

Desenvolvimento de software tradicional versus IA

Ivan Nicola Viragine



THE
DEVELOPER'S
CONFERENCE

Objetivo

Entender como a IA altera o
paradigma de desenvolvimento de
software e destacar as principais
diferenças no desenvolvimento
utilizando **modelos de IA e**
tradicional

Ivan Nicola Viragine



- AI Engineer em uma startup da Califórnia
- 16 anos de experiência em desenvolvimento de software
- 8 anos de experiência com Inteligência Artificial
- Engenheiro de IA, Cientista de Dados, Desenvolvedor Full Stack, Fundador de startup, etc.
- MBA em Ciência de Dados pela USP
- Bacharel em Ciência da Computação pela UNESP

Quando utilizar IA?

Problemas **complexos**, com **muitas variáveis e grandes volumes** e/ou **variações de dados**

Porque utilizar IA?

A IA **busca padrões e relações** que seriam **muito difíceis ou até impossíveis** de serem determinadas por humanos e **transformadas em código**



Exemplos

Reconhecimento de imagens

- Como escrever **regras** estáticas para identificar gatos?



Exemplos

Reconhecimento de voz

- Como escrever **regras** estáticas para identificar palavras e frases em um áudio?



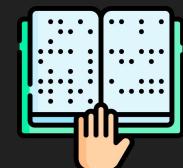
Onde usar modelos de IA em software?

- **Identificação e prevenção de fraudes**
 - apps de bancos, e-commerce/lojas virtuais
- **Recomendações**
 - e-commerce/lojas virtuais, ads, redes sociais
- **Análise de sentimentos**
 - reviews/resenhas, SAC
- **Tradução de idiomas**
 - apps de tradução, SAC
- **Diagnóstico médico**
 - apps de saúde, hospitais



Onde usar modelos de IA em software?

- **Reconhecimento de voz**
 - assistentes virtuais, apps de tradução, acessibilidade
- **Reconhecimento de imagens**
 - apps de segurança, redes sociais, acessibilidade
- **Sumarização de texto**
 - apps de notícias, SAC, pesquisas
- **Chatbots**
 - SAC, e-commerce/lojas virtuais
- **Otimização de processos**
 - indústrias, logística
- **Segurança**
 - controle de acesso, reconhecimento facial, prevenção de roubos

