import pandas as pd

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

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.naive\_bayes import GaussianNB

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

# Load the dataset from the UCI repository

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data"

columns = [

"ID", "Diagnosis", "Radius\_mean", "Texture\_mean", "Perimeter\_mean", "Area\_mean",

"Smoothness\_mean", "Compactness\_mean", "Concavity\_mean", "Concave\_points\_mean",

"Symmetry\_mean", "Fractal\_dimension\_mean", "Radius\_se", "Texture\_se", "Perimeter\_se",

"Area\_se", "Smoothness\_se", "Compactness\_se", "Concavity\_se", "Concave\_points\_se",

"Symmetry\_se", "Fractal\_dimension\_se", "Radius\_worst", "Texture\_worst",

"Perimeter\_worst", "Area\_worst", "Smoothness\_worst", "Compactness\_worst",

"Concavity\_worst", "Concave\_points\_worst", "Symmetry\_worst", "Fractal\_dimension\_worst"

]

# Read the data into a pandas DataFrame

data = pd.read\_csv(url, header=None, names=columns)

# Drop the ID column as it's not relevant for classification

data.drop("ID", axis=1, inplace=True)

# Encode the Diagnosis column (B = 0, M = 1)

label\_encoder = LabelEncoder()

data["Diagnosis"] = label\_encoder.fit\_transform(data["Diagnosis"])

# Separate features and target

X = data.drop("Diagnosis", axis=1)

y = data["Diagnosis"]

# Standardize the features

scaler = StandardScaler()

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

# Split the data into training and testing sets

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

# Initialize and train the Naive Bayes classifier

nb\_classifier = GaussianNB()

nb\_classifier.fit(X\_train, y\_train)

# Make predictions

y\_pred = nb\_classifier.predict(X\_test)

# Evaluate the model

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

print("Accuracy:", accuracy)

print("\nClassification Report:")

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