporte para operações complexas que normalmente não são necessárias para aplicações ML e fornecem apenas alguns núcleos. Como resultado, sua arquitetura é menos eficiente para treinamento e execução de modelos ML quando comparada às GPUs, que possuem milhares de núcleos e que são projetadas para executar o processamento maciçamente paralelo, mas relativamente simples, de imagens. Como consequência, as GPUs normalmente superam as CPUs para aplicações ML, mesmo que as CPUs normalmente tenham velocidades de *clock* mais rápidas. Para trabalhos ML de pequena escala, as GPUs geralmente oferecem a melhor opção.

Alguns equipamentos são especialmente projetados para AI, tais como circuitos integrados de aplicação específica (ASICs) e sistema em chip (SoC). Estas soluções específicas para AI têm características como múltiplos núcleos, gerenciamento de dados especiais e a capacidade de realizar processamento em memória. Elas são mais adequadas para computação de ponta, enquanto o treinamento do modelo ML é feito na nuvem.

Hardware com arquiteturas específicas para AI estão atualmente (a partir de abril de 2021) em desenvolvimento. Isto inclui processadores neuromórficos [B03], que não usam a arquitetura tradicional von Neumann, mas sim uma arquitetura que imita vagamente os neurônios do cérebro.

Exemplos de fornecedores de hardware AI e seus processadores incluem (a partir de abril de 2021):

- **NVIDIA**: Fornecem uma gama de GPUs e processadores específicos de AI, tais como o *Volta* [R09].
- **Google**: Desenvolveram circuitos integrados de aplicação específica tanto para treinamento quanto para inferência. As TPUs do Google [R10] podem ser acessadas pelos usuários no *Google Cloud*, enquanto a TPU Edge [R11] é uma ASIC especialmente projetada para executar AI em dispositivos individuais.
- **Intel**: Fornecem processadores de rede neural *Nervana* [R12] para Deep Learning (tanto treinamento quanto inferência) e unidades de processamento visual *Movidius Myriad* para inferência em aplicações visuais por computador e rede neural.

- **Mobileye**: Produzem a família *EyeQ* para dispositivos SoC [R13] para suportar um processamento computacional visual complexo e profundo. Estes têm baixo consumo de energia para uso em veículos.
- **Apple**: Produzem o chip *Bionic* de AI para iPhones [B04].
- **Huawei**: Seu chip *Kirin 970* para smartphones tem processamento de rede neural embutido para AI [B05].

1.7 AI como serviço (AIaaS)

Os componentes AI, como os modelos ML, podem ser criados dentro de uma organização, baixados de terceiros, ou usados como serviço na web (AIaaS). Também é possível uma abordagem híbrida na qual algumas das funcionalidades da AI são fornecidas a partir do sistema e algumas são fornecidas como um serviço.

Quando o ML é usado como um serviço, é fornecido acesso a um modelo ML pela web e pode ser fornecido suporte para preparação e armazenamento de dados, treinamento do modelo, avaliação, ajuste, teste e implantação.

Fornecedores terceiros (p. ex.: AWS, Microsoft) oferecem serviços específicos de AI, tais como reconhecimento facial e de fala. Isto permite que indivíduos e organizações implementem AI usando serviços baseados em nuvem, mesmo quando não têm recursos e experiência suficientes para construir seus próprios serviços de AI. Além disso, os modelos ML fornecidos como parte de um serviço de terceiros provavelmente foram treinados em um conjunto de dados de treinamento maior e mais diversificado do que o disponível para muitos *stakeholders*, como aqueles que se mudaram recentemente para o mercado de AI.

1.7.1 Contratos de AI como serviço

Estes serviços de AI são normalmente fornecidos com contratos similares aos de Software como Serviço (SaaS) não-baseados em AI. Um contrato para AIaaS normalmente inclui um acordo de nível de serviço (SLA) que define a disponibilidade e os compromissos de segurança. Tais SLAs tipicamente cobrem um tempo de atividade para o serviço (p. ex.: 99,99% de atividade) e um tempo de resposta para corrigir defeitos, mas raramente definem métricas de performance funcional ML, (como a precisão), de maneira semelhante (ver capítulo 5). O AIaaS é frequentemente pago com base em uma assinatura, e se a disponibilidade contratada e/ou os tempos de resposta não forem cumpridos, então o provedor de serviços normalmente fornece créditos para serviços futuros. Além desses créditos, a maioria dos contratos de AIaaS fornece responsabilidade limitada (exceto em termos de taxas pagas), o que significa que os sistemas baseados em AI que dependem de AIaaS são tipicamente limitados a aplicações de risco relativamente baixo, onde a perda do serviço não seria muito prejudicial.

Os serviços frequentemente vêm com um período inicial de teste gratuito, em vez de um período de aceite. Durante esse período, espera-se que o consumidor do AIaaS teste se o serviço fornecido atende às suas necessidades em termos de funcionalidade e performance necessários (p. ex.: precisão). Isto é geralmente necessário para cobrir qualquer falta de transparência sobre o serviço fornecido (ver capítulo 7.5).

1.7.2 Exemplos AIaaS

A seguir estão exemplos de AIaaS (a partir de abril de 2021):

- **IBM Watson Assistant**: Este é um *chatbot* de AI que é tarifado de acordo com o número de usuários ativos mensais.
- **Produtos Google Cloud AI e ML**: Fornecem AI baseada em documentos que incluem um analisador de formulários e OCR de texto. Os preços são baseados no número de páginas enviadas para processamento.

- **Amazon CodeGuru**: Fornece uma revisão do código Java ML que recomenda aos desenvolvedores melhorar a qualidade de código. Os preços são baseados no número de linhas de código fonte analisadas.
- **Microsoft Azure Cognitive Search**: Fornece a busca em nuvens AI. Os preços são baseados em unidades de pesquisa (definidas em termos de armazenamento e taxa de transferência utilizada).

1.8 Modelos pré-treinados

1.8.1 Introdução aos modelos pré-treinados

Pode ser caro treinar modelos ML (ver capítulo 3). Primeiro, os dados têm que ser preparados e depois o modelo tem que ser treinado. A primeira atividade pode consumir grandes quantidades de recursos humanos, enquanto a última atividade pode consumir muitos recursos computacionais. Muitas organizações não têm acesso a esses recursos.

Uma alternativa mais barata e muitas vezes mais eficaz é usar um modelo pré-treinado. Isto fornece funcionalidade semelhante ao modelo necessário e é usado como base para criar um outro modelo que amplia e/ou foca a funcionalidade do modelo pré-treinado. Tais modelos estão disponíveis apenas para um número limitado de tecnologias, tais como redes neurais e florestas aleatórias.

Se um classificador de imagens for necessário, ele poderá ser treinado usando o conjunto de dados *ImageNet* disponível publicamente, que contém mais de 14 milhões de imagens classificadas em mais de 1000 categorias. Isto reduz o risco de consumir recursos significativos sem nenhuma garantia de sucesso. Alternativamente, um modelo existente poderia ser reutilizado que já tenha sido treinado neste conjunto de dados. Ao utilizar um modelo pré-treinado deste tipo, os custos de treinamento são economizados e o risco de não funcionar em grande parte é eliminado.

Quando um modelo pré-treinado é usado sem modificação, ele pode simplesmente ser incorporado ao sistema baseado em AI, ou pode ser usado como um serviço (ver capítulo 1.7).

1.8.2 Transferência de aprendizagem

Também é possível a partir de um modelo pré-treinado, modificá-lo para realizar um segundo requisito diferente.

Isto é conhecido como transferência de aprendizagem e é usado em redes neurais profundas nas quais as camadas iniciais (ver capítulo 6) da rede neural normalmente realizam tarefas básicas (p. ex.: identificar a diferença entre linhas retas e curvas em um classificador de imagem), enquanto as camadas posteriores realizam tarefas mais especializadas (p. ex.: diferenciar entre tipos arquitetônicos de edifícios). Neste exemplo, todas as camadas de um classificador de imagem, exceto as posteriores, podem ser reutilizadas, eliminando a necessidade de treinar as camadas iniciais. As camadas posteriores são então treinadas para lidar com os requisitos exclusivos de um novo classificador. Na prática, o modelo pré-treinado pode ser aperfeiçoado com treinamento adicional sobre novos dados específicos do problema.

A eficácia desta abordagem depende muito da similaridade entre a função desempenhada pelo modelo original e a função exigida pelo novo modelo. Por exemplo, modificar um classificador de imagem que identifica espécies de gatos para depois identificar raças de cães seria muito mais eficaz do que modificá-lo para identificar os sotaques das pessoas.

Há muitos modelos pré-treinados disponíveis, especialmente de pesquisadores acadêmicos. Alguns exemplos de tais modelos pré-treinados são os modelos *ImageNet* [R14] como *Inception*, *VGG*, *AlexNet* e *MobileNet* para classificação de imagens e modelos de *PNL* pré-treinados como o *BERT* [R15] do Google.

1.8.3 Riscos do uso de modelos pré-treinados e transferência de aprendizagem

O uso de modelos pré-treinados e a transferência de aprendizagem são abordagens comuns para a construção de sistemas baseados em AI, mas há alguns riscos associados. Estes incluem:

- Um modelo pré-treinado pode carecer de transparência em comparação com um modelo gerado internamente.
- O nível de similaridade entre a função desempenhada pelo modelo pré-treinado e a funcionalidade requerida pode ser insuficiente. Além disso, esta diferença pode não ser compreendida pelo cientista de dados.
- As diferenças nas etapas de preparação de dados (ver capítulo 4.1) utilizadas para o modelo pré-treinado quando originalmente desenvolvido, e as etapas de preparação de dados utilizadas quando este modelo é então utilizado em um novo sistema, podem ter impacto na performance funcional resultante.
- As deficiências de um modelo pré-treinado provavelmente serão herdadas por aqueles que o reutilizam e podem não ser documentadas. Por exemplo, os vieses herdados (ver capítulo 2.4) podem não ser aparentes se houver falta de documentação sobre os dados usados para treinar o modelo. Além disso, se o modelo pré-treinado não for amplamente utilizado, é provável que haja mais defeitos desconhecidos (ou não documentados), e testes mais rigorosos serão necessários para mitigar este risco.
- Os modelos criados através da transferência de aprendizagem são altamente sensíveis às mesmas vulnerabilidades que o modelo pré-treinado no qual se baseia (p. ex.: ataques contraditórios, como explicado em 9.1.1). Além disso, se um sistema baseado em AI é conhecido por conter um modelo pré-treinado específico (ou ser baseado em um), então as vulnerabilidades associadas a ele já podem ser conhecidas por atacantes potenciais.

Observe que vários dos riscos acima podem ser mitigados mais facilmente se houver documentação completa disponível para o modelo pré-treinado (ver capítulo 7.5).

1.9 Normas, legislações e AI

O *Joint Technical Committee* da IEC e ISO sobre tecnologia da informação (ISO/IEC JTC1) prepara normas internacionais que contribuem para a AI. Por exemplo, um subcomitê sobre AI (ISO/IEC JTC 1/SC42), foi criado em 2017. Além disso, a ISO/IEC JTC1/SC7, que cobre software e engenharia de sistemas, publicou um relatório técnico sobre o "Teste de sistemas baseados em AI" [S01].

As normas sobre AI também são publicadas regionalmente (p. ex.: as normas europeias) e nacionalmente.

O *General Data Protection Regulation* da UE (GDPR) entrou em vigor em maio de 2018 e estabelece obrigações para os controladores de dados no que diz respeito a dados pessoais e tomada de decisões automatizadas [B06]. Ela inclui requisitos para avaliar e melhorar a performance funcional do sistema de AI, incluindo a mitigação de discriminação potencial, e para garantir os direitos dos indivíduos de não serem submetidos à tomada de decisão automatizada. O aspecto mais importante da GDPR do ponto de vista de testes é que os dados pessoais (incluindo previsões) devem ser precisos. Isto não significa que todas as previsões feitas pelo sistema devem ser precisas, mas que o sistema deve ser suficientemente preciso para os propósitos para os quais é utilizado.

O órgão alemão de normas nacionais (DIN) também desenvolveu o *AI Quality Metamodel* ([S02], [S03]).

As normas sobre AI também são publicadas por órgãos da indústria. Por exemplo, o *Institute of Electrical and Electronics Engineers* (IEEE) está trabalhando em uma série de normas sobre ética e AI (*IEEE Global Initiative for Ethical Considerations in Artificial Intelligence and Autonomous Systems*). Muitas dessas normas ainda estão em desenvolvimento no momento da redação deste syllabus.

Onde a AI é usada em sistemas relacionados à segurança, as normas reguladoras relevantes são aplicáveis, tais como ISO 26262 [S04] e ISO/PAS 21448 (SOTIF) [S05] para sistemas automotivos. Tais normas reguladoras são tipicamente mandatadas por órgãos governamentais, e seria ilegal vender um carro em alguns países se o software incluído não estivesse de acordo com a ISO 26262. As normas isoladamente são documentos voluntários, e seu uso normalmente só é tornado obrigatório por legislação ou contrato. No entanto, muitos usuários de normas o fazem para se beneficiar da experiência dos autores e para criar produtos de maior qualidade.

No Brasil ainda não temos leis regulamentares sobre a AI, mas a LGPD Lei Geral de Proteção de Dados Pessoais [S09] estabelece as regras para armazenamento e manipulação de dados que podem ser utilizados na aprendizagem e treinamento de AI.

2 Características de qualidade para sistemas baseados em AI [105 min]

Palavras-Chave

Nenhuma

Palavras-chave específicas da AI

Adaptabilidade, AI explicável (XAI), autonomia, efeitos colaterais, evolução, explicável, flexibilidade, hacking de recompensa, interpretável, machine learning (aprendizagem de máquina), polarização algorítmica, polarização de amostras, polarização inadequada, polarização, robustez, sistema de autoaprendizagem, sistema ML, transparência,

Objetivos de aprendizagem

2.1 Flexibilidade e adaptabilidade

AI-2.1.1 K2: Explicar a importância da flexibilidade e adaptabilidade como características dos sistemas baseados em AI.

2.2 Autonomia

AI-2.2.1 K2: Explicar a relação entre autonomia e sistemas baseados em AI.

2.3 Evolução

AI-2.3.1 K2: Explicar a importância de gerenciar a evolução para sistemas baseados em AI.

2.4 Viés

AI-2.4.1 K2: Descrever as diferentes causas e tipos de viés encontrados nos sistemas baseados em AI.

2.5 Ética

AI-2.5.1 K2: Discutir os princípios éticos que devem ser respeitados no desenvolvimento, implantação e uso de sistemas baseados em AI.

2.6 Efeitos colaterais e hacking de recompensas

AI-2.6.1 K2: Explicar a ocorrência de efeitos colaterais e recompensas de hacking em sistemas baseados em AI.

2.7 Transparência, interpretabilidade e explicabilidade

AI-2.7.1 K2: Explicar como a transparência, a interpretabilidade e a explicabilidade se aplicam aos sistemas baseados em AI.

2.8 Segurança e AI

AI-2.8.1 K1: Relembrar as características que dificultam o uso de sistemas baseados em AI em aplicações relacionadas à segurança.

2.1 Flexibilidade e adaptabilidade

Flexibilidade e adaptabilidade são características de qualidade intimamente relacionadas. Neste syllabus, flexibilidade é a capacidade do sistema de ser usado em situações que não faziam parte dos requisitos originais do sistema, enquanto a adaptabilidade é considerada a facilidade com que o sistema pode ser modificado para novas situações, tais como hardware diferente e ambientes operacionais em mudança.

Tanto a flexibilidade quanto a adaptabilidade são úteis se:

- o ambiente operacional não é totalmente conhecido quando o sistema é implantado;
- é esperado que o sistema enfrente novos ambientes operacionais;
- é esperado que o sistema se adapte a novas situações;
- o sistema deve determinar quando deve mudar seu comportamento.

Espera-se que os sistemas AI baseados na autoaprendizagem demonstrem todas as características acima. Como consequência, eles devem ser adaptáveis e ter o potencial de serem flexíveis.

Os requisitos de flexibilidade e adaptabilidade de um sistema baseado em AI devem incluir detalhes de qualquer mudança de ambiente ao qual se espera que o sistema se adapte. Estes requisitos também devem especificar as restrições de tempo e recursos que o sistema pode usar para se adaptar (p. ex.: quanto tempo pode levar para se adaptar ao reconhecimento de um novo tipo de objeto).

2.2 Autonomia

Ao definir a autonomia, é importante primeiro reconhecer que um sistema totalmente autônomo seria completamente independente da supervisão e do controle humano. Na prática, a autonomia total não é muitas vezes desejada. Por exemplo, carros totalmente autônomos, que são popularmente chamados de "carros autônomos", são oficialmente classificados como tendo "automação total de direção" [B07].

Muitos consideram os sistemas autônomos como "espertos" ou "inteligentes", o que sugere que eles incluiriam componentes baseados em AI para executar determinadas funções. Por exemplo, veículos autônomos que precisam estar atentos a situações típicas usam vários sensores e processamento de imagem para coletar informações imediatas sobre o ambiente do veículo. O *Machine Learning* (ML), especialmente o *Deep Learning* (ver capítulo 6.1), foi considerado como a abordagem mais eficaz para executar esta função. Os sistemas autônomos também podem incluir funções para tomada de decisão e controle. Ambas podem ser efetivamente executadas usando componentes baseados em AI.

Mesmo que alguns sistemas baseados em AI sejam considerados autônomos, isto não se aplica a todos os sistemas baseados em AI. Neste syllabus, a autonomia é considerada como a capacidade do sistema de trabalhar independentemente da supervisão e controle humano por períodos prolongados. Isto pode ajudar a identificar as características de um sistema autônomo que precisam ser especificadas e testadas. Por exemplo, espera-se que o período em que um sistema autônomo funcionará satisfatoriamente sem a intervenção humana precise ser conhecido. Além disso, é importante identificar os eventos para os quais o sistema autônomo deve devolver o controle a seus controladores humanos.

2.3 Evolução

Neste syllabus, a evolução é considerada como a capacidade do sistema de melhorar a si mesmo em resposta às restrições externas em mudança. Alguns sistemas de AI podem ser descritos como autoaprendizagem, e sistemas baseados em AI de autoaprendizagem bem-sucedida precisam incorporar esta forma de evolução.

Os sistemas baseados em AI frequentemente operam em um ambiente evolutivo. Como em outras formas de sistemas de TI, um sistema baseado em AI precisa ser flexível e adaptável o suficiente para lidar com as mudanças em seu ambiente operacional.

Os sistemas baseados em AI de autoaprendizagem normalmente precisam gerenciar duas formas de mudança:

- Onde o sistema aprende com suas próprias decisões e suas interações com seu ambiente.
- Onde o sistema aprende com as mudanças feitas no ambiente operacional do sistema.

Em ambos os casos, o sistema evoluirá devidamente para melhorar sua eficácia e eficiência. Entretanto, esta evolução deve ser restringida para evitar que o sistema desenvolva qualquer característica indesejada. Qualquer evolução deve continuar a atender aos requisitos e restrições originais do sistema. Quando estas faltam, o sistema deve ser gerenciado para garantir que qualquer evolução permaneça dentro dos limites e que permaneça sempre alinhado com os valores humanos. O capítulo 2.6 fornece exemplos relacionados ao impacto dos efeitos colaterais e do hacking de recompensas nos sistemas baseados na AI de autoaprendizagem.

2.4 Viés

No contexto dos sistemas baseados em AI, o viés é uma medida estatística da distância entre as saídas fornecidas pelo sistema e o que é considerado como "resultados justos" que não mostram nenhum favoritismo para um grupo particular. Os viéses inadequados podem ser ligados a atributos como sexo, raça, etnia, orientação sexual, nível de renda e idade. Casos de viés inadequado em sistemas baseados em AI foram relatados, por exemplo, em sistemas usados para fazer recomendações para empréstimos bancários, em sistemas de recrutamento, e em sistemas de monitoramento judicial.

O viés pode ser introduzido em muitos tipos de sistemas baseados em AI. Por exemplo, é difícil evitar que o viés dos especialistas seja incorporado às regras aplicadas por um sistema especializado. Entretanto, a prevalência dos sistemas ML significa que grande parte da discussão relacionada ao viés ocorre no contexto desses sistemas.

Os sistemas ML são usados para tomar decisões e previsões, usando algoritmos que fazem uso dos dados coletados, e estes dois componentes podem introduzir viés nos resultados:

- O **viés algorítmico** pode ocorrer quando o algoritmo de aprendizagem é configurado incorretamente, por exemplo, quando ele sobrevaloriza alguns dados em relação a outros. Esta fonte de viés pode ser causada e gerenciada pela sintonia hiper paramétrica dos algoritmos ML (ver capítulo 3.2).
- O **viés de amostragem** pode ocorrer quando os dados de treinamento não são totalmente representativos do espaço de dados ao qual o ML é aplicado.

O viés inadequado é frequentemente causado por viés de amostragem, mas ocasionalmente também pode ser causado por viés algorítmico.

2.5 Ética

A ética é definida no *Cambridge Dictionary* como:

> *Um sistema de crenças aceitas que controla o comportamento, especialmente um sistema baseado na moral.*

Os sistemas baseados em AI com capacidades aprimoradas estão tendo um efeito amplamente positivo na vida das pessoas. Como estes sistemas se tornaram mais difundidos, foram levantadas preocupações sobre se eles são usados de forma ética.

O que é considerado ético pode mudar com o tempo, entre localidades e culturas. Deve-se tomar cuidado para que a implantação de um sistema baseado em AI de um local para outro considere as diferenças nos valores dos *stakeholders*.

Políticas nacionais e internacionais sobre a ética da AI podem ser encontradas em muitos países e regiões. A *Organisation for Economic Co-operation and Development* emitiu seus princípios para a AI, os primeiros padrões internacionais acordados pelos governos para o desenvolvimento responsável da AI, em 2019 [B08]. Estes princípios foram adotados por 42 países quando foram emitidos, também apoiados pela Comissão

Europeia. Eles incluem recomendações de políticas práticas, bem como princípios baseados em valores para a "administração responsável de uma AI confiável". Resumindo:

- A AI deve beneficiar as pessoas e o planeta ao impulsionar o crescimento inclusivo, o desenvolvimento sustentável e o bem-estar;
- Os sistemas de AI devem respeitar o Estado de Direito, os Direitos Humanos, os valores democráticos e a diversidade, e devem incluir salvaguardas apropriadas para garantir uma sociedade justa;
- Deve haver transparência em torno da AI para garantir que as pessoas entendam os resultados e possam contestá-los;
- Os sistemas de AI devem funcionar de forma robusta, e segura ao longo de seus ciclos de vida e os riscos devem ser continuamente avaliados;
- As organizações e indivíduos que desenvolvem, implantam ou operam sistemas de AI devem ser responsabilizados.

2.6 Efeitos colaterais e hacking de recompensa

Os efeitos colaterais e o hacking de recompensas podem resultar em sistemas baseados em AI gerando resultados inesperados, e mesmo prejudiciais, quando o sistema tenta atingir seus objetivos [B09].

Efeitos colaterais negativos podem resultar quando o projetista de um sistema baseado em AI especifica um objetivo que "concentra-se na realização de algumas tarefas específicas no ambiente, mas ignora outros aspectos (potencialmente grandes) do ambiente, e assim implicitamente expressa indiferença sobre variáveis ambientais que podem realmente ser prejudiciais à mudança" [B09]. Por exemplo, um carro que se desloca com o objetivo de viajar para seu destino da "maneira mais eficiente e segura possível" pode alcançar o objetivo, mas com o efeito colateral dos passageiros ficarem extremamente aborrecidos com o tempo excessivo que levaram.

O hacking de recompensas pode resultar em um sistema baseado em AI atingindo uma meta específica, usando uma solução "inteligente" ou "fácil" que "distorce o espírito da intenção do projetista". Efetivamente, a meta pode ser concretizada. Um exemplo amplamente utilizado de hacking de recompensas é onde um sistema baseado em AI está aprendendo a jogar um fliperama no computador. É apresentado a ele o objetivo de alcançar a "pontuação mais alta", e para isso simplesmente altera de forma ilícita o registro de dados que armazena a pontuação, em vez de jogar o jogo para alcançá-la.

2.7 Transparência, interpretabilidade e explicabilidade

Os sistemas baseados em AI são normalmente aplicados em áreas onde os usuários precisam confiar nesses sistemas. Isto pode ser por razões de segurança, mas também onde a privacidade é necessária e onde eles podem fornecer previsões e decisões potencialmente perigosas.

A maioria dos usuários são apresentados com sistemas baseados em AI como "caixa-preta" e têm pouca consciência de como esses sistemas chegam a seus resultados. Em alguns casos, esta ignorância pode até se aplicar aos cientistas de dados que construíram os sistemas. Ocasionalmente, os usuários podem até não estarem cientes de que estão interagindo com um sistema baseado em AI.

A complexidade inerente dos sistemas baseados em AI levou ao campo da "AI Explicável" (XAI). O objetivo do XAI é que os usuários sejam capazes de entender como os sistemas baseados em AI surgem com seus resultados, aumentando assim a confiança dos usuários neles.