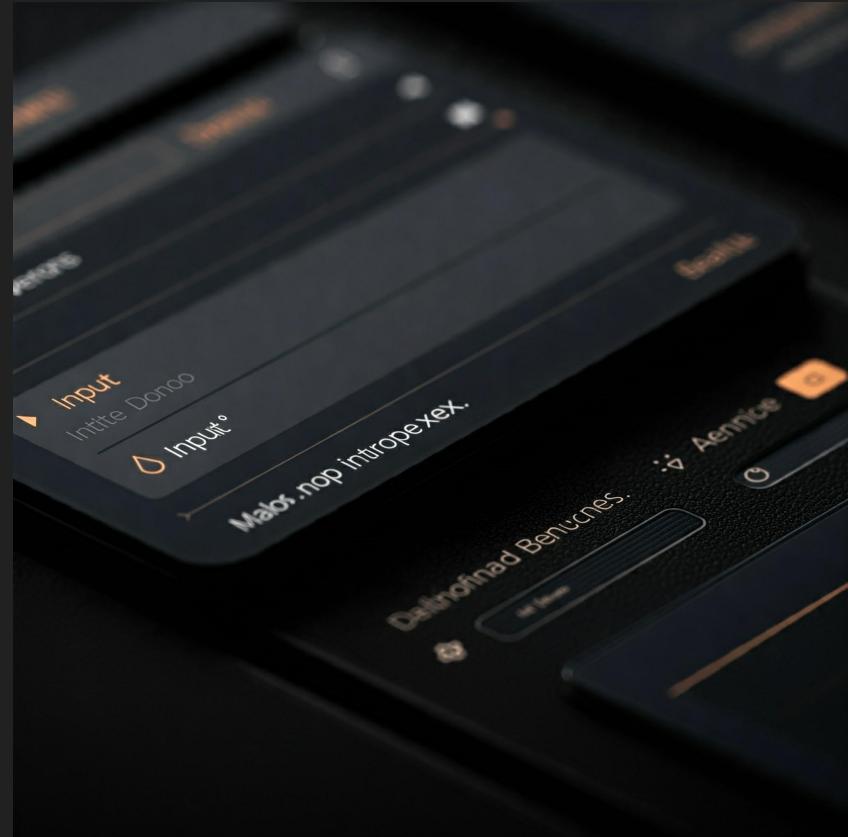


Diferenças

Natureza das entradas

Tradicional

- As entradas são mais **previsíveis e controladas**, normalmente através de uma GUI com campos e validações.



Natureza das entradas

IA

- As entradas podem ser mais variadas e até imprevisíveis, como **texto livre, imagens, áudio, etc.**

The screenshot shows a dark-themed ChatGPT interface. At the top right, there's a "Share" button with a downward arrow icon. In the top left corner, it says "ChatGPT 4o". The user message is: "Can you help me create a personalized morning routine that would help increase my productivity throughout the day? Start by asking me about my current habits and what activities energize me in the morning." Below this, the AI response is: "I'd be happy to help you create a personalized morning routine! To start, let me ask a few questions so I can better understand your current habits and what works best for you:

1. What time do you usually wake up in the morning? Do you use an alarm or wake up naturally?
2. How do you currently spend the first hour after waking up? What does your current routine look like?
3. Are there any specific activities or habits that help energize you in the morning? For example, do you enjoy exercise, meditation, reading, or having a certain type of breakfast?
4. What time do you typically start work or your main responsibilities for the day?

" At the bottom left, there's a "Message ChatGPT" button with a speech bubble icon. On the far right, there are two circular icons: one with an upward arrow and another with a downward arrow. At the very bottom center, it says "ChatGPT can make mistakes. Check important info."

Natureza das saídas

Tradicional

- **Determinística:** a mesma entrada *sempre* produz a mesma saída

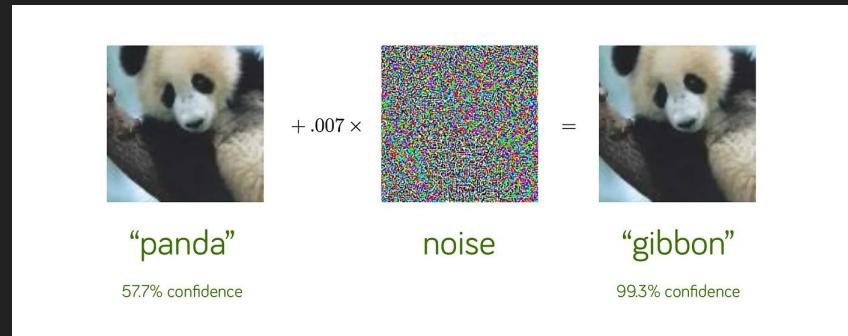


```
if this:  
    do_that()  
elif that:  
    do_this()  
else  
    do_something_else()
```

Natureza das saídas

IA

- **Não determinística (probabilística)**: a mesma entrada *pode ou não* produzir a mesma saída e/ou *pequenas variações* na entrada podem produzir variações na saída de maneira relativamente *imprevisível*.

