Construindo o
Meu Primeiro Chatbot

Flávio Luiz Seixas

Quem sou eu?

Flávio Luiz Seixas

Professor

Pós-Graduação em Computação *strictu-sensu:*

- Inteligência Artificial
- Sistemas Computacionais

MPEA
MESTRADO PROFISSIONAL EM ENFERMAGEM ASSISTENCIAL
uff

Doutorado/Mestrado Profissionais:

- Enfermagem assistencial
- Saúde materno-infantil

Colaborador:

- Ciências cardiovasculares

Objetivo

Abordar desde a programação básica em Python até a implementação de chatbots utilizando modelos de linguagem ampla (LLMs).

Módulo 1. Programação básica em Python (variáveis, tipos de dados, operadores, estruturas de controle e funções). Exercícios práticos usando o Google Colab.

Módulo 2. Funções básicos de NLP. Tokenização e segmentação de palavras. Filtragem e normalização, vetorização. Classificação e rotulação.

Módulo 3. Construção de um chatbot básico usando modelos de linguagem ampla aplicando uma informação de contexto (LLM-QA).

Módulo 4. Construção de um chatbot usando modelos de linguagem ampla aplicando geração aumentada por recuperação, ou seja, a combinação de modelos de geração de texto com técnicas de recuperação de informações, onde o modelo busca dados relevantes em uma base de conhecimento e os utiliza para gerar respostas mais precisas e informadas (LLM-QA + RAG).

Links importantes

Google Colab

https://colab.research.google.com/

Código do Curso

https://github.com/flavioluizseixas/curso-chatbot

Alinhando a metodologia de ensino

Modelo tradicional

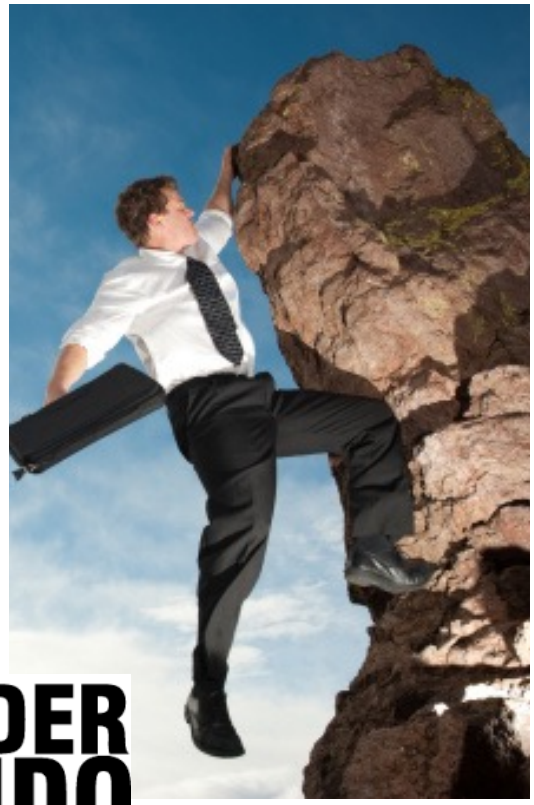
Ensino centrado no professor

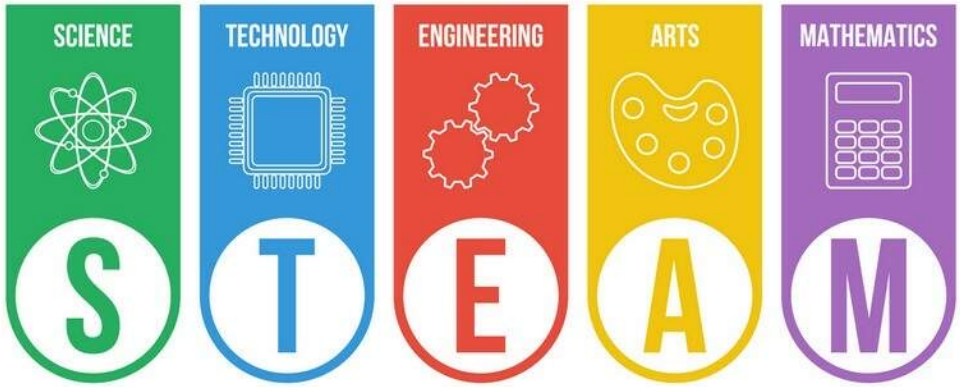
Metodologias ativas

Ensino centrado no estudante

Ensino baseado em desafios

- O estudante participa de forma **ativa** na resolução de um **problema real e relevante**.
- Por meio da prática e da investigação, os estudantes são envolvidos nas áreas de STEAM para concluir os seus **projetos**.
- O ensino se torna mais interessante e próximo à realidade do estudante, que incorpora os projetos de vida à sala de aula.





