Таблица1

Результаты оптимизации

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ширина окна | gaussian | tophat | epanechnikov | cosine | linear | exponential |
| 2 | -781.29 | -503.99 | -461.38 | -457.96 | -454.47 | -1031.69 |
| 3 | -929.38 | -634.95 | -581.86 | -577.47 | -572.59 | -1183.59 |
| 4 | -1041.27 | -738.03 | -677.95 | -672.81 | -666.66 | -1295.78 |
| 5 | -1131.59 | -822.99 | -757.55 | -751.85 | -744.59 | -1384.96 |
| 6 | -1207.24 | -894.27 | -825.12 | -819.00 | -810.78 | -1459.03 |
| 7 | -1272.23 | -957.08 | -883.89 | -877.41 | -868.32 | -1522.39 |
| 8 | -1329.12 | -1014.87 | -936.63 | -929.75 | -919.75 | -1577.77 |
| 9 | -1379.67 | -1066.80 | -984.62 | -977.33 | -966.41 | -1626.95 |
| 10 | -1425.12 | -1113.27 | -1028.45 | -1020.81 | -1008.98 | -1671.18 |
| 11 | -1466.40 | -1155.30 | -1068.61 | -1060.69 | -1048.01 | -1711.38 |

Согласно заданию, выполнил расчеты, результаты оптимизации приведены в таблице 1.

Из полученных данных можно сделать вывод, увеличивая ширину окна эффективность алгоритма теряется.

В моем случае лучший результат показал linear, особенно с шириной окна 2, но также не плохи epanechnikov, cosine.

**Код программы**

import pandas as pd

url = "C:/Users/fox/Desktop/ii/data/seeds\_dataset.data"

names = ["площадь","периметр","компактность", "длина","ширина","асимметрия","длина канавки ядра","сорт"] #название атрибутов

target\_names = { 1:"Кама" , 2:"Роза" , 2:"Канадка" }

dataset = pd.read\_csv(url, names=names)

dataset.head()

X = dataset.iloc[:,0:7]

y = dataset['сорт'].values

print(X)

from sklearn import preprocessing

scaler = preprocessing.StandardScaler().fit(X)

X\_scal = scaler.transform(X)

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

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X\_scal, y, test\_size = 0.30)

from sklearn.neighbors import KernelDensity

kernels=["gaussian", "tophat", "epanechnikov", "exponential", "linear", "cosine"]

for j in range(6):

    print( '\n' +(kernels[j-1]))

    kernelIndex = j-1

#выыод расчета с соответствующими kernel

    for i in range(2,12):

        kd = KernelDensity(bandwidth=i, kernel=kernels[kernelIndex])

        kd.fit(X\_train)

        print("%.2f" % kd.score(X\_test))