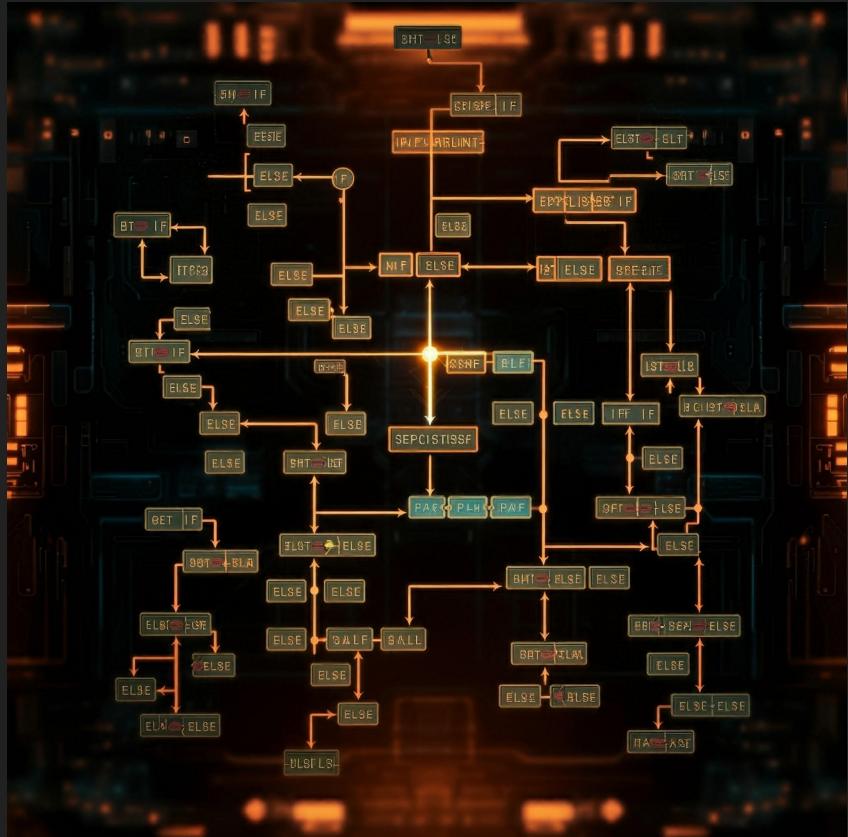


*Explaining and Harnessing Adversarial Examples,
Goodfellow et al, ICLR 2015*

Dependência de Dados

Tradicional

- Dependente de dados, mas não tão diretamente. O código pode ser escrito sem dados e **a qualidade é mais dependente da lógica e estrutura do código do que dos dados em si.**



Dependência de Dados

IA

- **Totalmente dependente de dados.** Sem dados, não há modelo. A qualidade do modelo é *diretamente proporcional à qualidade* e, de certa maneira, *quantidade* dos dados utilizados para treinamento. A avaliação dos modelos também tem alta dependência de qualidade e quantidade de dados.



Interpretabilidade

Tradicional

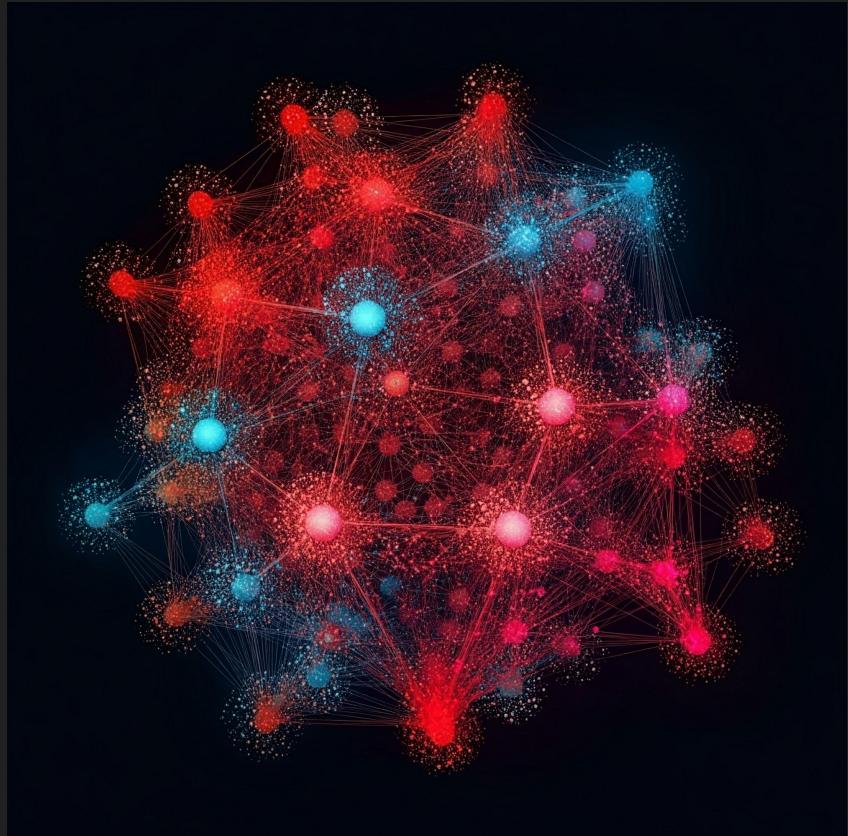
- É possível entender como o código funciona, prever a saída com precisão e depurar problemas com mais facilidade. **O caminho de execução é claro e previsível.**



