import numpy as np

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

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

import pandas as pd

class LogisticRegression:

def \_\_init\_\_(self, learning\_rate=0.01, iteration=1000):

self.learning\_rate = learning\_rate

self.iteration = iteration

self.theta = None

def add\_intercept(self, X):

# Step 1: Create a column vector of ones with the same number of rows as X

intercept\_column = np.ones((X.shape[0], 1))

# Step 2: Concatenate the intercept column with X horizontally

X\_with\_intercept = np.concatenate((intercept\_column, X),axis=1)

# Step 3: Return the feature matrix with the intercept column added

return X\_with\_intercept

def sigmoid(self, z):

return 1 / (1 + np.exp(-z))

def fit(self, X, y):

X = self.add\_intercept(X)

self.theta = np.zeros((X.shape[1]))

for \_ in range(self.iteration):

z = np.dot(X, self.theta)

h = self.sigmoid(z)

gradient = np.dot(X.T, (h - y)) / y.size

self.theta -= gradient \* self.learning\_rate

#print(self.theta)

def predict\_prob(self, X):

X = self.add\_intercept(X)

return self.sigmoid(np.dot(X, self.theta))

def predict(self, X, threshold=0.5):

return self.predict\_prob(X) >= threshold

data = pd.read\_csv("Breastcancer\_data.csv")

data.info()

data[4:]

X = data.iloc[:,2:-1].values

X = np.float64(X)

y = data.iloc[:,1].values

y = np.where(y == 'M', 1, 0)

y.shape, X.shape , y.size

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

model = LogisticRegression()

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

y\_test

val\_predictions = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, val\_predictions)

precision = precision\_score(y\_test, val\_predictions)

recall = recall\_score(y\_test, val\_predictions)

f1 = f1\_score(y\_test, val\_predictions)

print("Validation Set Metrics:")

print("Accuracy: {:.2f}".format(accuracy))

print("Precision: {:.2f}".format(precision))

print("Recall: {:.2f}".format(recall))

print("F1 Score: {:.2f}".format(f1))

from sklearn.metrics import confusion\_matrix

confusion = confusion\_matrix(y\_test, val\_predictions)

print(confusion)

print("Class 0 predicted and true : ")

print(confusion[0][0])

print("Class 0 predicted and false : ")

print(confusion[0][1])

print("Class 1 predicted and false : ")

print(confusion[1][0])

print("Class 1 predicted and true : ")

print(confusion[1][1])

import random

X\_valid = []

Y\_valid = []

for i in range(0, 20):

index = random.randint(0, 500)

X\_valid.append(X[index])

Y\_valid.append(y[index])

Y\_valid