import pandas as pd

import gradio as gr

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.linear\_model import LogisticRegression

# Load the dataset and preprocess it (same as before)

df = pd.read\_csv("WA\_Fn-UseC\_-Telco-Customer-Churn.csv")

df['TotalCharges'] = pd.to\_numeric(df['TotalCharges'], errors='coerce')

df.dropna(inplace=True)

df.drop('customerID', axis=1, inplace=True)

# Encode target variable

label\_encoder = LabelEncoder()

df['Churn'] = label\_encoder.fit\_transform(df['Churn'])

# One-hot encode categorical columns

categorical\_cols = df.select\_dtypes(include=['object']).columns

df\_encoded = pd.get\_dummies(df, columns=categorical\_cols, drop\_first=True)

# Scale features

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(df\_encoded.drop('Churn', axis=1))

y = df\_encoded['Churn']

# Train a simple model

model = LogisticRegression()

model.fit(X\_scaled, y)

# Define the prediction function

def predict\_churn(gender, SeniorCitizen, Partner, Dependents, tenure, PhoneService,

MultipleLines, InternetService, OnlineSecurity, OnlineBackup,

DeviceProtection, TechSupport, StreamingTV, StreamingMovies,

Contract, PaperlessBilling, PaymentMethod, MonthlyCharges, TotalCharges):

# Create the input dictionary

input\_data = {

'gender': gender,

'SeniorCitizen': SeniorCitizen,

'Partner': Partner,

'Dependents': Dependents,

'tenure': int(tenure),

'PhoneService': PhoneService,

'MultipleLines': MultipleLines,

'InternetService': InternetService,

'OnlineSecurity': OnlineSecurity,

'OnlineBackup': OnlineBackup,

'DeviceProtection': DeviceProtection,

'TechSupport': TechSupport,

'StreamingTV': StreamingTV,

'StreamingMovies': StreamingMovies,

'Contract': Contract,

'PaperlessBilling': PaperlessBilling,

'PaymentMethod': PaymentMethod,

'MonthlyCharges': float(MonthlyCharges),

'TotalCharges': float(TotalCharges)

}

# Create DataFrame from input data

input\_df = pd.DataFrame([input\_data])

# Combine with original data for encoding

df\_temp = pd.concat([df.drop('MonthlyCharges', axis=1), input\_df], ignore\_index=True)

# One-hot encode the new input data

df\_temp\_encoded = pd.get\_dummies(df\_temp, drop\_first=True)

# Reorder columns to match the training set

df\_temp\_encoded = df\_temp\_encoded.reindex(columns=df\_encoded.drop('MonthlyCharges', axis=1).columns, fill\_value=0)

# Scale the new input

scaled\_input = scaler.transform(df\_temp\_encoded.tail(1))

# Make prediction

prediction = model.predict(scaled\_input)

# Return the prediction

return round(prediction[0], 2)