AiU Certificação em Teste (CTAI) Syllabus

Versão 1.0R 2019

Inteligência Artificial United

Nota sobre Direitos Autorais

Agradecemos aos principais autores:

- Vipul Kocher, Saurabh Bansal, Srinivas Padmanabhuni e Sonika Bengani

Agradecemos aos co-autores

- Rik Marselis e José M. Díaz Delgado

Agradecemos aos tradutores

Melissa Pontes com resenha de Márcia Araújo Coelho

Agradecemos ao comitê de revisão

Albert Tort, Amit Dang, Ana Laura Ochoa Moreno, Andreas Hetz, Ángel Rayo Acevedo, Aurelio Gandarillas, Baris Sarialioglu, Björn Lemke, Christine Green, Daniel Garcia Castillo, Dario Iván Rosas Miranda, Durga Mohapatra, Emilie Potin-Suau, Erik van Veenendaal, Girish Nuli, Girts Baltaisbrencis, Guino,

Márcia Araújo Coelho, Henostroza, Gustavo Márquez Sosa, Héctor Ruvalcaba, Javier Alejandro Chávez Crivelli, Jeff Nyman, José Antonio Rodríguez Gómez, Juan Pablo Rios Alvarez, Julie Gardiner, Kimmo Hakala, Kristine Corbus, Kyle Alexander Siemens, Lorena Parra Rubio, Maarten-Jan van Gool, Márton Görög, Maximiliano Mannise, Miaomiao Tang, Michaël Pilaeten, Michel Dussouchaud, Nora Alriyes, Paweł Noga, Petr Neugebauer, Ralf Pichler, Ram Shanmugam, Sammy Kolluru, Samuel Ouko, Santiago de Jesús González Medellín, Sebastia Małyska, Sergio Emanuel Cusmai, Serge Wolf, Shantel Yanique Stewart, Silvia Nane, Sunil Godse, Siva Prasad. B, Soren Wassard, Tariq King, Tetsu Nagata, Vikas Dhaka, Tom Van Ongeval, Werner Lieblang, Wim Decoutere, Yogesh Ahuja, Young jae Choi

Histórico de Revisões

Versão	Data	Observações
AiU A 2019	2 de março de 2019	Primeira versão beta
AiU B 2019	8 de maio de 2019	Segunda versão beta
AiU 1.0R 2019	16 de julho de 2019	Primeira publicação

Índice

Objetivos de Negócio

BO-1	Compreender as tendências atuais, aplicações industriais da Inteligência Artificial (IA) usando Aprendizagem de Máquina (Machine Learning - ML).
BO-2	Comparar diferentes implementações de algoritmos de aprendizagem de máquina (ML) para auxiliar na escolha do mais adequado.
BO-3	Avaliar modelos para aprendizagem supervisionada e não supervisionado.
BO-4	Projetar e executar casos de testes para sistemas de IA.
BO-5	Utilizar vários métodos para agregar transparência ao funcionamento do modelo.
BO-6	Definir estratégias para testar sistemas de IA.
BO-7	Entender onde aplicar IA em testes manuais e em automação de testes.
BO-8	Utilizar ferramentas de execução de testes baseadas em IA para automatizar testes.

Objetivos de Aprendizagem/Níveis Cognitivos de Conhecimento

Os objetivos de aprendizagem (Learning objectives, LO) são pequenas sentenças que descrevem brevemente o que você deve saber depois de estudar cada capítulo. Os LOs são definidos com base na taxonomia modificada de Bloom da seguinte forma:

- K1: Lembrar. Alguns verbos de ação são Lembrar, Recordar, Escolher, Definir, Localizar, Combinar, Relacionar, Selecionar
- K2: Entender. Alguns verbos de ação são são Sumarizar, Generalizar, Classificar, Comparar, Contrastar, Demonstrar, Interpretar, Refrasear
- K3: Aplicar. Alguns verbos de ação são Implementar, Executar, Usar, Aplicar

Para mais detalhes sobre a taxonomia de Bloom, por favor, consulte [BT1] e [BT2] na seção de Referências.

Objetivos Práticos

Objetivos práticos (Hands-on Objectives, HOs) são pequenas sentenças que descrevem o que é esperado que seja realizado ou executado para entender o aspecto prático do Aprendizado.

Os HOs se definem da seguinte maneira:

- HO-0: Demonstração ao vivo de um exercício ou vídeo gravado
- HO-1: Exercício guiado. Os alunos seguem uma sequência de passos realizados pelo facilitador
- HO-2: Exercícios com dicas — Exercícios para ser resolvidos pelos alunos utilizando dicas fornecidas pelo facilitador
- HO-3: Exercícios não guiados e sem dicas

Pré-requisitos

Obrigatórios

- Nenhum

Recomendados

- ISTQB® nível Fundamental ou equivalente
- Conhecimento básico de alguma linguagem de programação - Java/Python/R
- Conhecimento básico de estatística
- Alguma experiência de desenvolvimento ou testes de software

Capítulo 1 - Introdução à Inteligência Artificial

Palavras-chave
inteligência artificial, aprendizagem de máquina, aprendizagem supervisionada, aprendizagem não supervisionada, classificação supervisionada, regressão supervisionada, agrupamento não supervisionado, associação não supervisionada, aprendizagem por reforço (Reinforcement Learning, RL), aprendizado profundo (Deep Learning, DL), redes neurais (Neural Networks, NN), CRISP-DM, ciclo de vida ML.

LO-1.1.1	K2	Explicar a inteligência artificial (AI).
LO-1.1.2	K1	Recordar vários tipos de IA — Limitada, Geral e Super IA.
LO-1.2	K2	Explicar o aprendizado automático (ML) e o fato de que ML é um caminho de chegar à IA.
LO-1.2.1.1	K2	Explicar a aprendizagem de máquina (ML) supervisionado e a diferença entre Classificação Supervisionada e Regressão.
LO-1.2.1.2	K2	Explicar a aprendizagem de máquina (ML) não-supervisionado e comparar o Agrupamento (Clustering) e Associação não-supervisionados.
LO-1.2.1.3	K1	Recordar a definição e aplicações do Aprendizagem por Reforço (Reinforcement Learning, RL).
LO-1.3	K2	Explicar Aprendizado Profundo (DL) e tipos de Redes Neurais (NN).
LO-1.4.1	K2	Explicar várias etapas de CRISP-DM — um processo para o ciclo de vida da ML.

LO-1.4.2	K3	Aplicar os passos necessários para identificar o tipo de problema de ML apropriado.

1.1 Inteligência Artificial (IA)

Inteligência artificial é a inteligência adquirida por uma máquina para resolver problemas geralmente resolvidos por seres humanos.
IBM Watson, MS Cortana, Siri da Apple, veículos autônomos são alguns exemplos dentre um grande número de aplicações de IA já conhecidas.

1.1.1 Definição de Inteligência Artificial (IA)

LO-1.1.1	K2	Explicar a inteligência artificial (IA).

A IA é a arte de criar máquinas que executam funções que exigem inteligência quando executadas por pessoas. [KUR] Al está desempenhando um papel de destaque nos setores de saúde, manufatura, comércio eletrônico, varejo, mídia social, logística e outros setores industriais. Algumas das razões o aumento do uso da IA são o aumento do poder de processamento, a disponibilidade de dados e as novas tecnologias. Os casos de uso de IA abrangem desde o monitoramento de casas, a determinação de quais ações investir, a decisão de qual receita fazer e a escolha de seu parceiro de vida!

AI é um termo genérico que abrange a ciência de tornar as máquinas inteligentes, seja um robô, uma geladeira, uma televisão, um carro, um firmware ou um componente de software. O aprendizado automático (ML) é subconjunto da IA. ML e AI são frequentemente usados de forma intercambiável, mas não são a mesma coisa.

ML é explicado em maiores detalhes em 1.2 Aprendizado Automático (Machine Learning, ML)

1.1.2 Tipos de IA

LO-1.1.2	K1	Recordar vários tipos de IA — Limitada, Geral and Super IA.

IA pode ser amplamente categorizada como Limitada, Geral ou Super.

- **IA Limitada**: Máquinas programadas para realizar uma tarefa específica com contexto limitado. Por exemplo, máquinas de jogos, assistentes de voz toda a IA atual.
- **IA Geral**: Máquinas com habilidades cognitivas gerais são popularmente chamadas de casos de IA Fortes. Estas IAs podem raciocinar e entender seu entorno como os humanos o fazem, e agir de acordo. Por exemplo, raciocínio de senso comum. Atualmente, IA Geral ainda não é uma realidade e ninguém sabe quando ou se irá e tornar.
- **Super IA**: Máquinas que são capazes de replicar pensamentos, ideias e emoções humanas. É este super estado de inteligência onde máquinas se tornarão mais inteligentes e sábias que os humanos. Considerando o estado de desenvolvimento da IA, Super IA não se tornará realidade em um futuro próximo.

1.2 Aprendizagem de Máquina (Machine Learning, ML)

1.2.1 Definição de ML

LO-1.2.1	K2	Explicar aprendizado automático (ML) e o fato de que ML é uma das maneiras de alcançar a IA.

Arthur Samuel definiu ML como, “Um campo de estudo que dá aos computadores a capacidade de aprender sem serem explicitamente programados.” Os sistemas de ML aprendem e melhoram com a experiência, e, com o tempo, refinam um modelo que pode ser usado para predizer respostas à perguntas, com base na aprendizagem anterior [JB1].

Além de ML, IA usa conceitos de representação do conhecimento e raciocínio para lidar com diversos cenários. Noções de busca, programação e otimização entram no âmbito da IA, mas não necessariamente do ML.

Algumas tecnologias usadas para alcançar IA são:

- Aprendizagem de máquina (Machine Learning, ML)
- Processamento de Linguagem Natural (Natural Language Processing, NLP)
- Robótica
- Processamento de voz
- Visão Computacional

Existem algumas maneiras nas quais algoritmos de ML podem ser categorizados:

- Aprendizagem supervisonada
- Aprendizagem não Supervisionada
- Aprendizagem por Reforço (Reinforcement Learning, RL)

1.2.2 Aprendizagem Supervisionada- Classificação e Regressão

LO-1.2.2	K2	Explicar o aprendizagem de Máquina (ML) Supervisionada e a diferença entre Classificação e Regressão Supervisionados.

Aprendizagem Supervisionada: Neste tipo de aprendizado, o modelo aprende a partir de dados rotulados durante a fase de treinamento. O dado rotulado age como um treinador/supervisor para a função de mapeamento que interfere a relação entre os dados de entrada e o rótulo de saída durante o treinamento. Durante a fase de testes, a função de mapeamento é aplicada a um novo subconjunto de dados invisíveis para predizer a saída, que também está rotulada. O modelo é implantado quando o nível de precisão da saída é satisfatório.

Problemas resolvidos por Aprendizagem Supervisionada se dividem, ainda, em duas categorias:

Classificação: Quando o problema requer a classificação de um dado em uma das poucas classes pré-definidas, se utiliza aprendizagem supervisionada. Este tipo de modelo é usado quando a saída de dados é discreta ou quando a saída fica entre o número de classes alimentadas durante o treinamento. Reconhecimento de face ou detecção de objetos em imagens são exemplos de problemas que podem usar classificação. Outras aplicações de classificação são detecção de spam (spam ou não spam), o diagnóstico de uma doença com base em um Raio-X, correta identificação identificação de sinais de trânsito por parte de um sistema de assistência ao motorista, etc. Alguns dos algoritmos

comumente utilizados para classificação são regressão logística, o vizinho mais próximo (nearest neighbor, NN), máquina de vetores de suporte (Support vector machine, SVM) e as redes neurais (Neural Networks, NN).

