# Futuristic Bias Detection Framework for AI Models on African Datasets

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

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

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

from fairlearn.metrics import MetricFrame, selection\_rate, demographic\_parity\_difference, equalized\_odds\_difference

from fairlearn.reductions import ExponentiatedGradient, EqualizedOdds

import shap

import tensorflow as tf

from sklearn.ensemble import RandomForestClassifier

from flask import Flask, request, jsonify

import os

# Module Name: AfricaFairAI

class AfricaFairAI:

def \_\_init\_\_(self, dataset\_path='african\_data.csv'):

self.dataset\_path = dataset\_path

self.data = None

self.model = None

self.sensitive\_feature = 'ethnicity'

def generate\_sample\_dataset(self):

np.random.seed(42)

size = 1000

ethnicity = np.random.choice(['Group A', 'Group B', 'Group C'], size=size, p=[0.5, 0.3, 0.2])

age = np.random.randint(18, 60, size=size)

income = np.random.randint(20000, 100000, size=size)

education = np.random.choice(['Primary', 'Secondary', 'Tertiary'], size=size)

target = np.random.choice([0, 1], size=size, p=[0.6, 0.4])

self.data = pd.DataFrame({'ethnicity': ethnicity, 'age': age, 'income': income, 'education': education, 'target': target})

self.data.to\_csv(self.dataset\_path, index=False)

return self.data

def load\_dataset(self):

if not os.path.exists(self.dataset\_path):

self.generate\_sample\_dataset()

self.data = pd.read\_csv(self.dataset\_path)

return self.data

def dataset\_bias\_analysis(self):

distribution = self.data[self.sensitive\_feature].value\_counts(normalize=True)

plt.figure(figsize=(10, 6))

sns.barplot(x=distribution.index, y=distribution.values, palette='viridis')

plt.title(f'Distribution of {self.sensitive\_feature} in Dataset')

plt.xlabel(self.sensitive\_feature)

plt.ylabel('Proportion')

plt.show()

return distribution

def train\_model(self):

features = ['age', 'income']

X = self.data[features]

y = self.data['target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42, stratify=y)

self.model = RandomForestClassifier(n\_estimators=100, random\_state=42)

self.model.fit(X\_train, y\_train)

y\_pred = self.model.predict(X\_test)

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

return self.model, accuracy

def evaluate\_model\_fairness(self, X\_test, y\_test):

y\_pred = self.model.predict(X\_test)

fairness\_metrics = MetricFrame(metrics={'Selection Rate': selection\_rate, 'Equalized Odds': equalized\_odds\_difference},

y\_true=y\_test, y\_pred=y\_pred, sensitive\_features=X\_test[self.sensitive\_feature])

disparity = demographic\_parity\_difference(y\_test, y\_pred, sensitive\_features=X\_test[self.sensitive\_feature])

return fairness\_metrics.by\_group, disparity

def bias\_mitigation(self, X\_train, y\_train):

mitigator = ExponentiatedGradient(EqualizedOdds())

mitigator.fit(X\_train, y\_train, sensitive\_features=X\_train[self.sensitive\_feature])

return mitigator

# Flask API for AfricaFairAI

app = Flask(\_\_name\_\_)

africa\_fair\_ai = AfricaFairAI()

data = africa\_fair\_ai.load\_dataset()

model, accuracy = africa\_fair\_ai.train\_model()

@app.route('/predict', methods=['POST'])

def predict():

data = request.get\_json()

df = pd.DataFrame([data])

prediction = model.predict(df)[0]

return jsonify({'prediction': int(prediction)})

@app.route('/fairness', methods=['POST'])

def fairness():

data = request.get\_json()

df = pd.DataFrame([data])

fairness\_results, disparity = africa\_fair\_ai.evaluate\_model\_fairness(df, africa\_fair\_ai.data['target'])

return jsonify({'fairness\_metrics': fairness\_results.to\_dict(), 'disparity': disparity})

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)