Sistema de Previsão de Enchentes

Integrantes:

- Douglas Souza Felipe
- Richard Marques
- Lucas Medeiros Leite
- Evelyn Z
- Luis Fernando dos Santos Costa

GIT: https://github.com/Douglas-Felipe/FIAP-GS
VIDEO : https://youtu.be/D-JmoUT5vF8
Participar da GS: SIM

Introdução

Este projeto tem como objetivo desenvolver um sistema capaz de prever a ocorrência de enchentes com base nos dados de precipitação dos últimos 7 dias, utilizando um modelo de Machine Learning. A intenção é fornecer uma ferramenta que possa auxiliar em sistemas de alerta precoce e na tomada de decisões para mitigação de danos.

Enchentes são desastres naturais que causam grandes prejuízos materiais e humanos. A capacidade de prever esses eventos com antecedência é crucial para minimizar seus impactos. Este projeto busca resolver esse problema analisando dados históricos de precipitação e ocorrência de enchentes para treinar um modelo preditivo robusto e eficaz.

Desenvolvimento

Arquitetura da Solução

O sistema de previsão de enchentes é composto por módulos interconectados, cada um desempenhando um papel crucial desde a coleta de dados até a apresentação da previsão ao usuário. A arquitetura foi desenhada para ser modular e escalável.

A visão geral do projeto compreende os seguintes componentes:

1. **Coleta de Dados (Hardware - ESP32):** Um microcontrolador ESP32 é responsável pela coleta (real ou simulada) de dados de precipitação.

2. **Processamento de Dados (Datasets):** Os dados coletados, juntamente com dados históricos de enchentes, são processados bom base no treinamento do modelo.
3. **Modelagem e Treinamento de Machine Learning (Modelo):** Um modelo de classificação é treinado utilizando os dados de precipitação para prever a probabilidade de enchentes.
4. **API de Previsão (API):** Uma API REST expõe o modelo treinado, permitindo que outros sistemas ou interfaces consumam as previsões.
5. **Dashboard de Visualização (Streamlit):** Uma interface web interativa permite aos usuários inserir dados de precipitação e visualizar as previsões do modelo.

A seguir, detalhamos cada um desses componentes.

1. Coleta de Dados (Hardware - ESP32)

O módulo de coleta de dados utiliza um microcontrolador ESP32 para medir a precipitação.

- **Sensor:** Foi utilizado um sensor de distância ultrassônico HC-SR04 para medir, de forma aproximada, a profundidade da água em um recipiente simples, convertendo essa medida em precipitação em milímetros.
  - **Justificativa da Escolha do Sensor:** O HC-SR04 é de fácil simulação no Wokwi (ambiente de simulação utilizado) e não requer hardware especializado em medição de precipitação, sendo adequado para um protótipo.
  - **Operação em 3.3V:** O ESP32 opera com pinos GPIO a 3.3V. Para garantir a compatibilidade e segurança dos níveis de tensão, o HC-SR04 foi alimentado com 3.3V. Isso reduz sua faixa máxima confiável para cerca de 200cm (comparado a 400cm em 5V), mas é suficiente para a maioria dos pluviômetros caseiros e elimina a necessidade de conversores de nível de tensão para o pino Echo.
- **Sincronização e Medição:** O relógio do ESP32 é sincronizado via NTP (Network Time Protocol) para operar em UTC. A medição da água no recipiente é realizada uma vez por dia ou ao pressionar um botão de acionamento manual.
- **Cálculo da Precipitação:** A precipitação diária é calculada usando a diferença entre a altura total do recipiente e a distância medida pelo sensor HC-SR04.
  - Fórmula da distância: $D_{\text{cm}} = \frac{\text{duration (µs)}}{58.0}$ (onde `duration` é o tempo do pulso Echo).
  - Profundidade da água: $h = \max\left(0, H_{\text{container}} - D_{\text{medida}}\right)$.
  - Precipitação: $P_{\text{mm}} = 10 \times h$.
- **Armazenamento Local:** O valor de precipitação de cada dia é armazenado no armazenamento não volátil (NVS) do ESP32, utilizando a biblioteca "Preferences", sob uma chave no formato `YYYY-MM-DD`.
- **Transmissão de Dados:** Diariamente (ou ao pressionar o botão), o ESP32 monta um array JSON com os dados de precipitação dos últimos 7 dias e envia esses dados via HTTP POST para a API de previsão.
  - **Feedback Visual (LEDs):**
    - Amarelo fixo: Durante a requisição HTTP.