Regressão: Quando os dados de saída são de natureza contínua ou numérica, por exemplo, predizendo a idade/peso de uma pessoa, predizendo o preço futuro de uma ação, etc., utiliza-se o aprendizado de regressão. O algoritmo mais utilizado para este tipo de problema é regressão linear, um algoritmo simples que explica a relação entre as entradas e saídas como uma equação linear. Outros algoritmos são regressão logística, máquina de vetores de suporte, regressão Lasso, etc.

1.2.3 Aprendizagem não Supervisionada — Agrupamento e Associação

LO-1.2.3	K2	Explicar a aprendizagem de máquina (ML) não supervisionada e comparar Agrupamento e Associação não Supervisionados.

Aprendizagem não supervisionada: Os dados limpos e rotulados não estão prontamente disponíveis a qualquer momento, assim, certos problemas precisam ser resolvidos sem um conjunto de dados de treinamento explicitamente rotulado. Este tipo de ML em que dados explicitamente rotulados não são fornecidos é chamado Aprendizagem não Supervisionada. O objetivo em tais problemas é aprender o padrão e estrutura dos dados de entrada sem nenhum rótulo associado.

A Aprendizagem não Supervisionada é ainda classificada nos dois métodos abaixo, de acordo com o tipo de saída:

Agrupamento (Clustering): Este modelo de Aprendizagem não Supervisionada agrupa os dados de entrada com base em características ou atributos similares. Os dados de entrada com atributos similares (não rotulados) são agrupados automaticamente (em um cluster). Portanto, os resultados obtidos grupos de dados de entrada com características similares. Por exemplo, segmentação de clientes que pode-se realizar por análise de mercado.

Associação: A Mineração de Regra de Associação encontra relações ou dependências interessantes entre atributos de dados. A descoberta de associações interessantes proporciona uma fonte de informações geralmente usadas para tomada de decisão. Por exemplo, análise de dados de carrinho de compras, sistema de recomendação de produtos baseado em aprendizados

derivados do comportamento de compra do cliente são bons exemplos de modelagem baseada em regras de associação.

1.2.4 Aprendizagem por Reforço

LO-1.2.4	K1	Recordar da definição e aplicações do Aprendizagem por Reforço (Reinforcement Learning, RL).

É um tipo de ML em que um agente (algoritmo) aprende interagindo com o ambiente de maneira iterativa e, assim, aprende com a experiência. O agente é recompensado quando toma uma decisão correta e penalizado quando toma uma decisão errada. Essa recompensa e o aprendizado baseado em penalidade são, portanto, definidos como "aprendizagem por reforço" (RL). Configurar o ambiente adequado, escolher a estratégia certa para o agente atingir o objetivo desejado e projetar uma função de recompensa, são alguns dos principais desafios na implementação da RL. Robótica, veículos autônomos e Chatbots são exemplos de aplicativos que podem usar RL.

1.3 Aprendizagem Profunda

1.3.1 Aprendizagem Profunda e os Tipos de Redes Neurais

LO-1.3.1	K2	Explicar a Aprendizagem Profunda (Deep Learning, DL) e os tipos de Redes Neurais (Neural Networks, NN).

Aprendizagem Profunda (Deep Learning, DL) refere-se aos sistemas que adquirem experiência com conjuntos de dados massivos. DL usa redes neurais artificiais (Artificial Neural Networks, ANN) para analisar grandes conjuntos de dados, por exemplo veículos autônomos, processamento de texto grande e aplicativos de visão computacional, entre outros. [AG1]

DL é um subconjunto da ML e ML é um subconjunto da AI. DL usa os mesmos tipos de aprendizagem (Supervisionada, Não Supervisionada e Aprendizagem por Reforço) que ML.

Redes Neurais Artificiais: As Redes Neurais Artificiais (Artificial Neural Networks, ANN) são inspiradas na arquitetura do cérebro humano. Os 'Neurônios', como unidades básicas das ANN, agem sob um estímulo de entrada

e produzem um sinal de saída. A entrada passa por camadas de funções de ativação para gerar a saída. Estas camadas formam uma rede em forma de malha.

Cada ANN tem pelo menos duas camadas — camadas de entradas e de saídas. Todas as camadas entre estas duas são chamadas camadas ocultas. Alguns dos vários tipos de redes neurais são:

Rede Neural Profunda: Rede Neural Profunda (Deep Neural Network, DNN) é uma ANN com duas ou mais camadas ocultas.

Rede Neural Convolucional: Rede Neural Convolucional (Convolutional Neural Network, CNN) é uma ANN que emerge do estudo do córtex visual do cérebro, e tem sido usada em reconhecimento de imagem desde a década de 1980. Diferente de outras redes neurais, CNNs trabalham diretamente sem serializar/ vetorizar uma imagem de entrada and extraindo características mediante filtros. As CNN proporcionam serviços de busca de imagens, veículos autônomos, sistemas de classificação automática de vídeos e mais.

Rede Neural Recorrente: Estas ANNs podem predizer o futuro de problemas de séries temporais. Elas seguem uma abordagem sequencial de uma série de dados de entrada de tamanhos arbitrários no lugar de entradas de tamanho fixo, como em outras redes neurais. Cada entrada e saída são independentes de todas as outras camadas. A retroalimentação da camada de saída é alimentado pela mesma rede recorrentemente, até que se alcance um nível de confiança adequado. RNNs podem analisar dados de séries temporais como preços de estoque, e informar quando comprar ou vender. Em veículos autônomos, podem antecipar trajetórias e ajudar a evitar acidentes.

1.4 Etapas do Processo de ML

Um projeto típico de ML segue todas as etapas do processo padrão entre indústrias para mineração de dados (CRISP-DM) — um padrão da indústria, e um framework flexível.

1.4.1 Etapas do processo de ML — Processo CRISP-DM

LO-1.4.1	K2	Explicar várias etapas de CRISP-DM — um processo para o ciclo de vida de ML.

CRISP-DM tem tradicionalmente seis etapas no ciclo de vida da mineração de dados. Foi personalizado para cumprir com os requisitos dos projetos ML, adicionando uma sétima etapa.

Os sete estágios do framework CRISP-DM para ML são: [DSC1] [SMU]

1. **Aquisição de Dados**: Reunir dados de fontes externas e internas (por exemplo: bancos de dados, arquivos CSV, mídias sociais, etc.)

2. **Preparação de Dados**: Limpar dados brutos e reformular. Novos atributos são criados com a engenharia de recursos, um processo para criar novas variáveis a partir dos dados existentes. Redução de dimensionalidade, imputação de dados, tratamento de valor nulo para os valores ausentes, etc., são alguns dos métodos envolvidos na preparação de dados.

3. **Modelagem**: Selecionar o modelo ou algoritmo, dividir os dados disponíveis em conjunto de treino e conjunto de testes. Modelos são obtidos executando algoritmos ML no conjunto de dados de treino. Usar o conjunto de dados de testes para avaliar e aprimorar o desempenho do modelo até a obtenção de um desempenho satisfatório.

4. **Avaliação**: Avaliar o modelo através de várias métricas (discutidas em 3.2 Métricas) e estabelecer a linha de base antes da entrega final.

5. **Implantação (Deploy)**: Realizar implantação e monitoramento do modelo em relação às métricas de produção.

6. **Operações**: Realizar manutenções e operações regulares. Regenerar e refinar o modelo quando as métricas caírem abaixo de um certo limite.

7. **Otimização**: A solução implantada deve ser substituída em caso de desvio de conceito (ver 6.1.3 Risco de Desvio de Conceito (Concept Drift, CD), à medida que novos algoritmos fiquem disponíveis, ou devido a falhas importantes de desempenho.

Os passos 1-4 podem ser classificados como partes da fase **offline**, cujo resultado é o modelo treinado. Os passos 5-6 são partes da fase **online**, onde o modelo treinado na fase offline é integrado com o resto do sistema e é implantado em um ambiente de produção. O passo de otimização envolve a re-execução dos passos 1-6.

1.4.2 Etapas para Identificação do Tipo de Problema de ML

LO-1.4.2	K3	Aplicar os passos envolvidos na identificação do tipo de problema de ML apropriado.

É importante entender os problemas que estamos tentando resolver e o tipo de aprendizagem necessário para resolvê-los. Uma maneira de identificar o tipo de problema de ML é discutido a seguir:

1. Se o problema envolve a noção de múltiplos estados, e envolve mudanças para cada estado, então explore RL.
2. Se existe uma variável de saída, trata-se de aprendizagem supervisionada.
   2.1. No caso em que a saída é discreta e categórica, trata-se de um problema de classificação.
   2.2. Em caso de saída numérica e contínua, trata-se de um problema de regressão.
3. Se a saída não for provida no conjunto de dados fornecidos, então explore a aprendizagem não supervisionada.
   3.1. Se o problema consiste em agrupar dados similares, trata-se de um problema de agrupamento (clustering).
   3.2. Se o problema consiste em encontrar dados co-existentes, então aplique mineração de regras de associação
   3.3. Se os dados brutos não estão estruturados, a extração automática de funções pode ser explorada com algoritmos de aprendizado profundo.

O pré-requisito para os passos acima é que devem existir dados suficientes disponíveis para analisar o tipo de problema de ML adequado.

Capítulo 2 - Visão Geral de Testes de Sistemas de Inteligência Artificial

Palavras-chave: fase offline, fase online, ISO25010

LO-2.1.1	K2	Explicar os testes realizados em sistemas IA-ML na fase de treinamento (offline) e fase de integração pós-treinamento (online).
LO-2.2.1	K2	Comparar os testes de sistemas IA com sistemas que não são IA.
LO-2.3.1	K1	Recordar as características de qualidade ISO 25010, especialmente adequação funcional, eficiência de desempenho, confiabilidade, capacidade de manutenção e outros parâmetros, como complexidade, escalabilidade e aprendizado contínuo, como parâmetros a serem utilizados para a avaliação do modelo treinado.
LO-2.3.2	K1	Recordar as características de qualidade específicas dos testes de IA, além daquelas mencionadas na ISO25010 - comportamento inteligente, moralidade e personalidade.

2.1 Fases de Testes de IA

2.1.1 Testes de Sistemas de IA Offline e Online

LO-2.1.1	K2	Explicar os testes realizados em sistemas de IA-ML na fase de treinamento (offline) e na fase de integração pós-treinamento (online).

O ciclo de vida de ML, conforme descrito nos estágios 1.4 do processo ML, pode ser dividido em duas fases: offline e online. Os diferentes tipos de testes realizados nestas fases são:

Testes da fase Offline: Nesta fase, o modelo treinado é testado. Várias métricas são usadas como parâmetros de avaliação de um modelo treinado para verificar até que ponto ele alcançou os objetivos. Existem parâmetros diferentes para o

aprendizado supervisionado e não supervisionado. Tipicamente, o modelo é testado quanto ao comportamento funcional, mas as características não funcionais não são testadas. Dado que o modelo é implantado em um ambiente diferente do ambiente de treinamento, fazer testes de desempenho do modelo nesta fase não faz muito sentido. O tempo de treinamento de um modelo é um parâmetro que é avaliado como não-funcional. Para os modelos que precisam ser re-treinados frequentemente, este pode ser um parâmetro importante. Testes Offline são tratados no Capítulo 3 - Testes Offline de Sistemas de IA.

Outro aspecto importante dos testes offline é se é possível explicar o comportamento do modelo. Existem vários métodos e algoritmos que podem ser usados para isso. Este aspecto dos testes são descritos no Capítulo 5 - IA Explicável.

Fase de testes Online: Nesta fase, a integração do modelo treinado com o resto do sistema, incluindo todos os outros componentes IA e não-IA, é testado. Podem ser realizados testes funcionais e não-funcionais, como desempenho.

