09

Faça como eu fiz: criando uma aplicação com Streamlit

Nesta aula, você aprendeu a construir uma aplicação web interativa com Streamlit, que utiliza um agente inteligente desenvolvido com LangChain para responder perguntas sobre dados. Essa aplicação:

- Carrega um arquivo CSV enviado pelo usuário,
- Exibe um resumo inicial dos dados,
- Permite gerar relatórios completos automaticamente,
- Permite criar gráficos com base em perguntas em linguagem natural,
- Permite fazer perguntas sobre os dados,
- Pode ser acessada via navegador e hospedada gratuitamente.

Passamos por todas as etapas, desde a criação do ambiente local até o deploy no Streamlit Cloud. Se você ainda não reproduziu o que foi feito em aula, chegou a hora de colocar tudo em prática!

Vamos relembrar o passo a passo?

Criamos uma pasta no computador projeto-langchain

Abrimos o VScode e clicamos em File > Open Folder

Inicializamos o ambiente virtual abrindo um terminal (Terminal > New Terminal) e executando o comando:

```
python3 -m venv .venv
```

COPIAR CÓDIGO

Ativamos o ambiente virtual com:

```
.venv\Scripts\Activate
```

COPIAR CÓDIGO

Criamos um arquivo `requirements.txt` e adicionamos os pacotes necessários para o projeto:

```
langchain==0.3.22
langchain-groq==0.3.2
langchain-core==0.3.50
langchain-experimental==0.3.4
langchain-community==0.3.20
pandas==2.2.3
tabulate==0.9.0
streamlit==1.44.1
```

```
matplotlib==3.10.1
seaborn==0.13.2
python-dotenv==1.0.1
```

COPIAR CÓDIGO

Instalamos todas as bibliotecas com o comando:

```
pip install -r requirements.txt
```

COPIAR CÓDIGO

Criamos o arquivo `.env` para armazenar com segurança a chave da API do Groq:

```
GROQ_API_KEY = "ADICIONE SUA CHAVE AQU
```