APRENDER FAZENDO

Módulo 1

Projeto 1. Calculadora da idade.

Crie um programa que peça ao usuário o seu nome e o ano de nascimento. O programa deve calcular a idade e exibir uma mensagem como “Olá [nome], você tem [idade] anos.”.

Projeto 2. Lista de compras simples.

Desenvolva um programa que permita ao usuário adicionar três itens a uma lista de compras. Depois de adicionar os itens, o programa deve exibir todos eles com uma mensagem como “Sua lista de compras contêm: [item1], [item2], [item3].”

Projeto 3. Jogo de adivinhação.

Crie um programa em que o usuário deve adivinhar um número entre 1 e 10. O programa deve comparar a entrada do usuário com um número definido pelo programador e exibir mensagens como “Você acertou!” ou “Tente novamente”.

Módulo 1. Para casa

Desenvolva um sistema de controle de tarefas em Python, onde o usuário pode adicionar tarefas, definir a prioridade de cada tarefa (alta, média ou baixa), listar as tarefas pendentes e marcar tarefas como concluídas. O sistema deve exibir as tarefas de acordo com a prioridade.

Funcionalidades:

1. Adicionar uma nova tarefa: O usuário insere o nome da tarefa e escolhe a prioridade (alta, média ou baixa).
2. Listar tarefas pendentes: Exibe todas as tarefas, organizadas por prioridade.
3. Marcar uma tarefa como concluída: Permite ao usuário remover uma tarefa da lista de pendentes.
4. Sair do programa: O usuário pode sair do sistema a qualquer momento.

Módulo 2

Projeto.

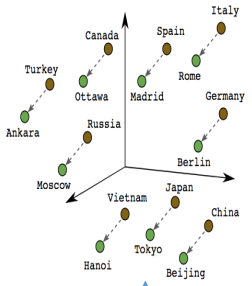
Desenvolva um analisador de sentimentos simples para classificar comentários de clientes como positivos ou negativos.

O projeto deve passar pelas seguintes etapas:

1. **Tokenização:** Separar o texto em palavras (tokens).
2. **Filtragem e Normalização:** Remover stop words, aplicar lowercasing e normalizar o texto (por exemplo, remover pontuação).
3. **Contagem de Palavras:** Implementar a contagem de palavras para entender a frequência de termos no texto.
4. **Vetorização:** Converter o texto em uma representação numérica (bag of words).
5. **Classificação:** Treinar um classificador simples (como Naive Bayes) para identificar o sentimento do texto.

2 NLP

Representação vetorial



Texto

Tokenização e segmentação de frases
- Palavras
- Sentenças
- Símbolos

Filtragem e normalização
- Stopwords
- Lematização
- Stemming

Vetorização
- TF-IDF
- Word2Vec
- BERT

Controle de contexto e sentido

Texto classificado

Módulo 2. Para casa

Projeto.

Desenvolva um classificador de reviews de filmes que analisa e classifica as críticas como positivas, negativas ou neutras. O projeto envolve processamento de linguagem natural (NLP) para transformar texto em dados estruturados e a construção de um modelo de aprendizado de máquina.

Etapas do Projeto:

1. **Coleta e Preparação dos Dados:** Obter um conjunto de dados de reviews de filmes (pode ser um dataset simples do NLTK ou um CSV).
2. **Tokenização e Filtragem:** Tokenizar as reviews, remover stop words, pontuação e aplicar técnicas de normalização.
3. **Vetorização:** Converter as reviews em vetores usando técnicas como Bag of Words ou TF-IDF.
4. **Classificação de Sentimento:** Treinar um classificador para prever o sentimento (positivo, negativo ou neutro) com base nos reviews vetorizados. Pode-se usar algoritmos como Naive Bayes, SVM ou Random Forest.
5. **Avaliação do Modelo:** Avaliar o desempenho do modelo usando métricas como acurácia, precisão, recall e F1-score.
6. **Teste e Demonstração:** Permitir que o usuário insira uma nova review e classificar automaticamente o sentimento.