Dado que as entradas no sistema pode ser não-textuais, não-estruturadas, bem como suporte de automação para alguns dos testes relacionados ao ML part, pode ser limitado, com base na ferramenta que está sendo utilizada.

Testes Online é abordado no Capítulo 4 - Testes Online de Sistemas de IA.

2.2 Testes de IA vs. Não-IA

2.2.1 Testes de Sistemas de IA vs. Sistemas Tradicionais (não-IA)

LO-2.2.1	K2	Comparar testes de sistemas IA com sistemas não-IA.

- Oráculos de teste para sistemas de IA não estão facilmente disponíveis
- Diferente dos testes tradicionais, os resultados dos testes de sistemas de IA são não-determinísticos. As saídas de um sistema de IA apresentam probabilidades, portanto, os resultados de um caso de teste são também probabilidades, ao invés de passar/falhar.
- Diferentemente dos testes tradicionais, os resultados dos testes de sistemas de IA são não determinísticos. A saída de um sistema de IA tem probabilidades, portanto, o resultado de um caso de teste também é probabilidades, e não uma aprovação / reprovação definida.

- A lógica dos sistemas de IA é gerada com base nos dados usados para treinar o modo. Essa lógica não está disponível para avaliação, principalmente redes neurais. Isso dificulta a compreensão de porque uma saída específica foi produzida. Uma resposta correta ou desejada não garante um funcionamento correto.
- Testes na fase offline é uma fase adicional que requer habilidades e técnicas especializadas, como aquelas relacionadas com a limpeza e pré-processamento de dados, e testes do modelo treinado.
- Testes em uma fase online requerem um conhecimento profundo de como sistemas de IA funcionam. A integração de sistemas de IA com outros sistemas de IA e não-IA aumentam a necessidade de técnicas de projeto de testes diferentes.
- Similar a sistemas não-IA, testes funcionais e testes não-funcionais precisam ser realizados em sistemas IA.
- A fase de testes online pode ser realizada como sistema de caixa-preta normal e teste de integração de sistemas sem se preocupar se há um ou mais componentes de IA no mix.

2.3 Características de Qualidade da IA

2.3.1 Características de Qualidade para Avaliação de Sistemas de IA

LO-2.3.1	K1	Recordar as características de qualidade ISO 25010, especialmente adequação funcional, eficiência de desempenho, confiabilidade, capacidade de manutenção e outros parâmetros, como complexidade, escalabilidade e aprendizado contínuo, como parâmetros a serem utilizados para a avaliação do modelo treinado.

As características de qualidade de sistemas de IA podem ser avaliados com base em uma combinação de características do ISO 25010 e outras características de qualidade. Algumas das características importantes das perspectivas dos testes de IA são

- **Adequação funcional** - correção funcional, completude e adequação.
- **Confiabilidade**

- Disponibilidade - Disponibilidade do sistemas durante operações normais.
- Tolerância à falhas - Capacidade do sistema de lidar com dados corrompidos, incompletos ou irrelevantes, sem quebrar.

- **Eficiência de Desempenho**
  - Comportamento temporal - com que rapidez o sistema responde a demandas feitas a ele.
  - Utilização de recursos - quais e quantos recursos são usados pelo sistema para desempenhar uma função.

- **Manutenção**
  - Observabilidade - O grau de eficácia e eficiência com o qual é possível avaliar o impacto em um produto ou sistema de uma alteração pretendida em uma ou mais de suas partes, ou diagnosticar um produto por deficiências ou causas de falhas ou identificar partes para ser modificado. No caso da IA, a analisabilidade também se refere à capacidade de entender porque o sistema tomou a decisão que tomou. Idealmente, a explicabilidade (ou examinabilidade) deveria ser uma característica de qualidade separada para a ISO 25010 para sistemas de IA.

  - Testabilidade - O grau com que um componente IA suporta testes em um dado contexto. Quanto maior o grau/a testabilidade, mais fácil encontrar bugs por meio de testes.

Outros parâmetros importantes são:

- **Complexidade** — Complexidade temporal e espacial;
- **Escalabilidade** — A habilidade do sistema de gerenciar uma maior carga mediante adição de recursos adicionais;
- **Aprendizado contínuo** — capacidade do sistema de aprender continuamente a partir de novos dados, especialmente de dados do ambiente em tempo real.

2.3.2 Características de Qualidade Extendidas Específicas para IA

LO-2.3.2	K1	Recordar as características de qualidade específicas dos testes de IA, além daquelas mencionadas na ISO25010 - comportamento inteligente, moralidade e personalidade.

O uso de IA vem requerendo uma extensão das características qualidade padrão.

Uma máquina inteligente também deve ter as seguintes características de qualidade, além das mencionadas na ISO25010. [TDA]

- Comportamento inteligente — Comportamento inteligente é a habilidade de compreender ou entender. É basicamente a combinação de raciocínio, memória, imaginação e julgamento; cada uma dessas faculdades dependem das outras. Inteligência é uma combinação de habilidades cognitivas e conhecimento que se fazem evidentes por comportamentos adaptativos. As sub-características são: Habilidade de aprender, Improvisação, Transparência nas escolhas, Colaboração e Interação Natural.
- Moralidade — Moralidade, em relação à IA, trata-se de princípios relativos à distinção entre certo e errado ou bom e mau comportamentos. As sub-características são: Ética, Privacidade e Amigabilidade Humana.
- Personalidade — Personalidade é a combinação de características ou qualidades que formam o caráter distinto de um indivíduo. As sub-características são: Estado de espírito, Empatia, Humor e Carisma.

Capítulo 3 - Testes Offline de Sistemas de IA

Palavras-chave: dados estruturados, dados não-estruturados, dimensionalidade, métricas, validação-cruzada, subajuste (underfitting), sobreajuste (overfitting), analítica (analytics).

LO-3.1.1	K1	Recordar das etapas envolvidas na preparação e pré-processamento de dados e na necessidade de limpeza exaustiva dos dados.
LO-3.1.2	K3	Aplicar a leitura e manipulação de dados, incluindo etapas de filtragem para dados estruturados.
HO-3.1.2	H2	Utilizar código (linguagem de sua escolha) para ler dados de várias fontes de dados, como um arquivo do Excel, um arquivo CSV ou uma base de dados, e limpe um determinado conjunto de dados descartando as colunas menos importantes, adicionando colunas e limpando os dados, em caso de dados estruturados de texto (manipulação de dados).
LO-3.1.3	K2	Resumir várias etapas de pré-processamento de dados para dados não estruturados (imagens).
LO-3.1.4	K2	Resumir várias etapas de pré-processamento de dados para dados não estruturados (texto).
HO-3.1.4	H1	Realizar passos de pré-processamento de dados em dados não-estruturados (text).
LO-3.1.5	K3	Aplicar vários métodos de imputação de dados a diferentes tipos de problemas.
HO-3.1.5	H3	Preencher valores de dados ausentes para um determinado conjunto de dados usando a média, moda e o método KNN.
LO-3.1.6	K1	Listar vários tipos de gráficos (uni, bi e multivariado) usadas para visualização de dados.
LO-3.1.7	K2	Explicar o conceito de *outliers* (valores discrepantes) e várias razões para sua presença.
LO-3.1.8	K3	Aplicar o método de plotagem de caixa visual para

		determinar *outliers* de um determinado conjunto de dados.
HO-3.1.8	H2	Desenhar um gráfico de caixa para o conjunto de dados fornecido para identificar *outliers*.
LO-3.1.9	K3	Aplicar técnicas de redução de dimensionalidade - eliminação irrelevante de recursos, análise de componentes principais (principal component analysis, PCA).
HO-3.1.9	H3	Executar a eliminação de recursos irrelevantes, PCA, para redução de dimensionalidade em um conjunto de dados.
LO-3.2.1	K2	Explicar o papel das métricas em sistemas de ML.
LO-3.2.2	K1	Listar vários parâmetros de avaliação de modelos para aprendizado supervisionado e não-supervisionado
LO-3.2.3	K2	Comparar inércia e a pontuação de Rand ajustada como métrica para agrupamento não-supervisionado.
HO-3.2.3	H3	Aplicar a inércia - soma dos quadrados dentro do cluster (within-cluster-sum-of-squares, WCSS) - e usar o método elbow para determinar o número ideal de clusters.
LO-3.2.4	K2	Compare as métricas de apoio, confiança e elevação para o mineração de regras de associação não-supervisionada.
HO-3.2.4	H3	Compare as métricas de apoio, confiança e elevação para o mineração de regras de associação não-supervisionada.
LO-3.2.5	K2	Explicar a matriz de confusão
LO-3.2.6	K3	Aplicar a fórmula para calcular acurácia, precisão, memória, especificidade e pontuação F1 para classificação supervisionada e compará-las.
HO-3.2.6	H3	Calcular acurácia, precisão, memória, especificidade e pontuação F1 para classificação supervisionada.
LO-3.2.7	K3	Aplicar a fórmula para calcular várias métricas de erro (erro quadrático médio - root mean square error (RMSE) e R-quadrado) para regressão supervisionada.
HO-3.2.7	H2	Calcular os erros RMSE e R-quadrado para regressão supervisionada.
LO-3.3.1	K1	Definir o método de validação dos modelos dividindo o conjunto de dados disponíveis em conjunto de dados para

		treinamento, validação e testes.
LO-3.3.2	K2	Resumir o conceito de subajuste e sobreajuste e compensação de desvio (bias-variance tradeoff).
HO-3.3.2	H0	Demonstrar subajuste e sobreajuste para mostrar a bias-variance tradeoff.
LO-3.3.3	K3	Aplicar os métodos de teste de divisão, validação cruzada K-fold, *bootstrap* e validação cruzada de exclusão única (leave-one-out cross-validation).
HO-3.3.3	H1	Aplicar o método de validação cruzada K-fold em um algoritmo e comparar os vários resultados distintos.
LO-3.4.1	K1	Recordar-se das características dos quatro tipos de análise - descritiva, exploratória, preditiva, prescritiva - e seu uso na ML.

3.1 Preparação e Pré-Processamento de Dados

3.1.1 Passo a Passo para Preparação e Pré-Processamento de Dados

LO-3.1.1	K1	Recordar os passos necessários na preparação e pré-processamento de dados e a necessidade de uma limpeza completa dos dados.

Os dados de entrada podem ser em forma de tabelas de banco de dados, arquivos CSV (comma separated values), ou dados não-estruturados como imagens, áudios, vídeos ou textos corridos. Os dados necessários são obtidos de várias fontes, internas e externas.
Após a aquisição dos dados, estes precisam de uma limpeza completa e pré-processamento, antes de alimentarem algoritmos para treinamento e testes.
Os seguintes passos são realizados para preparação e pré-processamento de dados:

- Manipulação de Dados
- Filtragem de dados
- Atividades de pré-processamento de dados incluem
    - Imputação de Dados, isto é, tratamento de valores que faltam
    - Visualização de dados para obtenção de uma visão geral e tratamento de Anomalias e Valores Discrepantes (Outliers)
    - Análise de Correlação e Redução de Dimensões

Etapas específicas de pré-processamento de dados em diversos formatos (por exemplo, imagem, texto) precisam ser executadas para tornar o formato de dados adequado para treinamento. Quando o volume de dados é muito grande, é preciso fazer redução de dados sem perder as informações. Para dados estruturados ausentes, talvez seja necessário preencher os valores dos dados. Todas essas etapas de pré-processamento de dados são necessárias para obter a precisão desejada e uma melhor previsibilidade nos modelos

3.1.2 Preparação de Dados

LO-3.1.2	K3	Aplicar a leitura e manipulação de dados, incluindo as etapas de filtragem para dados estruturados.
HO-3.1.2	H1	Utilize código (linguagem de sua escolha) para ler dados de várias fontes, como um arquivo Excel, um arquivo CSV ou uma base de dados e limpar um determinado conjunto de dados descartando a(s) coluna(s) menos importante(s), adicionando colunas e limpando os dados para obter dados de texto estruturados (manipulação de dados).

