**## Sistema Preditivo de Obesidade - Tech Challenge F4**

**## 1- Importação de Bibliotecas e Dados**

import pandas as pd

import pickle

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.pipeline import Pipeline

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

%matplotlib inline

# Carregar dados

df = pd.read\_csv('Obesity.csv')

df.head()

**## 2 - Análise Exploratória dos Dados**

# Verificar tipos de dados e valores nulos

df.info()

df.isnull().sum()

**## 3 - Visualização Inicial da Variável Alvo**

# Visualizar distribuição da variável alvo 'Obesity'

plt.figure(figsize=(10, 5))

sns.countplot(data=df, x='Obesity', palette='Set2')

plt.title('Distribuição dos Níveis de Obesidade')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

**## 4. Pré-processamento de Dados**

**## Objetivo**:

##Transformar os dados brutos em um formato que possa ser interpretado por algoritmos de Machine ##Learning, tratando variáveis categóricas, valores numéricos e preparando o conjunto de treino e teste.

# 2\_preprocessamento.py

import pandas as pd

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split

# Carregar a base de dados

df = pd.read\_csv('Obesity.csv')

# Visualizar colunas categóricas

cat\_cols = df.select\_dtypes(include='object').columns

print("Colunas categóricas:", list(cat\_cols))

# Verificar colunas

print("Colunas originais:", df.columns.tolist())

# Corrigir nome da coluna se estiver errado

if 'TUE' in df.columns:

    df.rename(columns={'TUE': 'TER'}, inplace=True)

    print("Coluna TUE renomeada para TER")

# Verificar novamente

print("Colunas após ajuste:", df.columns.tolist())

# Codificar variáveis categóricas

cat\_cols = df.select\_dtypes(include='object').columns

for col in cat\_cols:

df[col] = LabelEncoder().fit\_transform(df[col])

**## 5. Codificação de Variáveis Categóricas**

## Vamos aplicar LabelEncoder para converter os valores categóricos em números.

# Aplicar Label Encoding

label\_encoders = {}

for col in cat\_cols:

    le = LabelEncoder()

    df[col] = le.fit\_transform(df[col])

    label\_encoders[col] = le  # Guardar para possível uso posterior (ex: inversão)

# Visualizar resultado

print(df.head())

**## 6. Separar Features (X) e Target (y)**

# Separar variáveis preditoras e alvo

X = df.drop('Obesity', axis=1)

y = df['Obesity']

**## 7. Divisão em Conjuntos de Treino e Teste**

# Dividir os dados em treino e teste (80/20)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, y, test\_size=0.2, random\_state=42, stratify=y

)

# Verificar dimensões

print("Shape X\_train:", X\_train.shape)

print("Shape X\_test :", X\_test.shape)

print("Classes:", y.value\_counts())

**## O que terá ao final dessa etapa:**

# df tratado com todas as variáveis categóricas codificadas

# X\_train, X\_test, y\_train, y\_test prontos para treinar modelos

print("Exemplo de linha já codificada:")

print(X\_train.iloc[0])

**## 8. Treinamento do Modelo com Pipeline**

**## Objetivo**:

# Criar um modelo preditivo com um pipeline de Machine Learning que envolva:

# Padronização (StandardScaler)

# Algoritmo (RandomForestClassifier)

# Avaliação de desempenho

# Salvamento do modelo

# Criar pipeline: Padronização + Modelo

pipeline = Pipeline([

    ("scaler", StandardScaler()),

    ("model", RandomForestClassifier(random\_state=42))

])

# Treinar o modelo

pipeline.fit(X\_train, y\_train)

# Avaliar o modelo

y\_pred = pipeline.predict(X\_test)

acc = accuracy\_score(y\_test, y\_pred)

print("Acurácia:", acc)

print("\nRelatório de Classificação:")

print(classification\_report(y\_test, y\_pred))

# Salvar modelo treinado

with open("modelo\_obesidade.pkl", "wb") as f:

    pickle.dump(pipeline, f)

print("✅ Modelo salvo com sucesso!")

**## O que terá após essa etapa:**

# modelo\_obesidade.pkl: modelo salvo com pipeline completo

# Avaliação impressa no terminal:

# Acurácia

# Relatório de classificação

**## 8. Aplicativo com Streamlit para Previsão**

pip install streamlit

# 4\_app\_streamlit.py

import streamlit as st

import pandas as pd

import numpy as np

import pickle

# Carregar modelo treinado

with open('modelo\_obesidade.pkl', 'rb') as f:

    model = pickle.load(f)

st.title("Sistema Preditivo de Obesidade")

st.write("""

Este app permite prever o nível de obesidade com base nos dados do paciente.

Preencha os campos abaixo e clique em 'Prever'.