Módulo 3

Projeto.

Desenvolva um chatbot que possa responder a perguntas relacionadas à pandemia de COVID-19 com base em informações de um contexto fornecido. Utilizando a biblioteca transformers e um modelo pré-treinado em português, o projeto irá explorar técnicas de NLP para construir um sistema de perguntas e respostas eficiente e personalizado.

Etapas do Projeto:

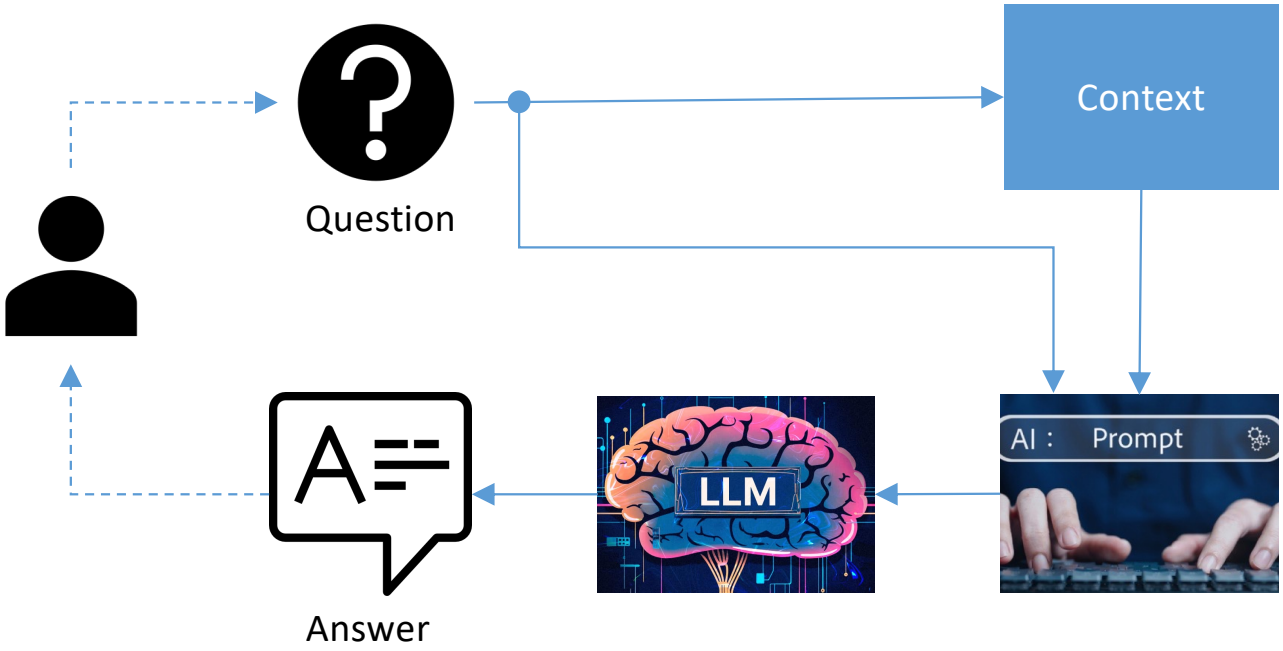
1. **Coleta e Preparação de Dados:** Obter textos em português sobre a pandemia de COVID-19 (como artigos ou trechos de notícias). Estruturar o contexto em que o modelo vai buscar as respostas.
2. **Exploração do Modelo:** Usar o modelo pré-treinado pierreguillou/bert-large-cased-squad-v1.1-portuguese para tarefas de question-answering. Compreender como o modelo processa perguntas e como extrai as respostas com base no contexto fornecido.
3. **Integração com Pipeline de Perguntas e Respostas:** Implementar o pipeline de perguntas e respostas usando transformers e o modelo BERT, permitindo ao chatbot identificar a resposta correta no texto.