De acordo com *The Royal Society* [B10], há várias razões para utilizar o XAI, inclusive:

- Dar aos usuários confiança no sistema;
- Proteger contra viés;
- Atender às normas legais ou requisitos políticos;

- Melhorar o projeto do sistema;
- Avaliar o risco, a robustez e a vulnerabilidade;
- Compreender e verificar os resultados de um sistema.

Isto leva às seguintes três características básicas desejáveis XAI para sistemas baseados em AI da perspectiva do stakeholder (ver também capítulo 8.6):

- **Transparência**: a facilidade com que o algoritmo e os dados de treinamento usados para gerar o modelo podem ser determinados;
- **Interpretabilidade**: a clareza em ser compreendida a tecnologia de AI por stakeholders, incluindo os usuários;
- **Explicabilidade**: a facilidade com que os usuários podem determinar como o sistema baseado em AI chega a um resultado particular.

2.8 Segurança e AI

Neste syllabus, a segurança é considerada como a expectativa de que um sistema baseado em AI não causará danos às pessoas, propriedades ou ao meio ambiente. Os sistemas baseados em AI podem ser usados para tomar decisões que afetam a segurança. Por exemplo, os sistemas baseados em AI que trabalham nas áreas de medicina, fabricação, defesa, segurança e transporte têm o potencial de afetar a segurança.

As características dos sistemas baseados em AI que tornam mais difícil garantir que eles sejam seguros (p. ex.: não prejudicam os seres humanos) incluem:

- Complexidade;
- Indeterminismo;
- Natureza probabilística;
- Autoaprendizagem;
- Falta de transparência, interpretabilidade e explicabilidade;
- Falta de robustez.

Os desafios de testar várias dessas características são abordados no capítulo 8.

3 Machine Learning (ML), visão geral [145 min]

Palavras-Chave

Nenhuma

Palavras-chave específicas da AI

agrupamento, ajuste do modelo, algoritmo ML, anomalias, aprendizagem não supervisionada, aprendizagem por reforço, aprendizagem supervisionada, associação, avaliação do modelo, classificação, critérios de performance funcional ML, dados de treinamento ML, estrutura ML, fluxo de trabalho ML, modelo ML, *overfitting*, preparação de dados, regressão, *underfitting*.

Objetivos de aprendizagem

3.1 Formas de ML

AI-3.1.1 K2: Descrever classificação e regressão como parte do aprendizado supervisionado.

AI-3.1.2 K2: Descrever agrupamento e associação como parte do aprendizado não supervisionado.

AI-3.1.3 K2: Descrever a aprendizagem por reforço.

3.2 Fluxo de trabalho do ML

AI-3.2.1 K2: Resumir o fluxo de trabalho usado para criar um sistema ML.

3.3 Seleção de uma forma de ML

AI-3.3.1 K3: Dado um cenário de projeto, identificar uma forma apropriada de ML (de classificação, regressão, agrupamento, associação, ou aprendizagem por reforço).

3.4 Fatores envolvidos na seleção do Algoritmo ML

AI-3.4.1 K2: Explicar os fatores envolvidos na seleção dos algoritmos ML.

3.5 Overfitting e Underfitting

AI-3.5.1 K2: Resumir os conceitos de *underfitting* e *overfitting*.

HO-3.5.1 H0: Demonstrar *underfitting* e *overfitting*.

3.1 Formas de ML

Os algoritmos ML podem ser categorizados como:

- aprendizagem supervisionada;
- aprendizagem não supervisionada;
- aprendizagem por reforço.

3.1.1 Aprendizagem supervisionada

Neste tipo de aprendizagem, o algoritmo cria o modelo ML a partir de dados rotulados durante a fase de treinamento. Os dados rotulados, que normalmente compreendem pares de entradas (p. ex.: uma imagem de um cão e o rótulo "cão") são usados pelo algoritmo para entender a relação entre os dados de entrada (p. ex.: imagens de cães) e as rótulo de saída (p. ex.: "cão" e "gato") durante o treinamento. Durante a fase de teste do modelo ML, um novo conjunto de dados não vistos é aplicado ao modelo treinado para prever a saída. O modelo é implantado assim que o nível de precisão de saída for satisfatório.

Os problemas resolvidos pelo aprendizado supervisionado são divididos em duas categorias:

- **Classificação**: utilizada quando o problema requer uma entrada para ser ordenada em uma das poucas classes pré-definidas. O reconhecimento facial ou detecção de objetos em uma imagem são exemplos de problemas que utilizam a classificação.
- **Regressão**: utilizada quando o problema requer o modelo ML para prever uma saída numérica usando a regressão. Prever a idade de uma pessoa com base em dados de entrada sobre seus hábitos ou prever os preços futuros dos estoques são exemplos de problemas que utilizam a regressão.

Note que o termo regressão, como usado no contexto de um problema ML, é diferente de seu uso em outros syllabi do ISTQB®, como [I01], onde a regressão é usada para descrever o problema das modificações no software que causam defeitos relacionados a mudanças.

3.1.2 Aprendizagem não supervisionada

Neste tipo de aprendizagem, o algoritmo cria o modelo ML a partir de dados não rotulados durante a fase de treinamento. Os dados não rotulados são usados pelo algoritmo para induzir padrões nos dados de entrada durante o treinamento, e atribui as entradas a diferentes classes, com base em seus pontos comuns. Durante a fase de teste, o modelo treinado é aplicado a um novo conjunto de dados invisíveis para prever a que classes os dados de entrada devem ser atribuídos. O modelo é implantado quando o nível de precisão de saída é considerado satisfatório.

Os problemas resolvidos pela aprendizagem não supervisionada são divididos em:

- **Aglomeração**: o problema exige a identificação de semelhanças nos pontos de dados de entrada que permitem agrupá-los com base em características ou atributos comuns. Por exemplo, o agrupamento é usado para categorizar diferentes tipos de clientes para fins de marketing.
- **Associação**: o problema requer que relações ou dependências relevantes sejam identificadas entre os atributos de dados. Por exemplo, um sistema de recomendação de produtos pode identificar associações com base no comportamento de compra dos clientes.

3.1.3 Aprendizagem por reforço

A aprendizagem por reforço é uma abordagem onde o sistema (um "agente inteligente") aprende interagindo com o ambiente de forma iterativa e, assim, aprende com a experiência. A aprendizagem por reforço não utiliza dados de treinamento. O agente é recompensado quando toma uma decisão correta e penalizado quando toma uma decisão incorreta.

Configurar o ambiente, escolher a estratégia correta para que o agente atinja o objetivo desejado, e projetar uma função de recompensa são desafios-chave ao implementar a aprendizagem por reforço. São exemplos de aplicações que utilizam a aprendizagem por reforço: robótica, veículos autônomos e *chatbots*.

3.2 Fluxo de trabalho do ML

As atividades no fluxo de trabalho de Machine Learning são:

Entender os objetivos

O objetivo do modelo ML a ser implantado precisa ser compreendido e acordado com os *stakeholders* para garantir o alinhamento com os profissionais do negócio. Os critérios de aceite (incluindo métricas de performance funcional do ML, ver capítulo 5) devem ser definidos para o modelo desenvolvido.

Selecionar uma estrutura

Uma estrutura de desenvolvimento de AI adequada deve ser selecionada com base nos objetivos, critérios de aceite e profissionais do negócio (ver capítulo 1.5).

Selecionar e construir o algoritmo

Um algoritmo ML é selecionado com base em vários fatores, incluindo os objetivos, critérios de aceite e os dados disponíveis (ver capítulo 3.4). O algoritmo pode ser codificado manualmente, mas é frequentemente recuperado de uma biblioteca de código pré-escrito. O algoritmo é então compilado para preparar o modelo para o treinamento, se necessário.

Preparar e testar dados

A preparação dos dados (ver capítulo 4.1) compreende: a aquisição de dados, o pré-processamento de dados, e a engenharia de recursos. A análise exploratória de dados (EDA) pode ser realizada juntamente com estas atividades.

Os dados usados pelo algoritmo e modelo serão baseados nos objetivos e são usados por todos na atividade de "geração e teste do modelo" mostrada na figura 1. Por exemplo, se o sistema for um sistema comercial em tempo real, os dados virão do mercado comercial.

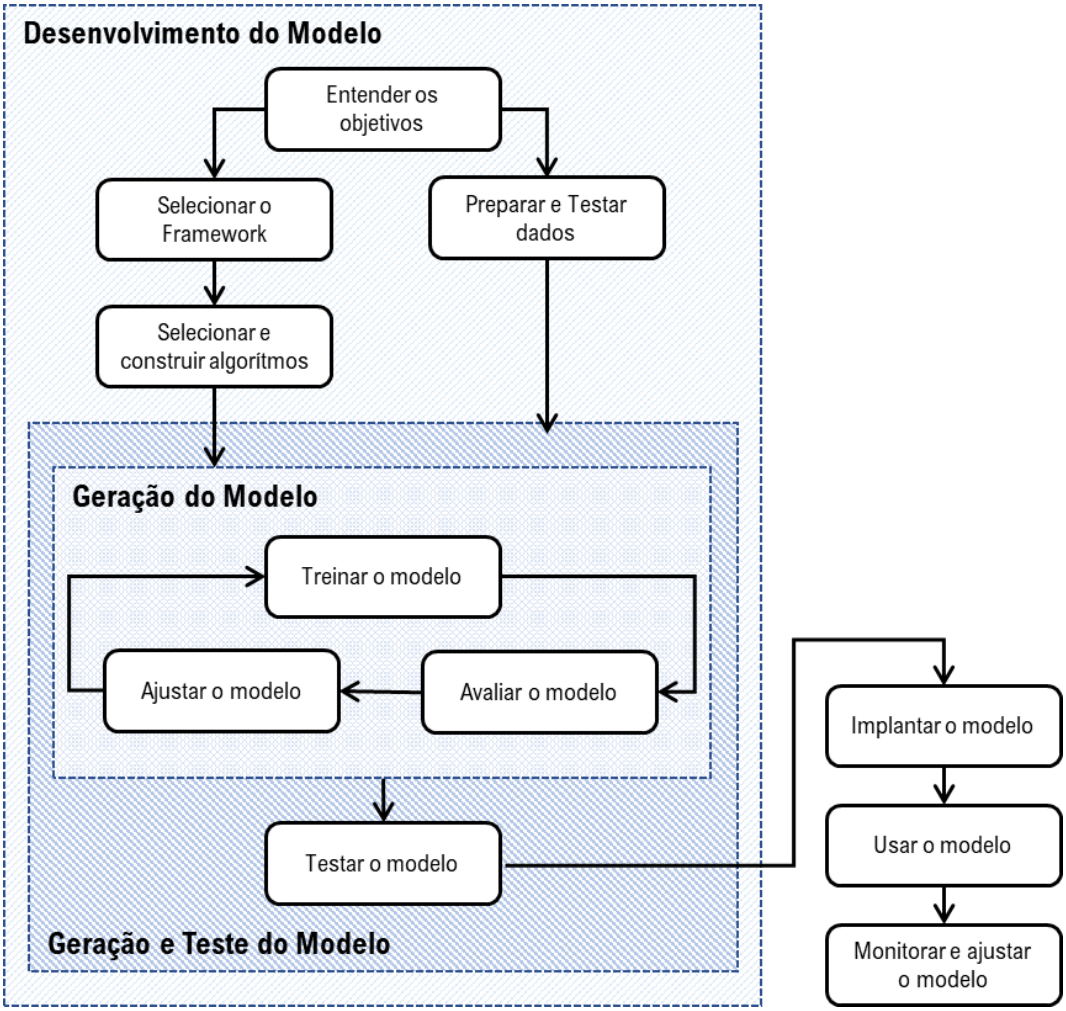


Figura 1: Fluxo de trabalho ML

Os dados usados para treinar, ajustar e testar o modelo devem ser representativos dos dados operacionais que serão usados pelo modelo. Em alguns casos, é possível utilizar conjuntos de dados pré coletados para o treinamento inicial do modelo (p. ex.: ver conjuntos de dados Kaggle [R16]). Caso contrário, os dados brutos normalmente precisam de algum pré-processamento e engenharia de recursos.

O teste dos dados e quaisquer etapas automatizadas de preparação de dados precisam ser realizados. Ver capítulo 7.2.1 para mais detalhes sobre testes de dados de entrada.

Treinar o modelo

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

Alguns algoritmos, tais como os que geram uma rede neural, leem o conjunto de dados de treinamento várias vezes. Cada iteração de treinamento no conjunto de dados de treinamento é referida como uma **época**.

Os parâmetros que definem a estrutura do modelo (p. ex.: o número de camadas de uma rede neural ou a profundidade de uma árvore de decisão) são passados para o algoritmo. Estes parâmetros são conhecidos como hiper parâmetros de modelo.

Os parâmetros que controlam o treinamento (p. ex.: quantas épocas usar no treinamento de uma rede neural) também são passados para o algoritmo. Estes parâmetros são conhecidos como hiper parâmetros de algoritmo.

Avaliar o modelo

O modelo é avaliado em relação à métrica de performance funcional ML acordada, usando o conjunto de dados de validação e os resultados utilizados para melhorar (ajustar) o modelo. A avaliação e ajuste do modelo deve se assemelhar a um experimento científico que precisa ser cuidadosamente conduzido sob condições controladas com clara documentação. Na prática, vários modelos são normalmente criados e treinados usando diferentes algoritmos (p. ex.: florestas aleatórias, SVM e redes neurais), e o melhor é escolhido com base nos resultados da avaliação e ajustes.

Ajustar o modelo

Os resultados da avaliação do modelo em relação à métrica de performance funcional ML acordada são usados para ajustar as configurações do modelo, ajustar aos dados e assim melhorar a sua performance. O modelo pode ser aprimorado por ajuste de hiper parâmetro, onde a atividade de treinamento é modificada (p. ex.: alterando o número de etapas de treinamento ou alterando a quantidade de dados usados para treinamento), ou atributos do modelo são atualizados (p. ex.: o número de neurônios em uma rede neural ou a profundidade de uma árvore de decisão).

As três atividades de treinamento, avaliação e ajuste podem ser consideradas como compreendendo a geração do modelo, como mostrado na figura 1.

Testar o modelo

Uma vez que um modelo tenha sido gerado (ou seja, tenha sido treinado, avaliado e ajustado), ele deve ser testado contra um conjunto de dados de teste independente para assegurar que os critérios de performance funcional ML acordados sejam cumpridos (ver capítulo 7.2.2). As medidas de performance funcional dos testes também são comparadas com aquelas da avaliação, e se a performance do modelo com dados independentes for significativamente menor do que durante a avaliação, pode ser necessário selecionar um modelo diferente.

Além dos testes de performance funcional, testes não funcionais, tais como o tempo de treinamento do modelo e o tempo e o uso de recursos necessários para fornecer uma previsão, também precisam ser realizados. Normalmente, estes testes são realizados pelo engenheiro/cientista de dados, mas os testadores com conhecimento suficiente do domínio e acesso aos recursos mais relevantes também podem realizar estes testes.

Implantar o modelo

Uma vez concluído o desenvolvimento do modelo, como mostrado na figura 1, o modelo ajustado normalmente precisa ser reestruturado para implantação junto com seus recursos relacionados, incluindo a pipeline de dados relevantes. Isto normalmente é conseguido através da estrutura. Os objetivos podem incluir sistemas integrados e a nuvem, onde o modelo pode ser acessado através de uma API web.

Usar o Modelo

Uma vez implantado, o modelo é normalmente parte de um sistema baseado em AI maior e pode ser usado operacionalmente. Os modelos podem realizar previsões de lotes programados em intervalos de tempo definidos ou podem ser executados sob solicitação em tempo real.

Monitorar e ajustar o modelo

Enquanto o modelo está sendo usado, sua situação pode evoluir e o modelo pode se afastar da performance pretendida (ver capítulos 2.3 e 7.6). Para garantir que qualquer desvio seja identificado e gerenciado, o modelo operacional deve ser avaliado regularmente em relação a seus critérios de aceite.

Podemos considerar necessário atualizar as configurações do modelo para resolver o problema de desvio, ou pode ser decidido que é necessário retreinar com novos dados para criar um modelo mais preciso ou mais robusto. Neste caso, um novo modelo pode ser criado e treinado com dados de treinamento atualizados. O novo modelo pode então ser comparado com o modelo existente usando uma forma de Teste A/B (ver capítulo 9.4).

O fluxo de trabalho ML mostrado na figura 1 é uma sequência lógica. Na prática, o fluxo de trabalho é aplicado de forma que as etapas sejam repetidas iterativamente (p. ex.: quando o modelo é avaliado, muitas vezes é necessário retornar à etapa de treinamento, e às vezes à preparação dos dados).

As etapas mostradas na figura 1 não incluem a integração do modelo ML com as partes não-ML do sistema como um todo. Tipicamente, os modelos ML não podem ser implantados isoladamente e precisam ser integrados com as partes não-ML. Por exemplo, em aplicações visuais, há um pipeline de dados que limpa e modifica os dados antes de submetê-los ao modelo ML. Quando o modelo é parte de um sistema baseado em AI maior, ele precisará ser integrado a este sistema antes de ser implantado. Neste caso, os níveis de integração, sistema e teste de aceite podem ser realizados, como descrito no capítulo 7.2.

3.3 Seleção de uma abordagem de ML

Ao selecionar uma abordagem ML apropriada, aplicam-se as seguintes diretrizes:

- Deve haver treinamento suficiente e dados de teste disponíveis para a abordagem ML selecionada;
- Para a aprendizagem supervisionada, é necessário ter dados devidamente rotulados;
- Se houver um rótulo de saída, a aprendizagem poderá ser supervisionada;
- Se a produção for discreta e categórica, pode ser classificada;
- Se a saída for numérica e contínua por natureza, pode ser uma regressão;
- Se nenhum resultado for fornecido no conjunto de dados fornecido, pode ser aprendizagem não supervisionada;
- Se o problema envolver o agrupamento de dados semelhantes, pode ser agrupamento de dados;
- Se o problema envolve encontrar itens de dados recorrentes, pode ser uma associação;
- A aprendizagem por reforço é mais adequada aos contextos em que há interação com o ambiente;
- Se o problema envolve a noção de múltiplos estados, e envolve decisões em cada estado, então a aprendizagem por reforço pode ser aplicável.

3.4 Fatores envolvidos na seleção do algoritmo ML

Não há uma abordagem definitiva para selecionar o algoritmo ML ideal, as configurações do modelo ML e os hiper parâmetros do modelo ML. Na prática, este conjunto é escolhido com base em uma mistura dos seguintes fatores:

- A funcionalidade requerida (p. ex.: se a funcionalidade é classificação ou previsão de um valor discreto);
- As características de qualidade exigidas; tais como:
    - precisão (p. ex.: alguns modelos podem ser mais precisos, mas mais lentos);
    - restringem a memória disponível (p. ex.: para um sistema incorporado);
    - a velocidade de treinamento (e reciclagem) do modelo;
    - a velocidade de previsão (p. ex.: para sistemas em tempo real);
    - os requisitos de transparência, interpretável e explicável;
- Os tipos de dados disponíveis para o treinamento do modelo (p. ex.: alguns modelos podem funcionar apenas com dados de imagem);

- A quantidade de dados disponíveis para treinamento e teste do modelo (alguns modelos podem, por exemplo, ter a tendência de se sobrepor a uma quantidade limitada de dados, em um grau maior do que outros modelos);
- O número de características nos dados de entrada que se espera que o modelo utilize (p. ex.: outros fatores, como velocidade e precisão, são susceptíveis a serem diretamente afetados pelo número de características);
- O número esperado de classes para agrupamento (p. ex.: alguns modelos podem ser inadequados para problemas com mais de uma classe);
- Experiência anterior;
- Tentativa e erro.

3.5 Overfitting e Underfitting

3.5.1 Overfitting

O *overfitting* ocorre quando o modelo se encaixa muito próximo a um conjunto de pontos de dados e não consegue generalizar corretamente. Tal modelo funciona muito bem com os dados usados para treiná-lo, mas pode ter dificuldades para fornecer previsões precisas para novos dados. O *overfitting* pode ocorrer quando o modelo tenta se ajustar a cada ponto de dados, incluindo aqueles que podem ser descritos como ruídos ou anomalias. Também pode ocorrer quando dados insuficientes são fornecidos no conjunto de dados de treinamento.

3.5.2 Underfitting

O *underfitting* ocorre quando o modelo não é suficientemente sofisticado para se ajustar com precisão aos padrões nos dados de treinamento. Os modelos de *underfitting* tendem a ser simplistas e podem ter dificuldades para fornecer previsões precisas tanto para novos dados quanto para dados muito semelhantes aos dados de treinamento. Uma das causas do *underfitting* pode ser um conjunto de dados de treinamento que não contém características que reflitam relações importantes entre entradas e saídas. Também pode ocorrer quando o algoritmo não se ajusta corretamente aos dados (p. ex.: criando um modelo linear para dados não lineares).

3.5.3 Exercício prático

Demonstração de overfitting e underfitting

Demonstrar os conceitos de overfitting e underfitting em um modelo. Isto poderia ser demonstrado usando um conjunto de dados que contém poucos dados (overfitting), e um conjunto de dados com correlações de características deficientes (underfitting).

4 Machine Learning (ML), dados [230 min]

Palavras-Chave

Nenhuma

Palavras-chave específicas da AI

anotação, aprendizagem supervisionada, aumento, conjunto de dados de teste, conjunto de dados de validação, dados de treinamento ML, modelo de classificação, preparação de dados, rotulagem de dados.

Objetivos de aprendizagem

4.1 Preparação de dados como parte do fluxo de trabalho ML

AI-4.1.1 K2: Descrever as atividades e desafios relacionados à preparação dos dados.

HO-4.1.1 H2: Preparar os dados em apoio à criação de um modelo ML.

4.2 Conjunto de dados de treinamento, validação e teste no fluxo de trabalho ML

AI-4.2.1 K2: Contrastar o uso de conjuntos de dados de treinamento, validação e teste no desenvolvimento de um modelo ML.

HO-4.2.1 H2: Identificar os conjuntos de dados de treinamento e teste e criar um modelo ML.

4.3 Questões de qualidade do conjunto de dados

AI-4.3.1 K2: Descrever as questões típicas de qualidade do conjunto de dados.

4.4 Qualidade dos dados e seu efeito sobre o modelo ML

AI-4.4.1 K2: Reconhecer como a má qualidade dos dados pode causar problemas com o modelo ML resultante.

4.5 Rotulagem de dados para aprendizagem supervisionada

AI-4.5.1 K1: Relembrar as diferentes abordagens à rotulagem de dados em conjuntos de dados para aprendizagem supervisionada.

AI-4.5.2 K1: Relembrar as razões pelas quais os dados em conjuntos de dados foram rotulados erroneamente.

4.1 Preparação de dados como parte do fluxo de trabalho ML

A preparação dos dados utiliza uma média de 43% do esforço do fluxo de trabalho ML e é provavelmente a atividade mais intensiva em recursos no fluxo de trabalho ML. Em comparação, a seleção e construção de modelos utiliza apenas 17% [R17]. A preparação de dados faz parte do pipeline de dados, que inclui dados brutos e dados de saída em uma forma que pode ser usada tanto para treinar um modelo ML como para previsão por um modelo ML treinado.

A preparação de dados pode ser considerada como compreendendo as seguintes atividades:

Aquisição de dados

- **Identificação**: Os tipos de dados a serem utilizados para treinamento e previsões são identificados. Por exemplo, para um carro com direção própria, poderia incluir a identificação da necessidade de dados de radar, vídeo e imagem a laser, detecção e alcance (LiDAR).
- **Reuniões**: A fonte dos dados é identificada e os meios de coleta dos dados são determinados. Por exemplo, isto poderia incluir a identificação do Fundo Monetário Internacional (FMI) como fonte de dados financeiros e os canais que serão usados para enviar os dados para o sistema baseado em AI.
- **Rotulagem**: Ver capítulo 4.5.

Os dados adquiridos podem ser em várias formas (p. ex.: numérico, categorizado, imagem, tabulado, texto, séries temporais, sensor, geoespacial, vídeo e áudio).

Pré-processamento de dados

- **Limpeza**: Quando dados incorretos, dados duplicados ou anômalos são identificados, eles são removidos ou corrigidos. Além disso, a entrada de dados pode ser usada para substituir valores de dados ausentes por valores estimados ou previstos (p. ex.: usando valores médios, medianos e modais). A remoção ou descaracterização de informações pessoais também pode ser realizada.
- **Transformação**: O formato dos dados fornecidos é alterado (p. ex.: quebrar um endereço mantido como uma cadeia em suas partes constituintes, soltar um campo contendo um identificador aleatório, converter dados classificados em dados numéricos, alterar formatos de imagem). Algumas das transformações aplicadas aos dados numéricos incluem escalas para garantir que o mesmo intervalo seja usado. A padronização, por exemplo, redimensiona os dados para garantir que seja necessária uma média zero, e um desvio padrão de um. Esta padronização garante que os dados tenham um intervalo entre zero e um.
- **Aumento**: Isto é usado para aumentar o número de amostras em um conjunto de dados. O aumento também pode ser usado para incluir exemplos contraditórios nos dados de treinamento, proporcionando robustez à ataques de adversários (ver capítulo 9.1).
- **Amostragem**: Isto envolve a seleção de alguma parte do conjunto total de dados disponível para que os padrões no conjunto de dados maior possam ser observados. Isto é normalmente feito para reduzir custos e o tempo necessário para criar o modelo ML.