Preparação de Dados inclui:

1. **Manipulação de Dados**: Mudar a estrutura de um dado, por exemplo, adicionando uma nova coluna, deletando algumas linhas, etc.
2. **Filtragem de Dados**: Reduzir o tamanho de dados estruturados (tabela/matriz) e não-estruturados (imagem/texto) para aprimorar a qualidade.

3.1.3 Processamento de Dados Não Estruturados (Imagens)

LO-3.1.3	K2	Resumir vários passos de pré-processamento de dados para dados não-estruturados (imagens).

Remover o ruído da imagem e redimensioná-la são operações comuns realizadas em imagens para projetar algoritmos de visão computacional. [UDI]

3.1.4 Processamento de Dados Não-estruturados (Texto)

LO-3.1.4	K2	Resumir vários passos de pré-processamento de dados para dados não-estruturados (texto).

HO-3.1.4	H1	Realizar etapas de pré-processamento de dados em dados não-estruturados (texto).

O pré-processamento de dados de texto pode ser realizado em várias etapas de alterações sintáticas, dependendo da necessidade do modelo de ML. Por exemplo, remoção de numerais, conversão de maiúsculas para minúsculas, remoção de pontuações, espaços em branco, remoção de palavras de parada, execução de stemização/lematização, etc. [UDT]

3.1.5 Imputação de Dados

LO-3.1.5	K3	Aplicar vários métodos de imputação de dados a diferentes tipos de problemas.
HO-3.1.5	H3	Preencher valores de dados ausentes para um dado conjunto de dados usando média, moda e método KNN.

Dados coletados de campo podem ter valores nulos ou ausentes, requerendo o preenchimento de valores nulos com alguns valores apropriados. Nulos ou valores ausentes podem ser inseridos com medidas de tendência central (médio, mediana ou moda), método K vizinhos mais próximos [DI1], ou uma abordagem baseada em regressão.

3.1.6 Visualização de Dados

LO-3.1.6	K1	Listar vários tipos de gráficos (uni, bi e multivariado) utilizados para visualização dos dados.

A visualização dos dados ajuda a compreender sua estrutura e a relação entre seus atributos, o que não é possível simplesmente observando os números ou o texto fornecido. Existem vários tipos de visualizações. Os métodos de visualização mais comumente usados são: gráficos de linhas para valores contínuos, histogramas para valores discretos, gráficos de caixas, gráficos de barras, gráficos de pizza, etc. Eles fornecem informações significativas sobre os dados disponíveis desde o início.

Tipos de gráficos:
Univariado: A forma mais simples de análise, em que o dados em análise são variáveis simples. Por exemplo, a idade da população ou peso de uma população, etc. São analisados individualmente e suas relações não são consideradas. Gráficos em linha, histogramas, distribuição de frequência,

gráfico de barras e gráfico de caixa (box plots) são utilizados para análise de dados univariados.

Bivariado: Esse tipo de análise é realizada para encontrar o relacionamento entre duas variáveis no conjunto de dados fornecido. Plotar uma variável contra outra em um plano XY ajuda a encontrar o relacionamento em primeira mão entre duas variáveis. Por exemplo, a relação entre a idade e o peso da população em consideração. Para esse tipo de análise, podemos usar gráficos de dispersão ou correlogramas.

Multivariado: Análise de três ou mais variáveis. Gráficos de malha e gráficos em 3D são algumas das maneiras pelas quais pode-se visualizar dados multivariados e descobrir relações entre eles.

3.1.7 Detecção de Anomalia/Discrepâncias

LO-3.1.7	K2	Explicar o conceito de dados discrepantes (outliers) e as diversas razões para a presença deles.

Observações que não seguem o padrão esperado para um dado conjunto de dados caem na categoria de dados discrepantes, por exemplo, detecção de fraudes ou ataques de hackers.
Se as discrepâncias (outliers) não são frequentes e não contribuem para eventos críticos, elas podem ser removidas. Mas na prática, elas devem ser investigadas a fundo antes de serem removidas do conjunto de dados.

Causas de Outliers

- Erro: Neste caso, outliers são resultados de erros de medição, entrada de dados e amostragem. Por exemplo: dados de temperatura registrados em graus Celsius para a maioria dos registros, mas para alguns outros, em Fahrenheit, por erro.
- Natural: Alguns outliers podem acontecer em situações naturais. Por exemplo: se uma inundação acontece apenas uma vez em 100 anos, isso é considerado um outlier natural.
- Intencional: Outliers fictícios feitos para validar modelos de detecção. Por exemplo: registros artificiais, cultivados em laboratório, usados para testar cenários corner.

3.1.8 Técnicas de Detecção de Discrepâncias (Outliers)

LO-3.1.8	K3	Aplicar o método de plotagem de caixa visual para

		determinar outliers de um determinado conjunto de dados.
HO-3.1.8	H2	Desenhar um diagrama caixa para o conjunto de dados fornecido para identificar outliers.

Ao calcular várias estatísticas com o conjunto de dados fornecido, geralmente é útil usar um diagrama de caixa para ver a distribuição dos dados. Os diagramas de caixa ajudam a determinar a posição atípica do conjunto de dados fornecido.

As extremidades da caixa são os quartis superior e inferior. Os bigodes são as duas linhas fora da caixa que se estendem aos limites de dados mais alto e mais baixo, além dos quais os pontos de dados são considerados outliers.

3.1.9 Redução da Dimensionalidade

LO-3.1.9	K3	Aplicar técnicas de redução de dimensionalidade - eliminação de caracteristicas irrelevantes, análise de componentes principais (*principal component análisis*, PCA).
HO-3.1.9	H3	Executar a eliminação de características irrelevantes, PCA, para redução de dimensionalidade em um conjunto de dados.

Os problemas de ML geralmente têm um grande número de características de entrada, mas nem todos contribuem para a classificação ou saída da regressão. Quanto maior o número de características, mais difícil é visualizar o conjunto de treinamento. A técnica de redução do número de variáveis em consideração é denominada redução de dimensionalidade.
A necessidade de trabalhar com menos dimensões do que as dimensões originais decorre de:

- Fatores de custo e velocidade
- Requisito de memória
- Evitar redundância
- Identificar a parte mais relevante dos dados para para processamento posterior

Métodos de Redução de Dimensionalidade

- Eliminação de características irrelevantes é a remoção de colunas que não contribuem para a variável de saída:

- Análise univariada ajuda a remover as colunas cujos valores não mudam nas linhas. Por exemplo, considere os dados da transação de vendas de uma única loja de varejo, então, colunas como nome e local da loja não mudam nas linhas e devem ser removidas.
- Dados de densidade muito baixa podem ser facilmente descartados, após investigação, para economizar espaço. Por exemplo, as colunas baseadas no ID da linha do banco de dados têm um valor exclusivo para cada linha e devem ser removidas.

- A análise bivariada remove um dentre o par de atributos de entrada altamente correlacionados (portanto, também é conhecido como análise de correlação), por exemplo, nos dados de inventário de uma loja, 'preço do item' e 'quantidade do item' são atributos altamente correlacionados.
- Análise de componentes principais: o PCA (Principal component analysis) reduz extensivamente as dimensões de conjuntos de dados maiores, mas preserva as informações ao máximo. Deduz um novo conjunto de variáveis independentes (chamadas componentes principais) e as coloca em ordem decrescente de importância. O número necessário de componentes principais mais importantes (portanto, número reduzido de variáveis ou dimensões) pode ser selecionado ainda preservando a informação máxima possível do conjunto de dados original.

3.2 Métricas

3.2.1 Métricas

LO-3.2.1	K2	Explicar o papel das métricas nos sistemas de ML.

Métricas são parâmetros de avaliação para um modelo treinado e podem ser vistas como a medida de quanto o modelo treinado fornece resultados precisos e confiáveis. Para um dado tipo de algoritmo, métricas podem ser usadas para comparar modelos treinados entre si.

3.2.2 Métricas para Aprendizado Supervisionado e Não Supervisionado

LO-3.2.2	K1	Enumerar vários modelos de avaliação de parâmetros para aprendizado supervisionado e não supervisionado.

Os objetivos do problema para modelos de aprendizado supervisionado e não supervisionado são diferentes. Assim, as métricas para avaliar os modelos são diferentes.

Tipo de Aprendizado	Tipo de Modelo	Métricas utilizadas
Não Supervisionado	Agrupamento (Clustering)	• Inércia • Pontuação Rand ajustada
	Associação	• Suporte • Confiança • Levantamento
Supervisionado	Classificação	• Acurácia • Precisão • Memória/Sensibilidade • Especificidade • Pontuação F1
	Regressão	• Erro quadrático médio (Root-mean-square-error, RMSE) • Erro R-quadrado (R-square error)

3.2.3 Inércia e Pontuação Rand Ajustada

LO-3.2.3	K2	Comparar a inércia e a pontuação de Rand ajustada como métricas para *clustering* não supervisionado.
HO-3.2.3	H3	Apply inertia — within-cluster-sum-of-squares (WCSS) — and use the elbow method to determine the optimum number of clusters. Aplicar a inércia - soma dos quadrados dentro do *cluster* (WCSS) - e usar o método elbow para determinar o número ideal de *clusters*.

Para um modelo baseado em cluster não supervisionado, inércia ou soma dos quadrados dentro do cluster (within-cluster-sum-of-squares, WCSS) é a dispersão média de um cluster através de todos clusters descobertos. O menor valor de inércia significa melhor cluster, pois significa que os pontos de dados em um cluster estão mais próximos um do outro. O tamanho do cluster (e,

portanto, o valor de Inércia) diminuirá naturalmente à medida que o número de clusters aumentar. No entanto, a inércia para de diminuir significativamente além de um certo número de clusters; esse ponto mostra o valor ideal para a inércia e o número de clusters para um determinado conjunto de dados - esse método é conhecido como método elbow.

Quando os valores reais dos rótulos estão disponíveis para todos os pontos de dados, a **pontuação de Rand ajustada** é preferida à inércia. É uma medida de similaridade entre as atribuições de cluster (pelo modelo) e as classes separadas reais.

3.2.4 Métricas de Suporte, Confiança e Levantamento

LO-3.2.4	K3	Aplicar a fórmula para calcular as métricas de suporte, confiança e levantamento para mineração de regras de associação não supervisionada.
HO-3.2.4	H3	Calcular métricas de suporte, confiança e levantamento para mineração de regras de associação não supervisionada.

O Suporte para um conjunto de elementos mede a frequência com que ele aparece nas transações. Por exemplo, se o item "pão" estiver presente em 7 de um total de 10 transações em uma loja de varejo, seu suporte será de 70%.

Confiança mede a probabilidade de um conjunto de dados Y aparecer, dado que X apareceu.

Lift é usado para eliminar cenários em que dois conjuntos de itens ocorrem juntos com muita frequência (portanto, o valor da métrica de confiança será alto). No entanto, o conjunto de dois itens pode não ter nenhuma interdependência. Além disso, essa métrica pode revelar se mais ocorrência do conjunto de itens X significa mais ou menos ocorrência do conjunto de itens Y (ou seja, associação positiva ou negativa entre X e Y).

3.2.5 Matriz de Confusão

LO-3.2.5	K2	Explicar a matriz de confusão

As métricas de classificação supervisionada são calculadas usando uma matriz de confusão composta pelas contagens de verdadeiros positivos, falsos positivos, verdadeiros negativos, falsos negativos.