Interpretabilidade

IA

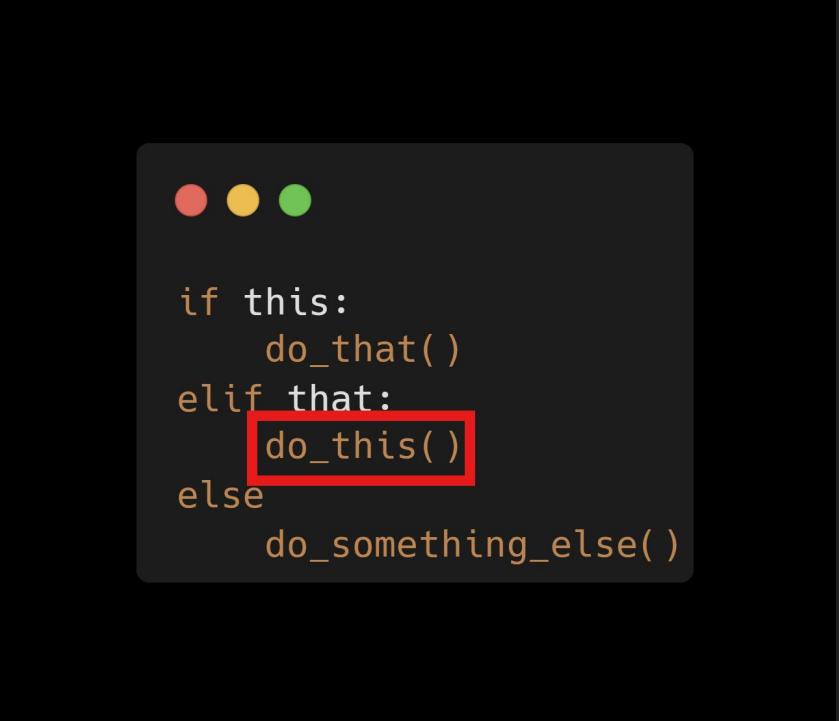
- IA é uma "**caixa preta**": é difícil e às vezes "impossível" entender como a IA chegou a uma determinada conclusão ou decisão ou prever uma saída através de uma entrada com precisão. Há ferramentas que ajudam no entendimento de uma ou mais saídas, porém, não de maneira generalizada.



Atualização e manutenção

Tradicional

- É possível atualizar o código diretamente, **modificar apenas as partes necessárias** e testar as mudanças de maneira mais direta e eficiente.



```
if this:  
    do_that()  
elif that:  
    do_this()  
else:  
    do_something_else()
```

A screenshot of a code editor window showing a snippet of Python code. The code contains an if-elif-else conditional block. The line ' do_this()' under the 'elif that:' condition is highlighted with a red rectangular box. The code editor has a dark theme with three colored circular icons (red, yellow, green) at the top left.

Atualização e manutenção

IA

- Para um modelo de IA ser atualizado, em geral, é necessário treinar um novo modelo com novos dados e/ou ajustar hiperparâmetros. Isso pode ser um processo complexo e caro e **pode não gerar os resultados esperados**. Mesmo em casos de atualização/otimização de prompts, nos quais não há a necessidade de retreinar o modelo, a atualização pode gerar resultados inesperados.



Depuração de Erros

Tradicional

- Erros são mais fáceis de identificar e corrigir, pois o caminho de execução é claro, **você tem acesso ao código e valores das variáveis a cada passo**. Uma vez feitos os ajustes, a atualização e correção pode começar a surtir efeito "imediatamente".