    - Verde fixo: Sucesso na comunicação e previsão (`success: true` na resposta da API).
    - Vermelho fixo: Servidor retornou `success: false`.
    - Laranja fixo: Erro na comunicação HTTP (timeout, erro 4xx, 5xx, etc.).
- **Estrutura do Código:** O código do ESP32 foi modularizado para melhor legibilidade, manutenibilidade e testabilidade, com módulos para Sensor, Storage, Network, LED, Button e TimeUtil.
- **Simulação (Wokwi):**
    - Placa: `board-esp32-devkit-c-v4`
    - Sensor: `wokwi-hc-sr04`
    - Componentes adicionais: Botão de pressão, quatro LEDs de 5mm (verde, vermelho, laranja, amarelo) com resistores de 220Ω.
- **Configuração da API no ESP32:** Para que o ESP32 envie dados para a API rodando localmente, é necessário configurar o IP do servidor no arquivo `src/Config.cpp`:

```
const char* SERVER_URL = "http://<SEU_IPV4_LOCAL>:8000";
```

2. Processamento e Armazenamento de Dados (Datasets)

Os dados utilizados para treinar o modelo de previsão de enchentes foram coletados e processados a partir de duas fontes principais:

1. **Dados de Enchentes:**
    - **Fonte:** The International Charter Space and Major Disasters.
    - **Escopo:** Eventos de enchentes ocorridos na América do Sul entre 2000 e 2025, incluindo data e localização (latitude, longitude).
2. **Dados de Precipitação:**
    - **Fonte:** Open-Meteo API.
    - **Metodologia:** Um script consultou a API do Open-Meteo para obter dados de precipitação diária para a localização de cada enchente, cobrindo o dia do evento e os 6 dias anteriores.

O dataset final (`dataset.csv`) possui as seguintes colunas: `Event_Date`, `Weather_Date`, `Latitude`, `Longitude`, `Precipitation`, `Flood`.

- **Abordagem de Modelagem (Sazonal - Escolhida):**
    - **Casos Positivos (`Flood = 1`):** Dados de precipitação dos 7 dias (dia do evento + 6 anteriores) para cada enchente registrada.
    - **Casos Negativos (`Flood = 0`):** Dados de precipitação para as mesmas datas e locais, porém do ano anterior, assumindo ausência de enchente.
    - **Justificativa:** Esta abordagem apresentou o melhor desempenho nos testes. Embora o modelo resultante tenda a ser mais preciso para enchentes sazonais,

essa característica foi considerada aceitável para um MVP focado na previsão de enchentes impulsionadas por padrões de chuva.
    - Uma abordagem alternativa (eventos aleatórios de não enchente) foi descartada, pois o modelo aprendia predominantemente a prever a ausência de enchentes devido ao desbalanceamento dos dados.

3. Modelagem e Treinamento de Machine Learning (Modelo)

Nesta etapa, os dados de precipitação são processados e utilizados para treinar um modelo de classificação. O processo detalhado encontra-se no notebook `modelo/classificacao_algoritmos.ipynb`.

3.1. Importando Dependências e Carregando Dados

Bibliotecas como Pandas, Matplotlib, Seaborn e Scikit-learn foram utilizadas.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC

# Carregar dados
df = pd.read_csv("../datasets/dataset.csv", parse_dates=['Event_Date',
'Weather_Date'])
df = df.sort_values('Weather_Date').reset_index(drop=True)
```

3.2. Engenharia de Características (Feature Engineering)

Novas colunas foram criadas para melhorar o treinamento do modelo:

- `Precip_7d`: Precipitação acumulada dos últimos 7 dias.
- `Precip_3d`: Precipitação acumulada dos últimos 3 dias.
- `Precip_1d`: Precipitação acumulada do último 1 dia (precipitação do dia).
- `Ratio_1d_7d`: Peso da chuva do dia atual em relação ao acumulado dos últimos 7 dias.
- `Acceleration`: Mede se a chuva de hoje está acima ou abaixo da média dos últimos 3 dias.

- `Impact_Score` : Métrica composta que dá mais peso à chuva recente, peso intermediário à chuva dos últimos 3 dias e peso linear ao acumulado dos 7 dias.

Justificativa das Features: Estas features foram criadas para capturar padrões temporais e de intensidade da chuva que podem ser mais indicativos de risco de enchente do que a precipitação isolada de um único dia.

```
df['Precip_7d'] = df['Precipitation'].rolling(window=7, min_periods=1).sum()
df['Precip_3d'] = df['Precipitation'].rolling(window=3, min_periods=1).sum()
df['Precip_1d'] = df['Precipitation']

epsilon = 1e-6  # Para evitar divisão por zero
df['Ratio_1d_7d'] = df['Precip_1d'] / (df['Precip_7d'] + epsilon)
df['Acceleration'] = df['Precip_1d'] - (df['Precip_3d'] / 3)
df['Impact_Score'] = (df['Precip_1d'] ** 2) + (df['Precip_3d'] ** 1.5) +
df['Precip_7d']
```

3.3. Filtrando Dados

Para evitar classes desbalanceadas, os dados foram filtrados, mantendo todos os dias com enchente (`Flood == 1`) e, para os dias sem enchente, apenas aqueles com o mesmo dia e mês da data em que ocorreram enchentes (abordagem sazonal, conforme descrito na seção de Datasets).

