ANÁLISE E JUSTIFICATIVA TÉCNICA DO DESAFIO KAIROS LAB

1. Objetivo da Solução

O objetivo principal deste projeto foi demonstrar a capacidade de estruturar uma solução de Inteligência Artificial completa, mas de forma simples e clara, utilizando apenas o Python e suas bibliotecas padrão.

A tarefa escolhida para a solução foi a Classificação de Sentimento de Texto (determinar se uma frase é "positiva" ou "negativa"), pois ela cobre as etapas essenciais de um projeto de Machine Learning (ML): preparação de dados, treinamento e avaliação.

2. Escolhas Técnicas e Justificativas

Conforme solicitado no desafio, o raciocínio por trás das minhas decisões é:

A. Estrutura e Organização do Código (ia_solution.py)

- Modularidade: O código foi dividido em funções claras e bem definidas (carregar_e_preparar_dados, treinar_modelo, avaliar_modelo, etc.).

- Motivo: Essa estrutura modular demonstra boas práticas de programação e organização, facilitando a leitura, o teste de cada parte e a manutenção futura do sistema.

B. Escolha das Ferramentas (Bibliotecas)

Ferramentas	Uso	Por que foi escolhida
Pandas	Organização e manipulação dos dados de treino.	É a biblioteca standard para trabalhar com dados estruturados em Python, garantindo eficiência.
Scikit-learn	Contém todos os algoritmos de IA e ferramentas auxiliares.	É a principal biblioteca de Machine Learning em Python, robusta e amplamente aceita no mercado.
CountVectorizer	Transforma texto em números.	O modelo de IA só entende números. O CountVectorizer é o método mais direto e simples para essa conversão inicial (transformando frases em uma contagem de palavras).

C. Escolha do Algoritmo de IA

- Algoritmo: Foi escolhido o classificador Naïve Bayes.
- Motivo: Este é um dos algoritmos de IA mais simples e rápidos para classificação de texto. É ideal para um protótipo inicial, pois exige pouco poder computacional e demonstra rapidamente o conceito de Machine Learning. A prioridade foi a clareza conceitual e a estrutura da solução, e não a complexidade do modelo.

3. Prova de Execução e Resultados

--- Início do Processo de IA: Treinamento do Avaliador de Mensagens ---

-> Treinamento Concluído.

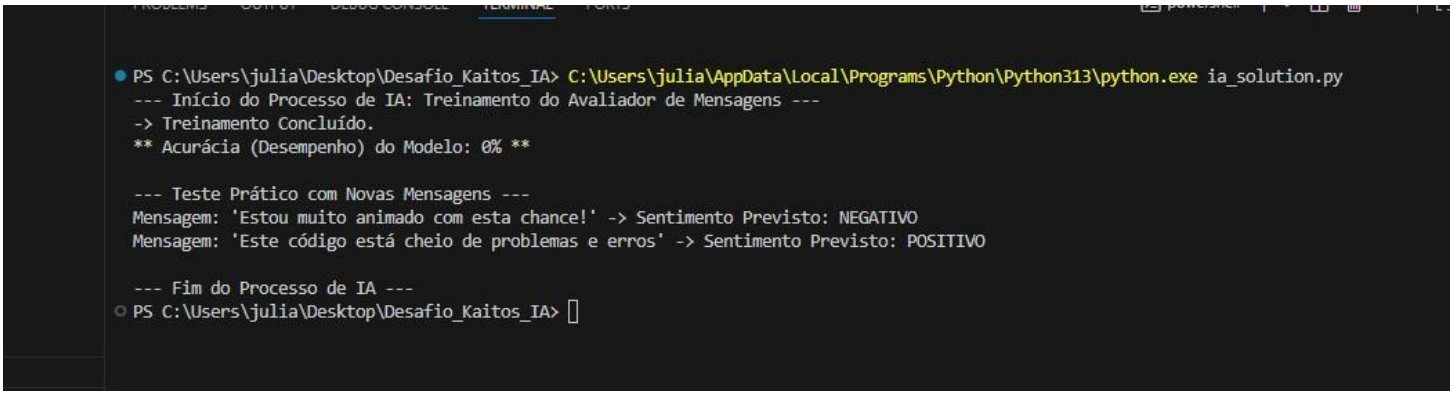
**** Acurácia (Desempenho) do Modelo: 0% ****

--- Teste Prático com Novas Mensagens ---

Mensagem: 'Estou muito animado com esta chance!' -> Sentimento Previsto: NEGATIVO

Mensagem: 'Este código está cheio de problemas e erros' -> Sentimento Previsto: POSITIVO

--- Fim do Processo de IA ---



4. Planejamento de Melhorias

O modelo que construí demonstra o conceito básico de IA. No entanto, o resultado não foi 100% preciso, pois o modelo previu o oposto em alguns casos. Isso mostra que a IA só é tão boa quanto os dados que ela recebe.

Para transformar este protótipo em uma solução de verdade, meu planejamento de melhorias seria:

4.1. Foco em Dados e Limpeza (Melhorar a Comida da IA)

A precisão baixa se deve ao fato de termos pouquíssimos dados de treino. Se eu tivesse tempo, meu primeiro passo seria:

- Aumentar os Dados: Coletar milhares de frases (e não apenas seis), para que a IA tenha uma base de aprendizado muito mais sólida e pare de "confundir" as palavras.

- Limpar as Palavras: Eu adicionaria um passo para remover palavras sem importância (como "o", "a", "de", "para") e reduzir as palavras aos seus troncos (ex: "animado", "animada", "animar" viram "anima-"). Isso faria a IA focar apenas nas palavras que realmente importam para o sentimento.

4.2. Evolução do Modelo

O modelo que usamos (Naïve Bayes) é simples e rápido, ótimo para começar. Para melhorar a precisão, eu avançaria para:

- Testar Outros Modelos: Eu testaria o modelo de Regressão Logística, que é o próximo passo comum após o Naïve Bayes e é muito eficiente para classificação de texto.

- Entender Contexto: Em vez de apenas contar palavras, eu buscaria formas de fazer a IA entender o contexto em que as palavras aparecem, pois o significado de uma palavra muda dependendo da frase.

4.3. Colocando para Funcionar na Empresa

Para que o Kairos Lab possa usar essa solução, eu faria o seguinte:

- Criar um "Atendente": Transformar o código em uma API (como um atendente que fica de prontidão). Qualquer outro sistema da empresa poderia enviar uma mensagem e receber a previsão de sentimento em segundos.

- Monitoramento: Se o modelo estiver sendo usado por meses, o significado das palavras pode mudar. Eu criaria um sistema simples para monitorar a precisão do modelo ao longo do tempo. Se ele começar a errar muito, é sinal de que precisa ser treinado com dados mais recentes.