Depuração de Erros

IA

- Erros podem ser difíceis de identificar e corrigir e **nem sempre estão explícitos nos dados de treinamento.** Eventualmente podem ser necessários ajustes nos dados, no modelo e/ou no processo de treinamento. Mesmo que você encontre exemplos nos dados de treinamento que podem ser causadores de erros, removê-los e retreinar o modelo **nem sempre garantirá que o erro será corrigido.**



Escalabilidade

Tradicional

- É possível escalar o código de maneira relativamente direta e eficiente, **adicionando mais recursos e/ou otimizando o código**. Há uma relação relativamente previsível entre entradas e recursos.



Escalabilidade

IA

- Não há tanta previsibilidade na escalabilidade de um modelo de IA pois as **"saídas" não são previsíveis e/ou diretamente proporcionais a algo tão mensurável da "entrada"**.



Viés e Justiça

Tradicional

- Todo **viés** está no código e pode ser identificado e corrigido diretamente e raramente serão introduzidos de maneira totalmente inconsciente.



```
if cat:  
    return "bad"  
elif dog:  
    return "good"  
else:  
    return "don't know"
```

Viés e Justiça

IA

- Viés pode ser introduzido nos dados de treinamento e/ou no modelo de **maneira inconsciente e pode ser difícil de identificar e corrigir.**
Devido ao alto volume, você nem sempre saberá que os dados poderão gerar vieses ou injustiças.



Alucinações

Tradicional

- Saídas inesperadas são geralmente resultado de **erros no código** e podem ser corrigidas diretamente. É possível entender o porquê dos erros.



```
if cat:  
    return "dog"  
elif dog:  
    return "dog"  
else  
    return ""
```

Alucinações

IA

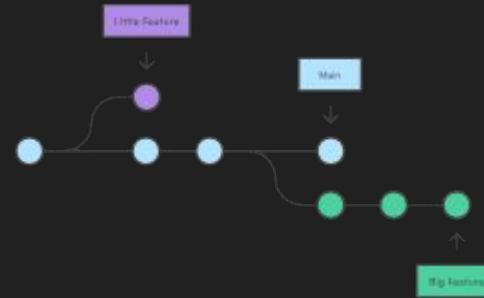
- Saídas inesperadas podem ser resultado de alucinações do modelo e podem ser **difícil**es de identificar e corrigir.



Versionamento

Tradicional

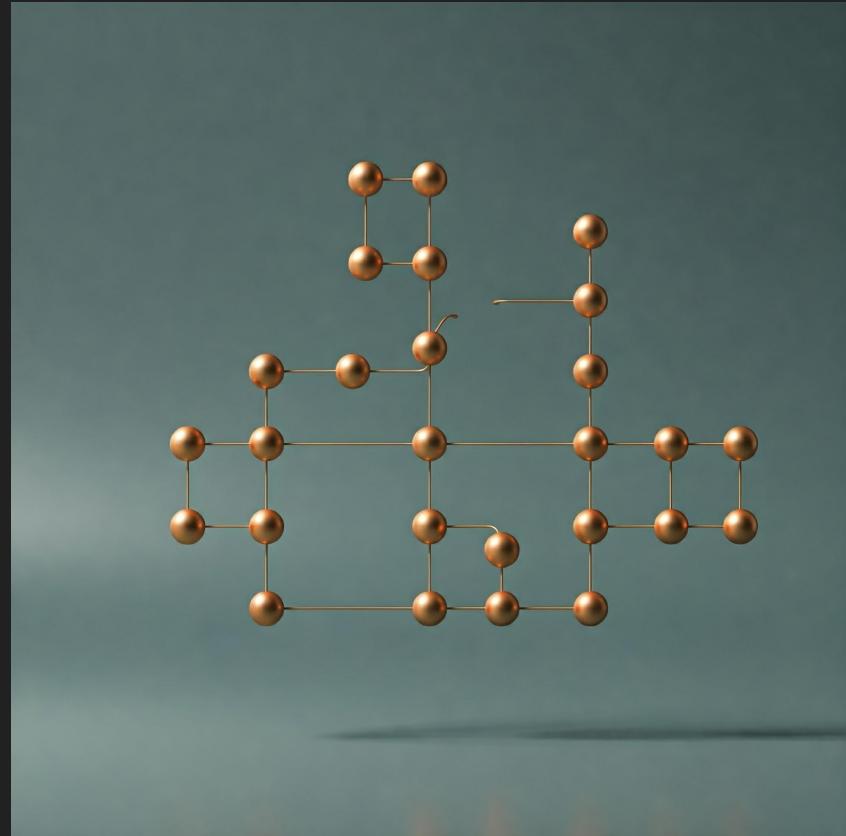
- É possível versionar o código e fazer **rollbacks de maneira relativamente direta e eficiente**. Se a versão anterior funcionava para determinados cenários e a nova versão não funciona para um cenário, voltar para a versão anterior **garantirá um comportamento similar ao passado**.