Matrix de Confusão	Alvo Positive	Alvo Positivo
Modelo Positivo	Verdadeiro Positivo VP	Falso Positivo FP
Modelo Negativo	Falso Negativo FN	Verdadeiro Negativo VN

3.2.6 Acurácia, Precisão, Memória, Especificidade e Pontuação de F1

LO-3.2.6	K3	Aplicar a fórmula para calcular métricas de acurácia, precisão, memória, especificidade e pontuação F1 para classificação supervisionada e compará-las.
HO-3.2.6	H3	Calcular acurácia, precisão, memória, especificidade e pontuação F1 para classificação supervisionada.

A **Acurácia** de um modelo mostra qual porcentagem ou fração do total de classificações foi feita com precisão pelo modelo treinado em um conjunto de dados de teste. Essa métrica se torna uma má escolha se uma classe de dados dominar sobre as outras.
Acurácia = (VP + VN) / (VP +NN + FP +FN).

Precisão mede com que precisão o modelo classifica os verdadeiros positivos.
Precisão = VP/ (VP+FP).

Recall mede até que ponto o modelo falhou ou errou ao detectar os pontos positivos.
Recall = VP/ (VP+FN).

Para minimizar os falsos positivos, é necessária alta precisão, enquanto que para minimizar os falsos negativos, o recall deve ser alto.

Como a precisão, mas oposta à recall, a **especificidade** mede com que precisão o modelo classifica os verdadeiros negativos.
Especificidade = VN / (VN + FP)

A pontuação F1 é calculada como a média harmônica de precisão e recall. Uma pontuação F1 baixa representa a má qualidade do modelo na detecção de

positivos. A pontuação F1 terá um valor entre 0 e 1. Perto de 1 significa que há boa qualidade e nenhum dado falso atrapalha o resultado.

3.2.7 RMSE e R-Quadrado

LO-3.2.7	K3	Aplicar a fórmula para calcular várias métricas (root mean square error, RMSE) e R-quadrado para a regressão supervisionada.
HO-3.2.7	H2	Calcular os erros RMSE and R-quadrado para a regressão supervisionada.

As métricas do modelo de regressão supervisionada representam quão bem a linha de regressão se ajusta aos pontos de dados reais.

RMSE (erro médio quadrático médio - root-mean-square-error) é uma medida de quão longe os pontos de dados estão da linha de regressão. É medido como o desvio padrão dos erros de previsão. O valor do RMSE muda se o mesmo conjunto de dados for medido em uma unidade diferente.

O quadrado R é uma medida de quão melhor as previsões pela linha de regressão são comparadas ao uso da média como preditor. Seu valor varia de 0 a 1 e é independente da unidade usada para medir pontos de dados.

3.3 Avaliação do Modelo

Os valores das métricas dependem de como os pontos de dados para treinamento e validação são escolhidos. Assim, o aspecto chave para a avaliação de modelos é que pontos de dados para treinamento e validação sejam selecionados de maneira completamente imparcial.

3.3.1 Dados de Treinamento, Validação e Testes

LO-3.3.1	K1	Definir a necessidade de validar os modelos dividindo o conjunto de dados disponível em conjuntos de dados de treinamento, validação e teste.

O conjunto de dados de treinamento contém os dados nos quais o modelo é treinado. O conjunto de dados de validação é usado pelo algoritmo de

aprendizado de máquina para avaliar se o treinamento foi eficaz. Em cada execução do ML (que é um processo de muitas iterações), o conjunto de dados de treinamento e o conjunto de dados de validação são combinados novamente e divididos de uma maneira diferente para que o algoritmo use combinações diferentes de dados para aprender.

O conjunto de dados de teste é um conjunto de dados separado, usado após o término do processo de ML para validar se o algoritmo foi treinado adequadamente. O conjunto de dados de teste não deve ser usado durante o processo de treinamento. [Wikil]

Na fase pós-treinamento, um modelo de ML é avaliado e testado com um conjunto de dados diferente de seu conjunto de dados de treinamento. No entanto, o conjunto de dados para as fases de treinamento, validação e teste devem vir da mesma fonte ou fontes semelhantes para garantir a eficácia das métricas. Um modelo treinado usando um tipo de conjunto de dados pode ter um desempenho muito ruim em um conjunto de dados originado de fontes muito diferentes.

3.3.2 Subajuste e Sobreajuste

LO-3.3.2	K2	Resumir o conceito de sub-ajuste (underfitting) e sobreajuste (overfitting) e tradeoff de viés e variância (bias-variance tradeoff).
HO-3.3.2	H0	Demonstrar sub-ajuste (underfitting) e sobreajuste (overfitting) para mostrar tradeoff de viés e variância (bias-variance tradeoff).

Se um modelo de ML supervisionado for muito simplista para ajustar os pontos de dados de treinamento (ou seja, não representa a tendência dos dados), é um exemplo de subajuste (underfitting). Por outro lado, um modelo de sobreajuste (overfitting) tenta ajustar demais os pontos de dados de treinamento e isso geralmente resulta em baixa precisão de previsão durante as fases subsequentes de validação ou teste.

A natureza de subajuste (underfitting) e sobreajuste (overfitting) de um modelo

também pode ser explicada em termos de viés e erros de variação. Se um modelo é simplificado demais e não aprende com todas as características fornecidos para representar, é dito que ele possui um viés alto e sofre com uma acurácia de predição ruim.

Se o desempenho da predição do modelo variar muito, alterando ligeiramente o conjunto de dados de treinamento, é considerado um modelo de alta variação (muito dependente do conjunto de dados de treinamento). Um bom modelo precisa atingir baixo viés e baixa variância. Isso é conhecido como tradeoff de viés e variância.

3.3.3 Métodos de Validação Cruzada

LO-3.3.3	K2	Explicar os métodos de split test, validação cruzada K-fold, bootstrap e validação cruzada leave-one-out.
HO-3.3.3	H1	Aplicar o método de validação cruzada K-fold em um algoritmo e comparar os vários resultados.

A maneira como o conjunto de dados disponível é dividido em conjuntos de dados de treinamento e validação pode levar a um alto viés ou alta variância. Para contornar isso, várias combinações de partições devem ser testadas antes de concluir as métricas do modelo. Alguns métodos úteis são: teste de divisão, bootstrap, validação cruzada K-fold e validação cruzada leave-one-out. Cada um desses métodos repete o processo de treinamento e validação várias vezes e o desempenho do modelo é calculado em todas as execuções.

Split-test divide os dados em partes para treinar e testar, mas em cada iteração em diferentes proporções. Isso ajuda a revelar como diferentes divisões podem produzir resultados diferentes.

Bootstrap funciona escolhendo pontos de dados aleatórios de todo o conjunto de dados para treinamento e usa o conjunto de dados restante para validação.

Validação cruzada K-fold divide o conjunto de dados em k partes e usa k-1 subconjuntos para treinar.

Validação cruzada com Leave-one-out é semelhante à validação cruzada K-fold, exceto que cada parte possui um ponto de dados, portanto, requer mais tempo de execução.

3.4 Analítica

3.4.1 Tipos de Análise

LO-3.4.1	K1	Lembrar as características dos quatro tipos de análise - descritiva, exploratória, preditiva, prescritiva - e seu uso na ML.

Análise é uma das principais tarefas quando se trata de entender o conjunto de dados disponível. A análise de dados pode ser feita em quatro níveis e cada nível subseqüente é uma extensão natural para o nível anterior.

Análise descritiva trata da dedução do resumo estatístico dos dados em termos de medidas para tendência central (média, mediana, moda), bem como medidas para dispersão (variância, desvio padrão). Isso ajuda a obter informações sobre o passado e a responder: "O que aconteceu?" [DA1]

Análise exploratória consiste em visualizar o conjunto de dados em alto nível para ver seus padrões e variações.

Análise preditiva consiste em modelar as variáveis de entrada e predizer a probabilidade de resultados.

Análise prescritiva compara todas as previsões viáveis resultantes da análise preditiva e elege / prescreve as melhores dentre elas.

Capítulo 4 - Teste Online de Sistemas IA

Keywords: Método de Análise Linguística, Chatbot

LO-4.1.1	K2	Explicar as partes IA e não IA de um aplicativo inteligente e suas necessidades de teste funcionais e não funcionais.
LO-4.1.2	K1	Mostrar as interações orientadas a informações e orientadas para a ação de partes de IA e não de IA.
LO-4.2.1	K3	Usar o método de design de teste de análise linguística para gerar cenários de teste.
HO-4.2.1	H1	Demonstrar o uso do método de design de teste de análise linguística para gerar cenários de teste.
LO-4.3.1	K3	Utilizar os casos de teste derivados dos requisitos e da arquitetura fornecidos para testar um Chatbot.
HO-4.3.1	H3	Testar um determinado Chatbot a nível de sistema e relatar erros.

4.1 Arquitetura de Uma Aplicação IA

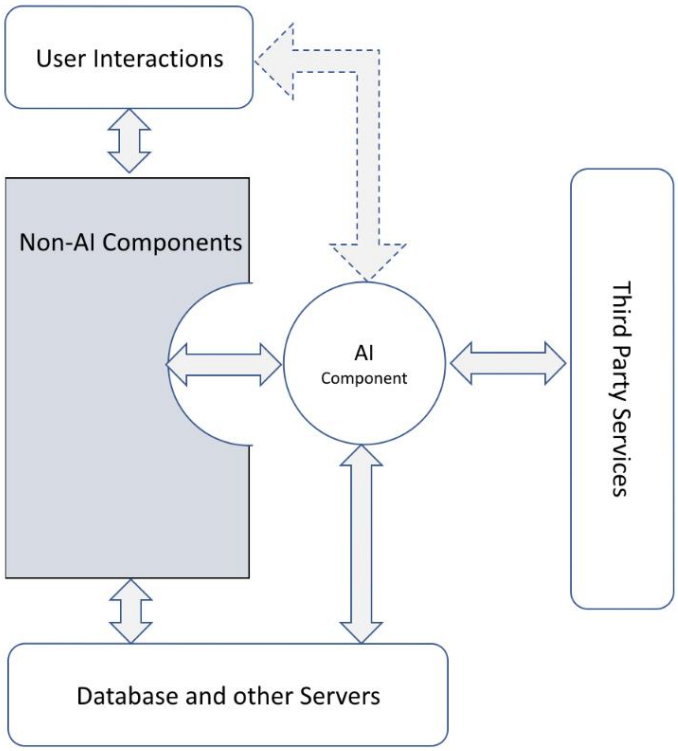
4.1.1 Componentes de uma Aplicação Inteligente e suas necessidades de Testes

LO-4.1.1	K2	Explicar as partes IA e não IA de um aplicativo inteligente e suas necessidades de teste funcionais e não funcionais.

Um aplicativo típico de IA precisa ser examinado, seja um aplicativo de IA monolítico ou um sistema geral composto por um conjunto menor de componentes de IA chamado de um aplicativo maior que não seja de IA. Essa análise é necessária para entender como vários componentes, IA e não-IA, trabalham juntos para fornecer a funcionalidade desejada. As interfaces entre esses vários componentes precisam ser entendidas do ponto de vista de:

- Dados de entrada sendo passados
- Resultado gerado
- Ações tomadas pelos respectivos componentes

- Interações diretas do usuário, se houver
- Interações diretas de terceiros, se houver
- Frequência de invocações
- Número de invocações paralelas, se possível
- Restrições como
  - Tempo
  - Duração
  - Intervalos (range) de entradas e saídas
- Precondições necessárias
- Premissas/configurações comuns
- Visualizar a cadeia de entradas e saídas com transformações relevantes, se houver
- Erros and exceções e seus tratamentos por parte
  - Do componente
  - Cadeia de componentes e manejo final
- Registro (Logging)



Além disso, é necessário explicar a razão do sistema chegar à solução fornecida. Adicionalmente, é necessário identificar cenários de teste para esses sistemas de IA usando técnicas como análise lingüística e teste exploratório.