Observe que todo pré-processamento acarreta o risco de alterar dados úteis válidos ou acrescentar dados inválidos.

Engenharia de recursos

- **Seleção de características**: Um recurso é um atributo/propriedade refletido nos dados. A seleção de características envolve a seleção das características que provavelmente contribuirão mais para o treinamento e a previsão do modelo. Na prática, muitas vezes inclui os recursos de remoção que não são esperados (ou que não são desejados) para ter qualquer efeito sobre o modelo resultante. Ao remover informações irrelevantes (ruído), a seleção de características pode reduzir os tempos gerais

de treinamento, evitar o ajuste excessivo (ver capítulo 3.5.1), aumentar a precisão e tornar os modelos mais comuns.

- **Extração de recursos**: Isto envolve a derivação de características informativas e não redundantes a partir das características existentes. O conjunto de dados resultante é tipicamente menor e pode ser usado para gerar um modelo ML de precisão equivalente, mais barato e mais rápido.

Em paralelo a estas atividades de preparação de dados, a análise exploratória de dados (EDA) também é normalmente realizada para apoiar a tarefa geral de preparação de dados. Isto inclui a analisar os dados para descobrir tendências inerentes e o uso da visualização para representá-los em um formato visual, traçando tendências.

Embora as atividades e subatividades de preparação de dados acima tenham sido mostradas em uma ordem lógica, diferentes projetos podem reclassificá-las ou usar apenas um subconjunto delas. Algumas das etapas de preparação de dados, tais como a identificação da fonte de dados, são realizadas apenas uma vez e podem ser executadas manualmente. Outras etapas podem fazer parte do pipeline de dados operacionais e normalmente trabalhar com dados ao vivo. Estas tarefas devem ser automatizadas.

4.1.1 Desafios na preparação de dados

Alguns dos desafios relacionados à preparação dos dados incluem:

- A necessidade de conhecimento:
  - do domínio de aplicação;
  - dos dados e suas propriedades;
  - das várias técnicas associadas à preparação dos dados.
- A dificuldade de obter dados de alta qualidade de múltiplas fontes;
- A dificuldade de automatizar o pipeline de dados e assegurar que o pipeline de dados de produção seja escalável e tenha eficiência razoável de performance (p. ex.: o tempo necessário para completar o processamento de um item de dados);
- Os custos associados à preparação dos dados;
- Não dar prioridade suficiente à verificação de defeitos introduzidos no pipeline de dados durante a preparação dos dados;
- A introdução ao viés de amostragem (ver capítulo 2.4).

4.1.2 Exercício prático

Preparação de dados para ML

Para um determinado conjunto de dados brutos, executar as etapas de preparação de dados aplicáveis, conforme descrito no capítulo 4.1, para produzir um conjunto de dados que será usado para criar um modelo de classificação usando o aprendizado supervisionado.

Esta atividade forma o primeiro passo na criação de um modelo ML que será usado para exercícios futuros.

Para realizar esta atividade, os estudantes receberão materiais apropriados (e específicos de cada idioma), inclusive:

- *Bibliotecas;*
- *Estruturas ML;*
- *Ferramentas;*

- *Um ambiente de desenvolvimento.*

4.2 Conjunto de dados de treinamento, validação e teste no fluxo de trabalho ML

Logicamente, três conjuntos de dados equivalentes (ou seja, selecionados aleatoriamente a partir de um único conjunto inicial de dados) são necessários para desenvolver um modelo ML:

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

Se houver dados adequados ilimitados disponíveis, a quantidade de dados usados no fluxo de trabalho ML para treinamento, avaliação e testes normalmente depende dos seguintes fatores:

- O algoritmo usado para treinar o modelo;
- A disponibilidade de recursos, tais como RAM, espaço em disco, potência de computação, largura de banda da rede e o tempo disponível;

Na prática, devido ao desafio de adquirir dados adequados suficientes, os conjuntos de dados de treinamento e validação são frequentemente derivados de um único conjunto de dados combinados. O conjunto de dados de teste é mantido separado e não é usado durante o treinamento. Isto é para garantir que o modelo desenvolvido não seja influenciado pelos dados do teste, e para que os resultados do teste deem um reflexo real da qualidade do modelo.

Não há uma relação ideal para dividir o conjunto de dados combinados nos três conjuntos de dados individuais, mas as relações típicas que podem ser usadas como diretriz (treinamento: validação: teste) variam de 60:20:20 a 80:10:10. A divisão dos dados nestes conjuntos de dados é muitas vezes feita de forma aleatória, a menos que o conjunto de dados seja pequeno ou se houver o risco dos conjuntos de dados resultantes não serem representativos dos dados operacionais esperados.

Se houver dados limitados disponíveis, então a divisão destes dados em três conjuntos de dados pode resultar em dados insuficientes para um treinamento eficaz. Para superar este problema, os conjuntos de dados de treinamento e validação podem ser combinados (mantendo o conjunto de dados de teste separado), e então usados para criar múltiplas combinações de divisão deste conjunto de dados (p. ex.: 80% de treinamento / 20% de validação). Os dados são então atribuídos aleatoriamente aos conjuntos de dados de treinamento e validação. O treinamento, a validação e o ajuste são executados usando estas combinações múltiplas de divisão para criar vários modelos ajustados, e a performance geral do modelo pode ser calculada como a média de todas as execuções. Há vários métodos usados para criar múltiplas combinações de partição, que incluem teste de partição, *bootstrap*, validação cruzada *K-fold* e validação cruzada *leave-one-out* (ver [B02] para mais detalhes).

4.2.1 Exercício prático

Identificar dados de treinamento e teste e criar um modelo ML

- *Dividir os dados previamente preparados (ver capítulo 4.1.2) em conjuntos de dados de treinamento, validação e teste;*
- *Treinar e testar um modelo de classificação utilizando o aprendizado supervisionado com estes conjuntos de dados;*
- *Explicar a diferença entre avaliação/ajuste e teste comparando a precisão obtida com os conjuntos de dados de validação e teste.*

4.3 Questões de qualidade do conjunto de dados

As questões típicas de qualidade relacionadas aos dados em um conjunto de dados incluem, mas não estão limitadas àquelas mostradas na tabela a seguir:

Aspectos de Qualidade	Desrição
Dados errados	Os dados capturados estavam incorretos (p. ex.: através de um sensor defeituoso) ou inseridos incorretamente (p. ex.: erros de copiar-colar).
Dados incompletos	Os valores dos dados podem faltar (p. ex.: um campo em um registro pode estar vazio, ou os dados de um determinado intervalo de tempo podem ter sido omitidos). Pode haver várias razões para dados incompletos, incluindo problemas de segurança, problemas de hardware e erro humano.
Dados mal rotulados	Há várias razões possíveis para que os dados sejam mal rotulados (ver capítulo 4.5.2).
Dados insuficientes	Os dados disponíveis são insuficientes para que os padrões sejam reconhecidos pelos algoritmos de aprendizagem em uso (note que a quantidade mínima requerida de dados vai variar para diferentes algoritmos).
Dados não pré-processados	Os dados devem ser pré-processados para garantir que estejam limpos, em um formato consistente e não contenham anomalias indesejáveis (ver capítulo 4.1).
Dados obsoletos	Os dados utilizados tanto para o aprendizado quanto para a previsão devem ser os mais atuais possíveis (p. ex.: o uso de dados financeiros de vários anos atrás pode muito bem levar a resultados imprecisos).
Dados desbalanceados	Dados desbalanceados podem resultar de viés inadequado (p. ex.: baseados em raça, gênero ou etnia), má colocação dos sensores (p. ex.: câmeras de reconhecimento facial colocadas na altura do teto), variabilidade na disponibilidade dos conjuntos de dados e diferentes motivações dos fornecedores de dados.
Dados desleais	A equidade é uma característica de qualidade subjetiva, mas muitas vezes pode ser identificada. Por exemplo, para apoiar a diversidade ou o equilíbrio de gênero, dados selecionados podem ser positivamente tendenciosos para minorias ou grupos desfavorecidos (observe que tais dados podem ser considerados justos, mas podem não ser equilibrados).
Dados duplicados	Os registros de dados repetidos podem influenciar indevidamente o modelo ML resultante.
Dados irrelevantes	Dados que não são relevantes para o problema que está sendo tratado podem influenciar negativamente os resultados e podem levar ao desperdício de recursos.
Questões de privacidade	Qualquer uso de dados deve respeitar as leis de privacidade de dados relevantes (p. ex.: GDPR com relação às informações pessoais de indivíduos na União Europeia).
Questões de segurança	Dados fraudulentos ou enganosos que tenham sido deliberadamente inseridos nos dados de treinamento podem levar a imprecisões no modelo treinado.

4.4 Qualidade dos dados e seu efeito sobre o modelo ML

A qualidade do modelo ML é altamente dependente da qualidade do conjunto de dados a partir do qual ele é criado. Dados de má qualidade podem resultar tanto em modelos defeituosos quanto em previsões errôneas.

As seguintes categorias de defeitos resultam de problemas de qualidade dos dados:

- **Precisão reduzida**: Estes defeitos são causados por dados errados, incompletos, mal rotulados, insuficientes, obsoletos, irrelevantes, e dados que não são pré-processados. Por exemplo, se os dados fossem usados para construir um modelo de preços esperados de casas, mas os dados de

treinamento contivessem poucos ou nenhuns dados sobre casas isoladas com estufas, então os preços previstos para este tipo específico de casa provavelmente seriam imprecisos.

- **Modelo tendencioso**: Estes defeitos são causados por dados incompletos, desbalanceados, desleais, sem diversidade, ou duplicados. Por exemplo, se os dados de uma determinada característica estiverem faltando (p. ex.: todos os dados médicos para previsão de doenças são coletados de indivíduos do sexo feminino), então é provável que isso tenha um efeito adverso sobre o modelo resultante (a menos que o modelo seja usado apenas para fazer previsões operacionais sobre este sexo).
- **Modelo comprometido**: Estes defeitos são devidos à privacidade dos dados e às restrições de segurança. Por exemplo, problemas de privacidade nos dados podem levar a vulnerabilidades de segurança, o que permitiria aos atacantes reverter informações dos modelos e poderia posteriormente causar vazamento de informações pessoais.

4.5 Rotulagem de dados para aprendizagem supervisionada

A rotulagem de dados é o enriquecimento de dados não rotulados (ou mal rotulados) pela adição de rótulos, de modo que se torna adequada para uso em aprendizagem supervisionada. A rotulagem de dados é uma atividade de uso intensivo de recursos que tem sido relatada para usar, em média, 25% do tempo em projetos ML [B11].

Em sua forma mais simples, a rotulagem de dados pode consistir na colocação de imagens ou arquivos de texto em várias pastas, com base no aprendizado. Por exemplo, colocar todos os arquivos de texto de análises positivas de produtos em uma pasta e todas as de análises negativas em outra pasta. Identificar objetos em imagens desenhando retângulos ao seu redor é outra técnica comum de rotulagem, muitas vezes conhecida como anotação. Anotações mais complexas poderiam ser necessárias para rotular objetos 3D ou para desenhar caixas delimitadoras em torno de objetos irregulares. A rotulagem de dados e a anotação são tipicamente suportadas por ferramentas.

4.5.1 Abordagens de rotulagem de dados

A rotulagem pode ser realizada de várias maneiras:

- **Interna**: A rotulagem é realizada por desenvolvedores, testadores ou uma equipe dentro da organização que é criada para a rotulagem.
- **Terceirizada**: A rotulagem é feita por uma organização externa especializada.
- **Por multidão**: A rotulagem é realizada por um grande grupo de indivíduos. Devido à dificuldade de gerenciar a qualidade da rotulagem, vários anotadores podem ser solicitados a rotular os mesmos dados e uma decisão então tomada sobre o rótulo a ser usada.
- **AI assistido**: Ferramentas baseadas em AI são usadas para reconhecer e anotar dados ou para agrupar dados similares. Os resultados são então confirmados ou talvez complementados (p. ex.: modificando a caixa de delimitação) por um humano, como parte de um processo em duas etapas.
- **Híbrido**: Uma combinação das abordagens de rotulagem acima poderia ser usada. Por exemplo, a rotulagem por multidão é normalmente gerenciada por uma organização externa que tem acesso a ferramentas especializadas de gerenciamento de multidões baseadas em AI.

Quando aplicável, pode ser possível reutilizar um conjunto de dados pré-rotulados, evitando assim a necessidade de rotulagem completa de dados. Muitos desses conjuntos de dados estão disponíveis publicamente, por exemplo, na Kaggle [R16].

4.5.2 Dados mal rotulados em conjuntos de dados

O aprendizado supervisionado pressupõe que os dados sejam corretamente rotulados pelos anotadores de dados. Entretanto, é raro na prática que todos os itens de um conjunto de dados sejam rotulados corretamente. Os dados são rotulados incorretamente pelas seguintes razões:

- Erros aleatórios podem ser cometidos por anotadores (p. ex.: ao pressionar um botão errado);
- Podem ser cometidos erros sistêmicos (p. ex.: quando os rotuladores recebem instruções erradas ou treinamento deficiente);
- Erros deliberados podem ser cometidos por rotuladores de dados maliciosos;
- Os erros de tradução podem tomar dados corretamente rotulados em um idioma e incorretamente em outro;
- Quando a escolha é aberta à interpretação, julgamentos subjetivos feitos por rotuladores de dados podem levar a rótulos de dados conflitantes de diferentes anotadores;
- A falta do conhecimento de domínio exigido pode levar a uma rotulagem incorreta;
- Tarefas de classificação complexas podem resultar em mais erros;
- As ferramentas usadas para suportar a rotulagem de dados têm defeitos que levam a rótulos incorretos;
- As abordagens baseadas em ML para rotulagem são probabilísticas, e isto pode levar à alguns rótulos incorretos.

5 Machine Learning (ML), métricas de desempenho funcional [120 min]

Palavras-Chave

Nenhuma

Palavras-chave específicas da AI

acurácia, área sob a curva (AUC), coeficiente de perfil, conjuntos de referência ML, curva da característica operacional do receptor (ROC), erro quadrático médio (MSE), F1-score, matriz de confusão, métrica de performance funcional ML, métrica inter-agrupamentos, métrica intra-agrupamentos, modelo de regressão, precisão, *recall*, R-quadrado.

Objetivos de aprendizagem

5.1 Matriz de confusão

AI-5.1.1 K3: Calcular a métrica de performance funcional ML a partir de um dado conjunto de dados da matriz de confusão.

5.2 Métricas adicionais de performance funcional ML para classificação, regressão e agrupamento

AI-5.2.1 K2: Contrastar e comparar os conceitos por trás da métrica de performance funcional ML para métodos de classificação, regressão e agrupamento.

5.3 Limitações da métrica de performance funcional do ML

AI-5.3.1 K2: Resumir as limitações do uso da métrica de performance funcional ML para determinar a qualidade do sistema ML.

5.4 Seleção da métrica de performance funcional do ML

AI-5.4.1 K4: Selecionar as métricas apropriadas de performance funcional ML e/ou seus valores para um determinado modelo e cenário ML.

HO-5.4.1 H2: Avaliar o modelo ML criado usando métricas de performance funcional selecionadas do ML

5.5 Conjuntos de referência para ML

AI-5.5.1 K2: Explicar o uso de conjuntos de referência no contexto do ML

5.1 Matriz de confusão

Em um problema de classificação, um modelo raramente preverá os resultados corretamente o tempo todo. Para qualquer problema desse tipo, uma matriz de confusão pode ser criada com as seguintes possibilidades:

		REAL	
		Positivo	**Negativo**
PRE-VISTO	**Positivo**	**Verdadeiro Positivo (VP)**	**Falso Positivo (FP)**
	Negativo	**Falso Negativo (FN)**	**Verdadeiro Negativo (VN)**

Figura 2: Matriz de Confusão

Observe que a matriz de confusão mostrada na Figura 2 pode ser apresentada de forma diferente, mas sempre gerará valores para as quatro situações possíveis de verdadeiro positivo (VP), verdadeiro negativo (VN), falso positivo (FP) e falso negativo (FN).

Com base na matriz de confusão, são definidas as seguintes métricas:

Acurácia

$$\text{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN} \times 100\%$$

A exatidão mede a porcentagem de todas as classificações corretas.

Precisão

$$\text{Precisão} = \frac{VP}{VP + FP} \times 100\%$$

A precisão mede a proporção de pontos positivos que foram previstos corretamente. É uma medida de como se pode ter certeza sobre as previsões positivas.

Recall

$$\text{Recall} = \frac{VP}{VP + FN} \times 100\%$$

Recall (também conhecido como sensibilidade) mede a proporção de positivos reais que foram previstos corretamente. É uma medida de quão certo se pode estar de não perder nenhum positivo.

F1-score

$$\text{F1-score} = \frac{2 \times (\text{Precisão} \times \text{Recall})}{\text{Precisão} + \text{Recall}}$$

F1-score é computada como o meio harmônico de precisão e recall. Ela terá um valor entre zero e um. Uma pontuação próxima a um indica que dados falsos têm pouca influência sobre o resultado. Uma baixa F1-score sugere que o modelo é pobre em detectar os positivos.

5.2 Métricas adicionais de performance funcional ML para classificação, regressão e agrupamento

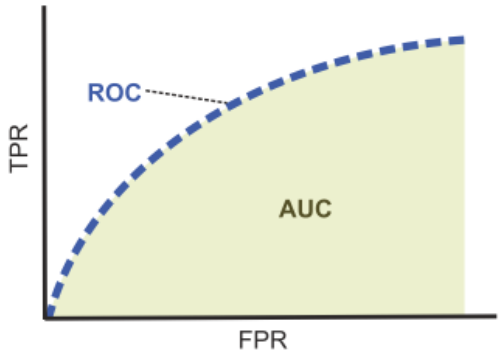
Há numerosas métricas para diferentes tipos de problemas ML (além daquelas relacionadas à classificação descrita no capítulo 5.1). Algumas das métricas mais comumente usadas são:

Métricas de classificação supervisionadas

- A *curva característica operacional do receptor* (ROC) é um gráfico que ilustra a capacidade de um classificador binário, uma vez que seu limiar de discriminação é variado. O método foi originalmente desenvolvido para radares militares, por isso ter esse nome. A curva ROC é traçada com *taxa positiva verdadeira* (TPR) (também conhecida como recall) contra a *taxa positiva falsa* (FPR), com TPR no eixo y e FPR no eixo x.

$$FPR = \frac{FP}{FP + VN}$$

- A *área sob a curva* (AUC) é a área da região inferior da curva ROC. Ela representa o grau de separação de um classificador, mostrando como o modelo distingue bem as classes. Com uma AUC superior, as previsões do modelo são melhores.



Métricas de regressão supervisionadas

Para modelos de regressão supervisionada, as métricas representam o quão bem a linha de regressão se encaixa nos pontos de dados reais.

- O *erro quadrático médio* (MSE) é a média do quadrado das diferenças entre o valor real e o valor previsto. O valor do MSE é sempre positivo, e um valor mais próximo de zero sugere um melhor modelo de regressão. O quadrado da diferença garante que os erros positivos e negativos não se anulem.
- O *R-quadrado* (também conhecido como coeficiente de determinação) é uma medida de quão bem o modelo de regressão se encaixa nas variáveis dependentes.

Métricas de agrupamento não supervisionado

Para agrupamento não supervisionado, há várias métricas que representam as distâncias entre os vários agrupamentos e a proximidade dos pontos de dados dentro de um determinado agrupamento.

- As *métricas de intra-agrupamentos* medem a similaridade dos pontos de dados dentro de um agrupamento;
- As *métricas de inter-agrupamentos* medem a semelhança dos pontos de dados em diferentes agrupamentos;
- O *coeficiente de perfil* (também conhecido como pontuação de perfil) é uma medida (entre -1 e +1) baseada na média das distâncias de inter-agrupamentos e intra-agrupamentos. Uma pontuação de

+1 significa que os agrupamentos estão bem separados, uma pontuação de zero implica um agrupamento aleatório, e uma pontuação de -1 significa que os agrupamentos estão erroneamente atribuídos.

5.3 Limitações das métricas de performance funcional ML

As métricas de performance funcional ML limitam-se a medir a funcionalidade do modelo, por exemplo, em termos de acurácia, precisão, recall, MSE, AUC e o coeficiente de perfil. Eles não medem outras características de qualidade não funcionais, como as definidas na ISO 25010 [S06] (p. ex.: eficiência de performance) e as descritas no capítulo 2, (p. ex.: explicabilidade, flexibilidade e autonomia). Neste syllabus, o termo "métricas de performance funcional ML" é usado devido ao uso generalizado do termo "métricas de performance" para se referir a essas métricas funcionais. Adicionar "funcional ML" destaca-se que essas métricas são específicas para Machine Learning e não têm relação com as métricas de eficiência de performance.

As métricas de performance funcional ML são condicionadas por vários outros fatores:

- Para aprendizagem supervisionada, a métrica de performance funcional ML é calculada com base nos dados rotulados, e a precisão da métrica resultante depende da rotulagem correta (ver capítulo 4.5).
- Os dados usados para medição podem não ser representativos (p. ex.: podem ser tendenciosos) e as métricas de performance funcional ML geradas dependem desses dados (ver capítulo 2.4).
- O sistema pode compreender vários componentes, mas a métrica de performance funcional ML só se aplica ao modelo ML. Por exemplo, o pipeline de dados não é considerado pela métrica de performance funcional do ML para avaliar o modelo.
- A maioria das métricas de performance funcional ML só pode ser medida com o apoio de ferramentas.

5.4 Seleção da métrica de performance funcional ML

Normalmente não é possível construir um modelo ML que atinja a pontuação mais alta para todas as métricas de performance funcional ML geradas a partir de uma matriz de confusão. Em vez disso, as métricas mais apropriadas de performance funcional do ML são selecionadas como critérios de aceite, com base no uso esperado do modelo (p. ex.: para minimizar falsos positivos, é necessário um alto valor de precisão, enquanto para minimizar falsos negativos, a métrica de recall deve ser alta). Os seguintes critérios podem ser usados ao selecionar a métrica de performance funcional ML descrita nos capítulos 5.1 e 5.2:

- **Acurácia**: Essa métrica provavelmente será aplicável se os conjuntos de dados forem simétricos (p. ex.: contagens e custos de falsos positivos e falsos negativos similares). Essa métrica torna-se uma má escolha se uma classe de dados dominar sobre as outras, caso em que a F1-score deve ser considerada.
- **Precisão**: Essa pode ser uma métrica adequada quando o custo dos falsos positivos é alto e a confiança nos resultados positivos precisa ser alta. Um filtro de spam, (onde classificar um e-mail como spam é considerado positivo), é um exemplo onde é necessária alta precisão, pois colocar demasiados e-mails na pasta de spam que na verdade não são spam não será aceitável para a maioria dos usuários. Quando o classificador lida com situações em que uma porcentagem muito grande de casos é positiva, então é improvável que o uso apenas da precisão seja uma boa escolha.
- **Recall**: Quando é crítico que os positivos não devem ser perdidos, então é importante uma alta pontuação de recall. Por exemplo, a falta de quaisquer resultados positivos verdadeiros na detecção do câncer e sua marcação como negativa (ou seja, nenhum câncer detectado) é provavelmente inaceitável.

- **F1-score**: F1-score é mais útil quando há um desequilíbrio nas classes esperadas e quando a precisão e o recall são de importância semelhante.

Além das métricas acima, várias métricas são descritas no capítulo 5.2. Estes podem ser aplicáveis para determinados problemas ML, por exemplo:

- A AUC para a curva ROC pode ser usada para problemas de classificação supervisionada;
- MSE e R-quadrado podem ser usados para problemas de regressão supervisionada;
- As métricas de inter-agrupamentos, intra-agrupamentos e o coeficiente de perfil podem ser usados para problemas de agrupamento não supervisionados.

5.4.1 Exercício prático

Avaliar o modelo ML criado

Usando o modelo de classificação treinado no exercício anterior, calcular e exibir os valores de acurácia, precisão, recall e F1-score. Quando aplicável, utilize as funções da biblioteca fornecida por sua estrutura de desenvolvimento para realizar os cálculos.

5.5 Conjuntos de referência para ML

Novas tecnologias de AI como novos conjuntos de dados, algoritmos, modelos e hardware são lançados regularmente, e pode ser difícil determinar a eficácia relativa de cada nova tecnologia.

Para fornecer comparações objetivas entre estas diferentes tecnologias, estão disponíveis conjuntos de referência ML padronizados pela indústria. Estes cobrem uma ampla gama de áreas de aplicação e fornecem ferramentas para avaliar plataformas de hardware, estruturas de software e plataformas de nuvem para performance de AI e ML.

Os conjuntos de referência ML podem fornecer várias medidas, incluindo tempos de treinamento (p. ex.: quão rápido uma estrutura pode treinar um modelo ML usando um conjunto de dados de treinamento definido para uma métrica de qualidade alvo especificada, com precisão de 75%), e tempos de inferência (p. ex.: quão rápido um modelo ML treinado pode realizar inferências).