Versionamento

IA

- É possível versionar modelos de IA e voltá-los de forma similar, porém, caso haja necessidade de retreino, mesmo com os mesmos dados e parâmetros é possível ter um novo modelo relativamente diferente.



Testes

Tradicional

- É possível testar os **caminhos pré-definidos e entradas e saídas esperadas** de maneira relativamente direta e eficiente.



Testes

IA

- É possível testar os modelos de IA, porém, como as **saídas não são previsíveis**, os testes são **probabilísticos** e usam métricas de avaliação para determinar a qualidade do modelo. **Mesmo que o modelo tenha uma boa métrica de avaliação, ele pode não funcionar bem em cenários reais.** Há total dependência da **qualidade e quantidade dos dados** utilizados para o teste para **testes efetivos**.



Feedback

Tradicional

- Não é tão comum que os usuários dêem feedbacks diretos para as interações, o mais comum são reportes de **bugs específicos** de uma forma mais direta.



Feedback

IA

- É comum que os usuários dêem feedbacks para as interações e esses feedbacks **são fundamentais** para a melhoria do modelo. Esse feedbacks podem ser de “curtir” ou “não curtir” ou até notas de 1 a 5 com comentários. Os feedbacks normalmente são **mais complexos**.



Recursos

Tradicional

- **Desenvolvimento:** notebooks e computadores comuns são suficientes para desenvolver e testar a maioria dos sistemas.
- **Servidores:** VMs comuns com CPU, RAM e storage são suficientes para servir a maioria dos sistemas.



Recursos

IA

- **Treino:** é necessário um grande volume de dados e poder computacional. Em geral, é necessário utilizar GPUs e/ou TPUs e/ou serviços de nuvem.
- **Inferência:** recursos necessários normalmente são menores, mas ainda mais exigentes que a maioria dos softwares tradicionais.
- **Desenvolvimento em geral:** Mesmo GPUs "consumer grade" dedicadas com 24GB de VRAM podem não ser suficientes.