A outra questão de complexidade que precisa ser garantida é a presença ou ausência de combinações de sistemas de IA. Sistemas independentes (standalone) de IA podem ser insuficientes para especificar completamente um problema do mundo real. Para lidar com esses cenários, combinações de sistemas de IA são usadas para modelar o problema. Para isso, é necessária uma estratégia de teste para lidar com combinações de sistemas de IA.

Na interação IA - não IA, é necessário garantir a cobertura do teste do componente IA, o componente não IA, e a interação envolvendo as duas partes.

Um exemplo de teste desse tipo de sistema é fornecido aqui:
Em um aplicativo de e-mail, a complementação de uma sentença é fornecida por um componente do IA. A interface do usuário do sistema de e-mail representa a parte que não é de IA e o componente de IA interage com ela para fornecer o texto sugerido. A parte não-IA exibe a sugestão e atua na entrada do usuário. Se a sugestão for aceita, isso pode resultar no armazenamento de alguns dados para aprendizado futuro.

O mesmo sistema de e-mail também fornece verificação ortográfica, fornecida por um componente de IA ou de não-IA.

Se analisarmos as interfaces, descobrimos que a interface do usuário (componente não IA) está passando o texto parcialmente escrito para o componente IA. Se houver uma sugestão do componente IA, o texto será exibido pela interface do usuário. As restrições relacionadas ao tempo e à duração são que a sugestão deve ser feita dentro de algumas centenas de milissegundos do texto digitado e isso deve ser um processo contínuo. O testador precisa determinar se o sistema está respondendo rápido o suficiente. Se houver problemas de tempo e duração, como as previsões do modelo sendo lentas, quando o sistema mostrar a frase sugerida, o usuário poderá ter escrito texto adicional. Isso pode tornar a frase sugerida incorreta gramaticalmente ou a sugestão pode estar completamente incorreta.

Um exemplo de problema na interação da GUI e das partes não GUI (IA) de uma saída pode estar relacionado à exibição do texto sugerido. O texto sugerido pode ser exibido em um local incorreto na GUI. A frase sugerida precisa ser visualmente diferenciada do texto atual que está sendo escrito e precisa ser

alterada com base nas ações do usuário. Quaisquer erros nestes comportamentos também se enquadram nas interações GUI e não GUI (IA).

Um exemplo da parte de tratamento de erro / exceção é o cenário em que o componente de complementação de sentença falha (por algum motivo) e a GUI é capaz de lidar com isso.

As configurações do sistema de e-mail incluem configurações de idioma e dicionário. Por exemplo, inglês como configuração de idioma e inglês dos EUA como configuração de dicionário. O sistema de conclusão de sentenças deve fornecer sugestões em inglês dos EUA. Idealmente, o sistema de verificação ortográfica e gramatical também deve usar o inglês dos EUA, neste caso. O que significa que as frases formadas pelo componente IA não devem ser marcadas como texto incorreto pelo componente de verificação ortográfica devido a uma incompatibilidade nas suposições/configurações da interação dos dois componentes, IA ou não-IA.

As imagens abaixo mostram alguns exemplos de problemas descobertos em um sistema assim, A e B mostram a marcação de um nome como erro de ortografia e não o outro, enquanto os dois nomes fazem parte do catálogo de endereços.

Por outro lado, B e C mostram a diferença de velocidade e cachê. Quando digitado pela primeira vez, o nome é completado, mas este não é marcado como erro por alguns segundos e, em seguida, é marcado como erro. No entanto, uma vez marcado como um erro, até a exclusão do nome e o preenchimento automático subsequente são marcados como um erro imediatamente.

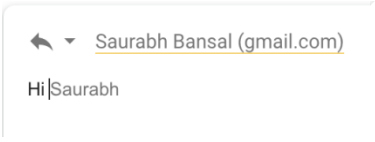
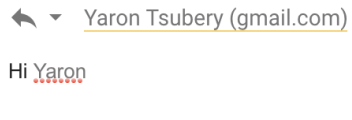
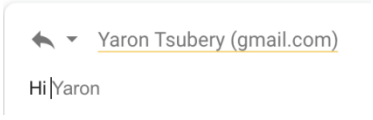


Image A



ImageB



ImageC

4.1.2 Interação de Partes IA e Não-IA

LO-4.1.2	K1	Mostrar a interação orientada a informações e orientada para a ação de peças de IA e de outras partes.

Nas chamadas orientadas a informações, a API ou o serviço do aplicativo IA é invocado a partir de um aplicativo final que pode ser não IA. Normalmente, o componente IA é chamado para uma resposta. Por exemplo, um algoritmo de detecção de fraude trabalha com a ideia de chamar um modelo treinado predefinido para retornar se uma transação de entrada é fraudulenta ou não.

Alguns dos testes importantes para testar as interações estão relacionados a:

- Testes baseados em valores limites — tanto de entrada como de saída
- Casos de testes não usuais (a.k.a. Corner test cases)
- Testes relacionados ao tamanho e tipo de dados passados
- Testes de gerenciamento de Exceções
    - Resposta não recebida
    - Loop de feedback de solicitação-resposta quebrado para ações
- Dados de entrada de testes errôneos
- Dados de saída de testes errôneos
- Requisições que não podem ser finalizadas
- Respostas não recebidas
- Testes relacionados a desempenho
- Testes relacionados a segurança
- Testes relacionados a robustez

As interações entre os componentes IA e não IA podem ter um impacto na experiência do usuário. As interações podem ser do seguinte tipo:

- Sinalizar apenas (basta alimentar informações para o sistema)
- Orientação para a ação, por exemplo, gerar uma resposta para o usuário OU classificar o problema
- Cenários de falha de interação entre APIs em que as APIs falham ao retornar um resultado
- Handover de componentes IA para componentes não-IA e vice-versa

Quando os testes são projetados para os sistemas de IA, estes são os seguintes níveis de teste:

- Testes que avaliam apenas a parte IA

- Testes que avaliam apenas a parte não-IA
- Testes de integração das duas partes
- Respostas em cache vs. respostas aprendidas

Em um sistema não autônomo (não standalone) maior, é necessário examinar a cobertura do modelo de IA implantado, bem como o gerenciamento de desempenho deste. Após o desenvolvimento do componente IA, o sistema precisa ser testado em um ambiente de implantação.

4.2 Método de Design de Teste de Análise Linguística

4.2.1 Design de Teste Baseado em Análise Linguística

LO-4.2.1	K3	Fazer uso do método de design de teste de análise linguística para gerar cenários de teste.

A análise linguística é usada para projetar um grande número de cenários, mesmo quando os requisitos são pouco documentados [LA1]. Uma análise lingüística dos requisitos identifica os objetos de teste e as ações a serem tomadas sobre eles. Esse método ajuda a encontrar casos de teste de canto (incomuns), um requisito essencial para testar sistemas de IA.

O método funciona como descrito abaixo:

1. Identificar os substantivos que representam os objetos de teste e os verbos que representam as ações.

2. Identificar as propriedades dos substantivos.

3. Identificar as propriedades das propriedades e repetí-las até que nenhuma outra propriedade possa ser descoberta ou se considere que foi alcançada uma profundidade suficiente.

4. Identificar advérbios e adjetivos aplicáveis aos substantivos e verbos para identificar mais propriedades

5. Usar 5W1H (O que, por quê, onde, quando, qual, como) nos verbos para identificar as combinações substantivo-verbo, fornecendo os testes

funcionais e não funcionais.

6. Realizar as seguintes perguntas para cada cenário:
    a. Quem/o que é é o agente?
    b. Quais são os instrumentos/meios para realizar o cenário?
    c. Qual é o propósito da ação?
    d. Qual é o direção da ação?
    e. Qual é a fonte da ação?
    f. Qual é o motivo da ação?
    g. Quem possui as médias e os resultados da ação?
    h. Onde a ação se origina?
    i. Quem é o receptor da ação?

Essas perguntas seguem as regras gramaticais do Sânscrito [LA2].
No caso de testar sistemas de IA, os dados são o caso de teste. O método fornecido permite a identificação de ambos, os dados e a combinação de ações (cenários).

4.3 Testes de Sistemas de Inteligência Artificial

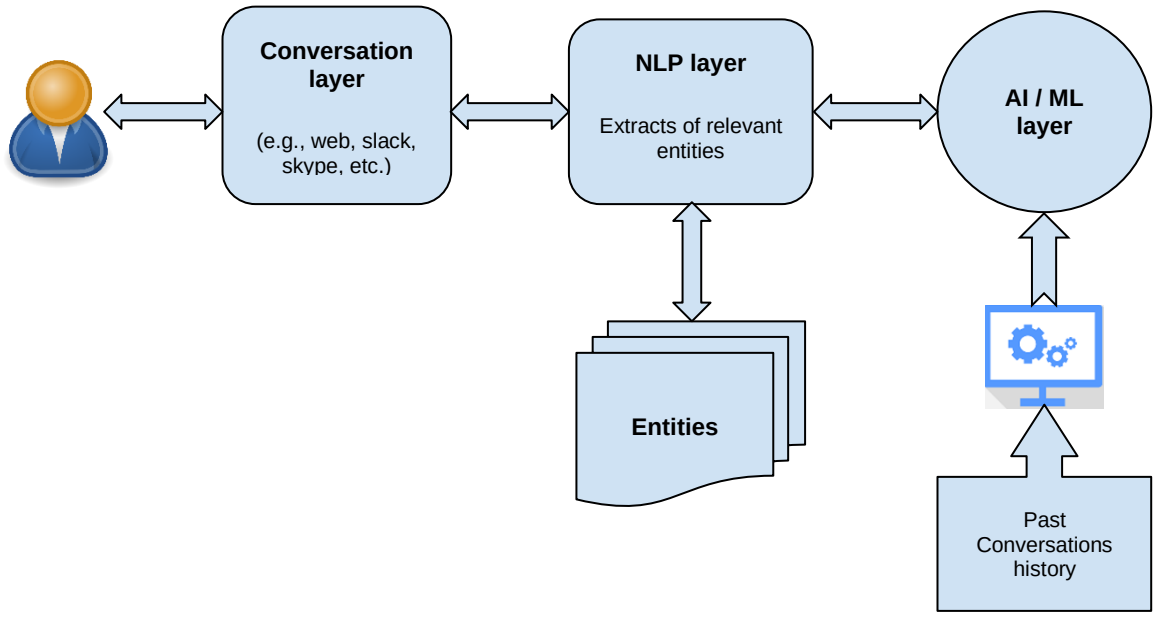
4.3.1 Testando um Chatbot

LO-4.3.1	K3	Utilizar os casos de teste derivados de determinados requisitos e arquitetura para testar um Chatbot.
HO-4.3.1	H3	Testar um determinado Chatbot no nível do sistema e reportar erros.

Hoje, os Chatbots são usados em vários contextos, desde bots de atendimento ao cliente, bots de informações até bots de suporte.

A arquitetura lógica mostra as principais camadas de um Chatbot, incluindo as de conversação, NLP, IA/ML e conectividade. A camada de conversação gerencia a interação do front-end com o usuário final, por exemplo, Skype, Slack, messenger do Facebook ou um front-end da web. A camada de NLP (Procesamiento del lenguaje natural, Natural Language Processing), extrai as entidades e informações relevantes das declarações do usuário final. A camada IA/ML decodifica a intenção real do enunciado, tendo sido treinada em

conversas históricas passadas fornecidas como dados para o sistema.



