**Lab-10**

Chandra kiran

BL.EN. U4AIE23141

Code:

import os

import numpy as np

import cv2

import matplotlib.pyplot as plt

import seaborn as sns

import shap

import lime.lime\_tabular

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

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from sklearn.feature\_selection import SequentialFeatureSelector

from sklearn.pipeline import Pipeline

from sklearn.ensemble import RandomForestClassifier

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

# === Dataset Path ===

DATASET\_PATH = "C:\\AIO\\Semster Files\\SEMSTER - 4\\ML\\Lab Work\\ML\_Assignment\_09\_BL.EN.U4AIE23138\\Dataset"

# === Load Glioma Dataset ===

def load\_glioma\_dataset(path, size=64):

    X = []

    for fname in os.listdir(path):

        img = cv2.imread(os.path.join(path, fname), cv2.IMREAD\_GRAYSCALE)

        if img is not None:

            img = cv2.resize(img, (size, size)).flatten()

            X.append(img)

    X = np.array(X)

    y = np.zeros(X.shape[0])

    y[X.shape[0] // 2:] = 1  # Artificial 2-class

    return X, y

# === A1: Correlation Heatmap ===

def plot\_correlation\_heatmap(X):

    X\_df = pd.DataFrame(X[:, :100])  # Limit to 100 features to keep plot readable

    corr = X\_df.corr()

    plt.figure(figsize=(12, 10))

    sns.heatmap(corr, cmap='coolwarm')

    plt.title("A1: Feature Correlation Heatmap")

    plt.show()

# === A2/A3: PCA-based Classification ===

def run\_pca\_classification(X, y, variance\_threshold=0.99):

    scaler = StandardScaler()

    X\_scaled = scaler.fit\_transform(X)

    pca = PCA(n\_components=variance\_threshold)

    X\_pca = pca.fit\_transform(X\_scaled)

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_pca, y, test\_size=0.3, random\_state=42, stratify=y)

    model = RandomForestClassifier(random\_state=42)

    model.fit(X\_train, y\_train)

    y\_pred = model.predict(X\_test)

    print(f"\n--- PCA {int(variance\_threshold\*100)}% Variance Retained ---")

    print(f"Number of PCA features: {X\_pca.shape[1]}")

    print("Accuracy:", accuracy\_score(y\_test, y\_pred))

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

    return model, X\_train, X\_test, y\_train, y\_test, pca

# === A4: Sequential Feature Selector ===

def run\_sequential\_selection(X, y, n\_features=50):

    scaler = StandardScaler()

    X\_scaled = scaler.fit\_transform(X)

    model = RandomForestClassifier(random\_state=42)

    sfs = SequentialFeatureSelector(model, n\_features\_to\_select=n\_features, direction='forward', cv=3, n\_jobs=-1)

    sfs.fit(X\_scaled, y)

    X\_reduced = sfs.transform(X\_scaled)

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_reduced, y, test\_size=0.3, random\_state=42, stratify=y)

    model.fit(X\_train, y\_train)

    y\_pred = model.predict(X\_test)

    print(f"\n--- A4: Sequential Feature Selection (Top {n\_features} Features) ---")

    print("Accuracy:", accuracy\_score(y\_test, y\_pred))

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

    return model, X\_train, X\_test, y\_train, y\_test

# === A5: LIME + SHAP Explanation ===

def explain\_with\_lime\_shap(model, X\_train, X\_test):

    print("\n--- A5: LIME & SHAP Explainability ---")

    # LIME

    lime\_exp = lime.lime\_tabular.LimeTabularExplainer(

        training\_data=X\_train,

        mode='classification',

        feature\_names=[f'pixel\_{i}' for i in range(X\_train.shape[1])],

        class\_names=['Class 0', 'Class 1']

    )

    lime\_explanation = lime\_exp.explain\_instance(X\_test[0], model.predict\_proba)

    lime\_explanation.save\_to\_file("lime\_explanation.html")

    print("LIME explanation saved to lime\_explanation.html")

    # SHAP

    explainer = shap.Explainer(model, X\_train)

    shap\_values = explainer(X\_test[:10])

    shap.summary\_plot(shap\_values, X\_test[:10], feature\_names=[f'pixel\_{i}' for i in range(X\_train.shape[1])])

# === MAIN EXECUTION ===

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

    import pandas as pd

    X, y = load\_glioma\_dataset(DATASET\_PATH)

    # A1: Correlation Heatmap

    plot\_correlation\_heatmap(X)

    # A2: PCA with 99% variance

    model\_99, X\_train\_99, X\_test\_99, y\_train\_99, y\_test\_99, \_ = run\_pca\_classification(X, y, 0.99)

    # A3: PCA with 95% variance

    model\_95, X\_train\_95, X\_test\_95, y\_train\_95, y\_test\_95, \_ = run\_pca\_classification(X, y, 0.95)

    # A4: Sequential Feature Selection

    model\_sfs, X\_train\_sfs, X\_test\_sfs, y\_train\_sfs, y\_test\_sfs = run\_sequential\_selection(X, y)

    # A5: Explainability

    explain\_with\_lime\_shap(model\_sfs, X\_train\_sfs, X\_test\_sfs)