# Budget\_Analysis.py

import streamlit as st

import pandas as pd

import joblib

import os

import matplotlib.pyplot as plt

import matplotlib.ticker as mtick

# Load Models & Encoders

MODEL\_DIR = "models"

try:

total\_budget\_model = joblib.load(os.path.join(MODEL\_DIR, "total\_budget\_model.pkl"))

used\_budget\_model = joblib.load(os.path.join(MODEL\_DIR, "used\_budget\_model.pkl"))

budget\_label\_encoder\_dept = joblib.load(os.path.join(MODEL\_DIR, "budget\_label\_encoder\_dept.pkl"))

budget\_label\_encoder\_area = joblib.load(os.path.join(MODEL\_DIR, "budget\_label\_encoder\_area.pkl"))

except Exception as e:

st.error(f"❌ Model loading failed: {e}")

st.stop()

# UI Setup

st.set\_page\_config(page\_title="Budget Analysis", page\_icon="💸", layout="wide")

st.title("💸 Department Budget Analysis & Visualization")

if 'department' not in st.session\_state:

st.warning("⚠️ Please first file a complaint using \*Complaint\_Management.py\*")

else:

st.subheader("💰 Step 4: Budget Prediction & Charts")

# Inputs

selected\_area = st.selectbox("• Select Area", budget\_label\_encoder\_area.classes\_,

index=list(budget\_label\_encoder\_area.classes\_).index(st.session\_state['area']))

department = st.session\_state['department']

actual\_total\_budget = st.number\_input("• Actual Total Budget (₹)", min\_value=0, value=1000000, step=10000)

actual\_used\_budget = st.number\_input("• Actual Used Budget (₹)", min\_value=0, value=400000, step=10000)

actual\_remaining\_budget = st.number\_input("• Actual Remaining Budget (₹)", min\_value=0, value=200000, step=10000)

if st.button("🧾 Predict Budget & Show Visuals", key="predict\_compare\_budget"):

try:

# Encode inputs

dept\_encoded = budget\_label\_encoder\_dept.transform([department])[0]

area\_encoded = budget\_label\_encoder\_area.transform([selected\_area])[0]

# Predict Total Budget

total\_input = pd.DataFrame([{

"used\_budget": actual\_used\_budget,

"remaining\_budget": actual\_remaining\_budget,

"department\_encoded": dept\_encoded,

"area\_encoded": area\_encoded

}])

predicted\_total = total\_budget\_model.predict(total\_input)[0]

# Predict Used Budget

used\_input = pd.DataFrame([{

"total\_budget": predicted\_total,

"remaining\_budget": actual\_remaining\_budget,

"department\_encoded": dept\_encoded,

"area\_encoded": area\_encoded

}])

predicted\_used = used\_budget\_model.predict(used\_input)[0]

predicted\_remaining = predicted\_total - predicted\_used

# Check for potential corruption if the difference is large

corruption\_threshold = 0.2 # 20% difference considered as corruption

corruption\_message = ""

total\_budget\_diff = abs(actual\_total\_budget - predicted\_total) / actual\_total\_budget

used\_budget\_diff = abs(actual\_used\_budget - predicted\_used) / actual\_used\_budget

if total\_budget\_diff > corruption\_threshold or used\_budget\_diff > corruption\_threshold:

corruption\_message = "⚠️ Potential Corruption Detected: Large budget discrepancy!"

# Show Prediction

st.success("✅ Budget Prediction Completed")

st.metric("Predicted Total Budget", f"₹{predicted\_total:,.2f}")

st.metric("Predicted Used Budget", f"₹{predicted\_used:,.2f}")

st.metric("Predicted Remaining Budget", f"₹{predicted\_remaining:,.2f}")

# Show corruption warning if applicable

# Show corruption status

if corruption\_message:

st.warning(corruption\_message)

st.markdown("### 🛑 \*\*:red[CORRUPTION DETECTED]\*\*")

else:

st.markdown("### ✅ \*\*:green[NO CORRUPTION DETECTED]\*\*")

# === 📊 Charts Section ===

st.header("📊 Visual Comparisons")

# 1️⃣ Bar Chart (Side-by-Side)

st.subheader("📊 Bar Chart: Actual vs Predicted")

view\_mode = st.radio("View Mode", ["Absolute (₹)", "Percentage (%)"])

def format\_inr(value):

if value >= 1e6:

return f'₹{value/1e6:.1f}M'

elif value >= 1e3:

return f'₹{value/1e3:.1f}K'

return f'₹{value:.0f}'

# Data for Chart

if view\_mode == "Absolute (₹)":

data\_actual = {"Total": actual\_total\_budget, "Used": actual\_used\_budget, "Remaining": actual\_remaining\_budget}

data\_pred = {"Total": predicted\_total, "Used": predicted\_used, "Remaining": predicted\_remaining}

formatter = format\_inr

ylabel = "₹ Amount"

else:

data\_actual = {

"Used": (actual\_used\_budget / actual\_total\_budget) \* 100 if actual\_total\_budget else 0,

"Remaining": (actual\_remaining\_budget / actual\_total\_budget) \* 100 if actual\_total\_budget else 0

}

data\_pred = {

"Used": (predicted\_used / predicted\_total) \* 100 if predicted\_total else 0,

"Remaining": (predicted\_remaining / predicted\_total) \* 100 if predicted\_total else 0

}

formatter = lambda x: f"{x:.1f}%"

ylabel = "Percentage"

fig1, ax1 = plt.subplots()

bar\_width = 0.35

x = range(len(data\_actual))

bars1 = ax1.bar(x, data\_actual.values(), width=bar\_width, label="Actual", color='skyblue')

bars2 = ax1.bar([i + bar\_width for i in x], data\_pred.values(), width=bar\_width, label="Predicted", color='orange')

# Labels

for b in bars1 + bars2:

height = b.get\_height()

ax1.text(b.get\_x() + b.get\_width() / 2, height \* 1.01, formatter(height), ha='center', fontsize=8)

ax1.set\_xticks([i + bar\_width/2 for i in x])

ax1.set\_xticklabels(data\_actual.keys())

ax1.set\_ylabel(ylabel)

ax1.set\_title("Actual vs Predicted Budget")

ax1.legend()

ax1.grid(True, linestyle="--", axis="y")

st.pyplot(fig1)

# 2️⃣ Pie Chart

st.subheader("🥧 Pie Chart: Budget Distribution")

pie\_col1, pie\_col2 = st.columns(2)

# Actual Pie

with pie\_col1:

fig2, ax2 = plt.subplots()

ax2.pie([actual\_used\_budget, actual\_remaining\_budget],

labels=["Used", "Remaining"],

autopct='%1.1f%%',

colors=["#ff9999", "#66b3ff"],

startangle=140)

ax2.set\_title("Actual Budget")

st.pyplot(fig2)

# Predicted Pie

with pie\_col2:

fig3, ax3 = plt.subplots()

ax3.pie([predicted\_used, predicted\_remaining],

labels=["Used", "Remaining"],

autopct='%1.1f%%',

colors=["#ffcc99", "#99ff99"],

startangle=140)

ax3.set\_title("Predicted Budget")

st.pyplot(fig3)

except Exception as e:

st.error(f"❌ Error during prediction or visualization: {e}")