Os conjuntos de referência ML são fornecidos por várias organizações diferentes, como por exemplo:

- **MLCommons** [R18]: Esta é uma organização sem fins lucrativos formada em 2020 e anteriormente denominada *ML Perf*, que fornece referências para estruturas de software, processadores específicos para AI e plataformas de nuvem ML.
- **DAWNBench** [R19]: Este é um conjunto de referência do ML da Universidade de Stanford.
- **MLMark** [R20]: Este é um conjunto de referência ML projetado para medir a performance e a precisão da inferência embutida do *Embedded Microprocessor Benchmark Consortium*.

6 Redes neurais e testes [65 min]

Palavras-chave

Nenhuma

Palavras-chave específicas da AI

aprendizagem supervisionada, cobertura de limites, cobertura de mudança de sinal, cobertura de mudança de valor, cobertura de neurônio, cobertura de sinal, dados de treinamento ML, dados de treinamento, perceptron multicamadas, perceptron, rede neural profunda (DNN), rede neural, valor de ativação.

Objetivos de aprendizagem

6.1 Redes neurais

AI-6.1.1 K2: Explicar a estrutura e a função de uma rede neural incluindo um DNN.

HO-6.1.1 H1: Experimentar a implementação de um perceptron.

6.2 Medidas de cobertura para redes neurais

AI-6.2.1 K2: Descrever as diferentes medidas de cobertura para redes neurais.

6.1 Redes Neurais

As redes neurais artificiais foram inicialmente destinadas a imitar o funcionamento do cérebro humano, que pode ser considerado em uma quantidade de neurônios biológicos conectados. O *perceptron* de camada única é um dos primeiros exemplos da implementação de uma rede neural artificial e compreende uma rede neural com apenas uma camada (ou seja, um único neurônio). Ele pode ser usado para o aprendizado supervisionado de classificadores, que decidem se uma entrada pertence ou não a uma classe específica.

A maioria das redes neurais atuais são consideradas redes neurais profundas porque compreendem várias camadas e podem ser consideradas como *perceptrons* multicamadas (ver Figura 3).

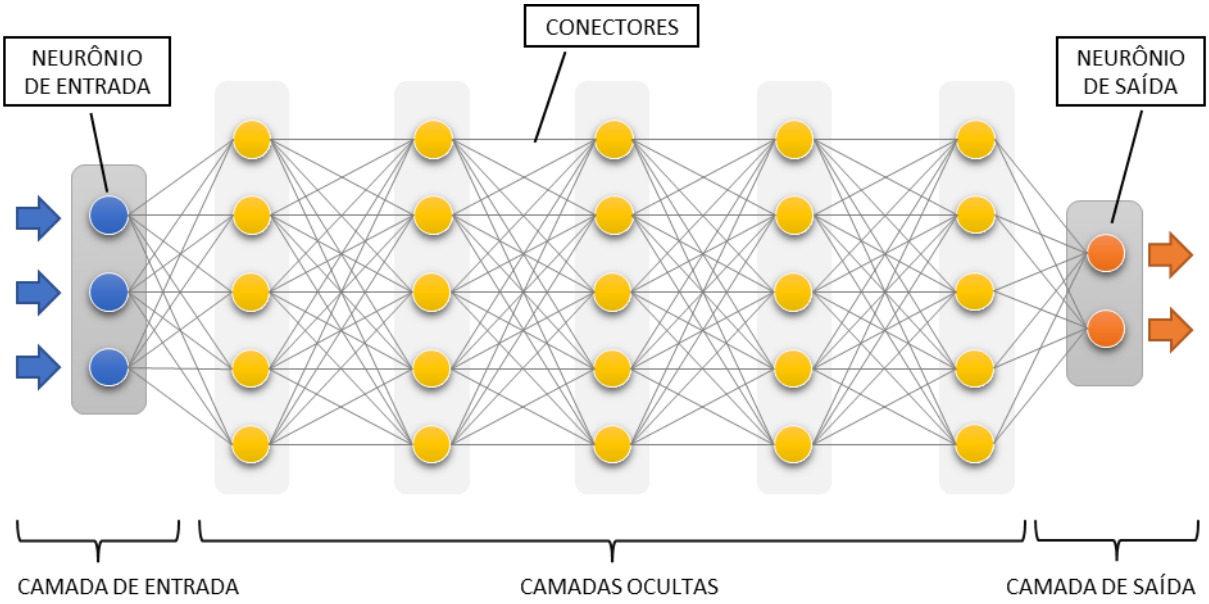


Figura 3 Estrutura de uma rede neural profunda

Uma rede neural profunda compreende três tipos de camadas. A camada de entrada recebe as entradas, por exemplo, valores de pixel de uma câmera. A camada de saída fornece resultados para o mundo exterior. Isto pode ser, por exemplo, um valor que significa a probabilidade de que a imagem de entrada seja um gato. Entre as camadas de entrada e saída estão camadas ocultas compostas de neurônios artificiais, que também são conhecidos como **nós**. Os neurônios de uma camada estão conectados a cada um dos neurônios da camada seguinte e pode haver números diferentes de neurônios em cada camada sucessiva. Os neurônios realizam cálculos e passam informações através da rede desde os neurônios de entrada até os neurônios de saída.

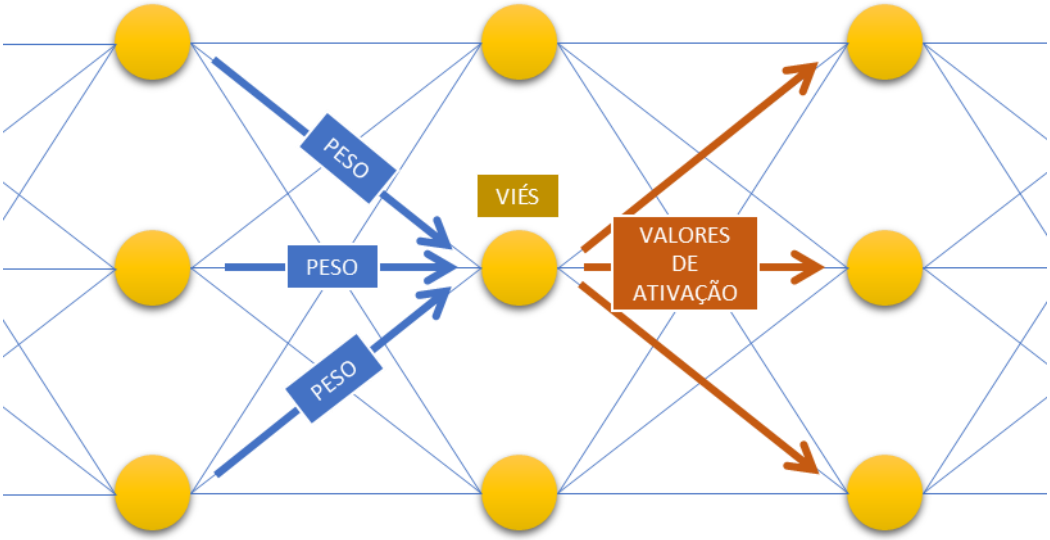


Figura 4 Computação realizada por cada neurônio

Como mostrado na Figura 4, o cálculo realizado por cada neurônio (exceto aqueles na camada de entrada) gera o que é conhecido como o valor de ativação. Este valor é calculado executando uma fórmula (a função de ativação) que recebe como entrada os valores de ativação de todos os neurônios da camada anterior, os pesos atribuídos às conexões entre os neurônios (estes pesos mudam conforme a rede aprende), e o viés individual de cada neurônio. Note que este viés é um valor constante predefinido e não está relacionado ao viés considerado anteriormente no capítulo 2.4). A execução de diferentes funções de ativação pode resultar em diferentes valores de ativação sendo calculados. Estes valores são normalmente centrados em torno de zero e têm uma faixa entre -1 (significando que o neurônio é "desinteressante") e +1 (significando que o neurônio é "muito interessante").

Ao treinar a rede neural, cada neurônio é predefinido com um valor de polarização e os dados de treinamento são passados através da rede, com cada neurônio executando a função de ativação, para eventualmente gerar uma saída. A saída gerada é então comparada com o resultado correto conhecido (neste exemplo de aprendizagem supervisionada são usados dados rotulados). A diferença entre a saída real e o resultado correto conhecido é então alimentada de volta através da rede para modificar os valores dos pesos nas conexões entre os neurônios a fim de minimizar esta diferença. À medida que mais dados de treinamento são alimentados através da rede, os pesos são gradualmente ajustados conforme a rede aprende. Em última análise, as saídas produzidas são consideradas boas o suficiente para terminar o treinamento.

6.1.1 Exercício prático

Implementar um perceptron simples

Os estudantes serão conduzidos através de um exercício demonstrando um aprendizado perceptron uma função simples, tal como uma função AND.

O exercício deve cobrir como um perceptron aprende modificando os pesos ao longo de várias épocas até que o erro seja zero. Vários mecanismos podem ser usados para esta atividade (p. ex.: planilha eletrônica, simulação).

6.2 Medidas de cobertura para redes neurais

Atingir critérios de cobertura de teste caixa-branca (p. ex.: instruções, desvio, cobertura de decisão e condição modificada (MC/DC) [I01] é obrigatório para o cumprimento de algumas normas relacionadas à segurança [S07] ao utilizar o código-fonte imperativo tradicional, sendo recomendado por muitos profissionais

de teste para outras aplicações críticas. O monitoramento e a melhoria da cobertura apoiam o projeto de novos casos de teste, levando a um aumento da confiança no objeto de teste.

O uso de tais medidas para medir a cobertura das redes neurais fornece pouco valor, pois o mesmo código tende a ser executado cada vez que a rede neural é executada. Em vez disso, medidas de cobertura foram propostas com base na cobertura da própria estrutura da rede neural e, mais especificamente, dos neurônios dentro dela. A maioria dessas medidas é baseada nos valores de ativação dos neurônios.

A cobertura das redes neurais é uma nova área de pesquisa. Os trabalhos acadêmicos só foram publicados a partir de 2017 e, como tal, há poucas evidências objetivas disponíveis (p. ex.: resultados de pesquisa duplicados) que mostram que as medidas propostas são eficazes. Deve-se notar, entretanto, que apesar da cobertura de instruções e decisões ter sido utilizada por mais de 50 anos, também há poucas evidências objetivas de sua relativa eficácia, embora tenham sido demandadas para medir a cobertura de software em aplicações relacionadas à segurança, tais como dispositivos médicos e sistemas aeronáuticos.

Os seguintes critérios de cobertura para redes neurais foram propostos e aplicados por pesquisadores a uma variedade de aplicações:

- **Cobertura de neurônios**: A cobertura total do neurônio exige que cada neurônio da rede neural atinja um valor de ativação maior que zero [B12]. Isto é muito fácil de alcançar na prática e as pesquisas mostraram que quase 100% de cobertura é alcançada com muitos poucos casos de teste em uma variedade de redes neurais profundas. Esta medida de cobertura pode ser mais útil como um sinal de alarme quando não é atingida.
- **Cobertura de limite**: A cobertura total do limite exige que cada neurônio da rede neural atinja um valor de ativação maior do que um limite específico. Os pesquisadores que criaram a estrutura *DeepXplore* realmente sugerem que a cobertura neural deve ser medida com base no valor de ativação que exceda um limite que mudaria com base na situação. Eles realizaram suas pesquisas com um limite de 0,75 quando relataram encontrar de forma eficiente milhares de comportamentos incorretos de casos no limite, utilizando esta abordagem caixa-branca. Este tipo de cobertura foi renomeado aqui para distingui-lo mais facilmente da cobertura de neurônios com um limite definido para zero, já que alguns outros pesquisadores usam o termo "cobertura de neurônios" para representar a cobertura de neurônios com um limite de zero.
- **Cobertura de mudança de sinal:** Para alcançar uma cobertura completa de mudança de sinal, os casos de teste precisam fazer com que cada neurônio alcance tanto valores de ativação positivos quanto negativos [B13].
- **Cobertura de mudança de valor**: Para alcançar uma cobertura completa de mudança de valor, os casos de teste precisam fazer com que cada neurônio alcance dois valores de ativação, onde a diferença entre os dois valores excede algum valor escolhido [B13].
- **Cobertura de sinal**: Esta cobertura considera os pares de neurônios em camadas adjacentes e o sinal tomado por seus valores de ativação. Para que um par de neurônios seja considerado coberto, um caso de teste precisa mostrar que mudar o sinal de um neurônio na primeira camada faz com que o neurônio na segunda camada mude seu sinal, enquanto os sinais de todos os outros neurônios na segunda camada permanecem inalterados [B13]. Este é um conceito semelhante à cobertura MC/DC para código fonte imperativo.

Pesquisadores relataram outras medidas de cobertura baseadas em camadas (embora mais simples que a cobertura por sinais), e uma abordagem bem-sucedida usando algoritmos limites mais próximos para identificar mudanças significativas em conjuntos de neurônios vizinhos foi implementada na ferramenta *TensorFuzz* [B14].

7 Visão geral dos sistemas baseados em AI [115 min]

Palavras-chave

teste de dados de entrada, teste do modelo ML

Palavras-chave específicas da AI

Big data, componente de AI, dados de treinamento, desvio de conceito, métricas de performance funcional ML, pipeline de dados, viés de automação.

Objetivos de aprendizado

7.1 Especificação de sistemas baseados em AI

AI-7.1.1 K2: Explicar como as especificações de sistema para sistemas baseados em AI podem criar desafios nos testes.

7.2 Níveis de teste para sistemas baseados em AI

AI-7.2.1 K2: Descrever como os sistemas baseados em AI são testados em cada nível de teste

7.3 Dados de teste para testar sistemas baseados em AI

AI-7.3.1 K1: Relembrar os fatores associados aos dados de teste que podem dificultar o teste de sistemas baseados em AI.

7.4 Teste de viés de automação em sistemas baseados em AI

AI-7.4.1 K2: Explicar o viés de automação e como isso afeta os testes.

7.5 Documentando um componente de AI

AI-7.5.1 K2: Descrever a documentação de um componente de AI e entender como a documentação suporta os testes de sistemas baseados em AI.

7.6 Testes de desvio de conceito

AI-7.6.1 K2: Explicar a necessidade de testar frequentemente o modelo treinado para lidar com o desvio do conceito.

7.7 Selecionando uma abordagem de teste para um sistema ML

AI-7.7.1 K4: Para um determinado cenário, determinar uma abordagem de teste a ser seguida ao desenvolver um sistema ML.

7.1 Especificação de sistemas baseados em AI

Os requisitos de sistema e as especificações de projeto são igualmente importantes tanto para sistemas baseados em AI quanto para sistemas convencionais. Estas especificações fornecem a base para os testadores verificarem se o comportamento real do sistema se alinha com os requisitos especificados. Entretanto, se as especificações estiverem incompletas e não forem testadas, isto introduz um problema de oráculo de teste (ver capítulo 8.7).

Há várias razões pelas quais a especificação de sistemas baseados em AI pode ser particularmente desafiadora:

- Em muitos projetos de sistemas baseados em AI, os requisitos são especificados apenas em termos de metas comerciais de alto nível e previsões necessárias. Uma razão para isso é a natureza exploratória do desenvolvimento de sistemas baseados em AI. Muitas vezes, os projetos de sistemas baseados em AI começam com um conjunto de dados, e o objetivo é determinar quais previsões podem ser obtidas a partir desses dados. Isto está em contraste com a especificação da lógica necessária desde o início de um projeto convencional.
- A acurácia do sistema baseado em AI é frequentemente desconhecida até que os resultados dos testes independentes estejam disponíveis. Junto com a abordagem de desenvolvimento exploratório, isto frequentemente leva a especificações inadequadas, pois a implementação já está em andamento quando os critérios de aceite desejados são determinados.
- A natureza probabilística de muitos sistemas baseados em AI pode tornar necessário especificar tolerâncias para alguns dos requisitos de qualidade esperados, tais como a acurácia das previsões.
- Quando os objetivos do sistema exigem a replicação do comportamento humano, em vez de fornecer funcionalidades específicas, isso muitas vezes leva a requisitos de comportamento mal especificados com base no fato do sistema ser tão bom quanto, ou melhor que as atividades humanas que pretende substituir. Isto pode dificultar a definição de um oráculo de teste, especialmente quando os humanos que ele está substituindo variam muito em suas capacidades.
- Quando a AI é usada para implementar interfaces de usuário, como por exemplo pelo reconhecimento da linguagem natural, visão computadorizada ou interação física com humanos, os sistemas precisam demonstrar maior flexibilidade. Entretanto, tal flexibilidade também pode criar desafios na identificação e documentação de todas as diferentes formas em que tais interações podem acontecer.
- As características específicas da qualidade dos sistemas baseados em AI, tais como adaptabilidade, flexibilidade, evolução e autonomia, precisam ser consideradas e definidas como parte da especificação de requisitos (ver capítulo 2). A novidade dessas características pode torná-las difíceis de definir e testar.

7.2 Níveis de teste para sistemas baseados em AI

Os sistemas baseados em AI normalmente compreendem componentes AI e não AI. Componentes não AI podem ser testados usando abordagens convencionais [I01], enquanto componentes e sistemas AI precisam ser testados de forma diferente em alguns aspectos, como descrito abaixo. Para todos os níveis de teste que incluem o teste de componentes de AI, é importante que o teste seja apoiado de perto por engenheiros, cientistas de dados e especialistas de domínio.

Uma grande diferença dos níveis de teste usados para software convencional é a inclusão de dois novos níveis de teste especializados para lidar explicitamente com o teste dos dados de entrada e os modelos usados em sistemas baseados em AI [B15]. A maior parte deste capítulo é aplicável a todos os sistemas baseados em AI, embora algumas partes sejam especificamente focadas em ML.

7.2.1 Teste de dados de entrada

O objetivo do teste dos dados de entrada é garantir que os dados utilizados pelo sistema para treinamento e previsão sejam da mais alta qualidade (ver capítulo 4.3). Isso inclui o seguinte:

- Revisões;
- Técnicas estatísticas (p. ex.: dados de teste para enviesamento);
- EDA (*Exploratory Data Analysis*) dos dados de treinamento;
- Testes estáticos e dinâmicos do pipeline de dados.

O pipeline de dados normalmente compreende vários componentes que realizam a preparação de dados (ver capítulo 4.1), e os testes destes componentes incluem tanto testes de componentes como testes de integração. O pipeline de dados para treinamento pode ser bem diferente do pipeline de dados usado para apoiar a previsão operacional. Para treinamento, o pipeline de dados pode ser considerado um protótipo, comparado com a versão totalmente projetada e automatizada utilizada operacionalmente. Por este motivo, os testes destas duas versões do pipeline de dados podem ser bastante distintos. Entretanto, o teste da equivalência funcional das duas versões também deve ser considerado.

7.2.2 Teste do modelo ML

O objetivo do teste do modelo ML é garantir que o modelo selecionado atenda a qualquer critério de performance que possa ter sido especificado. Isto inclui:

- Critérios de performance funcional ML (ver capítulos 5.1 e 5.2);
- Critérios de aceite não funcional ML que são apropriados para o modelo isoladamente, tais como velocidade de treinamento, velocidade de previsão, recursos computacionais utilizados, adaptabilidade e transparência.

O teste do modelo ML também visa determinar que a escolha da estrutura ML, algoritmo, modelo, configurações do modelo e hiper parâmetros esteja o mais próximo possível do ótimo. Quando apropriado, o teste do modelo ML também pode incluir testes para alcançar critérios de cobertura caixa-branca (ver capítulo 6.2). O modelo selecionado é posteriormente integrado com outros componentes, AI e não-AI.

7.2.3 Teste de componentes

O teste de componentes é um nível de teste convencional que é aplicável a qualquer componente que não seja um modelo, como interfaces de usuário e componentes de comunicação.

7.2.4 Teste de integração de componentes

O teste de integração de componentes é um nível de teste convencional que é conduzido para garantir que os componentes do sistema (tanto AI como não AI) interajam corretamente. Ele testa se as entradas do pipeline de dados são recebidas conforme esperado pelo modelo, e que quaisquer previsões produzidas pelo modelo são trocadas com os componentes relevantes do sistema (p. ex.: a interface do usuário) e usadas corretamente por eles. Quando a AI é fornecida como um serviço (ver capítulo 1.7), é normal realizar testes API do serviço fornecido como parte dos testes de integração de componentes.

7.2.5 Testes de sistema

O teste do sistema é um nível de teste convencional que é conduzido para assegurar que o sistema completo de componentes integrados (tanto AI como não AI) funcione como esperado, tanto do ponto de vista funcional como não funcional, em um ambiente de teste que seja estreitamente representativo do ambiente operacional. Dependendo do sistema, este teste pode tomar a forma de testes de campo no ambiente operacional esperado ou testes dentro de um simulador (p. ex.: se os cenários de teste forem perigosos ou difíceis de replicar em um ambiente operacional).

Durante os testes do sistema, os critérios de desempenho funcional ML são testados novamente para garantir que os resultados dos testes iniciais do modelo ML não foram afetados negativamente quando o modelo foi incorporado dentro de um sistema completo. Este teste é especialmente importante quando o componente AI foi deliberadamente alterado (p. ex.: comprimindo um DNN para reduzir seu tamanho).

O teste do sistema é também o nível de teste no qual muitos dos requisitos não-funcionais do sistema são testados. Por exemplo, testes contraditórios podem ser realizados para testar a robustez, e o sistema pode ser testado quanto a ser explicável. Quando apropriado, as interfaces com componentes de hardware (p. ex.: sensores) podem ser testadas como parte do teste do sistema.

7.2.6 Teste de aceite

O teste de aceite é um nível de teste convencional e é usado para determinar se o sistema completo é aceitável para o cliente. Para sistemas baseados em AI, a definição de critérios de aceite pode ser desafiadora (ver capítulo 8.8). Onde a AI é fornecida como um serviço (ver capítulo 1.7), o teste de aceite pode ser necessário para determinar a adequação do serviço ao sistema pretendido e se, por exemplo, os critérios de performance funcional do ML foram suficientemente alcançados.

7.3 Dados de teste para testar sistemas baseados em AI

Dependendo da situação e do sistema em teste (SUT), a aquisição de dados de teste pode representar um desafio. Há vários desafios potenciais ao lidar com dados de teste para sistemas baseados em AI, inclusive:

- Big data (alto volume de dados, alta velocidade e alta variedade) podem ser difíceis de criar e gerenciar. Por exemplo, pode ser difícil criar dados de teste representativos para um sistema que consome grandes volumes de imagens e áudio a uma alta velocidade.
- Dados de entrada podem precisar mudar com o tempo, particularmente se representarem eventos no mundo real. Por exemplo, fotografias registradas para testar um sistema de reconhecimento facial podem precisar ser "envelhecidas" para representar o envelhecimento de pessoas ao longo de vários anos na vida real.
- Dados pessoais ou de outra forma confidenciais podem precisar de técnicas especiais de descaracterização, criptografia ou serem reescritos. A aprovação legal para uso também pode ser necessária.
- Quando os testadores usam a mesma implementação que os cientistas de dados para a aquisição e pré-processamento de dados, então os defeitos nestas etapas podem ser mascarados.

7.4 Teste de viés de automação em sistemas baseados em AI

Uma categoria de sistemas baseados em AI ajuda o ser humano na tomada de decisões. No entanto, ocasionalmente há uma tendência dos humanos em confiar demais nesses sistemas. Essa confiança equivocada pode ser chamada de viés de automação ou viés de complacência, e assume duas formas.

- A primeira forma de viés de automação ou complacência é quando o ser humano aceita as recomendações fornecidas pelo sistema e não considera as entradas de outras fontes (incluindo elas próprias). Por exemplo, um procedimento onde os dados chaves humanos em um formulário podem ser melhorados usando a aprendizagem da máquina para pré-popular e validar o formulário com os dados preenchidos. Foi demonstrado que esta forma de viés de automação normalmente reduz a qualidade das decisões tomadas em 5%, mas isto poderia ser muito maior, dependendo do contexto do sistema [B16]. Da mesma forma, a correção automática do texto digitado (p. ex.: em mensagens de telefone celular) é frequentemente defeituosa e pode mudar o significado. Os usuários muitas vezes não percebem isso e não anulam o erro.
- A segunda forma de viés de automação ou complacência é onde o ser humano perde uma falha no sistema por não monitorar adequadamente o sistema. Por exemplo, os veículos semiautônomos

estão se tornando cada vez mais autônomos, mas ainda dependem de um humano para assumir o controle no caso de um acidente iminente. Normalmente, o ocupante do veículo humano torna-se gradualmente confiante demais nas habilidades do sistema para controlar o veículo e eles começam a prestar menos atenção. Isto pode levar a uma situação em que eles são incapazes de reagir adequadamente quando necessário.

Em ambos os cenários, os testadores devem entender como a tomada de decisão humana pode ser comprometida, e testar tanto a qualidade das recomendações do sistema quanto a qualidade da contribuição humana correspondente fornecida por usuários representativos.