""")

def user\_input\_features():

    gender = st.selectbox('Gênero', ['Female', 'Male'])

    age = st.number\_input('Idade', min\_value=1, max\_value=120, value=30)

    height = st.number\_input('Altura (metros)', min\_value=0.5, max\_value=2.5, value=1.7, format="%.2f")

    weight = st.number\_input('Peso (kg)', min\_value=10, max\_value=300, value=70)

    family\_history = st.selectbox('Histórico familiar de excesso de peso?', ['No', 'Yes'])

    favc = st.selectbox('Você come alimentos altamente calóricos com frequência?', ['No', 'Yes'])

    fcvc = st.selectbox('Você costuma comer vegetais nas refeições?', ['No', 'Sometimes', 'Frequently'])

    ncp = st.slider('Quantas refeições principais você faz por dia?', 1, 5, 3)

    caec = st.selectbox('Você come algo entre as refeições?', ['No', 'Sometimes', 'Frequently'])

    smoke = st.selectbox('Você fuma?', ['No', 'Yes'])

    ch2o = st.slider('Quantos litros de água você bebe por dia?', 1, 5, 2)

    scc = st.selectbox('Você monitora a quantidade de calorias que ingere?', ['No', 'Yes'])

    faf = st.selectbox('Com que frequência você pratica atividade física?', ['Never', 'Sometimes', 'Frequently'])

    ter = st.slider('Tempo diário com dispositivos tecnológicos (horas)', 0, 10, 3)

    calc = st.selectbox('Frequência de consumo de álcool', ['Never', 'Sometimes', 'Frequently'])

    mtrans = st.selectbox('Meio de transporte habitual', ['Bike', 'Car', 'Motorbike', 'Public Transport', 'Walking'])

    data = {

        'Gender': gender,

        'Age': age,

        'Height': height,

        'Weight': weight,

        'family\_history': family\_history,

        'FAVC': favc,

        'FCVC': fcvc,

        'NCP': ncp,

        'CAEC': caec,

        'SMOKE': smoke,

        'CH2O': ch2o,

        'SCC': scc,

        'FAF': faf,

        'TER': ter,

        'CALC': calc,

        'MTRANS': mtrans

    }

    return pd.DataFrame(data, index=[0])

input\_df = user\_input\_features()

# Pré-processamento das entradas: aplicar os mesmos LabelEncoders usados no treino

from sklearn.preprocessing import LabelEncoder

encoders\_info = {

    'Gender': ['Female', 'Male'],

    'family\_history': ['No', 'Yes'],

    'FAVC': ['No', 'Yes'],

    'FCVC': ['No', 'Sometimes', 'Frequently'],

    'CAEC': ['No', 'Sometimes', 'Frequently'],

    'SMOKE': ['No', 'Yes'],

    'SCC': ['No', 'Yes'],

    'FAF': ['Never', 'Sometimes', 'Frequently'],

    'CALC': ['Never', 'Sometimes', 'Frequently'],

    'MTRANS': ['Bike', 'Car', 'Motorbike', 'Public Transport', 'Walking']

}

for col, categories in encoders\_info.items():

    le = LabelEncoder()

    le.classes\_ = np.array(categories)

    input\_df[col] = le.transform(input\_df[col])

# Previsão

prediction = model.predict(input\_df)

prediction\_proba = model.predict\_proba(input\_df)

st.subheader('Previsão do nível de obesidade:')

st.write(prediction[0])

st.subheader('Probabilidades por classe:')

probs\_df = pd.DataFrame(prediction\_proba, columns=model.classes\_)

st.write(probs\_df.T)

**## A - Atualização no seu app Streamlit**

# Adicionar ao início do script

import seaborn as sns

import matplotlib.pyplot as plt

**## B - Depois do carregamento do modelo, adicione o seletor de página:**

# Menu lateral para alternar entre páginas

pagina = st.sidebar.selectbox("Escolha a página", ["Previsão", "Análise dos Dados"])

**## C - Parte de Análise dos Dados**

# Página: ANÁLISE DOS DADOS

if pagina == "Análise dos Dados":

    st.header("📊 Análise dos Dados de Obesidade")

    # Carregar dados originais

    df = pd.read\_csv("Obesity.csv")

    st.subheader("Distribuição da variável alvo (Obesity)")

    fig1, ax1 = plt.subplots()

    sns.countplot(data=df, x="Obesity", palette="Set2", ax=ax1)

    ax1.set\_xticklabels(ax1.get\_xticklabels(), rotation=45)

    st.pyplot(fig1)

    st.subheader("Matriz de Correlação")

    df\_encoded = df.copy()

    for col in df\_encoded.select\_dtypes(include='object').columns:

        df\_encoded[col] = LabelEncoder().fit\_transform(df\_encoded[col])

    fig2, ax2 = plt.subplots(figsize=(10, 6))

    sns.heatmap(df\_encoded.corr(), annot=False, cmap='coolwarm', ax=ax2)

    st.pyplot(fig2)

    st.subheader("Distribuição das variáveis numéricas")

    num\_cols = df\_encoded.select\_dtypes(include=np.number).columns.drop("Obesity")

    for col in num\_cols:

        fig, ax = plt.subplots()

        sns.histplot(df\_encoded[col], kde=True, bins=20, ax=ax)

        ax.set\_title(f'Distribuição de {col}')

        st.pyplot(fig)

**## D - Página de Previsão (não precisa alterar)**

# A lógica de previsão pode permanecer como está, apenas garanta que ela só execute se pagina == "Previsão":

if pagina == "Previsão":

    st.title("Sistema Preditivo de Obesidade")

    st.write("""Este app permite prever o nível de obesidade com base nos dados do paciente.""")

    input\_df = user\_input\_features()

    # Reaplicar os encoders

    for col, categories in encoders\_info.items():

        le = LabelEncoder()

        le.classes\_ = np.array(categories)

        input\_df[col] = le.transform(input\_df[col])

    prediction = model.predict(input\_df)

    prediction\_proba = model.predict\_proba(input\_df)

    st.subheader('Previsão do nível de obesidade:')

    st.write(prediction[0])

    st.subheader('Probabilidades por classe:')

    st.write(pd.DataFrame(prediction\_proba, columns=model.classes\_).T)

## Abrir App no terminal: streamlit run 4\_app\_streamlit.py