Módulo 3

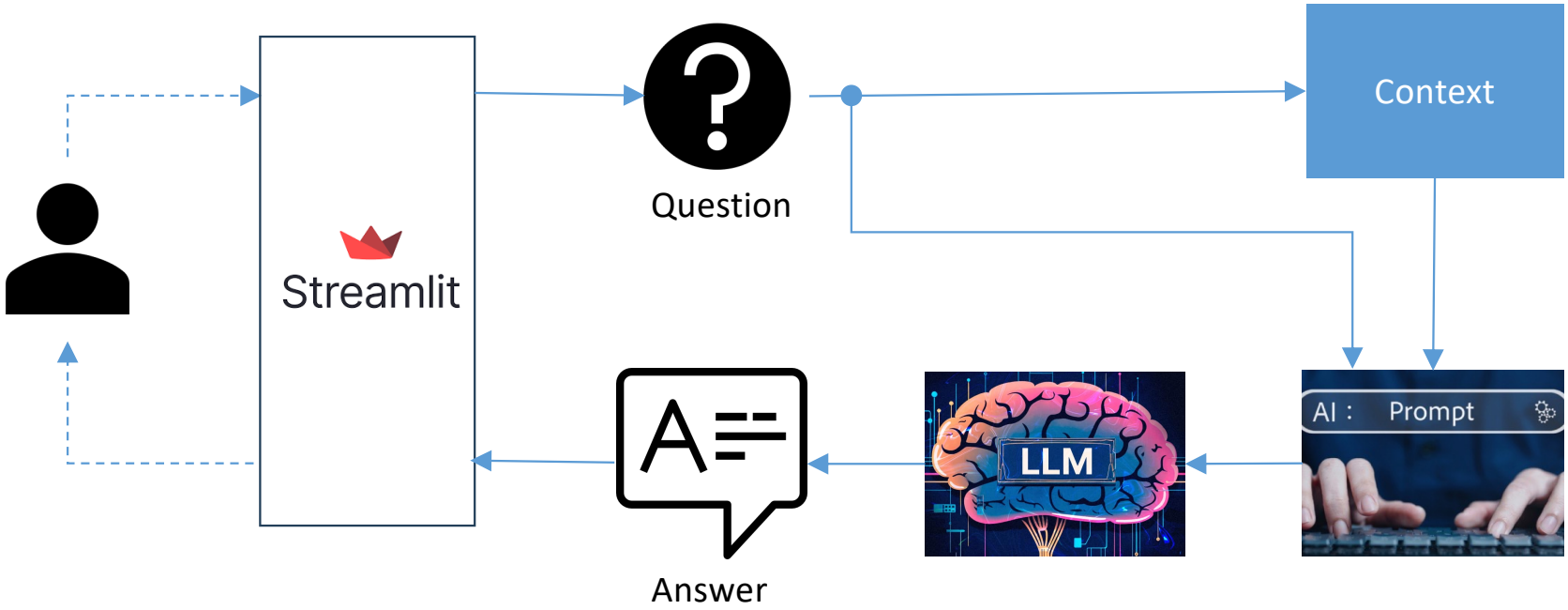
Etapas do Projeto:

4. **Construção de uma Interface Simples:** Criar uma interface básica (pode ser via terminal, interface gráfica ou até em uma aplicação web) onde o usuário possa digitar perguntas e receber respostas.
5. **Testes e Refinamento:** Testar o chatbot com várias perguntas relacionadas à pandemia para garantir que ele retorna respostas precisas. Ajustar o modelo ou o contexto, se necessário, para melhorar a precisão.

3 LLM.QA



3 LLM.QA + UI



Módulo 3. Para casa

Expandir o contexto, adicionando mais informações sobre outras pandemias ou tópicos de saúde. Implementar suporte para múltiplos contextos, com o chatbot identificando automaticamente o tópico relevante.

Módulo 4

Empresas e analistas frequentemente precisam extrair informações específicas de relatórios financeiros ou documentos extensos, como PDFs de resultados de empresas. No entanto, esses documentos são densos e podem demandar muito tempo para a obtenção de dados específicos. A proposta deste projeto é desenvolver um sistema automatizado que realize a recuperação de informações de documentos PDF, utilizando técnicas de inteligência artificial e processamento de linguagem natural (NLP), integrando modelos de embeddings e um pipeline de perguntas e respostas (QA).

Objetivo: Desenvolver uma aplicação de perguntas e respostas baseada em documentos PDF, permitindo consultas sobre informações específicas contidas nesses documentos. O sistema utilizará técnicas de split e vetorização de texto, juntamente com um modelo de linguagem para responder às perguntas com base em conteúdo embasado e relevante.

Método: RAG é uma técnica que combina a recuperação de informações com a geração de texto utilizando modelos de linguagem. Ela é amplamente usada em sistemas de perguntas e respostas para lidar com grandes volumes de informações não estruturadas, como documentos extensos, e fornecer respostas precisas e contextuais.