COPIAR CÓDIGO

Criamos o arquivo ferramentas.py, responsável por definir todas as ferramentas que o agente poderá utilizar:

```
import os from dotenv import load_dote

# Obtenção da chave de api
load_dotenv() GROQ_API_KEY = os.getenv

# Configurações do LLM
llm = ChatGroq( api_key=GROQ_API_KEY,
```

```
#Relatório informações
@tool
def informacoes_dataframe(pergunta: st
# Coleta de informações
shape = df.shape
columns = df.dtypes
nulos = df.isnull().sum()
nans_str = df.apply(lambda col: col[~c
duplicados = df.duplicated().sum()

Prompt de resposta
template_resposta = PromptTemplate(

    template="""
    Você é um analista de dados encarr
    a partir de uma {pergunta} feita p

    A seguir, você encontrará as infor

    ================ INFORMAÇÕES DO D

    Dimensões: {shape}

    Colunas e tipos de dados: {columns

    Valores nulos por coluna: {nulos}

    Strings 'nan' (qualquer capitaliza

    Linhas duplicadas: {duplicados}

    ==================================
```

```
    Com base nessas informações, escre

    1. Um título: ## Relatório de info
    2. A dimensão total do DataFrame;
    3. A descrição de cada coluna (inc
    4. As colunas que contêm dados nul
    5. As colunas que contêm strings '
    6. E a existência (ou não) de dado
    7. Escreva um parágrafo sobre anál
    esses dados.
    8. Escreva um parágrafo sobre trat
    """,
    input_variables=["pergunta","shape

cadeia = template_resposta | llm | Str

resposta = cadeia.invoke({
          "pergunta": pergunta,
          "shape": shape,
          "columns": columns,
          "nulos": nulos,
          "nans_str": nans_str,
          "duplicados": duplicados
    })

return resposta

# Relatório estatístico

@tool
def resumo_estatistico(pergunta: str,
""" Utilize esta ferramenta sempre que
```

```
# Coleta de estatísticas descritivas
estatisticas_descritivas = df.describe

# Prompt de resposta
template_resposta = PromptTemplate(
    template="""
    Você é um analista de dados encarr
    a partir de uma {pergunta} feita p

    A seguir, você encontrará as estat

    ================ ESTATÍSTICAS DES

    {resumo}

    ==================================

    Com base nesses dados, elabore um
    os principais pontos dos resultado

    1. Um título: ## Relatório de esta
    2. Uma visão geral das estatística
    3. Um paráfrago sobre cada uma das
    4. Identificação de possíveis outl
    5. Recomendações de próximos passo
    """,
    input_variables=["pergunta", "resu
)

cadeia = template_resposta | llm | Str

resposta = cadeia.invoke({"pergunta":
```

```
return resposta

# Gerador de gráficos
@tool
def gerar_grafico(pergunta: str, df: p
""" Utilize esta ferramenta sempre que

# Captura informações sobre o datafram
colunas_info = "\n".join([f"- {col} ({
amostra_dados = df.head(3).to_dict(ori

#Template otimizado para geração de có
template_resposta = PromptTemplate(
        template="""
        Você é um especialista em visu

        ## Solicitação do usuário:
        "{pergunta}"

        ## Metadados do DataFrame:
        {colunas}

        ## Amostra dos dados (3 primei
        {amostra}

        ## Instruções obrigatórias:
        1. Use as bibliotecas `matplot
        2. Defina o tema com `sns.set_
        3. Certifique-se de que todas
        4. Escolha o tipo de gráfico a
        - **Distribuição de variáveis
        - **Distribuição de variáveis
        - **Comparação entre categoria
```

```
        - **Relação entre variáveis**:
        - **Séries temporais**: `linep
        5. Configure o tamanho do gráf
        6. Adicione título e rótulos (
        7. Posicione o título à esquer
        8. Mantenha os ticks eixo X se
        9. Remova as bordas superior e
        10. Finalize o código com `plt

        Retorne APENAS o código Python

        Código Python:
        """, input_variables=["pergunta
    )

    # Gera o código
cadeia = template_resposta | llm | StrO
codigo_bruto = cadeia.invoke({
        "pergunta": pergunta,
        "colunas": colunas_info,
        "amostra": amostra_dados
    })

    # Limpa o código gerado
codigo_limpo = codigo_bruto.replace("`

    # Tenta executar o código para val
exec_globals = {'df': df, 'plt': plt,
exec_locals = {}
exec(codigo_limpo, exec_globals, exec_

    # Mostra o gráfico
fig = plt.gcf()
```

```
st.pyplot(fig)

return ""

# Função para criar ferramentas
def criar_ferramentas(df):
ferramenta_informacoes_dataframe = Too
name="Informações Dataframe", func=lam
description="""Utilize esta ferramenta
ferramenta_resumo_estatistico = Tool(
name="Resumo Estatístico",
func=lambda pergunta:resumo_estatistic
description="""Utilize esta ferramenta
return_direct=True)
ferramenta_gerar_grafico = Tool(
name="Gerar Gráfico",
func=lambda pergunta:gerar_grafico.run
description="""Utilize esta ferramenta
'crie um gráfico', 'plote', 'visualize
return_direct=True)
ferramenta_codigos_python = Tool(
name="Códigos Python",
func=PythonAstREPLTool(locals={"df": d
description="""Utilize esta ferramenta
Exemplos de uso incluem: "Qual é a méd

return [
    ferramenta_informacoes_dataframe,
    ferramenta_resumo_estatistico,
    ferramenta_gerar_grafico,
    ferramenta_codigos_python
]
```

COPIAR CÓDIGO

Criamos o arquivo App.py, que define a interface da aplicação com Streamlit. Esse arquivo:

