# ✅ Streamlit App with Custom Theme, Visuals, Tooltips, and Grand Title

import streamlit as st

import pandas as pd

import pickle

from sklearn.metrics.pairwise import cosine\_similarity

import matplotlib.pyplot as plt

# 🎨 Set custom page config with new title and emoji

st.set\_page\_config(

    page\_title="✈️ Travel Intelligence Hub",

    page\_icon="🌍",

    layout="wide",

    initial\_sidebar\_state="expanded"

)

# 🌟 New appealing title

st.markdown("""

    <h1 style='text-align: center; color: #2E86C1;'>🌍 Travel Intelligence Hub</h1>

    <h4 style='text-align: center; color: #566573;'>Predict ratings, understand traveler behavior, and deliver smart attraction recommendations — all powered by Machine Learning.</h4>

""", unsafe\_allow\_html=True)

# --- Load models

reg\_model = pickle.load(open("regression\_model.pkl", "rb"))

clf\_model = pickle.load(open("classification\_model.pkl", "rb"))

le = pickle.load(open("label\_encoder.pkl", "rb"))

@st.cache\_data

def load\_data():

    return pd.read\_csv("Tourism\_Master.csv")

df = load\_data()

df.dropna(inplace=True)

@st.cache\_data

def get\_user\_item\_matrix(n\_users=1000):

    matrix = df.pivot\_table(index="UserId", columns="Attraction", values="Rating").fillna(0)

    return matrix.head(n\_users)

@st.cache\_data

def get\_user\_similarity(matrix):

    sim = cosine\_similarity(matrix)

    return pd.DataFrame(sim, index=matrix.index, columns=matrix.index)

def recommend\_attractions(user\_id, matrix, sim\_df, top\_n=5):

    if user\_id not in sim\_df.index:

        return ["User ID not in sample."]

    similar\_users = sim\_df[user\_id].sort\_values(ascending=False).drop(user\_id).head(10).index

    mean\_ratings = matrix.loc[similar\_users].mean().sort\_values(ascending=False)

    already\_rated = matrix.loc[user\_id]

    return mean\_ratings[already\_rated == 0].head(top\_n).index.tolist()

# --- Sidebar

st.sidebar.header("⚙️ Controls")

sample\_size = st.sidebar.slider(

    "🔢 User sample size for recommendations",

    min\_value=200, max\_value=3000, step=200, value=1000,

    help="Controls how many users are considered for similarity calculations"

)

user\_item\_matrix = get\_user\_item\_matrix(sample\_size)

user\_sim\_df = get\_user\_similarity(user\_item\_matrix)

# --- Tabs

tab1, tab2, tab3 = st.tabs(["🎯 Predict Rating", "🧭 Predict Visit Mode", "🤝 Recommend Attractions"])

with tab1:

    st.subheader("🎯 Predict User Rating for an Attraction")

    col1, col2 = st.columns(2)

    with col1:

        country = st.selectbox("🌍 Select Country", sorted(df["Country"].unique()), help="Choose where the user is from")

        attraction = st.selectbox("🏝️ Select Attraction", sorted(df["Attraction"].unique()), help="Pick the attraction to rate")

    with col2:

        season = st.selectbox("📅 Visit Season", sorted(df["VisitSeason"].unique()), key="season\_regression", help="Season of visit")

        popularity = st.slider("📊 Attraction Popularity", 0.0, 5.0, 3.0, help="Average rating from all users")

        avg\_rating = st.slider("⭐ User Avg Rating", 0.0, 5.0, 3.5, help="User's historical average rating")

    input\_df = pd.DataFrame({

        "Country": [country],

        "Attraction": [attraction],

        "VisitSeason": [season],

        "AttractionPopularity": [popularity],

        "UserAvgRating": [avg\_rating]

    })

    if st.button("🚀 Predict Rating"):

        pred = reg\_model.predict(input\_df)[0]

        st.success(f"🌟 Predicted Rating: {round(pred, 2)}")

        # Optional visual

        fig, ax = plt.subplots()

        ax.bar(["Predicted"], [pred], color="#2E86C1")

        ax.set\_ylim(0, 5)

        st.pyplot(fig)

with tab2:

    st.subheader("🧭 Predict Visit Mode")

    col1, col2 = st.columns(2)

    with col1:

        continent = st.selectbox("🌍 Continent", sorted(df["Continent"].unique()), help="User's continent")

        attraction\_type = st.selectbox("🎡 Attraction Type", sorted(df["AttractionType"].unique()), help="Type of attraction")

    with col2:

        region = st.selectbox("🗺️ Region", sorted(df["Region"].unique()), help="User's region")

        season = st.selectbox("📆 Visit Season", sorted(df["VisitSeason"].unique()), key="season\_classifier", help="Time of year visited")

    class\_df = pd.DataFrame({

        "Continent": [continent],

        "Region": [region],

        "VisitSeason": [season],

        "AttractionType": [attraction\_type]

    })

    if st.button("🔍 Predict Visit Mode"):

        pred\_label = clf\_model.predict(class\_df)[0]

        pred\_decoded = le.inverse\_transform([pred\_label])[0]

        st.success(f"🧭 Predicted Visit Mode: {pred\_decoded}")

with tab3:

    st.subheader("🤝 Personalized Attraction Recommendations")

    user\_id = st.number\_input("👤 Enter User ID", min\_value=0, step=1, help="User ID must exist in the dataset")

    if st.button("🎁 Get Recommendations"):

        recs = recommend\_attractions(user\_id, user\_item\_matrix, user\_sim\_df)

        st.write("🎯 \*\*Top Recommendations:\*\*")

        for i, r in enumerate(recs):

            st.markdown(f"\*\*{i+1}. {r}\*\*")