```
filtered_df = df[
    (df['Flood'] == 1) |
    ((df['Weather_Date'].dt.month == df['Event_Date'].dt.month) &
    (df['Weather_Date'].dt.day == df['Event_Date'].dt.day))
].reset_index(drop=True)
```

3.4. Preparação para Treinamento

- **Seleção de Features e Alvo:**

```
features = ['Precip_1d', 'Precip_3d', 'Precip_7d', 'Ratio_1d_7d',
'Acceleration', 'Impact_Score']
X = filtered_df[features]
y = filtered_df['Flood']
```

- **Divisão Treino/Teste e Escalonamento:**

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

3.5. Avaliação de Modelos

Uma função `avaliar_modelo` foi criada para treinar, prever e exibir métricas (acurácia, relatório de classificação, matriz de confusão) para diferentes algoritmos.

```
def avaliar_modelo(modelo, X_tr, y_tr, X_te, y_te, nome):
    modelo.fit(X_tr, y_tr)
    y_pred = modelo.predict(X_te)
    print(f"\n--- {nome} ---")
    print("Acurácia:", accuracy_score(y_te, y_pred))
    print(classification_report(y_te, y_pred))
    cm = confusion_matrix(y_te, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'Matriz de Confusão - {nome}')
    plt.ylabel('Verdadeiro')
    plt.xlabel('Previsto')
    plt.show()
```

Foram testados: KNN, Regressão Logística, Árvore de Decisão, Floresta Aleatória e SVM.

3.6. Comparação e Escolha do Modelo

O KNN apresentou a maior acurácia (aproximadamente 70.59%) entre os modelos testados. **Justificativa da Escolha do KNN:** Além da acurácia, o KNN é um modelo relativamente simples de implementar e interpretar para este problema de classificação. Sua sensibilidade à escala das features foi tratada com o `StandardScaler`.

3.7. Exportação do Modelo

O modelo KNN treinado (`knn.joblib`) e o normalizador (`scaler.joblib`) foram salvos para serem utilizados pela API e pelo dashboard Streamlit.