Módulo 4

Metodologia:

1. **Carregamento e Processamento do Documento:**

- Utilizar o módulo PyPDFLoader para carregar documentos PDF e extrair o texto.
- Realizar a divisão do texto em segmentos utilizando o RecursiveCharacterTextSplitter, mantendo a integridade do conteúdo.

2. Criação de Embeddings e Armazenamento Vetorial:

- Utilizar o modelo GPT4AllEmbeddings para gerar embeddings vetoriais dos segmentos de texto.
- Armazenar esses vetores em uma base de dados vetorial gerenciada pelo Chroma, permitindo buscas semânticas.

3. Mecanismo de Perguntas e Respostas:

- Implementar um pipeline que inclui busca por similaridade, formatação do contexto relevante, e invocação de um modelo de linguagem (LlamaCpp) para gerar respostas precisas baseadas nas informações extraídas.

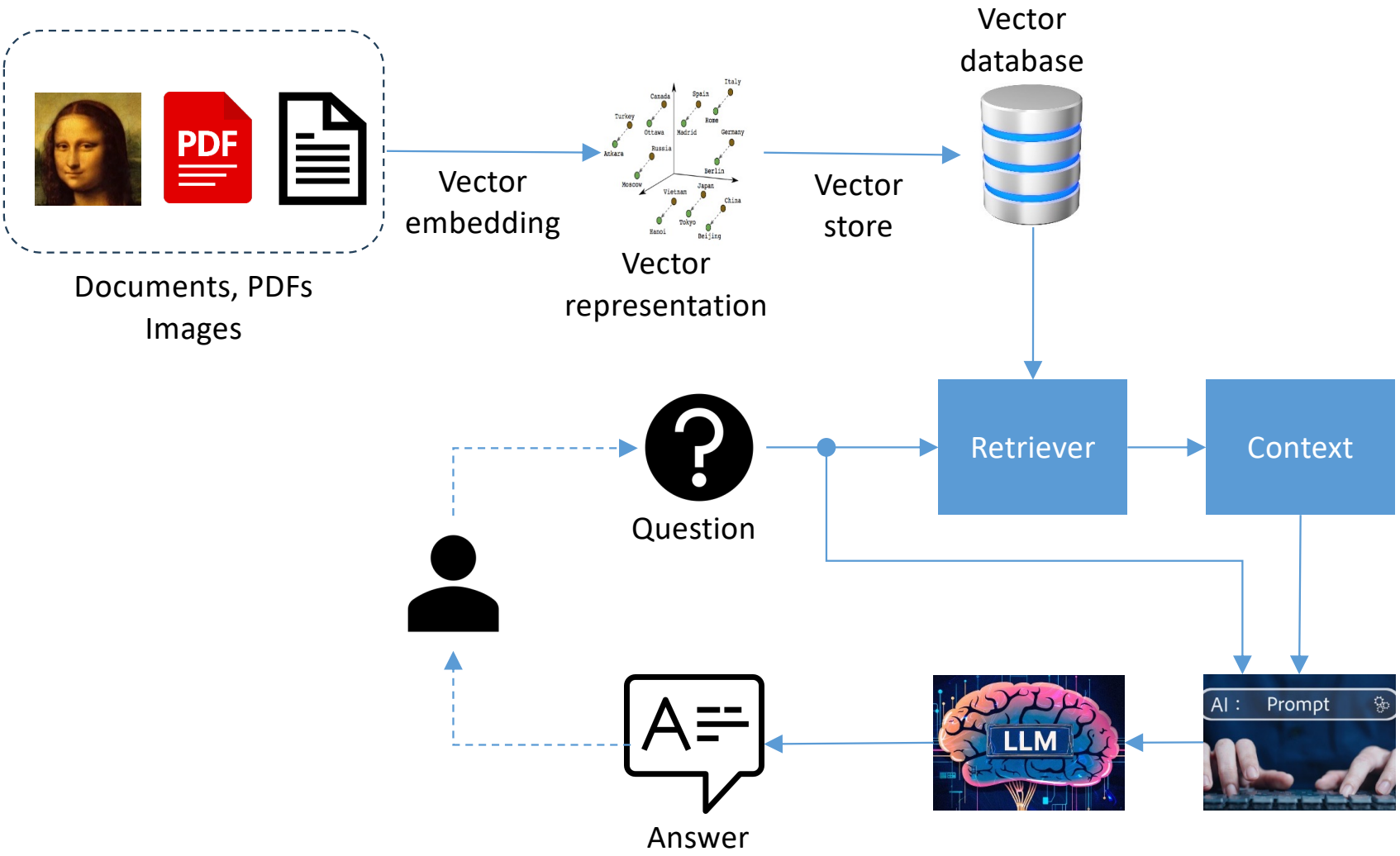
Módulo 4

Metodologia:

4. Template de de Prompt e Cadeia de Execução:

- Criar um template de prompt dinâmico utilizando o PromptTemplate, adaptado para formatar as respostas com base no contexto extraído e na pergunta feita.
- Integrar o fluxo completo em uma cadeia de execução (qa_chain) para permitir consultas automatizadas.

4 LLM.QA + RAG


PDF
Documents, PDFs
Images
Vector
embedding
Vector
representation
Vector
database
Vector
store
Retriever
Context
Question
Answer
LLM
AI : Prompt

Módulo 4. Para casa

Empresas que oferecem produtos tecnológicos complexos, como softwares ou equipamentos, enfrentam o desafio de manter uma base de suporte técnico eficiente. Muitas vezes, a documentação de produtos é extensa e os usuários, ao buscarem suporte, encontram dificuldades para localizar respostas específicas. Este projeto propõe a criação de um sistema automatizado de perguntas e respostas que utilize a técnica RAG (Retrieval-Augmented Generation) para fornecer respostas precisas e contextualizadas a partir de manuais e guias técnicos.

Objetivo: Desenvolver uma aplicação de suporte técnico automatizado que permita a consulta de informações específicas em documentos técnicos. A solução utilizará técnicas de recuperação de informações combinadas com modelos de linguagem para gerar respostas rápidas e precisas.

Resultados Esperados:

- Um sistema que possibilite consultas a manuais e documentos de suporte técnico, com respostas geradas de forma precisa e contextualizada.
- Redução do tempo de atendimento a usuários, otimizando processos de suporte.
- Aumento da eficiência e precisão na recuperação de informações técnicas.


</>
CMS
CSS
Js
C++
PHP
HTML
(:

Aprendizado de máquina supervisionado

Flávio Luiz Seixas

Aprendizado de máquina supervisionado

Desenvolver um modelo de aprendizado de máquina que possa prever se um paciente com diabetes será readmitido no hospital de 30 dias após a alta. A previsão visa ajudar hospitais a implementar intervenções preventivas e otimizar recursos.

1. Coleta de Dados:

- Utilizar um conjunto de dados público, como o Diabetes Readmission Dataset disponível no UCI Machine Learning Repository.
- O conjunto de dados inclui informações sobre hospitalizações, condições de saúde dos pacientes, medicações e resultados.

2. Pré-processamento dos Dados:

- Limpeza dos dados, tratando valores nulos e categóricos.
- Engenharia de atributos: Seleção e criação de novas features relevantes (como frequência de hospitalizações anteriores, controle glicêmico, medicações prescritas, etc.).

Aprendizado de máquina supervisionado

3. Divisão dos Dados:

- Dividir o dataset em dados de treino e teste (geralmente 70%/30%).

4. Treinamento do Modelo:

- Implementar algoritmos de classificação como:
  - ❑ Random Forest
  - ❑ Logistic Regression
  - ❑ Support Vector Machines (SVM)
- Comparar o desempenho dos modelos usando métricas como acurácia, precisão, recall, F1-score, e curva ROC.

5. Avaliação e Ajuste de Hiperparâmetros:

- Realizar validação cruzada para ajustar hiperparâmetros usando técnicas como GridSearchCV ou RandomizedSearchCV.

Aprendizado de máquina supervisionado

6. Validação e Teste:

- Avaliar o modelo no conjunto de dados de teste e verificar seu desempenho.

7. Interpretação dos Resultados:

- Identificar as features mais importantes para a decisão do modelo, como tipo de medicação, número de internações anteriores e níveis de glicose.

8. Implementação de Ações Preventivas:

- Propor estratégias baseadas nos resultados do modelo para reduzir as readmissões hospitalares.

Aprendizado de máquina supervisionado

Tecnologias e Bibliotecas Utilizadas:

- Python
- Pandas e NumPy para manipulação de dados
- Scikit-learn para construção dos modelos
- Matplotlib e Seaborn para visualização de dados
- Jupyter Notebook para documentação e execução do projeto