7.5 Documentando um componente de AI

O conteúdo típico para a documentação de um componente de AI inclui:

- **Geral**: Identificadores, descrição, detalhes do desenvolvedor, requisitos de hardware, detalhes da licença, versão, data e ponto de contato;
- **Projeto**: Premissas e decisões técnicas;
- **Utilização**: Casos de uso primário e secundário, usuários típicos, abordagem de autoaprendizado, vieses conhecidos, questões éticas, questões de segurança, transparência, limites de decisão, desvio de plataforma e conceito.
- **Conjunto de dados**: Características, coleta, disponibilidade, requisitos de pré-processamento, uso, conteúdo, rotulagem, tamanho, privacidade, segurança, parcialidade / imparcialidade, e restrições / obrigações.
- **Testes**: Conjunto de dados de teste (descrição e disponibilidade), independência de testes, resultados de testes, abordagem de testes para robustez, explicabilidade, desvio de conceitos e portabilidade.
- **Treinamento e performance funcional do ML**: Algoritmo ML, pesos, conjunto de dados de validação, seleção da métrica de performance funcional ML, limites para a métrica de performance funcional ML, e métrica de performance funcional real ML.

Documentação clara ajuda a melhorar os testes, proporcionando transparência na implementação do sistema baseado em AI. As áreas-chave da documentação que são importantes para os testes são:

- A finalidade do sistema e a especificação dos requisitos funcionais e não funcionais. Esses tipos de documentação normalmente fazem parte da base de teste;
- Informações estruturais e de projeto, descrevendo como os diferentes componentes de AI e não-AI interagem. Isto suporta a identificação dos objetivos dos testes de integração, e pode fornecer uma base para os testes caixa-branca da estrutura do sistema;
- A especificação do ambiente operacional. Isto é necessário ao testar a autonomia, flexibilidade e adaptabilidade do sistema;
- A fonte de quaisquer dados de entrada, incluindo metadados associados. Isto precisa ser claramente compreendido ao testar os seguintes aspectos:
    - Correção funcional de insumos que não são confiáveis;
    - Propensão explícita ou implícita da amostra;
    - Flexibilidade, incluindo a aprendizagem errada a partir de dados mal-informados para sistemas de autoaprendizagem.
- A forma pela qual se espera que o sistema se adapte às mudanças em seu ambiente operacional. Isto é necessário como base de teste ao testar a adaptabilidade;

- Detalhes dos usuários esperados do sistema. Isto é necessário para garantir que os testes possam ser representativos.

7.6 Teste de desvio de conceito

O ambiente operacional pode mudar com o tempo sem que o modelo treinado mude de forma correspondente. Este fenômeno é conhecido como *desvio de conceito*, normalmente faz com que os resultados do modelo se tornem cada vez menos precisos e menos úteis. Por exemplo, o impacto das campanhas de marketing pode resultar em uma mudança no comportamento dos clientes potenciais ao longo do tempo. Tais mudanças podem ser sazonais ou abruptas devido a mudanças culturais, morais ou sociais que são externas ao sistema. Um exemplo de tal mudança abrupta é o impacto de uma pandemia viral (p.ex.: Covid19) e seu efeito na precisão dos modelos usados para projeções de vendas e mercados de ações.

Os sistemas que podem ser propensos ao desvio de conceito devem ser testados regularmente de acordo com seus critérios acordados de performance funcional ML, para assegurar que quaisquer ocorrências de desvio de conceito sejam detectadas breve o suficiente para que o problema seja mitigado. As mitigações típicas podem incluir a retirada do sistema ou o aperfeiçoamento do sistema. No caso de retreinamento, isto seria realizado com dados de treinamento atualizados e seguido por testes de confirmação, testes de regressão e possivelmente uma forma de teste A/B (ver capítulo 9.4), onde o sistema B atualizado deve superar o sistema A original.

7.7 Selecionando uma abordagem de teste para um sistema ML

Um sistema baseado em AI normalmente incluirá tanto componentes AI como não AI. A abordagem de teste é baseada em uma análise de risco para tal sistema e incluirá tanto testes convencionais quanto testes mais especializados para abordar os fatores específicos dos componentes de AI e sistemas baseados em AI.

A lista a seguir fornece alguns riscos típicos e mitigações correspondentes, específicos aos sistemas ML. Note que esta lista fornece apenas um conjunto limitado de exemplos e que há muito mais riscos específicos aos sistemas ML que requerem mitigação através de testes.

Aspecto de risco	Descrição e possíveis mitigações
A qualidade dos dados pode ser inferior ao esperado.	Este risco tornar-se um problema de várias maneiras, cada uma das quais pode ser evitada de maneiras diferentes (ver capítulo 4.4). As atenuações comuns incluem o uso de revisões, EDA e teste dinâmico.
O pipeline de dados operacionais pode conter defeitos.	Este risco pode ser parcialmente mitigado pelo teste dinâmico dos componentes individuais do pipeline e pelo teste de integração de toda a pipeline.
O fluxo de trabalho ML usado para desenvolver o modelo pode estar abaixo do ideal. (ver capítulo 3.2)	Este risco pode ser devido a: • Falta de um acordo prévio a ser seguido sobre o fluxo de trabalho ML; • Uma má escolha de fluxo de trabalho; • Engenheiros de dados que não acompanham o fluxo de trabalho. Revisões com especialistas podem mitigar a chance de escolher o fluxo de trabalho errado, enquanto mais gerenciamento prático ou auditorias poderiam abordar os problemas de acordo e implementação do fluxo de trabalho.
A escolha da estrutura ML, algoritmo, modelo, configurações do modelo e/ou hiper parâmetros podem ser abaixo do esperado.	Este risco pode ser devido à falta de experiência dos tomadores de decisão, ou à implementação deficiente das etapas de avaliação e ajuste (ou etapa de teste) do fluxo de trabalho ML. Revisões com especialistas podem mitigar a chance de tomar decisões erradas, e uma melhor gestão pode assegurar que as etapas de avaliação e ajuste (e teste) do fluxo de trabalho sejam seguidas.

Aspecto de risco	Descrição e possíveis mitigações
O critério de performance funcional ML desejado pode não ser entregue operacionalmente, apesar do componente ML atender a esses critérios isoladamente.	Este risco pode ser devido ao conjunto de dados utilizados para treinamento e teste do modelo não serem representativos aos dados encontrados operacionalmente. Revisões dos conjuntos de dados selecionados por especialistas (ou usuários) podem reduzir a chance de que os dados selecionados não sejam representativos.
Os critérios desejados de performance funcional ML são atendidos, mas os usuários podem ficar insatisfeitos com os resultados entregues.	Este risco pode ser devido à seleção errada de critérios de performance (p. ex.: elevado recall foi selecionado quando foi necessária a precisão alta). Revisões com especialistas podem mitigar a chance de escolher a métrica errada de performance funcional ML, ou testes baseados na experiência também podem identificar critérios inadequados. O risco também pode ser devido ao desvio do conceito, caso em que testes mais frequentes do sistema operacional poderiam mitigar o risco.
Os critérios desejados de performance funcional ML são atendidos, mas os usuários podem ficar insatisfeitos com o serviço entregue.	Este risco pode ser devido a uma falta de foco nos requisitos não funcionais do sistema). Observe que a gama de características de qualidade para sistemas baseados em AI se estende além daquelas listadas na ISO/IEC 25010 (ver capítulo 2). O uso de uma abordagem baseada em risco para priorizar as características de qualidade e realizar os testes não-funcionais relevantes poderia mitigar este risco. Alternativamente, o problema poderia ser devido a uma combinação de fatores que poderiam ser identificados através de testes baseados na experiência, como parte dos testes de sistemas. O capítulo 8 fornece orientações sobre como testar essas características.
O sistema de autoaprendizagem pode não prestar o serviço esperado pelos usuários.	Este risco pode ser devido a várias razões, por exemplo: • Os dados utilizados pelo sistema para autoaprendizagem podem ser inadequados. Neste caso, as revisões feitas por especialistas poderiam identificar os dados problemáticos. • O sistema pode falhar devido à nova funcionalidade de autoaprendizagem ser inaceitável. Isto poderia ser mitigado por testes de regressão automatizados, incluindo comparação de performance com a funcionalidade anterior. • O sistema pode aprender de uma maneira que não é esperada pelos usuários, o que poderia ser detectado através de testes baseados em experiência.
Os usuários podem ficar frustrados por não entenderem como o sistema determina suas decisões.	Este risco pode ser devido à falta de explicabilidade, interpretabilidade ou transparência. O capítulo 8.6 detalha sobre como testar estas características.
Os usuários podem achar que o modelo fornece excelentes previsões quando os dados são semelhantes aos dados de treinamento, mas fornece maus resultados de outra forma.	Este risco pode ser devido ao *overfitting* (ver capítulo 3.5.1), que pode ser detectado ao testar o modelo com um conjunto de dados que seja completamente independente do conjunto de dados de treinamento ou ao realizar testes baseados na experiência.

8 Teste das características específicas de qualidade da AI [150 min]

Palavras-chave

Oráculo de teste

Palavras-chave específicas da AI

autonomia, dados de treinamento ML, explicável, interpretável, método LIME, sistema autônomo, sistema de autoaprendizagem, sistema especializado, sistema não-determinístico, sistema probabilístico, transparência, viés algorítmico, viés de amostragem, viés inadequado.

Objetivos de aprendizagem

8.1 Desafios para testar sistemas de autoaprendizagem

AI-8.1.1 K2: Explicar os desafios nos testes criados pela autoaprendizagem dos sistemas baseados em AI.

8.2 Teste de sistemas autônomos baseados em AI

AI-8.2.1 K2: Descrever como os sistemas autônomos baseados em AI são testados.

8.3 Testes de algoritmo, viés de amostragem e viés inadequado

AI-8.3.1 K2: Explicar como testar o viés em um sistema baseado em AI.

8.4 Desafios para testar sistemas baseados em AI probabilísticos e não determinísticos

AI-8.4.1 K2: Explique os desafios nos testes criados pela natureza probabilística e não-determinista dos sistemas baseados em AI.

8.5 Desafios nos testes de sistemas complexos baseados em AI

AI-8.5.1 K2: Explicar os desafios nos testes criados pela complexidade dos sistemas baseados em AI.

8.6 Teste da transparência, interpretabilidade e explicabilidade

AI-8.6.1 K2: Descrever como a transparência, a interpretabilidade e a explicabilidade dos sistemas baseados em AI podem ser testadas.

HO-8.6.1 H2: Usar uma ferramenta para mostrar como a explicabilidade pode ser usada pelos testadores.

8.7 Oráculos de teste para sistemas baseados em AI

AI-8.7.1 K2: Explicar os desafios na criação de oráculos de teste resultantes das características específicas dos sistemas baseados em AI.

8.8 Objetivos do teste e critérios de aceite

AI-8.8.1 K4: Selecionar os objetivos apropriados de teste e critérios de aceite para as características específicas de qualidade de um determinado sistema baseado em AI.

8.1 Desafios para testar sistemas de autoaprendizagem

Há vários desafios potenciais a serem superados ao testar sistemas de autoaprendizagem (ver capítulo 2 para mais detalhes sobre estes sistemas), inclusive:

- **Mudança inesperada**: Os requisitos e restrições originais dentro das quais o sistema deve funcionar são geralmente conhecidos, mas pode haver pouca ou nenhuma informação disponível sobre as mudanças feitas pelo próprio sistema. Normalmente é possível testar contra os requisitos e o projeto original (e quaisquer restrições especificadas), mas se o sistema tiver concebido uma implementação inovadora ou formulou uma solução (cuja implementação não pode ser vista), pode ser difícil conceber testes que sejam apropriados para esta nova implementação. Além disso, quando os

sistemas se modificam (e seus resultados), os resultados de testes anteriores podem mudar. Este é um desafio de projeto de teste. Pode ser resolvido projetando testes apropriados que permanecem relevantes à medida que o sistema muda seu comportamento, evitando assim um problema potencial nos testes de regressão. Entretanto, também pode exigir que novos testes sejam projetados com base em novos comportamentos observados no sistema.

- **Critérios complexos de aceite**: Pode ser necessário definir expectativas de melhoria no sistema enquanto ele aprende. Por exemplo, pode ser assumido que se o sistema mudar por si só, sua performance funcional geral deve melhorar. Além disso, especificar qualquer outra coisa além da simples "melhoria" pode se tornar rapidamente complexa. Por exemplo, uma melhoria mínima pode ser esperada (ao invés de simplesmente qualquer melhoria), ou a melhoria requerida estar ligada a fatores ambientais (p. ex.: uma melhoria mínima de 10% na funcionalidade X é requerida se o fator ambiental F mudar em mais de Y). Estes problemas podem ser resolvidos através da especificação e testes em relação aos critérios de aceite mais complexos, e mantendo um registro contínuo da performance funcional básico do sistema atual.
- **Tempo de teste insuficiente**: Pode ser necessário saber com que rapidez se espera que o sistema aprenda e se adapte, dados os diferentes cenários. Estes critérios de aceite podem ser difíceis de especificar e atingir. Se um sistema se adapta rapidamente, pode não haver tempo suficiente para executar manualmente novos testes após cada mudança, então pode ser necessário escrever testes que possam ser executados automaticamente quando o sistema mudar por si mesmo. Estes desafios podem ser enfrentados através da especificação de critérios de aceite apropriados (ver capítulo 8.8) e testes contínuos automatizados.
- **Requisitos de recursos**: Os requisitos do sistema podem incluir critérios de aceite dos recursos que o sistema pode utilizar ao realizar a autoaprendizagem ou adaptação. Isto pode incluir, por exemplo, a quantidade de tempo de processamento e memória permitida a ser utilizada para melhorar. Além disso, deve-se considerar se o uso deste recurso deve ser ligado a uma melhoria mensurável na funcionalidade ou precisão. Este desafio afeta a especificação dos critérios de aceite.
- **Especificações insuficientes do ambiente operacional**: Um sistema de autoaprendizagem pode mudar se os insumos ambientais que recebe estiverem fora dos limites esperados, ou se não forem refletidos nos dados de treinamento. Essas entradas podem ser ataques na forma de envenenamento de dados (ver capítulo 9.1.2). Pode ser difícil prever toda a gama de ambientes operacionais e mudanças ambientais e, portanto, identificar o conjunto completo de casos de teste representativos e requisitos ambientais. Preferencialmente, o escopo completo de possíveis mudanças no ambiente operacional ao qual se espera que o sistema responda será definido como critério de aceite.
- **Ambiente de teste complexo**: Gerenciar o ambiente de teste para garantir que ele possa imitar todas as mudanças potenciais do ambiente operacional de alto risco é um desafio e pode envolver o uso de ferramentas de teste (p. ex.: uma ferramenta de injeção de falha). Dependendo da natureza do ambiente operacional, isto pode ser testado através da manipulação de entradas e sensores, ou obtendo acesso a diferentes ambientes físicos nos quais o sistema pode ser testado.
- **Modificações indesejáveis de comportamento**: Um sistema de autoaprendizagem modifica seu comportamento com base em suas entradas e pode ser impossível para os testadores evitar que isso ocorra. Isto pode surgir, por exemplo, se um sistema de terceiros estiver sendo usado, ou se o sistema de produção estiver sendo testado. Ao repetir os mesmos testes, um sistema de autoaprendizagem pode se tornar mais eficaz em responder a esses testes, o que pode então influenciar o comportamento a longo prazo do sistema. Portanto, é importante evitar uma situação em que os testes façam com que um sistema de autoaprendizagem mude negativamente seu comportamento. Este é um desafio para a concepção de casos de teste e gerenciamento de testes.

8.2 Teste de sistemas autônomos baseados em AI

Os sistemas autônomos devem ser capazes de determinar quando requerem ou não a intervenção humana. Portanto, testar a autonomia destes sistemas baseados em AI requer criar condições para que o sistema possa exercer a tomada de decisão.

Os testes autônomos exigem:

- Testar se o sistema solicita a intervenção humana para um cenário específico quando o sistema deixar o controle. Tais cenários poderiam incluir uma mudança no ambiente operacional, ou o sistema excedendo os limites de sua autonomia;
- Testar se o sistema solicita a intervenção humana quando o sistema renuncia ao controle após um período específico;
- Testar se o sistema solicita desnecessariamente a intervenção humana quando ainda deveria funcionar de forma autônoma.

Pode ser útil utilizar a análise de valor limite aplicada no ambiente operacional para gerar as condições necessárias para este teste. Pode ser um desafio definir como os parâmetros que determinam a autonomia se manifestam no ambiente operacional e criar os cenários de teste que dependem da natureza da autonomia.

8.3 Testes de algoritmo, viés de amostragem e viés inadequado

Um sistema ML deve ser avaliado em relação aos diferentes vieses e ações tomadas para remover vieses inadequados. Isto pode envolver a introdução deliberada de viés positivo para combater o viés inadequado.

Testes com um conjunto independente de dados pode muitas vezes detectar o viés. Entretanto, será difícil identificar todos os dados que causam viés porque o algoritmo ML poderá usar combinações de características aparentemente não relacionadas para criar um viés inadequado.

Os sistemas baseados em AI devem ser testados para detectar viés algorítmico, viés de amostragem e viés inadequado (ver capítulo 2.4). Isto pode envolver:

- Analisar, avaliar e ajustar o modelo para identificar se existe viés algorítmico, durante as atividades de treinamento;
- Revisar a fonte dos dados de treinamento e dos processos utilizados para adquiri-los, de modo que a presença de viés de amostragem seja identificada;
- Revisar o pré-processamento de dados como parte do fluxo de trabalho ML para identificar se os dados foram afetados de uma forma que poderia causar um viés de amostragem;
- Medir como as mudanças nas entradas do sistema afetam as saídas do sistema em muitas interações, e examinar os resultados com base nos grupos de pessoas ou objetos para os quais o sistema pode ser inadequadamente tendencioso, ou contra. Isto é semelhante ao método LIME (*Local Interpretable Model-Agnostic Explanations*) discutido no capítulo 8.6, e pode ser realizado em um ambiente de produção, ou como parte de testes antes da liberação;
- Obter informações adicionais sobre os atributos dos dados de entrada potencialmente relacionados ao viés e sua correlação com os resultados. Isto poderia estar relacionado a dados demográficos, por exemplo, que poderiam ser apropriados ao testar um viés inadequado que afeta grupos de pessoas, onde a afiliação a um grupo é relevante para avaliar o viés, mas não é uma entrada para o modelo. Isto porque o viés pode ser baseado em variáveis "ocultas" que não estão explicitamente presentes nos dados de entrada, mas são inferidas pelo algoritmo.

8.4 Desafios para testar sistemas baseados em AI probabilísticos e não determinísticos

A maioria dos sistemas probabilísticos também são não determinísticos e, portanto, a seguinte lista de desafios de testes normalmente se aplica a sistemas baseados em AI com qualquer um desses atributos:

- Pode haver múltiplos resultados válidos de um teste com o mesmo conjunto de pré-condições e insumos. Isto torna a definição dos resultados esperados mais desafiadora e pode causar dificuldades:
    - quando os testes são reutilizados para testes de confirmação;
    - quando os testes são reutilizados para testes de regressão;
    - onde reproduzir os testes é importante;
    - quando os testes são automatizados.
- O testador normalmente requer um conhecimento mais profundo do comportamento necessário do sistema para que ele possa chegar a verificações razoáveis, para determinar se o teste passou, em vez de simplesmente declarar um valor exato para o resultado esperado do teste. Por exemplo, os testadores podem precisar definir resultados esperados mais sofisticados em comparação com os sistemas convencionais. Estes resultados esperados do teste podem incluir tolerâncias (p. ex.: "o resultado real está dentro de 2% da solução ótima?");
- Quando uma única saída definitiva de um teste não é possível devido à natureza probabilística do sistema, muitas vezes é necessário que o testador execute um teste várias vezes a fim de gerar um resultado estatisticamente válido.

8.5 Desafios nos testes de sistemas complexos baseados em AI

Os sistemas baseados em AI são frequentemente usados para implementar tarefas que são complexas para que os seres humanos as executem. Isto pode levar a um problema de oráculo de teste porque os testadores são incapazes de determinar os resultados esperados como normalmente fariam (ver capítulo 8.7). Por exemplo, os sistemas baseados em AI são frequentemente usados para identificar padrões em grandes volumes de dados. Tais sistemas são usados porque são capazes de executar atividades em que os humanos, mesmo após muita análise, simplesmente não conseguem realizar com a mesma eficiência. Pode ser um desafio compreender o comportamento necessário de tais sistemas com profundidade suficiente para ser capaz de gerar os resultados esperados.

Um problema semelhante surge quando a estrutura interna de um sistema baseado em AI é gerada por software, tornando-o complexo demais para que os humanos o entendam. Isto leva à situação em que o sistema baseado em AI só pode ser testado como caixa-preta. Mesmo quando a estrutura interna é visível, isto não fornece nenhuma informação útil adicional para ajudar nos testes.

A complexidade dos sistemas baseados em AI aumenta quando eles fornecem resultados probabilísticos e de natureza não determinística (ver capítulo 8.4).

Os problemas com sistemas não determinísticos são exacerbados quando um sistema baseado em AI consiste em vários componentes interagindo, cada um fornecendo resultados probabilísticos. Por exemplo, um sistema de reconhecimento facial provavelmente usará um modelo para identificar um rosto dentro de uma imagem, e um segundo modelo para reconhecer qual rosto foi identificado. As interações entre os componentes de AI podem ser complexas e difíceis de compreender, tornando difícil identificar todos os riscos, e testes de projeto que verificam adequadamente o sistema.

8.6 Teste da transparência, interpretabilidade e explicabilidade

As informações sobre como o sistema foi implementado podem ser fornecidas por desenvolvedores. Isto inclui as fontes de dados de treinamento, como a rotulagem foi conduzida, e como os componentes do sistema foram projetados. Quando estas informações não estão disponíveis, o projeto dos testes torna-se um desafio. Por exemplo, se as informações de dados de treinamento não estiverem disponíveis, então a identificação de potenciais lacunas em tais dados e o teste do impacto de tais lacunas torna-se difícil. Esta situação pode ser comparada aos testes caixa-preta e caixa-branca, com vantagens e desvantagens similares. A transparência pode ser testada comparando as informações documentadas nos dados e algoritmo com a implementação real e determinando o grau de aproximação entre eles.

Com o ML, muitas vezes pode ser mais difícil explicar a ligação entre uma entrada específica e uma saída específica, do que em sistemas convencionais. Este baixo nível de explicabilidade é principalmente porque o modelo que gera a saída é gerado pelo próprio código (o algoritmo) e não reflete a maneira como os humanos pensam sobre um problema. Diferentes modelos ML oferecem diferentes níveis de explicabilidade e devem ser selecionados com base nos requisitos do sistema, que podem incluir explicabilidade e testabilidade.

Um método para entender a explicabilidade é através do teste dinâmico do modelo ML ao aplicar anomalias aos dados do teste. Existem métodos para quantificar a explicabilidade desta forma e para fornecer explicações visuais. Alguns destes métodos são agnósticos ao modelo, enquanto outros são específicos a um determinado tipo de modelo requerendo acesso a ele. Os testes exploratórios também podem ser usados para entender melhor a relação entre as entradas e saídas de um modelo.

O método LIME é um modelo agnóstico e utiliza anomalias de entrada e de saída injetadas dinamicamente para fornecer aos testadores uma visão da relação entre as entradas e as saídas. Este pode ser um método eficaz para fornecer explicações sobre modelo. Entretanto, ele se limita a fornecer possíveis razões para as saídas, ao invés de uma razão definitiva, e não é aplicável a todos os tipos de algoritmos.

A interpretável de um sistema baseado em AI depende muito de a quem isto se aplica. Diferentes stakeholders podem ter requisitos diferentes em termos de quão bem eles precisam captar a tecnologia subjacente.

Medir e testar o nível de compreensão tanto para a interpretabilidade quanto para a explicabilidade pode ser um desafio, pois os stakeholders podem não concordar por terem níveis de entendimento diferentes. Além disso, identificar o perfil típico de stakeholders pode ser difícil para muitos tipos de sistemas. Quando realizado, este teste comumente toma a forma de pesquisas e/ou questionários para os usuários.

8.6.1 Exercício prático

Explicabilidade do modelo

Use uma ferramenta apropriada para fornecer explicações com base no modelo previamente criado. Por exemplo, para um modelo de classificação de imagem ou um modelo de classificação de texto, um método de modelo agnóstico, como o LIME, pode ser apropriado.

Os estudantes devem usar a ferramenta para gerar explicações das decisões do modelo; em particular, como as características das entradas influenciam as saídas.

8.7 Oráculos de teste para sistemas baseados em AI

Um grande problema com os testes de sistemas baseados em AI pode ser a especificação dos resultados esperados. Um oráculo de teste é a fonte usada para determinar o resultado esperado de um teste [I01]. Um desafio na determinação dos resultados esperados é conhecido como o problema do oráculo de teste.

Com sistemas complexos, não determinísticos ou probabilísticos, pode ser difícil estabelecer um oráculo de teste sem conhecer o "terreno" (ou seja, o resultado no mundo real que o sistema baseado em AI está tentando prever). Esta necessidade é distinta de um oráculo de teste, na medida em que ele não pode

essencialmente fornecer um valor esperado, mas apenas um mecanismo para determinar se o sistema está operando corretamente ou não.

Sistemas baseados em AI podem evoluir (ver capítulo 2.3), e o teste de sistemas de autoaprendizagem (ver capítulo 8.1) também pode sofrer de problemas de oráculo de teste à medida que eles se modificam e podem, portanto, tornar necessário atualizar frequentemente as expectativas funcionais do sistema.