```
import joblib
joblib.dump(knn, './knn.joblib')
joblib.dump(scaler, './scaler.joblib')
```

4. API de Previsão

Uma API construída com FastAPI expõe o modelo treinado.

- **Endpoint:** `/predict`

- **Método:** `POST`
- **Entrada:** Uma lista de 7 números (`List[float]`) representando a precipitação dos últimos 7 dias (índice 0 = dia mais recente).
  - Exemplo: `[10.5, 5.0, 0.0, 2.1, 15.8, 3.0, 0.0]`
- **Saída:** Um JSON com a chave `"enchente"` (`bool`).
  - `{"enchente": true}` : Risco de enchente.
  - `{"enchente": false}` : Sem risco de enchente.
- **Funcionamento Interno:** A API recebe os dados de precipitação, calcula as mesmas 6 features (`Precip_1d` , `Precip_3d` , `Precip_7d` , `Ratio_1d_7d` , `Acceleration` , `Impact_Score`) usadas no treinamento, aplica o `scaler.joblib` e, em seguida, usa o `knn.joblib` para fazer a previsão.

5. Dashboard de Visualização (Streamlit)

Uma aplicação web interativa desenvolvida com Streamlit permite a visualização e previsão.

- **Funcionalidades:**
  - Interface para inserção de dados de precipitação diária dos últimos 7 dias.
  - Cálculo e exibição das features (mesma "fórmula" do treinamento e da API).
  - Previsão do risco de enchente usando o `knn.joblib` e `scaler.joblib` carregados da pasta `../modelo/` .
  - Apresentação do resultado: "Alto risco de enchente" ou "Sem risco de enchente".
  - Explicação detalhada das features calculadas.

Justificativas Gerais da Solução

- **Modularidade:** A separação em componentes (ESP32, Modelo, API, Streamlit) permite desenvolvimento, teste e manutenção independentes.
- **Machine Learning:** A abordagem de ML (especificamente KNN após testes) foi escolhida pela capacidade de aprender padrões complexos a partir de dados históricos de precipitação e enchentes.
- **Engenharia de Features:** A criação de features customizadas foi fundamental para extrair informações relevantes dos dados brutos de precipitação, melhorando o desempenho do modelo.
- **API para Escalabilidade:** A API permite que a lógica de previsão seja consumida por diversas aplicações ou sistemas.
- **Streamlit para Interação:** O dashboard facilita a interação do usuário com o modelo de forma intuitiva.
- **Abordagem Sazonal de Dados:** Dada a natureza dos dados de enchentes e a complexidade de obter um dataset perfeitamente balanceado para todos os tipos de eventos, a abordagem sazonal foi uma escolha pragmática para o MVP, focando em eventos com precedentes anuais.

Resultados Esperados

Com base na arquitetura e desenvolvimento propostos, os resultados esperados são:

1. **Sistema Funcional de Previsão:** Um sistema completo capaz de receber dados de precipitação (simulados via ESP32 ou inseridos manualmente no Streamlit) e fornecer uma previsão de risco de enchente.
2. **Modelo Preditivo com Desempenho Aceitável:** O modelo KNN, com uma acurácia em torno de 70% nos dados de teste (considerando a abordagem sazonal), é esperado que forneça previsões úteis, especialmente para enchentes com padrões de chuva semelhantes aos dados de treinamento.
3. **Interface de Usuário Intuitiva:** O dashboard Streamlit deve permitir que usuários, mesmo sem conhecimento técnico profundo, possam utilizar o sistema para obter previsões.
4. **API Robusta:** A API FastAPI deve ser capaz de lidar com requisições de previsão de forma eficiente e retornar os resultados de maneira padronizada.
5. **Coleta de Dados (Protótipo):** O módulo ESP32, mesmo em simulação, deve demonstrar a viabilidade da coleta de dados de precipitação e sua integração com a API.
6. **Transparência no Cálculo:** A exibição das features calculadas no Streamlit e a documentação do processo de engenharia de features visam dar transparência ao funcionamento do modelo.
7. **Identificação de Limitações:** Espera-se que o sistema demonstre sua eficácia, mas também que suas limitações (como a tendência à previsão sazonal) sejam compreendidas, abrindo caminho para futuras melhorias.

Conclusões

O desenvolvimento do "Sistema de Previsão de Enchentes" demonstrou a viabilidade de integrar hardware para coleta de dados, modelagem de machine learning e interfaces de usuário para criar uma solução prática para um problema complexo.

Principais Conclusões:

1. **Viabilidade da Solução Integrada:** A arquitetura modular proposta, conectando o ESP32 (coleta), o modelo KNN (predição), a API FastAPI (serviço) e o Streamlit (interface), provou ser uma abordagem eficaz para construir o sistema.
2. **Importância da Engenharia de Características:** A criação de features como `Impact_Score`, `Acceleration` e os acumulados de precipitação foi crucial para

que o modelo KNN pudesse capturar os padrões relevantes nos dados de chuva e alcançar um desempenho preditivo satisfatório para o MVP.

3. **Desempenho do Modelo KNN:** O modelo K-Nearest Neighbors, após escalonamento dos dados, apresentou a melhor acurácia (aproximadamente 70.59%) entre os algoritmos testados, tornando-se a escolha para esta versão do sistema.
4. **Abordagem Sazonal e Suas Implicações:** A estratégia de utilizar dados do ano anterior para representar cenários de "não enchente" resultou em um modelo com bom desempenho para eventos sazonais. Embora isso seja uma limitação para eventos atípicos, é uma simplificação válida e útil para um MVP focado em enchentes impulsionadas por padrões de chuva conhecidos.
5. **Modularidade como Facilitador:** A divisão do projeto em componentes facilitou o desenvolvimento, teste e a compreensão de cada parte do sistema.
6. **Potencial de Alerta Precoce:** O sistema, mesmo como protótipo, demonstra o potencial de fornecer alertas precoces que podem auxiliar na mitigação de danos causados por enchentes.

Desafios e Aprendizados:

- A obtenção e o tratamento de um dataset balanceado e representativo para todos os tipos de enchentes é um desafio contínuo em problemas dessa natureza.
- A simulação da coleta de dados com o ESP32 forneceu insights valiosos sobre a integração de hardware com sistemas de software.

Próximos Passos e Melhorias Futuras:

- Expandir o dataset com mais eventos de enchente e períodos de não enchente mais diversificados.
- Explorar algoritmos de machine learning mais avançados e técnicas de ensemble.
- Implementar a coleta de dados com sensores reais em campo.
- Aprimorar a interface do usuário com mais visualizações e informações contextuais.
- Integrar outras variáveis meteorológicas (e.g., nível de rios, umidade do solo) para enriquecer o modelo.
- Permitir que usuário coloque sua localização na interface streamlit e que através do local o sistema busque a precipitação dos últimos 7 dias e emita uma previsão de enchente

Em suma, este projeto estabelece uma base sólida para um sistema de previsão de enchentes, com potencial significativo para evolução e aplicação prática na prevenção e gestão de desastres naturais.