Os casos de teste devem cobrir variações de dados de entrada em cada camada individual. A camada de conversação precisa ser testada para a transmissão de dados correta do front-end ao back-end e vice-versa. Na camada da NLP, deve-se testar se a limpeza de dados e o pré-processamento dos textos brutos de entrada estão ocorrendo corretamente e se as entidades extraídas são suficientes para transmitir o contexto em uma conversa em andamento. A camada ou modelo de IA /ML (treinado em conversas históricas) deve ser testado quanto à precisão de predizer a intenção correta com base nas entidades extraídas (pela camada da NLP), por exemplo, para um texto de entrada 'Oi' / 'Olá' / ' Saudações 'etc. A intenção de 'bem-vindo' deve ser prevista.

Além disso, o chatbot deve ser capaz de saltar entre várias intenções sem problemas se a sequência do fluxo da conversa mudar abruptamente. Portanto, os casos de teste que abrangem a troca dinâmica de intenção devem ser identificados.

Os requisitos não funcionais devem ser testados para o desempenho do front-end principalmente, se é capaz de suportar muitas chamadas simultâneas de clientes.

Capítulo 5 - Inteligência Artificial Explicável

Palavras-chave: interpretar/explicar o modelo de ML, explicável, LIME (Explicações Independentes de Interpretação Local, Local Interpretable Model-Agnostic Explanations), CAM (Manufatura Assistida por Computador, Computer-Aided Manufacturing)

LO-5.1.1	K2	Explicar a necessidade de IA explicável (XAI).
LO-5.1.2	K3	Utilizar LIME para explicar modelos de AI.
HO-5.1.2	H1	Utilizar LIME em um classificador de imagens e em um classificador de texto.
LO-5.1.3	K3	Utilizar Grad-CAM para explicar modelos CNN (Convolutional neural network).
HO-5.1.3	H1	Utilizar Grad-CAM para tornar CNN mais transparente.

5.1 IA Explicável (XAI)

5.1.1 IA Explicável e Sua Necessidade

LO-5.1.1	K2	Explicar a necessidade de IA explicável (XAI).

Uma vez que um modelo de ML é treinado, ele deve funcionar com um nível definido de acurácia para todas as variações definidas de cenários. Se a qualidade das predições do modelo for insuficiente para alguns cenários e funcionar de maneira fantástica para outros, é provável que seja um modelo tendencioso. Um conjunto de dados para o qual a qualidade do modelo fica abaixo do nível de qualidade definido representa um defeito.

Como testador, é muito difícil descobrir todos esses defeitos sem usar uma abordagem sistemática. É preciso examinar o comportamento do modelo e sua variação com a mudança nos fatores de entrada para deduzir uma aproximação da relação entre entrada e saída. Esse tipo de dedução é chamado de interpretação ou explicação do modelo de ML. Esse relacionamento aproximado pode não ser uma substituição do modelo real, mas pode ser bom o suficiente para revelar um possível viés. Uma visão do comportamento do modelo ajuda a

avaliar sua qualidade geral e a viabilidade do modelo para implantação. Outras razões para exigir explicação ou interpretação de modelos podem ser medidas de segurança, aceitação social, viés de detecção ou curiosidade humana e aprendizado sobre o modelo [EA1].

Nem todos os modelos podem ser explicados. Quanto mais complexo o modelo, menor a probabilidade de interpretá-lo ou explicá-lo. A saída de modelos não DL (Deep Learning), por exemplo floresta aleatória [Wiki2], árvores de decisão [Wiki3], regressão linear [Wiki4] etc., podem ser convenientemente explicadas em termos das variáveis de entrada.

Os modelos de DL são inerentemente complexos em sua implementação; portanto, é melhor examinar o modelo como uma caixa-preta, ou seja, observar as variações dos resultados por pequenas perturbações das variáveis de entrada e aproximar o modelo subjacente por um modelo simples e interpretável.

Alguns dos algoritmos, ferramentas e abordagens populares e fáceis de usar para interpretação de modelos incluem: LIME [EA2] e CAM [EA3].

5.1.2 LIME

LO-5.1.2	K3	Utilizar LIME para explicar um modelo de IA
HO-5.1.2	H1	Utilizar LIME em um classificador de imagens e em um classificador de texto.

O LIME é alimentado com uma amostra para estudar as predições do modelo e suas variações mais próximas e revela as features de entrada responsáveis pela saída prevista do modelo. LIME gera variantes próximas suficientes da entrada da amostra e obtém resultado para cada uma das variantes. Portanto, ele tenta aproximar como pequenas variações na entrada estão alterando a variável de saída. Como todas as variantes geradas e estudadas pelo LIME estão próximas da amostra fornecida, as explicações do LIME são chamadas de 'local' (para a entrada de amostra). Além disso, é um método independente de modelo, pois o LIME não exige que se olhe o modelo ou algoritmo.

LIME pode ser usado em um classificador de imagens para gerar explicações. Deduz partes da imagem que desempenham um papel significativo na determinação do resultado. Ele permite que o testador relacione e exclua que o modelo não está baseando suas deduções em recursos irrelevantes.
Da mesma forma, para um classificador de texto, o LIME pode apontar as palavras (partes do texto) que levam à categorização do texto de amostra em uma das classes predefinidas. Isso ajuda um testador a avaliar se o modelo está baseando suas deduções em termos irrelevantes.

5.1.3 CAM para Redes Neurais

LO-5.1.3	K3	Utilizar o Grad-CAM para explicar modelos CNN.
HO-5.1.3	H1	Utilizar o Grad-CAM para tornar CNN mais transparente.

As CNNs (Redes neurais convolucionais) são muito úteis, mas não são transparentes sobre porque chegaram a uma conclusão específica. Para trazer transparência a esses modelos, o Mapeamento de Ativação de Classes com base em Gradiente (Grad-CAM) visualiza as regiões de entrada que são importantes para as previsões desses modelos. Ele usa as informações de gradiente específicas da classe que fluem para a camada convolucional final de uma CNN e produz um mapa de localização aproximado das regiões importantes da imagem.

Capítulo 6 - Riscos e Estratégias de Testes para Sistemas de IA

Palavras-chave: modelo pretreinado, derivação conceitual

LO-6.1.1	K1	Recordar os riscos de testar sistemas de IA.
LO-6.1.2	K2	Explicar os problemas associados ao uso de modelos pré-treinados de terceiros em sistemas de produção.
LO-6.1.3	K2	Explicar a necessidade de testar o modelo treinado novamente devido à derivação do conceito.
LO-6.1.4	K1	Definir ambiente de testes para sistemas de IA e recordar os desafios.
LO-6.2.1	K1	Recordar-se de estratégias de teste para aplicações de IA,

		especialmente considerando tipos, fases e níveis de teste.

6.1 Riscos em Testes IA

6.1.1 Riscos de Testes de Sistemas de IA

LO-6.1.1	K1	Recordar os riscos de testes de sistemas de IA.

Os testes de sistemas de IA apresentam alguns riscos adicionais, além dos riscos associados a sistemas não-IA. Alguns desses riscos são

- Desafios relacionados à condição de teste
  - Com um grande número de testes (cada conjunto de dados sendo um teste), como verificar
    - corretude dos testes
    - completude dos testes
  - Falta de um oráculo de teste confiável para indicar qual deve ser a saída correta para uma entrada arbitrária
  - Identificação de falsos positivos e falsos negativos. Os falsos positivos ainda são mais fáceis de identificar, porque as falhas devem ser investigadas. Os falso negativos são difíceis de identificar porque esses testes são mostrados como aprovados
- Desafios relacionados a dados de testes
  - Em IA, os dados de teste são iguais a um caso de teste, especialmente para testes offline. Uma quantidade enorme de dados de teste é idealmente necessária para o teste
  - A disponibilidade de dados pode ser um problema.
  - A qualidade dos dados pode ser ruim, sendo necessária a limpeza de dados
  - Dados marcados/anotados são necessários para treinamento e teste. Obter esse tipo de dados é caro.
  - As caixas de canto dos sistemas ML são difíceis e caras de gerar
  - Desafios relacionados a custos
  - Para aplicações complexas, o tipo de hardware necessário para treinar e testar o modelo offline pode ser caro
  - Custos de energia para treinar DNN (rede neural profunda) é geralmente muito alto
- Desafios relacionados a habilidades e ferramentas

    - As habilidades necessárias para testar sistemas de IA são muitas. O testador precisa entender como os sistemas de IA são construídos e como eles precisam ser testados
    - Falta de informações estruturadas, ferramentas e frameworks para testes de sistemas de IA
- Desafios relacionados à compreensão do domínio e viés relacionado
    - Provas formais da qualidade ideal de um algoritmo não garantem que um aplicativo implemente ou use o algoritmo corretamente
    - É necessária uma cobertura completa dos testes de entrada, com base no domínio, para evitar vieses (tendências), cobertura incompleta e possibilidade de acidentes

6.1.2 Riscos de Utilizar Modelos Prétreinados

LO-6.1.2	K2	Explicar os problemas associados ao uso de modelos prétreinados de terceiros em sistemas de produção.

Algumas organizações disponibilizaram seus modelos prétreinados e muitos desenvolvedores de IA os utilizam.

Como são desconhecidos ou não documentados, os sistemas construídos com modelos prétreinados podem falhar de maneiras imprevistas ou produzir resultados que podem ser sub-ideais em algumas situações.

Por exemplo, modelos ImageNet, como Inception, VGG, AlexNet etc. Os modelos podem ter seus próprios vieses e deficiências, que podem precisar ser encontrados pelo teste. Como são desconhecidos ou não documentados, os sistemas construídos com modelos prétreinados podem falhar de maneiras imprevistas ou produzir resultados que podem ser subótimos em algumas situações.

6.1.3 Risco de Desvio de Conceito (Concept Drift, CD)

LO-6.1.3	K2	Explicar a necessidade de testar o modelo treinado novamente devido à desvio do conceito.

Os modelos de trabalho podem se degradar ao longo do tempo devido à mudança no relacionamento entre as features de entrada e saída. Por exemplo, o impacto de comerciais e vários outros tipos de campanhas de marketing

resultam em uma mudança no comportamento de clientes em potencial a cada poucos meses. Isso é conhecido como desvio de conceito (CD). Para descobrir a ocorrência de desvio de conceito, são necessários testes periódicos dos modelos em funcionamento e modelos integrados com amostras de dados recentes. Para esse fim, o novo treinamento do modelo e a repetição do ciclo de vários testes e validações são realizados nas fases offline e integrada. [JB2]

Existem várias maneiras de lidar com o desvio de conceito, tais como:

- Não fazer nada
- Retreinar o modelo com dados atualizados
- Periodicamente, atualizar o modelo usando os últimos dados do modelo atual
- Aprender a mudança - uma abordagem em conjunto em que o modelo atual permaneça inalterado. Um novo modelo pega a saída do modelo atual e aprende a corrigir as predições
- Detectar e escolher modelo - detecte a desvio de conceito, se possível, e escolha um modelo apropriado

6.1.4 Desafios do Ambiente de Testes de IA

LO-6.1.4	K1	Definir o ambiente de teste para sistemas de IA e recordar-se de seus desafios.

Os ambientes de teste para sistemas de IA podem ser muito complexos, devido a possíveis casos de uso, contextos e várias maneiras e etapas de pré-processamento de dados. Do ponto de vista de testes offline, as necessidades do ambiente são mais exigentes do que do ponto de vista de teste online. É necessário um grande espaço de armazenamento de dados, alta largura de banda da rede e também maior poder computacional para treinar / executar o modelo.