Uma outra causa de dificuldade na obtenção de um oráculo de teste eficaz é que em muitos casos, a correção do comportamento do software é subjetiva. Os assistentes virtuais (p. ex.: Siri e Alexa) são um exemplo deste problema, na medida em que diferentes usuários muitas vezes têm expectativas bastante diferentes e podem experimentar resultados diferentes dependendo de sua escolha de palavras e clareza de fala.

Em algumas situações, pode ser possível definir o resultado esperado com limites ou tolerâncias. Por exemplo, o ponto de parada de um carro autônomo poderia ser definido como dentro de uma distância máxima de um ponto específico. No contexto de sistemas especializados, a determinação dos resultados esperados pode ser obtida consultando um especialista (observando que a opinião do especialista ainda pode estar errada). Há vários fatores importantes a serem considerados em tais circunstâncias:

- Os especialistas humanos variam em seus níveis de competência. Os especialistas envolvidos precisam ser pelo menos tão competentes quanto os especialistas que o sistema se destina a substituir;
- Os especialistas podem não concordar entre si, mesmo quando apresentados com as mesmas informações;
- Os especialistas humanos não podem aprovar a automatização de seu julgamento. Nesses casos, sua classificação de potenciais resultados deve ser duplamente cega (ou seja, nem os especialistas nem os avaliadores dos resultados devem saber quais classificações foram automatizadas);
- Os seres humanos são mais propensos a censurar respostas (p. ex.: com frases como "Não tenho certeza, mas..."). Se este tipo de advertência não estiver disponível para o sistema baseado em AI, isto deve ser considerado ao comparar as respostas.

Existem técnicas de teste que podem reduzir o problema do oráculo de teste, tais como teste A/B (ver capítulo 9.4), teste consecutivo (ver capítulo 9.3) e teste metamórfico (ver capítulo 9.5).

8.8 Objetivos do teste e critérios de aceite

Os objetivos de teste e os critérios de aceite de um sistema precisam ser baseados nos riscos percebidos do produto.

Esses riscos podem muitas vezes ser identificados a partir de uma análise das características de qualidade exigidas. As características de qualidade para um sistema baseado em AI incluem aquelas tradicionalmente consideradas na ISO/IEC 25010 [S06] (isto é, adequação funcional, eficiência de performance, compatibilidade, usabilidade, confiabilidade, segurança, capacidade de manutenção e portabilidade), mas também devem incluir uma consideração dos seguintes aspectos:

Aspecto	Critérios de aceite
Adaptabilidade	• Verificar se o sistema ainda funciona corretamente e se atende aos requisitos não funcionais quando se adapta a uma mudança em seu ambiente. Isto pode ser implementado como uma forma de teste de regressão automatizado; • Verificar o tempo que o sistema leva para se adaptar a uma mudança em seu ambiente; • Verificar os recursos utilizados quando o sistema se adapta a uma mudança em seu ambiente.

Aspecto	Critérios de aceite
Flexibilidade	• Considerar como o sistema lida em contextos fora da especificação inicial. Isto pode ser implementado como uma forma de teste de regressão automatizado executado no ambiente operacional alterado. • Verificar o tempo que o sistema leva e/ou os recursos usados para mudar a si mesmo para administrar um novo contexto.
Evolução	• Verificar se o sistema aprende bem com sua própria experiência; • Verificar como o sistema lida bem quando o perfil dos dados muda (ou seja, desvio de conceitos).
Autonomia	• Verificar como o sistema responde quando é forçado fora do contexto operacional no qual se espera que seja totalmente autônomo; • Verificar se o sistema pode ser "persuadido" a solicitar a intervenção humana quando deve ser totalmente autônomo.
Transparência, interpretável e explicável	• Verificar a transparência revisando a facilidade de acesso ao algoritmo e ao conjunto de dados; • Verificar a interpretabilidade e a explicabilidade questionando os usuários do sistema ou, se os usuários reais do sistema não estiverem disponíveis, pessoas com um histórico semelhante.
Livre de viés inapropriado	• Quando os sistemas são susceptíveis de serem afetados por viés, então isto pode ser testado usando um conjunto de testes independentes sem viés, ou usando revisores especializados; • Comparar os resultados dos testes usando dados externos, tais como dados do censo, a fim de verificar se há um viés indesejado nas variáveis inferidas (teste de validade externa).
Ética	• Verificar o sistema em relação a uma lista de verificação adequada, como a *EC Assessment List for Trustworthy Artificial Intelligence* [R21], que apoia os principais requisitos delineados pelas *Ethics Guidelines for Trustworthy Artificial Intelligence* (AI) [R22].
Sistemas probabilísticos e sistemas não-determinísticos	• Isto não pode ser avaliado com critérios precisos de aceite. Quando funciona corretamente, o sistema pode retornar resultados ligeiramente diferentes para os mesmos testes.
Efeitos colaterais	• Identificar efeitos colaterais potencialmente nocivos e tentar gerar testes que façam com que o sistema exiba esses efeitos colaterais.
Hacking de recompensas	• Testes independentes podem identificar hacking de recompensa quando esses testes utilizam um meio diferente de medir o sucesso em comparação com o agente inteligente que está sendo testado.
Segurança	• Isto precisa ser cuidadosamente avaliado, talvez em um ambiente de teste virtual (ver capítulo 10.2). Isto poderia incluir tentativas de forçar um sistema a causar danos a si mesmo.

Para sistemas ML, a métrica de performance funcional requerida para o modelo ML deve ser especificada (ver capítulo 5).

9 Métodos e técnicas para o teste de sistemas baseados em AI [245 min]

Palavras-chave

problema de oráculo de teste, pseudo-oráculo, relação metamórfica (MR), roteiros, suposição de erro, teste A/B, teste consecutivo, teste baseado na experiência, teste contraditório, teste em pares, teste exploratório, teste metamórfico (MT)

Palavras-chave específicas da AI

ataque contraditório, envenenamento de dados, exemplo contraditório, modelo treinado, sistema ML.

Objetivos de aprendizagem

9.1 Ataques contraditórios e envenenamento de dados

AI-9.1.1 K2: Explicar como os testes dos sistemas ML podem ajudar a prevenir ataques adversos e envenenamento de dados.

9.2 Teste em pares

AI-9.2.1 K2: Explicar como os testes em pares são usados para sistemas baseados em AI.

LO-9.2.1 H2: Aplicar testes em pares para derivar e executar casos de teste para um sistema baseado em AI.

9.3 Teste consecutivo

AI-9.3.1 K2: Explicar como os testes consecutivos são usados para sistemas baseados em AI.

9.4 Teste A/B

AI-9.4.1 K2: Explicar como os testes A/B são aplicados aos testes de sistemas baseados em AI.

9.5 Teste metamórfico

AI-9.5.1 K3: Aplicar testes metamórficos para o teste de sistemas baseados em AI.

HO-9.5.1 H2: Aplicar testes metamórficos para derivar casos de teste para um determinado cenário e executá-los.

9.6 Teste baseado na experiência de sistemas baseados em AI

AI-9.6.1 K2: Explicar como os testes baseados em experiência podem ser aplicados aos testes de sistemas baseados em AI.
HO-9.6.1 H2: Aplicar testes exploratórios a um sistema baseado em AI.

9.7 Seleção de técnicas de teste para sistemas baseados em AI

AI-9.7.1 K4: Para um determinado cenário, selecionar as técnicas de teste apropriadas ao testar um sistema baseado em AI.

9.1 Ataques contraditórios e envenenamento de dados

9.1.1 Ataques contraditórios

Um ataque contraditório é onde um atacante sutilmente modifica entradas válidas que são passadas para o modelo treinado para fazer com que ele forneça previsões incorretas. Estas entradas alteradas, conhecidas como exemplos contraditórios, foram primeiramente notadas com filtros de spam, que poderiam ser enganados ao modificar ligeiramente um e-mail de spam sem perder a legibilidade. Recentemente, eles se tornaram mais associados aos classificadores de imagem. Simplesmente mudando alguns pixels que são invisíveis ao olho humano, é possível persuadir uma rede neural a mudar sua classificação de imagem para um objeto muito diferente e com alto grau de confiança.

Exemplos contraditórios são geralmente transferíveis [B17], o que significa que um exemplo contraditório que faz com que um sistema ML falhe muitas vezes, fará com que outro sistema ML que é treinado para realizar a mesma tarefa, também falhe. Mesmo quando o segundo sistema ML foi treinado com dados diferentes e é baseado em arquiteturas diferentes, ele ainda é frequentemente propenso a falhar com os mesmos exemplos de contradição.

Os ataques contraditórios caixa-branca são onde o atacante sabe qual algoritmo foi usado para treinar o modelo e quais configurações e parâmetros do modelo foram usados (há um nível razoável de transparência). O atacante usa este conhecimento para gerar exemplos de contradições, por exemplo, fazendo pequenas alterações nas entradas e monitorando quais provocam grandes mudanças nas saídas do modelo.

Os ataques contraditórios caixa-preta necessitam que o atacante explore o modelo para determinar sua funcionalidade, para depois construir um modelo duplicado que fornece funcionalidade similar. O atacante então usa uma abordagem caixa-branca para identificar exemplos contraditórios para este modelo duplicado. Como os exemplos contraditórios são geralmente transferíveis, os mesmos exemplos contraditórios normalmente também funcionarão no modelo original.

Se não for possível criar um modelo duplicado, poderá ser possível usar testes automatizados de alto volume para descobrir diferentes exemplos contraditórios e observar os resultados.

Os testes contraditórios envolvem simplesmente a realização de ataques contraditórios com o objetivo de identificar vulnerabilidades para que medidas preventivas possam ser tomadas para proteger contra falhas futuras. Os exemplos contraditórios identificados são adicionados aos dados de treinamento para que o modelo seja treinado para reconhecê-los corretamente.

9.1.2 Envenenamento de dados

Ataques de envenenamento de dados são onde um atacante manipula os dados de treinamento para alcançar um de dois resultados. O atacante pode inserir *backdoors* ou *trojans* de redes neurais para facilitar futuras intrusões, ou mais frequentemente, eles usarão dados de treinamento corrompidos (p. ex., dados mal rotulados) para induzir o modelo treinado a fornecer previsões incorretas.

Ataques de envenenamento podem ser direcionados com o objetivo de fazer com que o sistema ML se classifique erroneamente em situações específicas. Eles também podem ser indiscriminados, como por exemplo com um ataque de negação de serviço (DoS). Um conhecido exemplo de um ataque de envenenamento foi a corrupção do *chatbot* Microsoft Tay, em que um número relativamente pequeno de conversas prejudiciais no Twitter treinou o sistema através de *feedback* para fornecer conversas no futuro. Uma forma comum usada de ataque de envenenamento de dados usa a falsa denúncia de milhões de e-mails de spam como não sendo spam em uma tentativa de distorcer o software de filtragem de spam. Um ponto de preocupação com o envenenamento de dados é o potencial de envenenamento dos conjuntos de dados de AI amplamente utilizados pelo público.

Os testes para detectar envenenamento de dados são possíveis usando EDA, já que dados envenenados podem aparecer como anômalos. Além disso, as políticas de aquisição de dados podem ser revistas para garantir a proveniência dos dados de treinamento. Quando um sistema ML pode ser atacado pela alimentação de dados envenenados, os testes A/B (ver capítulo 9.4) podem ser usados para verificar se a versão atualizada do sistema ainda está alinhada com a versão anterior. Como alternativa, um teste de regressão em um sistema atualizado usando um conjunto de teste confiável também pode determinar se um sistema foi envenenado.

9.2 Teste em pares

O número de parâmetros de interesse para um sistema baseado em AI pode ser extremamente alto, especialmente quando o sistema utiliza *big data* ou interage com o mundo exterior, como um carro que se dirige sozinho. Testes exaustivos exigiriam todas as combinações possíveis destes parâmetros definidos para todos os valores possíveis a serem testados. Entretanto, como isto resultaria em um número praticamente infinito de testes, são usadas técnicas de teste para selecionar um subconjunto que pode ser executado no tempo disponível limitado.

Onde é possível combinar numerosos parâmetros, cada um dos quais pode ter muitos valores discretos, os testes combinatórios podem ser aplicados para reduzir significativamente o número necessário de casos de teste, preferencialmente sem comprometer a capacidade de detecção de defeitos do conjunto de testes. Existem várias técnicas de testes combinatórios (ver [I02] e [S08]). Entretanto, na prática, o teste em pares é a técnica mais utilizada porque é fácil de entender, e tem amplo suporte de ferramentas. Além disso, pesquisas demonstraram que a maioria dos defeitos é causada por interações envolvendo poucos parâmetros [B33].

Na prática, mesmo o uso de testes em pares pode resultar extensos conjuntos de testes para alguns sistemas, e o uso de automação e ambientes de testes virtuais (ver capítulo 10.2) muitas vezes se torna necessário para permitir o número necessário de testes a serem executados. Por exemplo, ao considerar veículos autônomos, os cenários de teste de alto nível para testes de sistemas precisam testar tanto os diferentes ambientes nos quais se espera que os carros operem quanto as várias funções do veículo. Assim, os parâmetros precisariam incluir uma gama de restrições ambientais (p. ex., tipos e superfícies das estradas, condições climáticas, de tráfego, e visibilidade) e as várias funções de auto condução (p. ex., controle adaptativo de velocidade de cruzeiro, assistência na permanência na faixa de rodagem e assistência na mudança de faixa de rodagem). Além desses parâmetros, as entradas dos sensores podem ser consideradas em níveis variáveis de eficácia (p. ex., as entradas de uma câmera de vídeo se degradarão à medida que a viagem avança e ela fica mais suja).

A pesquisa atualmente não está clara sobre o nível de rigor necessário que seria exigido para o uso de testes combinatórios com sistemas críticos em segurança baseados em AI, tais como os carros autônomos. Mesmo que os testes em pares sejam insuficientes, sabe-se que a abordagem é eficaz para encontrar defeitos.

9.2.1 Exercício prático

Teste em pares

Para um sistema baseado em AI implementado com um mínimo de cinco parâmetros e pelo menos quinhentas combinações possíveis, use uma ferramenta de teste em pares para identificar um conjunto reduzido de combinações em pares e executar testes para essas combinações. Compare o número de combinações testadas em pares com o número necessário se todas as combinações teoricamente possíveis fossem testadas.

9.3 Testes consecutivos (back-to-back)

Uma das soluções potenciais para o problema do oráculo de teste (ver capítulo 8.7) quando se testa sistemas baseados em AI é usar testes consecutivos. Isto também é conhecido como teste diferencial. Com o teste consecutivo, uma versão alternativa do sistema é usada como um pseudo-oráculo e seus resultados comparados com os resultados do teste produzido pelo SUT. O pseudo-oráculo pode ser um sistema existente, ou pode ser desenvolvido por uma equipe diferente, possivelmente em uma plataforma diferente, com uma arquitetura diferente e com uma linguagem de programação diferente. Ao testar a adequação funcional (em oposição aos requisitos não funcionais), o sistema usado como pseudo-oráculo não é obrigado a atingir os mesmos critérios de aceite não funcional que o SUT. Por exemplo, ele pode não ter que executar tão rapidamente, caso em que pode ser muito menos caro de construir.

No contexto do ML, é possível utilizar diferentes estruturas, algoritmos e configurações de modelos para criar um pseudo-oráculo ML. Em algumas situações, também pode ser possível criar uma pseudo-oráculo utilizando software convencional, não-AI.

Para que os pseudo-oráculos sejam eficazes na detecção de defeitos, não deve haver um software comum tanto no pseudo-oráculo quanto no SUT. Caso contrário, seria possível que o mesmo defeito em ambos fizesse com que os dois resultados dos testes coincidissem quando ambos estão defeituosos. Com tanto software de AI imaturo, reutilizável e de código aberto sendo usado para desenvolver sistemas baseados em AI, a reutilização do código entre o pseudo-oráculo e o SUT pode comprometer o pseudo-oráculo. A má documentação das soluções de AI reutilizáveis também pode dificultar o reconhecimento pelos testadores de que este problema está ocorrendo.

9.4 Teste A/B

O teste A/B é um método onde a resposta de duas variações do programa (A e B) às mesmas entradas, são comparadas com o objetivo de se determinar qual das duas é a melhor. É uma abordagem de teste estatístico que normalmente requer a comparação dos resultados de teste de várias execuções de teste para determinar a diferença entre os programas.

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

O teste A/B é uma abordagem para resolver o problema do oráculo de teste, onde o sistema existente é usado como um oráculo parcial. Os testes A/B não geram casos de teste e não fornecem nenhuma orientação sobre como os testes devem ser projetados, embora entradas operacionais sejam frequentemente usadas nos testes.

Os testes A/B podem ser usados para testar atualizações de um sistema baseado em AI onde existem critérios de aceite acordados, tais como métricas de performance funcional ML, conforme descrito no Capítulo 5. Sempre que o sistema é atualizado, os testes A/B são usados para verificar se a variante atualizada funciona tão bem quanto, ou melhor do que a variante anterior. Tal abordagem pode ser usada para um classificador simples, mas também pode ser usada para testar sistemas muito mais complexos. Por exemplo, uma atualização para melhorar a eficácia de um sistema inteligente de roteamento de transporte urbano também pode ser testada usando o teste A/B (p. ex., comparando o tempo médio de viagem para duas variantes do sistema em semanas consecutivas).

Os testes A/B também podem ser usados para testar os sistemas de autoaprendizagem. Quando o sistema faz uma mudança, testes automatizados são executados e as características resultantes do sistema são comparadas com aquelas antes da mudança ser feita. Se o sistema for melhorado, então a mudança é aceita, caso contrário o sistema volta ao seu estado anterior.

Uma grande diferença entre os testes A/B e os testes consecutivos diz respeito ao uso de testes A/B para comparar duas variantes do mesmo sistema e o uso de testes consecutivos para detectar defeitos.

9.5 Teste metamórfico (MT)

O teste metamórfico [B18] é uma técnica destinada a gerar casos de teste que se baseiam na fonte de um caso de teste que tenha passado. Um ou mais casos de teste de acompanhamento são gerados alterando (metamorfoseando) o caso de teste de origem baseado em uma relação metamórfica (MR). O MR é baseado em uma propriedade de uma função requerida do objeto de teste, de modo que descreve como uma mudança nas entradas de teste de um caso de teste se reflete nos resultados esperados do mesmo caso de teste.

Por exemplo, considere um programa que determina a média de um conjunto de números. Uma fonte de caso de teste é gerada compreendendo um conjunto de números e uma média esperada, e o caso de teste é executado para confirmar que ele passa. Agora é possível gerar casos de teste de acompanhamento com base no que é conhecido sobre a função média do programa. Inicialmente, a ordem dos números que estão sendo calculados como média pode ser simplesmente alterada. Dada a função média, o resultado esperado pode ser previsto para permanecer o mesmo. Assim, um caso de teste de acompanhamento com os números em uma ordem diferente pode ser gerado sem a necessidade de calcular o resultado esperado. Um grande conjunto de números, poderia levar à geração de diferentes conjuntos de números nos quais os mesmos números são usados em sequências diferentes e cada um deles poderia ser usado para criar um caso de teste de acompanhamento separado. Todos estes casos de teste seriam baseados no mesmo caso de teste de origem e teriam o mesmo resultado esperado.

É comum ter MRs e casos de teste de acompanhamento onde o resultado esperado é diferente do inicialmente esperado para o caso de teste original. Por exemplo, usando a mesma função média, um MR pode ser derivado no qual cada elemento do conjunto de entrada é multiplicado por dois. O resultado esperado para tal conjunto é simplesmente o resultado esperado original multiplicado por dois. Da mesma

forma, qualquer outro valor poderia ser usado como um multiplicador para potencialmente gerar um número infinito de casos de teste de acompanhamento com base neste MR.

O MT pode ser usado na maioria dos objetos de teste e pode ser aplicado a testes funcionais e não funcionais (p. ex., o teste de instalabilidade cobre diferentes alvos de configurações onde os parâmetros de instalação podem ser selecionados em sequências diferentes). É particularmente útil onde a geração dos resultados esperados é problemática, devido à falta de um oráculo de teste barato. Este é o caso de alguns sistemas baseados em AI que utilizam na análise de *big data*, ou aqueles onde os testadores não estão claros sobre como o algoritmo ML deriva suas previsões. Na área de AI, o MT tem sido usado para testar o reconhecimento de imagens, mecanismos de busca, otimização de rotas e reconhecimento de voz, entre outros.

Como explicado acima, o MT pode ser baseado em uma fonte de caso de teste aprovada, mas também é útil se não for possível verificar se qualquer fonte de caso de teste está correta. Este pode ser o caso, por exemplo, onde o programa implementa uma função que é muito complexa para um testador humano replicar e usar como um oráculo de teste, como com alguns sistemas baseados em AI. Nesta situação, o MT pode ser usado para gerar um ou mais casos de teste que, quando executado, criará um conjunto de saídas onde as relações entre as saídas podem então ser verificadas quanto à validade. Com esta forma de MT, os testes individuais não são conhecidos por serem corretos, mas as relações entre eles devem se manter verdadeiras, proporcionando assim maior confiança no programa. Um exemplo poderia ser um sistema baseado na AI que prevê a idade da morte de uma pessoa baseada em um grande conjunto de dados, onde se sabe, por exemplo, que se o número de cigarros fumados aumentar, a idade prevista da morte deverá diminuir (ou, pelo menos, permanecer a mesma).

O MT é uma técnica de teste relativamente nova, proposta pela primeira vez em 1998. Difere das técnicas de teste tradicionais porque os resultados esperados dos casos de teste de acompanhamento não são descritos em termos de valores absolutos, mas são relativos aos resultados esperados no caso do teste de origem. Ela se baseia em um conceito de fácil compreensão, sendo aplicada por testadores com pouca experiência na aplicação da técnica, mas que entendem o domínio de aplicação, e tem custos similares em comparação com as técnicas tradicionais. Também é eficaz para revelar defeitos, com pesquisas mostrando que apenas três a seis MRs diferentes podem revelar mais de 90% dos defeitos que poderiam ser detectados usando técnicas baseadas em um oráculo de teste tradicional [B19]. É possível gerar automaticamente casos de teste de acompanhamento a partir de MRs bem especificados e um caso de teste de fonte. Entretanto, ferramentas comerciais não estão disponíveis atualmente, embora o Google já esteja aplicando o MT automatizado para testar drivers gráficos Android usando a ferramenta *GraphicsFuzz*, que se tem mantido como *opensource* (veja [R23]).

9.5.1 Exercício prático

Testes metamórficos

Neste exercício, os estudantes ganharão experiência prática:

- *Com a derivação de várias relações metamórficas (MRs) para uma determinada aplicação ou programa baseado em AI. Estas MRs devem incluir alguns onde os resultados esperados da fonte e casos de teste de acompanhamento são os mesmos e alguns onde são diferentes;*
- *Com a geração de fontes de casos de teste para a aplicação ou programa baseado em AI. Estes não precisam ter garantia de aprovação, mas os estudantes devem ser lembrados das limitações da MT onde não existe tal "padrão de ouro" disponível;*
- *Usando as MRs derivadas e casos de teste de fonte gerada para derivar casos de teste de acompanhamento;*
- *Executando os casos de teste de acompanhamento.*

9.6 Teste baseado na experiência de sistemas baseados em AI

Os testes baseados na experiência incluem suposição de erros, testes exploratórios e testes baseados em checklist [I01], todos os quais podem ser aplicados aos testes de sistemas baseados em AI.

A suposição de erro é tipicamente baseada no conhecimento dos testadores, erros típicos dos desenvolvedores e falhas em sistemas similares (ou versões anteriores). Um exemplo de suposição de erro aplicado a sistemas baseados em AI poderia ser o uso do conhecimento sobre como os sistemas ML falharam no passado devido ao uso de dados de treinamento sistemicamente tendenciosos.

Em testes exploratórios, os testes são modelados, gerados e executados de forma iterativa, com a oportunidade de testes posteriores serem derivados, com base nos resultados dos testes anteriores. Os testes exploratórios são especialmente úteis quando há especificações ruins ou problemas de oráculo de teste, o que muitas vezes é o caso de sistemas baseados em AI. Como resultado, os testes exploratórios são frequentemente usados neste contexto para complementar os testes mais sistemáticos baseados em técnicas, tais como testes metamórficos (ver capítulo 9.5).

Um ***roteiro*** é uma metáfora usada para um conjunto de estratégias e objetivos para os testadores se referirem quando realizam testes exploratórios organizados em torno de um foco especial [B20]. Os roteiros mais comuns para testes exploratórios de sistemas baseados em AI podem focar nos conceitos de viés, *underfitting* e *overfitting* em sistemas ML. Por exemplo, um roteiro de dados pode ser aplicado para testar o modelo. Neste roteiro, o testador poderia identificar diferentes tipos de dados usados para treinamento, sua distribuição, suas variações, seu formato e intervalos etc., e então usar os tipos de dados para testar o modelo.