- Define a estrutura visual da página, com título e descrição do assistente;
- Permite o upload de arquivos .csv pelo usuário;
- Exibe as primeiras linhas do dataset;
- Oferece botões para gerar relatórios automáticos;
- Permite fazer perguntas em linguagem natural e gerar gráficos com apenas uma instrução;
- Usa o `AgentExecutor` para orquestrar o comportamento do agente e conectar com as ferramentas definidas.

```
import streamlit as st
import pandas as pd
import os
from langchain_groq import ChatGroq
from langchain.prompts import PromptTe
from langchain.agents import create_re
from langchain.agents import AgentExec
from ferramentas import criar_ferramen

# Inicia o app
st.set_page_config(page_title="Assiste
```

```
# Descrição da ferramenta
st.info(""" Este assistente utiliza um
📄 Gerar relatórios automáticos:
Relatório de informações gerais: apres
Relatório de estatísticas descritivas:
🔎 Fazer perguntas simples sobre os da
📊 Criar gráficos automaticamente com
Ideal para analistas, cientistas de da

# Upload do CSV
st.markdown("### 📁 Faça upload do seu
arquivo_carregado = st.file_uploader("
if arquivo_carregado:
df = pd.read_csv(arquivo_carregado) st

# LLM
GROQ_API_KEY = os.getenv("GROQ_API_KEY
llm = ChatGroq(
    api_key=GROQ_API_KEY,
    model_name="llama3-70b-8192",
    temperature=0
)

# Ferramentas
tools = criar_ferramentas(df)

# Prompt react
df_head = df.head().to_markdown()

prompt_react_pt = PromptTemplate(
    input_variables=["input", "agent_s
    partial_variables={"df_head": df_h
    template="""
```

```
    Você é um assistente que sempre re

    Você tem acesso a um dataframe pan
    Aqui estão as primeiras linhas do

    {df_head}

    Responda às seguintes perguntas da

    Para isso, você tem acesso às segu

    {tools}

    Use o seguinte formato:

    Question: a pergunta de entrada qu
    Thought: você deve sempre pensar n
    Action: a ação a ser tomada, deve
    Action Input: a entrada para a açã
    Observation: o resultado da ação
    ... (este Thought/Action/Action In
    Thought: Agora eu sei a resposta f
    Final Answer: a resposta final par
    Quando usar a ferramenta_python: f

    Comece!

    Question: {input}
    Thought: {agent_scratchpad}"""
)

# Agente
agente = create_react_agent(llm=llm, t
```

```
orquestrador = AgentExecutor(agent=agen
                             tools=tools
                             verbose=Tru
                             handle_pars

# AÇÕES RÁPIDAS
st.markdown("---")
st.markdown("## ⚡ Ações rápidas")

# Relatório de informações gerais
if st.button("📄 Relatório de informaç
    with st.spinner("Gerando relatório
        resposta = orquestrador.invoke
        st.session_state['relatorio_ger

# Exibe o relatório com botão de downlo
if 'relatorio_geral' in st.session_stat
    with st.expander("Resultado: Relato
        st.markdown(st.session_state['r

        st.download_button(
            label="📥 Baixar relatório
            data=st.session_state['rela
            file_name="relatorio_inform
            mime="text/markdown"
        )

# Relatório de estatísticas descritivas
if st.button("📄 Relatório de estatíst
    with st.spinner("Gerando relatório
        resposta = orquestrador.invoke
        st.session_state['relatorio_est
```

```
# Exibe o relatório salvo com opção de
if 'relatorio_estatisticas' in st.sess
    with st.expander("Resultado: Relat
        st.markdown(st.session_state['

        st.download_button(
            label="📥 Baixar relatóric
            data=st.session_state['rel
            file_name="relatorio_estat
            mime="text/markdown"
        )

PERGUNTA SOBRE OS DADOS
st.markdown("---")
st.markdown("## 🔎 Perguntas sobre os
pergunta_sobre_dados = st.text_input("
if st.button("Responder pergunta", key=
    with st.spinner("Analisando os dad
        resposta = orquestrador.invoke
        st.markdown((resposta["output"


# GERAÇÃO DE GRÁFICOS
st.markdown("---")
st.markdown("## 📊 Criar gráfico com b

pergunta_grafico = st.text_input("Digi
if st.button("Gerar gráfico", key="ger
    with st.spinner("Gerando o gráfico
        orquestrador.invoke({"input": 
```

COPIAR CÓDIGO

Executamos a aplicação localmente, digitando o comando:

```
streamlit run App.py
```

COPIAR CÓDIGO

O Streamlit abrirá a aplicação no navegador, permitindo que você interaja com ela como se fosse um site pronto.

Publicamos a aplicação na nuvem, seguindo os passos:

- Criamos um repositório no GitHub contendo os arquivos `App.py`, `ferramentas.py` e `requirements.txt`;
- Acessamos https://share.streamlit.io (https://share.streamlit.io) e conectamos nosso repositório;
- Após o deploy, acessamos as configurações e adicionamos a chave da API do Groq no menu Secrets;
- A aplicação foi disponibilizada publicamente via link, pronta para ser usada ou apresentada.

Opinião do instrutor

Essa aula mostrou na prática como transformar uma análise de dados em uma ferramenta real acessível por qualquer navegador. Ao integrar IA com uma interface simples como a do Streamlit, conseguimos oferecer uma solução poderosa que acelera diagnósticos, análises e tomadas de decisão. É um ótimo exemplo de como o LangChain pode ser usado para aplicações funcionais prontas para uso.