6.2 Estratégia de Testes

6.2.1 Estratégia de Testes para Testar Aplicações de IA

LO-6.2.1	K1	Recordar as estratégias de teste para aplicativos de IA, especialmente considerando tipos, fases e níveis de teste.

Um aplicativo baseado em IA do mundo real pode empregar um ou mais componentes de IA e não-IA. A estratégia de teste para esse sistema incluirá

dimensões convencionais de teste, bem como novos fatores específicos para os componentes de IA e sua integração com outros componentes do sistema. Algumas das considerações convencionais da estratégia de teste são:

- **Nível de testes requeridos para o sistema** - testes de unidade, testes de integração, teste de sistema, testes de aceitação e testes de integração de sistemas como descrito no syllabus ISTQB® FL. [ISTQB-FL2018]
- **Técnicas de Teste** - caixa-branca, caixa-preta e caixa-cinza
- **Testes Funcionais e Não-funcionais** - especialmente testes de segurança, performance, robustez e escalabilidade
- **Uso de automação de testes**

Para testar aplicações baseadas em IA, é necessário levar em consideração os seguintes aspectos adicionais:

- **Teste Offline (funcional)** - Testar o modelo de IA treinado é uma etapa adicional que precisa ser executada como parte do SDLC (ciclo de vida de desenvolvimento do software). As habilidades necessárias para realizar os testes nesta fase também precisam ser consideradas.

- **Teste Offline (não-funcional)** - Testar o modelo de IA treinado para vários aspectos não-funcionais. O modelo precisa ser testado quanto à velocidade, utilização de recursos, concorrência e carga e escalabilidade antes da implantação.

- **Testes Caixa-Preta** - a maioria dos modelos de IA se apresentam como APIs e são invocadas como caixa-preta em geral. Um componente crítico ausente nos sistemas de IA é a falta de oráculo de teste, dificultando o teste da caixa-preta. Técnicas como testes metamórficos estão ficando populares para ML, pois não requer um oráculo. Dada uma execução bem-sucedida de casos de teste, são realizadas variações nos casos de teste e os resultados são examinadas. A necessidade do resultado satisfazer apenas os relacionamentos não requer nenhum oráculo. [Wiki5]

- **Testes Caixa-Branca** - Em geral, o teste caixa-branca para sistemas ML é difícil, pois, exceto em alguns modelos simples, o funcionamento interno do modelo não é claro nem acessível. No entanto, recentemente, surgiram algumas técnicas de caixa branca para sistemas de ML. Um exemplo é o

DeepXplore, um produto de teste de caixa-branca automatizado. Usando a cobertura de neurônios, os pesquisadores foram capazes de expor milhares de comportamentos únicos de casos de canto incorretos. [DX1]

- **Aquisição e Processamento de Dados** - Os dados disponíveis nem sempre podem estar em condições de alimentar um modelo para treinamento ou teste. Isso foi discutido em detalhes em 3.1 Preparação e processamento de dados.

- **Conversão de implementações de de desenvolvimento em produção** - por exemplo, convertendo o notebook Jupyter em código Python que é executado no servidor dentro de um container Docker em uma instância da nuvem. Uma vez feito, o modelo gerado pode ser salvo como um arquivo e utilizado no aplicativo.

Capítulo 7 - IA em Testes

Keywords: Automação de Testes

LO-7.1.1	K2	Explicar como a IA pode ajudar em várias atividades de teste - planejamento e estimativa de teste, análise de risco, design de caso de teste e geração de dados de teste, atribuição de defeitos, análise de impacto, análise de cobertura.
LO-7.1.2	K2	Explain how AI can assist in reporting and smart dashboards. Explicar como a IA pode ajudar nos relatórios e nos painéis inteligentes.
LO-7.2.1	K2	Ilustrar o uso de várias ferramentas de automação de execução de teste baseadas em IA disponíveis comercialmente.
HO-7.2.1	H2	Fazer uso de versões demo / trial das ferramentas de execução de testes automáticos baseadas em IA.

A IA pode ser usada para melhorar os processos de teste existentes no SDLC. A ideia é fazer uso dos dados gerados nos processos de teste para fornecer análises detalhadas, mais automação, insights e padrões mais profundos e capacidade de prever e realizar ações corretivas.

7.1 IA para o Ciclo de Vida de Testes (Software Testing Life Cycle, STLC)

7.1.1 Métodos de IA para STLC

LO-7.1.1	K2	Explicar como a IA pode ajudar em várias atividades de teste - planejamento e estimativa de teste, análise de risco, design de caso de teste e geração de dados de teste, atribuição de defeitos, análise de impacto, análise de cobertura.

No contexto do planejamento e estimativa de teste, o custo geral do projeto de software pode ser aproximado, levando em consideração diferentes informações relacionadas aos projetos de IA, incluindo tamanho dos dados, esforço envolvido, escolha da plataforma, tipo de aplicativo, tempo de preparação dos dados, tempo de treinamento e testes. Tempo. Dados os dados históricos dessas

entradas, os modelos de ML podem ser preparados para fornecer estimativas mais precisas.

A IA pode ajudar a priorizar riscos, obter métricas mais precisas associadas à aderência ao cronograma e ajudar a identificar as métricas de desempenho dos aplicativos com mais precisão.

Para o design de teste, o uso de tecnologias de IA, como processamento de linguagem natural (NLP) e mineração de texto, pode ajudar a geração automatizada de casos de teste a partir de documentos de requisitos textuais. Adicionalmente, a IA aplicada na análise de código (estática e dinâmica), juntamente com a análise dos dados coletados nos testes, pode sinalizar possíveis problemas de desempenho e outros requisitos não-funcionais. Além disso, a execução de ML em dados anteriores pode ajudar a identificar padrões de dados de teste e a gerar dados de teste automatizados para testes de componentes e testes de sistema.

Em particular, para dados de imagem e elementos da GUI, IA pode ajudar a identificar elementos renderizados incorretamente automaticamente. Além disso, por uma análise centrada em dados baseada em ML de diferentes fluxos possíveis, os fluxos corretos de dados podem ser automatizados.

Para predição automática de defeitos, os modelos que usam ML podem prever defeitos com base nas métricas de qualidade do código.

A análise de impacto usando ML no código pode ajudar a automatizar a identificação de módulos e arquivos impactados com base na mudança.

Em termos de análise de cobertura, o uso da IA pode ajudar a obter uma cobertura abrangente de teste e código por meio da análise dos fluxos de dados capturados.

7.1.2 IA para Relatórios e Paineis (Dashboards) Inteligentes

LO-7.1.2	K2	Explicar como a IA pode ajudar nos relatórios e nos dashboards inteligentes.

No contexto de relatórios e dashboards, o uso do ML ajuda a gerar insights focados e dados resumidos para apresentação em dashboards inteligentes, em vez de análises simples.

7.2 IA Baseada em Ferramentas de Automação

7.2.1 Ferramentas

LO-7.2.1	K2	Ilustrar o uso de várias ferramentas de execução de testes automáticos baseadas em IA disponíveis comercialmente.
HO-7.2.1	H2	Fazer uso de versões demo/trial das ferramentas execução de testes automáticos baseadas em IA.

Há um grande número de ferramentas de automação disponíveis no mercado. Algumas dessas ferramentas usam a IA para tornar a automação mais fácil ou mais sustentável.

As ferramentas de IA podem usar GUI Spiders que atravessam a GUI completa e registram o aplicativo. Em iterações, eles são capazes de aprender, comparar e identificar bugs.

Algumas ferramentas combinam elementos de testes visuais da GUI e usam o ML para descobrir as alterações e a possível correlação entre localizadores e elementos e se uma alteração é um bug ou uma alteração esperada.

Algumas ferramentas usam a NLP, algumas funcionam no nível do DOM, outras na interface do usuário, outras vão para o log, algumas combinam testes funcionais e de desempenho e outras combinam algumas das abordagens mencionadas acima.

As ferramentas baseadas em imagem podem usar ferramentas de classificação de imagem baseadas em AI para sinalizar defeitos da interface do usuário. Elas podem estar funcionando como um plug-in de navegador da Web com gravação e reprodução. A comparação de imagens baseada em IA pode ser superior à comparação simples de imagens.

Referências

[AG1] Neural Networks and Deep Learning - By Aurélien Géron (O'Reilly)

[BT1] A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives - By L. Anderson, P. W. Airasian, and D. R. Krathwohl (Allyn & Bacon 2001)

[BT2]
https://www.apu.edu/live_data/files/333/blooms_taxonomy_action_verbs.pdf

[DA1] 4 types of data analytics to improve decision-making
https://www.scnsoft.com/blog/4-types-of-data-analytics

[DI1] Introduction to k-Nearest Neighbors
https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

[DSC1] https://www.datasciencecentral.com/profiles/blogs/the-data-science-project-lifecycle

[DX1] DeepXplore: Automated Whitebox Testing of Deep Learning Systems, SOSP '17 Proceedings of the 26th Symposium on Operating Systems Principles (Pages 1-18) http://www.cs.columbia.edu/~junfeng/papers/deepxplore-sosp17.pdf

[EA1] Interpretable Machine Learning: A Guide for Making Black Box Models Explainable (Section 2.1 Importance of Interpretability) - By Christoph Molnar
https://christophm.github.io/interpretable-ml-book/agnostic.html

[EA2] "Why Should I Trust You?" Explaining the Predictions of Any Classifier
https://arxiv.org/pdf/1602.04938v1.pdf

[EA3] Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization
https://arxiv.org/pdf/1610.02391.pdf

[ISTQB-FL 2018] ISTQB Foundation Level Syllabus version 2018. Available at
https://www.istqb.org/downloads/category/51-ctfl2018.html

[JB1] Machine Learning: Hands-On for Developers and Technical Professionals - By Jason Bell (WILEY 2014)

[JB2] A Gentle Introduction to Concept Drift in Machine Learning
https://machinelearningmastery.com/gentle-introduction-concept-drift-machine-learning

[KUR] The Age of Intelligent Machines - By Ray Kurzweil (MIT Press 1990)

[LA1] Visual Modeling for Test Idea Generation - By Vipul Kocher
https://www.testnet.org/testnet/download/preview-voorjaar-2015/visual-modeling-for-test-idea-generation-v2.pdf

[LA2] Vibhakti - By Ujjwol Lamichhane
https://www.scribd.com/doc/30377592/Vibhakti-%E0%A4%B5%E0%A4%BF%E0%A4%AD%E0%A4%95-%E0%A4%A4%E0%A4%BF

[SMU]The CRISP-DM User Guide
https://s2.smu.edu/~mhd/8331f03/crisp.pdf

[TDA] Testing in the digital age: AI makes the difference - By Tom van de Ven, Rik Marselis and Humayun Shaukat (Sogetibooks 2018)

[UDI] Image Preprocessing
https://towardsdatascience.com/image-pre-processing-c1aec0be3edf

[UDT] Text Preprocessing in Python: Steps, Tools, and Examples
https://medium.com/@datamonsters/text-preprocessing-in-python-steps-tools-and-examples-bf025f872908

[ULA] Unsupervised Learning: Association Rules - By Krzysztof J. Cios, Roman W. Swiniarski, Witold Pedrycz, Lukasz A. Kurgan
[Wiki1] https://en.wikipedia.org/wiki/Training,_validation,_and_test_sets

[Wiki2] Random forest
https://en.wikipedia.org/wiki/Random_forest

[Wiki3] Decision tree learning
https://en.wikipedia.org/wiki/Decision_tree_learning

[Wiki4] Linear regression

https://en.wikipedia.org/wiki/Linear_regression

[Wiki5] Metamorphic testing
https://en.wikipedia.org/wiki/Metamorphic_testing

GLOSSÁRIO

Dashboard

Feature

Framework

GUI