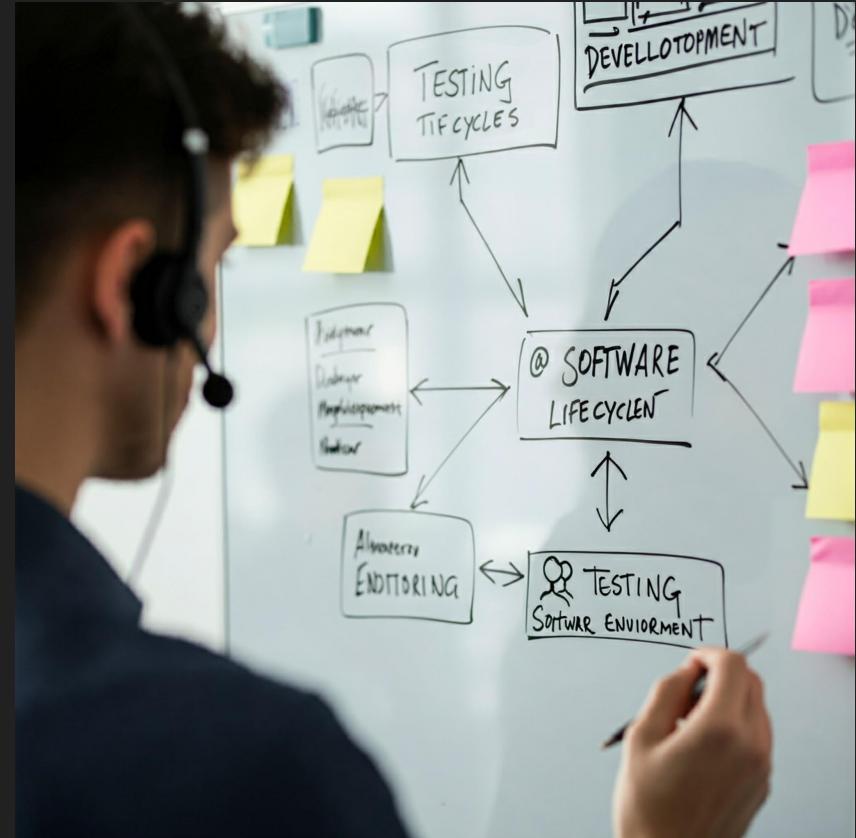


Ciclo de vida

Tradicional

1. Análise de requisitos
2. Desenho de arquitetura
3. Desenvolvimento
4. Testes
5. Deploy
6. Manutenção e atualizações

Os maiores esforços normalmente estão em 1, 3 e 6.

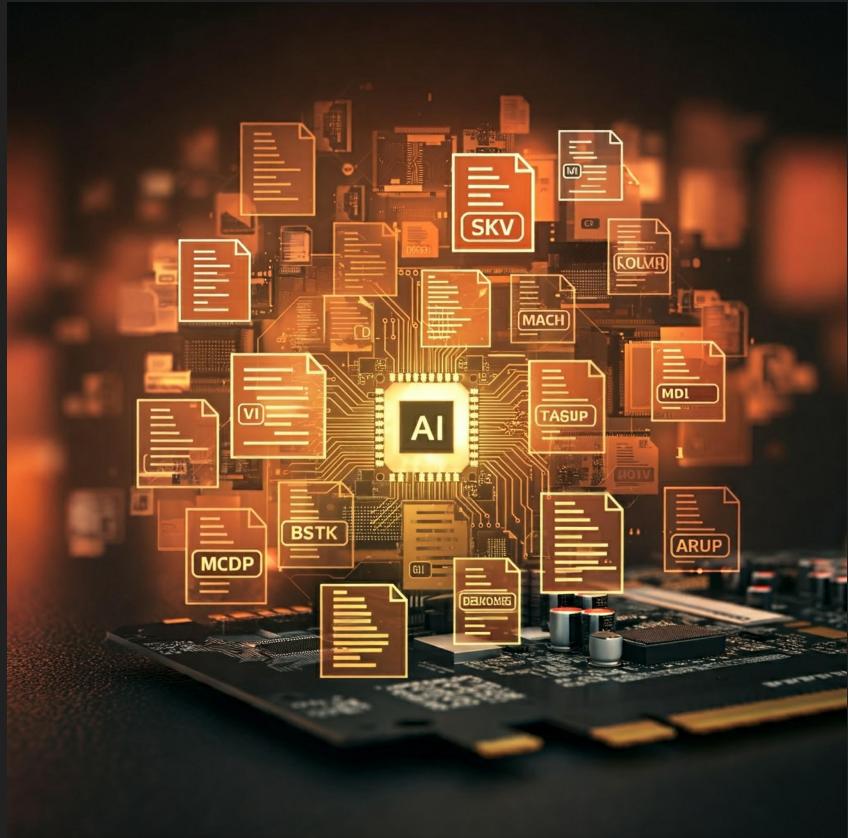


Ciclo de vida

IA

1. Análise de requisitos
2. Coleta e preparação de dados (treino, re-treino e validação/avaliação)
3. Treinamento/Fine-tuning do modelo
4. Avaliação e validação do modelo
5. Deploy do modelo
6. Monitoramento e re-treinamento

Os maiores esforços normalmente estão em 2, 3, 4 e 6.



Ciclo de vida

IA (LLMs)

1. Análise de requisitos
2. Coleta e preparação de dados
(validação/avaliação e possivelmente
fine-tuning)
3. Fine-tuning do modelo ou prompt engineering
4. Avaliação e validação do modelo
5. Deploy do modelo/API
6. Monitoramento e re-fine-tuning e/ou prompt
engineering

*Os maiores esforços normalmente estão em 3, 4 e
6.*



Obrigado!