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 DistanceMetric

from sklearn.neighbors import KNeighborsClassifier

DATA = pd.read\_csv(CSV\_URL, decimal='.', delimiter=",")

pulsar\_stars = DATA[(DATA.MIP >= MIP[0]) & (DATA.MIP <= MIP[1])]

print(f"Число строк в выборке: {len(pulsar\_stars)}")

print(f"Среднее для столбца MIP: {pulsar\_stars.MIP.mean()}")

print(f"Максимум для столбца SIP: {pulsar\_stars.SIP.max()}\n")

X = pulsar\_stars.drop(["TG"], axis=1).values

y = pulsar\_stars["TG"].values.ravel()

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

scaler = StandardScaler().fit(X\_train)

data\_scaler\_train = scaler.transform(X\_train)

data\_scaler\_test = scaler.transform(X\_test)

# MIP = 0, STDIP = 1, EKIP = 2, SIP = 3, MC = 4, STDC = 5, EKC = 6, SC = 7

print(f"MAX SIP в тенировочном наборе: {data\_scaler\_train.transpose()[3].max()}")

print(f"MAX SIP в тестовом наборе: {data\_scaler\_test.transpose()[3].max()}\n")

reg\_train = LogisticRegression(random\_state=random\_state, solver=solver).fit(data\_scaler\_train, y\_train)

reg\_test = LogisticRegression(random\_state=random\_state, solver=solver).fit(data\_scaler\_test, y\_test)

# MIP = 0, STDIP = 1, EKIP = 2, SIP = 3, MC = 4, STDC = 5, EKC = 6, SC = 7

print(f"Коэффициент при MIP, если тренировочный набор: {reg\_train.coef\_[0][0]}")

print(f"Коэффициент при MIP, если тестовый набор: {reg\_test.coef\_[0][0]}\n")

print(f"Accuracy на тестовых данных, если тренировочный набор: {reg\_train.score(data\_scaler\_test, y\_test)}")

print(f"Accuracy на тестовых данных, если тестовый набор: {reg\_test.score(data\_scaler\_test, y\_test)}\n")

STAR = scaler.transform([STAR])

predict = reg\_train.predict(STAR)[0] # Получаем назначенный класс

print(f"Ввероятность отнесения звезды к назначенному классу: {reg\_train.predict\_proba(STAR)[0][predict]}")

neigh = KNeighborsClassifier(n\_neighbors=k, p=2) # p=1 - manhattan, p=2 - euclidean

neigh.fit(data\_scaler\_train, y\_train)

print(f"При k = {k} относится к классу: {neigh.predict(STAR)[0]}\n")

for m in ["euclidean", "manhattan"]:

    dist = DistanceMetric.get\_metric(m)

    distances = [dist.pairwise(np.concatenate(([i], STAR)))[0][1] for i in list(data\_scaler\_train)]

    print("Расстояние до ближайшего по метрике \"" + m + "\":", min(distances))