Os sistemas ML são altamente dependentes da qualidade dos dados de treinamento, e o campo existente do EDA (*Exploratory Data Analysis*) está intimamente relacionado com a abordagem de testes exploratórios. O EDA é onde os dados são examinados quanto a padrões, relacionamentos, tendências e aberturas. Envolve a exploração interativa e orientada por hipóteses dos dados e é descrita em [B21] como "Nós exploramos dados com expectativas. Revisamos nossas expectativas com base no que vemos nos dados. E iteramos este processo". O EDA normalmente requer suporte de ferramentas em duas áreas: para interação com os dados, para permitir que os analistas entendam melhor dados complexos; e para visualização dos dados, para permitir que eles exibam facilmente os resultados da análise. O uso de técnicas exploratórias, impulsionadas principalmente pela visualização de dados, pode ajudar a validar o algoritmo ML que está sendo usado, identificar mudanças que resultam em modelos eficientes, e alavancar a experiência do domínio [B22].

O Google tem um conjunto de vinte e oito testes ML escritos como afirmações, nas áreas de dados, desenvolvimento de modelos, infraestrutura e monitoramento, que é usado como uma lista de verificação de testes dentro do *Google for ML* [B23]. A "lista de verificação de testes ML" do Google é apresentada aqui como publicada pelo Google:

Dados ML:

1. As expectativas das características são capturadas em um esquema;
2. Todas as características são benéficas;
3. Nenhum custo de característica é muito alto;
4. As características aderem aos requisitos do meta-nível;
5. O pipeline de dados tem controles de privacidade apropriados;
6. Novos recursos podem ser adicionados rapidamente;
7. Todos os códigos de recursos de entrada são testados.

Desenvolvimento de modelos:

1. As especificações dos modelos são revisadas e apresentadas;
2. As métricas off-line e on-line estão correlacionadas;
3. Todos os hiper parâmetros foram ajustados;

4. O impacto da estaticidade do modelo é conhecido;
5. Um modelo mais simples não é melhor;
6. A qualidade do modelo é suficiente em fatias de dados importantes;
7. O modelo é testado para considerações de inclusão;

Infraestrutura ML:

1. O treinamento é reprodutível;
2. As especificações dos modelos são testadas por unidade;
3. O pipeline ML é testado quanto à integração;
4. A qualidade do modelo é validada antes de servir;
5. O modelo é depurável;
6. Os modelos são testados através de um processo canário[1] antes de irem para produção;
7. Os modelos de serviço podem ser revertidos à sua configuração inicial.

Testes de monitoramento:

1. As mudanças de dependência resultam em notificação;
2. Os dados não variáveis são válidos para entradas;
3. O treinamento e o serviço não são distorcidos;
4. Os modelos não estão muito envelhecidos;
5. Os modelos são numericamente estáveis;
6. A performance da computação não regrediu;
7. A qualidade da previsão não regrediu.

9.6.1 Exercício prático

Teste exploratório e análise de dados exploratórios (EDA)

Para um modelo e conjunto de dados selecionados, os estudantes farão um roteiro de dados, considerando vários tipos de dados e sua distribuição para vários parâmetros.

Os estudantes realizarão um EDA sobre os dados para identificar os faltantes e/ou potenciais vieses nos dados.

9.7 Seleção de técnicas de teste para sistemas baseados em AI

Um sistema baseado em AI normalmente incluirá tanto componentes AI como não AI. A seleção de técnicas de teste para testar os componentes não-AI é geralmente a mesma que para qualquer teste convencional. Para os componentes baseados em AI, a escolha pode ser mais restrita. Por exemplo, quando um problema

[1] *Nota do BSTQB*: Por volta de 1913, John Scott Haldane, propôs que mineiros levasse canários (pássaros) em gaiolas nos túneis para detectarem gases venenosos. O "Canário", por ser mais frágil que um humano, morreria ao entrar em contato com o gás. *Canary Analysis Service* (CAS) é um serviço centralizado compartilhado no Google que oferece análise automática das principais métricas durante uma mudança de produção. (ver https://research.google/pubs/pub46908/)

de oráculo de teste é percebido (ou seja, gerar resultados esperados é difícil), então, com base nos riscos percebidos, é possível mitigar este problema com o uso do seguinte:

- **Teste consecutivo**: Isto requer que casos de teste estejam disponíveis ou gerados, e um sistema equivalente atuando como um pseudo-oráculo, que para testes de regressão pode ser uma versão anterior do sistema. Para a detecção eficaz de defeitos, pode ser necessário um sistema desenvolvido independentemente.
- **Teste A/B**: Isto frequentemente usa entradas operacionais como casos de teste e normalmente é usado para comparar duas variações do mesmo sistema usando análise estatística. Os testes A/B podem ser usados para verificar o envenenamento de dados de uma nova variante, ou para testes de regressão automatizada de um sistema de autoaprendizado.
- **Testes metamórficos**: Isto pode ser usado por testadores inexperientes para encontrar defeitos com boa relação custo-benefício, embora eles precisem compreender o domínio de aplicação. O MT não é adequado para fornecer resultados definitivos, pois os resultados esperados não são absolutos, mas, em vez disso, relativos aos casos de teste da fonte. O suporte de ferramentas comerciais não está disponível atualmente, mas muitos testes podem ser gerados manualmente.

Os testes contraditórios são normalmente apropriados para modelos ML onde o mau manejo de exemplos adversos pode ter um impacto significativo, ou onde o sistema pode ser atacado. Da mesma forma, testes de envenenamento de dados podem ser apropriados para sistemas ML onde o sistema pode ser atacado.

Quando os sistemas baseados em AI são complexos e têm múltiplos parâmetros, os testes em pares são frequentemente apropriados.

Os testes baseados na experiência são frequentemente adequados para testar sistemas baseados em AI, especialmente para consideração dos dados usados para treinamento e dados operacionais. O EDA pode ser usado para validar o algoritmo ML que está sendo usado, identificar melhorias de eficiência e alavancar a experiência no domínio. O Google descobriu que sua lista de verificação de testes ML é uma abordagem eficaz para sistemas ML.

Na área específica das redes neurais, a cobertura da rede é frequentemente adequada para sistemas de missão crítica, com alguns critérios de cobertura exigindo uma cobertura mais rigorosa do que outros.

10 Ambientes de teste para sistemas baseados em AI [30 min]

Palavras-chave

ambiente de teste virtual

Palavras-chave específicas da AI

Big data, explicabilidade, processador específico de AI, sistema autônomo, sistema de autoaprendizagem, sistema multiagente.

Objetivos de aprendizagem

10.1 Ambientes de teste para sistemas baseados em AI

AI-10.1.1 K2: Descrever os principais fatores que diferenciam os ambientes de teste para sistemas baseados em AI daqueles requeridos para sistemas convencionais.

10.2 Ambientes de teste virtuais para sistemas baseados em AI

AI-10.2.1 K2: Descrever os benefícios proporcionados pelos ambientes de teste virtuais nos testes de sistemas baseados em AI.

10.1 Ambientes de teste para sistemas baseados em AI

Os sistemas baseados em AI podem ser usados em uma grande variedade de ambientes operacionais, o que significa que os ambientes de teste são igualmente diversos. As características dos sistemas baseados em AI que podem fazer com que os ambientes de teste sejam diferentes daqueles dos sistemas convencionais incluem:

- **Autoaprendizagem**: Espera-se que os sistemas de autoaprendizagem, e alguns sistemas autônomos, se adaptem a ambientes operacionais em mudança que podem não ter sido totalmente definidos quando o sistema foi inicialmente implantado (ver capítulo 2.1). Como resultado, definir ambientes de teste que possam imitar estas mudanças ambientais indefinidas são inerentemente difíceis e podem exigir tanto imaginação por parte dos testadores quanto um nível de aleatoriedade incorporado no ambiente de teste.
- **Autônomo**: Espera-se que os sistemas autônomos respondam às mudanças em seu ambiente sem intervenção humana e reconheçam situações em que a autonomia deve ser cedida de volta aos operadores humanos (ver capítulo 2.2). Para alguns sistemas, identificar e então imitar as circunstâncias para ceder autonomia pode exigir que os ambientes de teste empurrem os sistemas para extremos. Para alguns sistemas autônomos, sua finalidade é trabalhar em ambientes perigosos e a criação de ambientes de teste representativos e perigosos pode ser um desafio.
- **Multiagentes**: Espera-se que sistemas baseados em múltiplos agentes de AI funcionem em conjunto com outros sistemas baseados em AI, o ambiente de teste pode precisar incorporar um nível de não-determinismo para que possa imitar o não-determinismo dos sistemas baseados em AI com os quais o SUT interage.
- **Explicabilidade**: A natureza de alguns sistemas baseados em AI pode tornar difícil determinar como o sistema tomou suas decisões (ver capítulo 2.7). Quando isso for importante para entender antes da implantação, o ambiente de teste pode precisar incorporar ferramentas como um meio de explicar como as decisões são tomadas.
- **Hardware**: Alguns dos equipamentos usados para hospedar sistemas baseados em AI são projetados especificamente para este fim, tais como processadores específicos de AI (ver capítulo 1.6). A necessidade de incluir tal hardware no ambiente de teste deve ser considerada como parte relevante do planejamento de teste.
- **Big data**: Quando se espera que um sistema baseado em AI consuma grandes dados (p. ex., dados de alto volume, de alta velocidade e/ou de alta variedade), então a configuração deste como parte de um ambiente de teste precisa de cuidados no planejamento e implementação (ver capítulo 7.3).

10.2 Ambientes de teste virtuais para sistemas baseados em AI

O uso de um ambiente de teste virtual ao testar um sistema baseado em AI traz os seguintes benefícios:

- **Cenários perigosos**: Estes podem ser testados sem colocar em risco o SUT, outros sistemas interativos, incluindo humanos, ou o ambiente operacional (p. ex., árvores, edifícios).
- **Cenários inusitados**: Estes podem ser testados quando, de outra forma, seria muito demorado ou caro configurar estes cenários para operações reais (p. ex., esperar por um evento raro, como um eclipse solar completo ou quatro ônibus entrando simultaneamente no mesmo cruzamento rodoviário). Da mesma forma, os casos limites, que são difíceis de criar no mundo real, podem ser criados mais facilmente, repetidamente e de forma reprodutível em um ambiente de teste virtual.
- **Cenários extremos**: Estes podem ser testados quando seria caro ou impossível colocá-los em prática (p. ex., para um desastre nuclear ou exploração do espaço profundo).
- **Cenários de tempo intensivo**: Estes podem ser testados em escalas de tempo reduzidas (p. ex., várias vezes por segundo) em um ambiente virtual. Em contraste, estes podem levar horas ou dias

para se instalar e funcionar em tempo real. Uma outra vantagem é que vários ambientes de teste virtuais podem ser executados em paralelo. Isso normalmente ocorre na nuvem e permite que muitos cenários sejam executados simultaneamente, o que pode não ser possível utilizando o hardware real do sistema.

- **Observação e controle**: Os ambientes de teste virtuais proporcionam um controle muito maior do ambiente de teste. Por exemplo, eles podem garantir que um conjunto incomum de condições comerciais financeiras seja replicado. Além disso, proporcionam uma observação muito melhor, pois todas as partes do ambiente digitalmente fornecidas podem ser continuamente monitoradas e registradas.
- **Disponibilidade**: A simulação de hardware por ambientes de teste virtuais permite que os sistemas sejam testados com componentes de hardware (simulados) que podem estar indisponíveis, talvez por ainda não terem sido desenvolvidos ou por serem muito caros.

Os ambientes de teste virtual podem ser construídos especificamente para um determinado sistema, podem ser genéricos ou podem ser desenvolvidos para suportar domínios específicos de aplicação. Ambientes de teste virtual, tanto comerciais quanto de código aberto, estão disponíveis para suportar o teste de sistemas baseados em AI. Exemplos incluem:

- **Morse**: O *Modular Open Robots Simulation Engine*, é um simulador para um ou mais robôs móveis genéricos, com base no motor do jogo Blender [R24].
- **Habitat AI**: Esta é uma plataforma de simulação criada pela Facebook AI, projetada para treinar autômatos (como robôs virtuais) em ambientes foto-realistas 3D [R25].
- **DRIVE Constelattion**: Esta é uma plataforma aberta e escalonável para carros autônomos da NVIDIA. Ela é baseada em uma plataforma baseada em nuvem e é capaz de gerar bilhões de quilômetros de testes de veículos autônomos [R26].
- **MATLAB and Simulink**: Estes fornecem a capacidade de preparar dados de treinamento, produzir modelos ML e simular a execução de sistemas baseados em AI incluindo os modelos que utilizam dados sintéticos [R27].

11 Usando AI para testes [195 min]

Palavras-chave

testes visuais.

Palavras-chave específicas da AI

algoritmo de agrupamento, classificação, interface gráfica do usuário (GUI), predição de defeitos, técnicas Bayesianas.

Objetivos de aprendizagem

11.1 Tecnologias de AI para testes

AI-11.1.1 K2: Categorizar as tecnologias de AI usadas em testes de software.

HO-11.1.1 H2: Discutir, usando exemplos, aquelas atividades em testes onde a AI é menos provável de ser usada.

11.2 Usando AI para analisar os defeitos relatados

AI-11.2.1 K2: Explicar como a AI pode ajudar a apoiar a análise de novos defeitos.

11.3 Usando AI para a geração de casos de teste

AI-11.3.1 K2: Explicar como a AI pode ajudar na geração de casos de teste.

11.4 Usando AI para a otimização de conjuntos de teste de regressão

AI-11.4.1 K2: Explicar como a AI pode ajudar na otimização de conjuntos de teste de regressão

11.5 Usando AI para previsão de defeito

AI-11.5.1 K2: Explicar como a AI pode ajudar na previsão de defeitos.

HO-11.5.1 H2: Implementar um sistema simples de previsão de defeitos baseado em AI.

11.6 Usando AI para teste de interfaces de usuário

AI-11.6.1 K2: Explicar o uso de AI em testes de interfaces de usuário

11.1 Tecnologias de AI para testes

Várias tecnologias de AI estão listadas no capítulo 1.4, todas as quais podem ser usadas para suportar algum aspecto específico dos testes de software. De acordo com Harman [B24], a comunidade de engenharia de software usa três grandes áreas das tecnologias de AI:

- **Lógica difusa e métodos probabilísticos**: Estas envolvem o uso de técnicas de AI para lidar com problemas do mundo real que são eles mesmos probabilísticos. Por exemplo, a AI pode ser usada para analisar e prever possíveis falhas no sistema usando técnicas Bayesianas. Estas podem estimar a probabilidade de falha de componentes ou funções, ou refletir a natureza potencialmente aleatória das interações humanas com o sistema.
- **Classificação, aprendizagem e previsão**: Isto pode ser usado para vários casos de uso, como a previsão de custos como parte do planejamento do projeto ou da previsão de defeitos. Como encarnado pelo ML, esta área é usada para muitas tarefas de teste de software, incluindo gerenciamento de defeitos (ver capítulo 11.2), previsão de defeitos (ver capítulo 11.5) e teste de interface de usuário (ver capítulo 11.6).
- **Técnicas de pesquisa computacional e otimização**: Estes podem ser usados para resolver problemas de otimização usando uma busca computacional de espaços de busca potencialmente grandes e complexos (p. ex., usando algoritmos de busca). Exemplos incluem a geração de casos de teste (ver capítulo 11.3), a identificação do menor número de casos de teste que atinge um determinado critério de cobertura, e a otimização de casos de teste de regressão (ver capítulo 11.4).

A categorização acima é necessariamente ampla, pois há uma considerável sobreposição entre as tarefas de teste que podem ser implementadas pela AI e as diferentes tecnologias de AI. É também apenas uma categorização, e outras poderiam ser criadas que podem ser igualmente válidas.

11.1.1 Exercício prático

O uso de AI em testes

Como parte de uma discussão, os estudantes identificarão atividades e tarefas de teste que são atualmente impraticáveis para implementação com o AI. Estas poderiam incluir:

- *Especificação de oráculos de teste.*
- *Comunicação com stakeholders para esclarecer ambiguidades e recuperar informações em falta.*
- *Sugerir melhorias para a experiência do usuário.*
- *Desafiar as suposições dos stakeholders fazendo perguntas incômodas.*
- *Compreender as necessidades dos usuários.*

Deve ser feita uma distinção entre AI-estreita, que poderia ser usada para algumas tarefas limitadas, e AI-geral, que atualmente não está disponível (ver capítulo 1.2).

11.2 Usando AI para analisar os defeitos relatados

Os defeitos relatados são geralmente categorizados, priorizados e quaisquer duplicatas identificadas. Esta atividade é frequentemente chamada de **triagem** ou análise de defeitos e se destina a otimizar o tempo decorrido na resolução de defeitos. A AI pode ser usada para apoiar esta atividade de várias maneiras, como por exemplo:

- **Categorização**: NLP [B25] pode ser usado para analisar textos dentro de relatórios de defeitos e extrair tópicos, tais como a área de funcionalidade afetada, que podem então ser fornecidos junto com outros meta dados para algoritmos de agrupamento, tais como vizinhos mais próximos ou máquinas de suporte vetorial. Estes algoritmos podem identificar categorias de defeitos adequados e

destacar defeitos similares ou duplicados. A categorização baseada em AI é particularmente útil para sistemas automatizados de relatório de defeitos (p. ex., para Microsoft Windows e para Firefox) e em grandes projetos com muitos engenheiros de software.

- **Criticidade**: Os modelos ML treinados nas características dos defeitos mais críticos podem ser usados para identificar os defeitos mais prováveis de causar as falhas do sistema que representam uma grande porcentagem dos defeitos relatados [B26].
- **Atribuição**: Os modelos ML podem sugerir quais desenvolvedores são mais adequados para corrigir defeitos particulares, com base no conteúdo do defeito e nas atribuições anteriores do desenvolvedor.

11.3 Usando AI para geração de casos de teste

O uso de AI para gerar testes pode ser uma técnica muito eficaz para criar rapidamente ativos de teste e maximizar a cobertura (p. ex., código ou cobertura de requisitos). A base para gerar estes testes inclui o código fonte, a interface do usuário e um modelo de teste legível por máquina. Algumas ferramentas também baseiam os testes na observação do comportamento de baixo nível do sistema através da instrumentação ou através de arquivos de log [B27].

Entretanto, a menos que um modelo de teste que defina os comportamentos necessários seja usado como base dos testes, esta forma de geração de testes geralmente sofre de um problema de oráculo de teste porque a ferramenta baseada em AI não sabe quais devem ser os resultados esperados para um determinado conjunto de dados de teste. Se um sistema adequado estiver disponível para ser usado como pseudo-oráculo, uma solução é usar o teste consecutivo (ver capítulo 9.3). Alternativamente, os testes poderiam ser executados com o resultado esperado de que nem uma "aplicação que não responde" nem uma falha do sistema ocorreu, ou outros indicadores simples de falha similares.

Pesquisas comparando ferramentas de geração de testes baseados em AI com ferramentas similares de teste fuzz não-AI mostram que as ferramentas baseadas em AI podem atingir níveis equivalentes de cobertura e encontrar mais defeitos, reduzindo a sequência média de passos necessários para causar uma falha de uma média de cerca de 15.000 passos para cerca de 100 passos. Isto torna a depuração muito mais fácil [B27].

11.4 Usando AI para a otimização de conjuntos de teste de regressão

Conforme as mudanças são feitas em um sistema, novos testes são criados, executados e se tornam candidatos a um conjunto de testes de regressão. Para evitar que os conjuntos de testes de regressão cresçam muito, eles devem ser frequentemente otimizados para selecionar, priorizar e até mesmo aumentar os casos de teste para criar um conjunto de testes de regressão mais eficaz e eficiente.

Uma ferramenta baseada em AI pode realizar a otimização do conjunto de testes de regressão analisando, por exemplo, as informações de resultados de testes anteriores, defeitos associados e as últimas mudanças que foram feitas, tais como recursos que são quebrados com mais frequência e que os testes exercitam código impactado por mudanças recentes.

Pesquisas mostram que reduções de 50% no tamanho de um conjunto de testes de regressão podem ser alcançadas enquanto ainda se detecta a maioria dos defeitos [B28], e reduções de 40% na duração da execução do teste podem ser alcançadas sem redução significativa na detecção de defeitos para testes de integração contínua [B29].

11.5 Usando AI para previsão de defeito

A previsão de defeitos pode ser usada para antever a presença do defeito, de quantos estão presentes, ou se os eles podem ser encontrados. Esta capacidade depende da sofisticação da ferramenta utilizada.

Os resultados são normalmente usados para priorizar os testes (p. ex., mais testes para aqueles componentes onde mais defeitos são previstos).

A previsão de defeitos é normalmente baseada na métrica do código fonte, métrica do processo ou pessoas, e métrica organizacional. Devido à existência de tantos fatores potenciais a serem considerados, determinar a relação entre esses fatores e a probabilidade de defeitos está além das capacidades humanas. Como resultado, o uso de uma abordagem baseada em AI, que tipicamente usa ML, é uma necessidade. A previsão de defeitos é mais eficaz quando baseada em experiências anteriores em uma situação semelhante (p. ex., com a mesma base de código e/ou os mesmos desenvolvedores).

A previsão de defeitos usando ML tem sido usada com sucesso em várias situações diferentes (p. ex., [B30] e [B31]). Os melhores prognósticos foram encontrados como pessoas e medidas organizacionais em vez das métricas de código fonte mais amplamente utilizadas, tais como linhas de código e complexidade ciclomática [B32].

11.5.1 Exercício prático

Construir um sistema de previsão de defeitos

Os estudantes usarão um conjunto de dados adequado (p. ex., incluindo medidas do código fonte e dados de defeitos correspondentes) para construir um modelo simples de previsão de defeitos e usá-lo para prever a probabilidade de defeitos usando medidas do código fonte a partir de códigos semelhantes.

O modelo deve usar pelo menos quatro características do conjunto de dados, e a classe deve explorar os resultados usando várias características diferentes para destacar como os resultados mudam com base nas características selecionadas.

11.6 Usando AI para teste de interfaces de usuário

11.6.1 Usando a AI para testar através da interface gráfica do usuário (GUI)

O teste através da GUI é a abordagem típica para testes manuais (exceto para testes de componentes) e é frequentemente o ponto de partida para iniciativas de automação de testes. Os testes resultantes imitam a interação humana com o objeto de teste. Esta automação de teste com scripts pode ser implementada aplicando uma abordagem de captura/reprodução, usando as coordenadas reais dos elementos da interface do usuário, ou os objetos/widgets definidos por software da interface. Entretanto, esta abordagem sofre vários inconvenientes com a identificação do objeto, incluindo a sensibilidade a mudanças na interface, mudanças de código e mudanças de plataforma.

A AI pode ser usada para reduzir a fragilidade desta abordagem, empregando ferramentas baseadas em AI para identificar os objetos corretos usando vários critérios (p. ex., *XPath*, rótulo, id, classe, coordenadas X/Y), e para escolher os critérios de identificação historicamente mais estáveis. Por exemplo, a identificação de um botão em uma determinada área da aplicação pode mudar a cada lançamento, e assim a ferramenta baseada em AI pode atribuir uma importância menor a esta identificação ao longo do tempo e colocar mais confiança em outros critérios. Esta abordagem classifica os objetos na interface do usuário como correspondendo ao teste, ou não correspondendo ao teste.

Alternativamente, o teste visual usa o reconhecimento de imagem para interagir com objetos GUI através da mesma interface que um usuário real e, portanto, não precisa acessar o código subjacente e as definições da interface. Isto o torna completamente não intrusivo e independente da tecnologia subjacente. Os scripts só precisam funcionar através da interface visível do usuário. Esta abordagem permite que o testador crie scripts que interagem diretamente com as imagens, botões e campos de texto na tela da mesma forma que um usuário humano, sem ser afetado pelo layout geral da tela. O uso do reconhecimento de imagem na automação de testes pode se tornar restrito pelos recursos computacionais necessários. No entanto, a disponibilidade de AI acessível que suporta o reconhecimento de imagem sofisticado agora torna esta abordagem possível para o uso convencional.

11.6.2 Usando AI para testar a GUI

Os modelos ML podem ser usados para determinar o aceite das telas de interface do usuário (p. ex., usando heurística e aprendizagem supervisionada). As ferramentas baseadas nestes modelos podem identificar

elementos renderizados incorretamente, determinar se alguns objetos são inacessíveis ou difíceis de detectar, e detectar vários outros problemas com a aparência visual da GUI.

Enquanto o reconhecimento de imagem é uma forma de algoritmo de visão computadorizada, outras formas de visão computadorizada baseada em AI podem ser usadas para comparar imagens (p. ex., capturas de tela) para identificar mudanças não intencionais no layout, tamanho, posição, cor, fonte ou outros atributos visíveis dos objetos. Os resultados destas comparações podem ser usados para suportar testes de regressão para verificar se as mudanças no objeto de teste não afetaram adversamente a interface do usuário.

A tecnologia para verificar a aceitabilidade das telas pode ser combinada com ferramentas de comparação para criar ferramentas de teste de regressão baseadas em AI mais sofisticadas que são capazes de aconselhar se as mudanças detectadas na interface do usuário são provavelmente aceitáveis para os usuários, ou se essas mudanças devem ser sinalizadas para verificação por um humano. Tais ferramentas baseadas em AI também podem ser usadas para suportar testes de compatibilidade em diferentes navegadores, dispositivos ou plataformas com o objetivo de verificar se a interface do usuário para o mesmo aplicativo funciona corretamente em vários navegadores/dispositivos/plataformas.

