ML Assignment 2

import pandas as pd

import numpy as np

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

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score, r2\_score

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.neural\_network import MLPClassifier

nhanes\_data = pd.read\_csv('/content/P\_DEMO.csv')

demographics\_cols = ['SEQN', 'Age', 'Gender']

cardio\_cols = ['SEQN', 'HeartConditionVar1', 'HeartConditionVar2', ...]

merged\_data = pd.merge(nhanes\_data[demographics\_cols], nhanes\_data[cardio\_cols], on='SEQN', how='inner')

age\_ranges = [(10, 20), (21, 30), (31, 40), (41, 50), (51, 60), (61, 70), (70, 200)]

for age\_range in age\_ranges:

filtered\_data = merged\_data[(merged\_data['Age'] >= age\_range[0]) & (merged\_data['Age'] <= age\_range[1])]

print(f"Patients in age range {age\_range}: {len(filtered\_data)}")

X = merged\_data.drop(['SEQN', 'HeartConditionVar1', 'HeartConditionVar2', ...], axis=1)

y = merged\_data['HeartConditionVar1']

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

scaler = StandardScaler()

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

X\_test\_scaled = scaler.transform(X\_test)

models = {

'Logistic Regression': LogisticRegression(),

'K-NN': KNeighborsClassifier(),

'SVM': SVC(),

'Decision Tree': DecisionTreeClassifier(),

'ANN': MLPClassifier()

}

for name, model in models.items():

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

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

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

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

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

f1 = f1\_score(y\_test, y\_pred)

auc = roc\_auc\_score(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f"Metrics for {name}:")

print(f"Accuracy: {accuracy}")

print(f"Precision: {precision}")

print(f"Recall: {recall}")

print(f"F1 Score: {f1}")

print(f"AUC: {auc}")

print(f"R2 Score: {r2}")

print("--------------")

male\_count = merged\_data[merged\_data['Gender'] == 'Male'].shape[0]

female\_count = merged\_data[merged\_data['Gender'] == 'Female'].shape[0]

print(f"Males in the dataset: {male\_count}")

print(f"Females in the dataset: {female\_count}")

heart\_condition\_count = merged\_data[merged\_data['HeartConditionVar1'] == 1].shape[0]

males\_with\_heart\_condition = merged\_data[(merged\_data['Gender'] == 'Male') & (merged\_data['HeartConditionVar1'] == 1)].shape[0]

females\_with\_heart\_condition = merged\_data[(merged\_data['Gender'] == 'Female') & (merged\_data['HeartConditionVar1'] == 1)].shape[0]

print(f"People with heart conditions: {heart\_condition\_count}")

print(f"Males with heart conditions: {males\_with\_heart\_condition}")

print(f"Females with heart conditions: {females\_with\_heart\_condition}")