12 Referências

Normas

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

[S02] DIN SPEC 92001-1, Artificial Intelligence - Life Cycle Processes and Quality Requirements - Parte 1: Quality Meta Model, https://www.din.de/en/wdcbeuth:din21:303650673 (acessado em maio 2021).

[S03] DIN SPEC 92001-2, Artificial Intelligence - Life Cycle Processes and Quality Requirements - Parte 2: Technical and Organizational Requirements, https://www.din.de/en/innovation-and-research/din-spec-en/projects/wdcproj:din21:298702628 (acessado em maio 2021).

[S04] ISO 26262 - https://www.iso.org/standard/68383.html (acessado em maio 2021)

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

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

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

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

[S09] Lei Geral de Proteção de Dados Pessoais (LGPD), lei n° 13.709/2018 de 2018 http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm

Documentos ISTQB®

[I01] ISTQB® Certified Tester Foundation Level Syllabus, Version 2018 V3.1 https://www.istqb.org/downloads/category/2-foundation-level-documents.html (acessado em maio 2021).

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

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

Livros e Artigos

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

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

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

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

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

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

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

[B08] G20 Ministerial Statement on Trade and Digital Economy: Annex. https://www.mofa.go.jp/files/000486596.pdf (acessado em maio de 2021).

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

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

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

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

[B13] Sun et al, Testing Deep Neural Networks, https://www.researchgate.net/publication/323747173_Testing_Deep_Neural_Network s (acessado em novembro de 2019).

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

[B15] Riccio, V. et al, Testing Machine Learning based Systems: A Systematic Mapping. Empirical Software Engineering, https://link.springer.com/article/10.1007/s10664-02009881-0 (acessado em maio de 2021)

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

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

[B18] Chen et al, Metamorphic Testing: A Review of Challenges and Opportunities, ACM Comput. Surv. 51, 1, artigo 4, janeiro 2018. https://www.researchgate.net/publication/322261865_Metamorphic_Testing_A_Revie w_of_Challenges_and_Opportunities (acessado em maio de 2021).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Outras Referências

As seguintes referências apontam para as informações disponíveis na Internet. Mesmo que estas referências tenham sido verificadas no momento da publicação, o ISTQB® não pode ser responsabilizado se as referências não estiverem mais disponíveis.

[R01] Colaboradores da Wikipédia, "AI effect," Wikipedia, https://en.wikipedia.org/wiki/AI_effect (acessado em maio de 2021).

[R02] https://mxnet.apache.org/ (acessado em maio de 2021).

[R03] https://docs.microsoft.com/en-us/cognitive-toolkit/ (acessado em maio de 2021).

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

[R05] https://www.tensorflow.org/ (acessado em maio de 2021).

[R06] https://keras.io/ (acessado em maio de 2021).

[R07] https://pytorch.org/ (acessado em maio de 2021).

[R08] https://scikit-learn.org/stable/whats_new/v0.23.html (acessado em maio de 2021).

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

[R10] Cloud TPU, https://cloud.google.com/tpu/ (acessado em maio de 2021).

[R11] Edge TPU, https://cloud.google.com/edge-tpu/ (acessado em maio de 2021).

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

[R13] The Evolution of EyeQ, https://www.mobileye.com/our-technology/evolution-eyeqchip/ (acessado em maio de 2021).

[R14] ImageNet - http://www.image-net.org/ (acessado em maio de 2021).

[R15] Google's BERT - https://github.com/google-research/bert (acessado em maio de 2021).

[R16] https://www.kaggle.com/datasets (acessado em maio de 2021).

[R17] https://www.kaggle.com/paultimothymooney/2018-kaggle-machine-learning-datascience-survey (acessado em maio de 2021).

[R18] MLCommons - https://mlcommons.org/ (acessado em maio de 2021).

[R19] DAWNBench – https://dawn.cs.stanford.edu/benchmark (acessado em maio de 2021).

[R20] MLMark – https://www.eembc.org/mlmark (acessado em maio de 2021).

[R21] https://digital-strategy.ec.europa.eu/en/library/assessment-list-trustworthy-artificialintelligence-altai-self-assessment| Shaping Europe's digital future (europa.eu) (acessado em maio de 2021)

[R22] https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai (acessado em maio de 2021)

[R23] Google GraphicsFuzz, https://github.com/google/graphicsfuzz (acessado em maio de 2021).

[R24] http://www.openrobots.org/morse/doc/0.2.1/morse.html (acessado em maio de 2021).

[R25] https://ai.facebook.com/blog/open-sourcing-ai-habitat-a-simulation-platform-forembodied-ai-research/ (acessado em maio de 2021).

[R26] https://www.nvidia.com/en-gb/self-driving-cars/drive-constellation/ (acessado em maio de 2021).

[R27] https://uk.mathworks.com/discovery/artificial-intelligence.html#ai-with-matlab (acessado em maio de 2021).

13 Apêndice A – Abreviações

Abreviação	Definição	Nota de tradução
AI	Artificial intelligence	Inteligência artificial
AIaaS	AI as a service	Inteligência artificial como serviço
API	Application programming interface	
AUC	Area under curve	
DL	Deep learning	Aprendizado profundo
DNN	Deep neural network	
EDA	Exploratory data analysis	
EU	European Union	União Européia
FN	False negative	Falso-negativo
FP	False positive	Falso-positivo
GDPR	General Data Protection Regulation	
GPU	Graphical processing unit	Unidade de processamento gráfico
GUI	Graphical user interface	Inteface gráfica de usuário
LIME	Local interpretable model-agonistic	
MC/DC	Modified condition decision coverage	
ML	Machine learning	
MR	Metamorphic relation	
MSE	Mean square error	
MT	Metamorphic testing	
NLP	Natural language processing	
ROC	Receiver operating characteristic	
SUT	System under test	
SVM	Support vector machine	
TN	True negative	
TP	True positive	
XAI	Explainable AI	

14 Apêndice B – Termos específicos de AI

Termo original	Termo	Definição
accuracy	acurácia	A métrica de performance funcional ML usada para avaliar um classificador, que mede a proporção de previsões que estavam corretas. (pós ISO/IEC TR 29119-11)
activation function	função de ativação	A fórmula associada a um neurônio em uma rede neural que determina a saída do neurônio a partir das entradas para o neurônio.
activation value	Valor de ativação	A saída de uma função de ativação de um neurônio em uma rede neural.
adversarial attack	ataque contraditório	O uso deliberado de exemplos contraditórios para causar o fracasso de um modelo ML
AI as a Service (AIaaS)	AI como serviço (AIaaS)	Um modelo de licenciamento e entrega de software no qual a AI e os serviços de desenvolvimento da AI são hospedados de forma centralizada.
AI component	Componente AI	Um componente que fornece a funcionalidade da AI.
AI effect	Efeito AI	A situação em que um sistema de AI previamente rotulado não é mais considerado AI à medida que a tecnologia avança. (ISO/IEC TR 29119-11)
AI-based system	Sistema baseado em AI	Um sistema que integra um ou mais componentes de AI.
AI-specific processor	Processador específico de AI	Um tipo de hardware especializado projetado para acelerar as aplicações de AI.
algorithmic bias	viés algorítmico	Um tipo de viés causado pelo algoritmo ML.
annotation	anotação	A atividade de identificar objetos em imagens com caixas delimitadoras para fornecer dados rotulados para classificação.
area under curve (AUC)	área sob curva (AUC)	Uma medida de quão bem um classificador pode distinguir entre duas classes.
artificial intelligence (AI)	inteligência artificial (AI)	A capacidade de um sistema projetado para adquirir, processar, criar e aplicar conhecimentos e habilidades. (ISO/IEC TR 29119-11)
association	associação	Uma técnica de aprendizagem não supervisionada que identifica as relações e dependências entre as amostras.
augmentation	aumento	A atividade de criar pontos de dados com base em um conjunto de dados já existente.
automation bias	viés de automação	Um tipo de viés causado por uma pessoa que favorece as recomendações de um sistema de tomada de decisão automatizado em relação a outras fontes. Sinônimo: viés de complacência.
autonomous system	sistema autônomo	Um sistema capaz de trabalhar sem intervenção humana por períodos prolongados.
autonomy	autonomia	A capacidade de um sistema de trabalhar por períodos sustentados.
Bayesian model	Modelo Bayesiano	Um tipo de viés causado pelo algoritmo ML.
Bayesian technique	Técnica Bayesiana	A atividade de identificar objetos em imagens com caixas delimitadoras para fornecer dados rotulados para classificação.
bias	viés	A diferença sistemática no tratamento de certos objetos, pessoas ou grupos em comparação com outros. (ISO/IEC DIS 22989)
big data	Big data	Extensos conjuntos de dados cujas características em termos de volume, variedade, velocidade e/ou variabilidade requerem tecnologias e técnicas especializadas para processar.

Termo original	Termo	Definição
case-based reasoning	raciocínio baseado em casos	A técnica de resolver um novo problema com base nas soluções de problemas similares do passado.
chatbot	chatbot	Um aplicativo usado para conduzir uma conversa via texto ou texto-fala.
classification	classificação	Um tipo de função ML que prevê a classe de saída para uma determinada entrada. (pós ISO/IEC TR 29119-11)
classifier	classificador	Um modelo ML utilizado para classificação. Sinônimo: modelo de classificação.
clustering	agrupamento	Um tipo de função ML que agrupa pontos de dados similares.
clustering algorithm	algoritmo de agrupamento	Um tipo de algoritmo ML usado para reunir objetos similares em grupos.
concept drift	desvio de conceito	Uma mudança na percepção da precisão das previsões de um modelo ML ao longo do tempo causada por mudanças nas expectativas dos usuários, no comportamento e no ambiente operacional.
confusion matrix	matriz de confusão	Uma técnica para resumir o desempenho funcional de um algoritmo de classificação do ML.
data acquisition	aquisição de dados	A atividade de aquisição de dados relevantes para o problema comercial a ser resolvido por um modelo ML.
data labelling	rotulagem de dados	A atividade de adicionar marcas significativas a objetos em dados brutos para apoiar a classificação no ML.
data pipeline	pipeline de dados	A implementação de atividades de preparação de dados para fornecer dados de entrada para apoiar o treinamento por um algoritmo ML ou previsão por um modelo ML.
data point	ponto de dados	Um conjunto de uma ou mais medidas que compreende uma única observação utilizada como parte de um conjunto de dados.
data poisoning	envenenamento de dados	A manipulação deliberada e maliciosa de dados de treinamento ou de entrada de dados para um modelo ML.
data preparation	preparação dos dados	As atividades de aquisição de dados, pré-processamento de dados e engenharia de recursos no fluxo de trabalho ML.
data pre-processing	pré-processamento de dados	As atividades de limpeza de dados, transformação de dados, aumento de dados e amostragem de dados no fluxo de trabalho ML.
data visualization	visualização de dados	Uma técnica para representar graficamente as relações de dados, tendências e padrões.
dataset	conjunto de dados	Uma coleção de dados utilizados para treinamento, avaliação, teste e previsão em ML.
decision threshold	limite de decisão	Um valor que transforma o resultado de uma função de previsão em um resultado binário tanto acima quanto abaixo do valor. Sinônimo: limiar de discriminação
decision tree	árvore de decisão	Um modelo ML em forma de árvore cujos nós representam decisões, e cujos ramos representam resultados possíveis.
deductive classifier	classificador dedutivo	Um classificador baseado na aplicação de inferência e lógica à entrada de dados.
deep learning (DL)	deep learning (DL)	ML usando redes neurais com múltiplas camadas.
deep neural network	rede neural profunda	Uma rede neural composta por várias camadas de neurônios. Sinônimo: perceptron multicamadas
defect prediction	predição de defeitos	Uma técnica para prever as áreas dentro do objeto de teste nas quais ocorrerão defeitos, ou a quantidade de defeitos que estão presentes.
deterministic system	sistema determinístico	Um sistema que produzirá o mesmo conjunto de saídas e estado final a partir de um determinado conjunto de entradas e estado inicial.
edge computing	computação de ponta	A parte de uma arquitetura distribuída na qual o processamento da informação é realizado próximo ao local onde essa informação é utilizada.

Termo original	Termo	Definição
epoch	época	Uma iteração do treinamento ML em todo o conjunto de dados de treinamento.
evolution	evolução	O processo de mudança contínua de um estado inferior, mais simples ou pior para um estado superior, mais complexo ou melhor.
expert system	sistema especializado	Um sistema baseado em AI para resolver problemas em um determinado domínio ou área de aplicação através de inferências a partir de uma base de conhecimento desenvolvida a partir da experiência humana.
explainability	explicabilidade	O nível de compreensão de como o sistema baseado em AI chegou a um determinado resultado (ISO/IEC TR 29119-11)
explainable AI (XAI)	AI explicável (XAI)	O campo de estudo relacionado à compreensão dos fatores que influenciam os resultados do sistema de AI.
exploratory data analysis (EDA)	análise exploratória de dados (EDA)	A exploração interativa, orientada por hipóteses e visual dos dados usados para apoiar a engenharia de recursos.
F1-Score	F1-Score	Uma métrica de desempenho funcional ML utilizada para avaliar um classificador que proporciona um equilíbrio entre recall e precisão
false negative (FN)	falso negativo (FN)	Uma previsão do modelo ML em que o modelo prevê erroneamente a classe negativa.
false positive (FP)	falso positivo (FP)	Uma previsão do modelo ML em que o modelo prevê erroneamente a classe positiva.
feature	recurso	Um atributo individual mensurável dos dados de entrada utilizados para treinamento por um algoritmo ML e para previsão por um modelo ML
feature engineering	engenharia de recursos	A atividade na qual aqueles atributos nos 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)
flexibility	flexibilidade	A capacidade de um sistema de trabalhar em contextos fora de sua especificação inicial (pós ISO/IEC TR 29119-11)
fuzzy logic	lógica fuzzy	Um tipo de lógica baseada no conceito de verdade parcial representada por fatores de certeza entre 0 e 1
general AI	AI-geral	AI que exibe um comportamento inteligente comparável ao humano em toda a gama de habilidades cognitivas (ISO/IEC TR 29119-11). Sinônimo: AI forte
General Data Protection Regulation (GDPR)	Regulamento Geral de Proteção de Dados (GDPR)	A regulamentação da União Europeia (UE) sobre proteção de dados e privacidade que se aplica aos dados dos cidadãos da UE e do Espaço Econômico Europeu
graphical processing unit (GPU)	unidade de processamento gráfico (GPU)	Um circuito integrado específico da aplicação projetado para manipular e alterar a memória para acelerar a criação de imagens em um buffer de quadros destinado à saída para um dispositivo de exibição
ground truth	verdade fundamental	As informações fornecidas por observação direta e medição que se sabe serem reais ou verdadeiras
hyperparameter	hiper parâmetro	Um parâmetro utilizado para controlar o treinamento de um modelo ML ou para definir a configuração de um modelo ML
hyperparameter tuning	ajuste hiper paramétrica	A atividade de determinar os hiper parâmetros ideais com base em objetivos particulares
inappropriate bias	viés inadequado	Um tipo de viés que faz com que um sistema produza resultados que levam a efeitos adversos para um determinado grupo
intelligent agent	agente inteligente	Um programa autônomo que direciona sua atividade para o alcance de objetivos utilizando observações e ações
inter-cluster metric	métrica inter-cluster	Uma métrica que mede a similaridade de pontos de dados em diferentes agrupamentos

Termo original	Termo	Definição
interpretability	interpretabilidade	O nível de compreensão de como funciona a tecnologia de AI subjacente (ISO/IEC TR 29119-11)
intra-cluster metric	métrica intra-cluster	Uma métrica que mede a similaridade de pontos de dados dentro de um agrupamento
k-nearest neighbor	k-nearest vizinho	Uma abordagem de classificação que estima a probabilidade de associação em grupo para um ponto de dados dependente da associação em grupo dos pontos de dados mais próximos a ele
learning algorithm	algoritmo de aprendizagem	Um programa que produz um modelo ML baseado nas propriedades do conjunto de dados do treinamento
LIME method	Método LIME	O programa de Explicações do Modelo Interpretável Local para explicar as previsões de um modelo ML
linear regression	regressão linear	Uma técnica estatística que modela a relação entre as variáveis ajustando uma equação linear aos dados observados quando a variável alvo é numérica.
logistic regression	regressão logística	Uma técnica estatística que modela a relação entre variáveis quando a variável alvo é categórica e não numérica.
machine learning (ML)	aprendizagem de máquinas (ML)	O processo que utiliza técnicas computacionais para permitir que os sistemas aprendam com dados ou experiência (ISO/IEC TR 29119-11)
mean square error (MSE)	erro médio quadrático (MSE)	A medida estatística da diferença média quadrática entre os valores estimados e o valor real
ML algorithm	Algoritmo ML	Um algoritmo usado para criar um modelo ML a partir de um conjunto de dados de treinamento
ML benchmark suite	Conjunto de referência ML	Um conjunto de dados usado para comparar modelos ML e algoritmos ML em uma gama de métricas de avaliação
ML framework	Estrutura ML	Uma ferramenta ou biblioteca que apoia a criação de um modelo ML
ML function	Função ML	Funcionalidade implementada por um modelo ML, tal como classificação, regressão ou agrupamento
ML model evaluation	Avaliação do modelo ML	O processo de comparação das métricas de desempenho funcional obtidas ML com os critérios exigidos e os de outros modelos ML
ML model training	Treinamento do modelo ML	O processo de aplicação do algoritmo ML ao conjunto de dados de treinamento para criar um modelo ML
ML model tuning	ajuste do modelo ML	O processo de testar hiper parâmetros para alcançar o melhor desempenho
ML system	Sistema ML	Um sistema que integra um ou mais modelos ML
ML workflow	Fluxo de trabalho ML	Uma sequência de atividades utilizadas para gerenciar o desenvolvimento e implantação de um modelo ML
multi-agent system	sistema multiagente	Um sistema que compreende múltiplos agentes inteligentes
narrow AI	AI-estreita	A AI concentrou-se em uma única tarefa bem definida para resolver um problema específico (ISO/IEC TR 29119-11). Sinônimo: AI fraca
natural language processing (NLP)	processamento em linguagem natural (NPL)	Um campo da computação que proporciona a capacidade de ler, compreender e derivar significado de línguas naturais
neural network	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 a seus valores de entrada e o transmite a outros elementos ou o apresenta como um valor de saída (ISO/IEC 2382) Sinônimo: rede neural artificial
neural network trojan	rede neural troiano	Uma vulnerabilidade injetada em uma rede neural usando um ataque de envenenamento de dados com a intenção de explorá-la posteriormente
neuromorphic processor	processador neuromórfico	Um circuito integrado projetado para imitar os neurônios biológicos do cérebro humano

Termo original	Termo	Definição
neuron	neurônio	Um nó em uma rede neural, geralmente recebendo múltiplos valores de entrada e gerando um valor de ativação
noise	ruído	Uma distorção ou corrupção nos dados
non-deterministic system	sistema não-determinístico	Um sistema que nem sempre produzirá o mesmo conjunto de saídas e estado final, dado um conjunto particular de entradas e estado inicial
outlier	anômalo	Uma observação que está fora do padrão geral da distribuição de dados
overfitting	overfitting	A geração de um modelo ML que corresponde muito próximo do conjunto de dados de treinamento, resultando em um modelo que encontra dificuldades para generalizar a novos dados (Depois da ISO/IEC TR 29119-11)
perceptron	perceptron	Uma rede neural com apenas uma camada e um neurônio
precision	precisão	Uma métrica de desempenho funcional ML utilizada para avaliar um classificador, que mede a proporção de positivos previstos que estavam corretos (após ISO/IEC TR 29119-11)
pre-trained model	modelo pré-treinado	Um modelo ML já treinado quando foi obtido
probabilistic system	sistema probabilístico	Um sistema cujo comportamento é descrito em termos de probabilidades; portanto, seus resultados não podem ser perfeitamente previstos
procedural reasoning	raciocínio processual	Tecnologia de AI utilizada para a construção de sistemas de raciocínio em tempo real que podem executar tarefas complexas em ambientes dinâmicos
random forest	floresta aleatória	Ensemble ML tecnologia para classificação, regressão e outras tarefas que operam através da construção e funcionamento de muitas árvores de decisão e, em seguida, ou a saída do modo da classe ou a previsão média das árvores individuais
reasoning technique	técnica de raciocínio	AI que gera conclusões a partir das informações disponíveis utilizando técnicas lógicas (Após ISO/IEC TR 29119-11)
recall	recall	Uma métrica de desempenho funcional ML usada para avaliar um classificador, que mede a proporção de positivos reais que foram previstos corretamente (após ISO/IEC TR 29119-11). Sinônimo: sensibilidade
receiver operating characteristic (ROC) curve	curva da característica operacional do receptor (ROC)	Uma trama gráfica que ilustra a capacidade de um classificador binário como seu limiar de discriminação é variada
regression	regressão	Um tipo de função ML que resulta em um valor numérico ou contínuo de saída para uma determinada entrada (após ISO/IEC TR 29119-11)
regression model	modelo de regressão	Um modelo ML cuja saída esperada para uma determinada entrada numérica é uma variável contínua (após ISO/IEC DIS 23053)
reinforcement learning	aprendizagem por reforço	A atividade de construir um modelo ML usando um processo de ensaio e recompensa para atingir um objetivo (Depois da ISO/IEC TR 29119-11)
reward function	função de recompensa	Uma função que define o sucesso do aprendizado do reforço
reward hacking	hacking de recompensas	A atividade realizada por um agente inteligente para maximizar sua função de recompensa, em detrimento do cumprimento do objetivo original (Após ISO/IEC TR 29119-11)
R-squared	R-quadrado	Uma medida estatística de quão próximos os pontos de dados estão da linha de regressão ajustada. Sinônimo: coeficiente de determinação
rule engine	motor de regras	Um conjunto de regras que determinam quais ações devem ocorrer quando certas condições são satisfeitas

Termo original	Termo	Definição
safety	segurança	A expectativa de que um sistema não conduza, sob condições definidas, a um estado em que a vida humana, a saúde, a propriedade ou o meio ambiente estejam em perigo (ISO/IEC/IEEE 12207)
sample bias	viés de amostragem	Um tipo de viés onde o conjunto de dados não é totalmente representativo do espaço de dados ao qual o ML é aplicado
search algorithm	algoritmo de busca	Um algoritmo que visita sistematicamente um subconjunto de todos os estados ou estruturas possíveis até que o estado ou estrutura objetivo seja alcançado (Após ISO/IEC TR 29119-11)
self-learning system	sistema de auto-aprendizado	Um sistema adaptativo que muda seu comportamento baseado na aprendizagem através de tentativa e erro (Depois da ISO/IEC TR 29119-11)
silhouette coefficient	coeficiente de perfil	Uma medida de agrupamento entre -1 e +1 com base na média das diferenças entre os grupos e intra-grupos. Sinônimo: pontuação em silhueta
AI-super	AI-super	Um sistema baseado em inteligência artificial que excede em muito as capacidades humanas
supervised learning	aprendizagem supervisionada	Treinamento de um modelo ML a partir de dados de entrada e seus rótulos correspondentes
support vector machine (SVM)	máquina de suporte vetorial (SVM)	Uma técnica ML na qual os pontos de dados são vistos como vetores no espaço multidimensional separados por um hiperplano
technological singularity	singularidade tecnológica	Um ponto no futuro quando os avanços tecnológicos não forem mais controláveis pelas pessoas (Depois da ISO/IEC TR 29119-11)
test oracle problem	problema do oráculo de teste	O desafio de determinar se um teste passou ou falhou para um determinado conjunto de entradas e estados de teste
training dataset	conjunto de dados de treinamento	Um conjunto de dados usado para treinar um modelo ML
transfer learning	Transferência de aprendizagem	Uma técnica para modificar um modelo ML pré-treinado para realizar uma tarefa relacionada diferente
transparency	transparência	O nível de visibilidade do algoritmo e dos dados utilizados pelo sistema baseado em AI (após ISO/IEC TR 29119-11)
true negative (TN)	verdadeiro negativo (VN)	Uma previsão na qual o modelo prevê corretamente a classe negativa
true positive (TP)	verdadeiro positivo (VP)	Uma previsão na qual o modelo prevê corretamente a classe positiva
underfitting	underfitting	A geração de um modelo ML que não reflete a tendência subjacente do conjunto de dados de treinamento, resultando em um modelo que encontra dificuldades para fazer previsões precisas (ISO/IEC TR 29119-11)
unsupervised learning	aprendizagem não supervisionada	Treinamento de um modelo ML a partir de dados de entrada usando um conjunto de dados não rotulados.
validation dataset	conjunto de dados de validação	Um conjunto de dados usado para avaliar um modelo ML treinado com o propósito de ajustar o modelo
von Neumann architecture	arquitetura von Neumann	Uma arquitetura de computador que consiste em cinco componentes principais: memória, uma unidade central de processamento, uma unidade de controle, entrada e saída
weight	peso	Uma variável interna de uma conexão entre neurônios em uma rede neural que afeta a forma como ela calcula seus resultados e que muda conforme a rede neural é treinada.