Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра: «Вычислительная техника»

Дисциплина: «Системы искусственного интеллекта»

Лабораторная работа №5

Алгоритмы кластеризации данных

Вариант 20

Выполнил

Студент группы ИВТАСбд-32

Хасанов И. Н.

Проверил:

преподаватель кафедры «ВТ»

Хайруллин И. Д.

Ульяновск 2024

**Оглавление**

[Постановка задачи 3](#_Toc167466900)

[Реализация 4](#_Toc167466901)

[Вывод 5](#_Toc167466902)

[Приложение. Source.cpp 6](#_Toc167466903)

# Постановка задачи

1. Произвести масштабирование признаков (scaling).
2. С использованием библиотеки scikit-learn написать программу с использованием алгоритмов кластеризации данных, позволяющую разделить исходную выборку на классы, соответствующие предложенной вариантом задаче (http://scikit-learn.org/stable/modules/clustering.html).
3. Провести эксперименты и определить наилучший алгоритм кластеризации, параметры алгоритма. Необходимо использовать не менее 3-х алгоритмов. Данные экспериментов необходимо представить в отчете (графики, ход проведения эксперимента, выводы).

Вариант: wiki4HE.

# Реализация

Код написан на языке Python. Вначале считываются данные из файла и записываются в датафрейм. После производится масштабирование признаков с помощью preprocessing.StandardScaler(). Масштабирование производится с помощью стандартизации - математическое ожидание каждой колонки становится равным 0, а его стандратное отклонение становится равным 1.

Далее используются алгоритмы кластеризации KMeans, MiniBatchKMeans и AffinityPropagation, оцениваются их точности через normalized\_mutual\_info\_score().

Нахождение лучших параметров (n\_init и n\_clusters для KMeans и MiniBatchKMeans, damping и preference для AffinityPropagation) находятся путем перебора.







Лучшим оказался метод AffinityPropagation.

# Вывод

Проделав данную лабораторную работу, были изучены алгоритмы кластеризации и подбор параметров для них.

# Приложение. Source.cpp

|  |
| --- |
| import numpy as np  import pandas as pd  from sklearn import preprocessing  from sklearn.metrics import normalized\_mutual\_info\_score  from sklearn.cluster import KMeans  from sklearn.cluster import MiniBatchKMeans  from sklearn.cluster import AffinityPropagation  Загрузка данных.  df = pd.read\_csv('wiki4HE.csv', sep=';')  df = df.apply(pd.to\_numeric, errors='coerce')  df.dropna(how="any", inplace=True, axis=0)  df.head()  x\_data = np.array(df.drop('YEARSEXP', axis=1))  y\_data = np.array(df['YEARSEXP'])  scaler = preprocessing.StandardScaler()  x\_data = scaler.fit\_transform(x\_data)  #KMeans  kmeans = KMeans(n\_clusters=3)  kmeans.fit(x\_data)  kmeans\_score = normalized\_mutual\_info\_score(y\_data, kmeans.labels\_)  print(kmeans.labels\_)  print("Точность кластеризации KMeans: ", kmeans\_score)  #MiniBatchKMeans  mini\_batch\_kmeans = MiniBatchKMeans(n\_clusters=3)  mini\_batch\_kmeans.fit(x\_data)  mini\_batch\_kmeans\_score = normalized\_mutual\_info\_score(y\_data, mini\_batch\_kmeans.labels\_)  print("Точность кластеризации MiniBatchKMeans: ", kmeans\_score)  #AffinityPropagation  affinity\_propagation = AffinityPropagation()  affinity\_propagation.fit(x\_data)  affinity\_propagation\_score = normalized\_mutual\_info\_score(y\_data, affinity\_propagation.labels\_)  print("Точность кластеризации AffinityPropagation: ", affinity\_propagation\_score)  best\_n\_clusters = 1  best\_n\_init = 1  best\_kmeans\_score = -1e20  for n\_clusters in range(1, 5):  for n\_init in range(1, 20):  kmeans = KMeans(n\_clusters=n\_clusters, n\_init=n\_init)  kmeans.fit(x\_data)  kmeans\_score = normalized\_mutual\_info\_score(y\_data, kmeans.labels\_)    if kmeans\_score > best\_kmeans\_score:  best\_kmeans\_score = kmeans\_score  best\_n\_clusters = n\_clusters  best\_n\_init = n\_init  print('Лучшая точность KMeans: ', best\_kmeans\_score)  print('Лучшее значение параметра n\_clusters: ', best\_n\_clusters)  print('Лучшее значение параметра n\_init: ', best\_n\_init)  best\_n\_clusters = 1  best\_n\_init = 1  best\_mini\_batch\_kmeans\_score = -1e20  for n\_clusters in range(1, 5):  for n\_init in range(1, 20):  mini\_batch\_kmeans = MiniBatchKMeans(n\_clusters=n\_clusters, n\_init=n\_init)  mini\_batch\_kmeans.fit(x\_data)  mini\_batch\_kmeans\_score = normalized\_mutual\_info\_score(y\_data, mini\_batch\_kmeans.labels\_)    if mini\_batch\_kmeans\_score > best\_mini\_batch\_kmeans\_score:  best\_mini\_batch\_kmeans\_score = mini\_batch\_kmeans\_score  best\_n\_clusters = n\_clusters  best\_n\_init = n\_init  print('Лучшая точность MiniBatchKMeans: ', best\_mini\_batch\_kmeans\_score)  print('Лучшее значение параметра n\_clusters: ', best\_n\_clusters)  print('Лучшее значение параметра n\_init: ', best\_n\_init)  best\_damping = 1  best\_preference = 1  best\_affinity\_propagation\_score = -1e20  for damping in np.linspace(0.5, 0.8, 20):  for preference in np.linspace(-50, 50, 20):  affinity\_propagation = AffinityPropagation(damping=damping, preference=preference)  affinity\_propagation.fit(x\_data)  affinity\_propagation\_score = normalized\_mutual\_info\_score(y\_data, affinity\_propagation.labels\_)    if affinity\_propagation\_score > best\_affinity\_propagation\_score:  best\_affinity\_propagation\_score = affinity\_propagation\_score  best\_damping = damping  best\_preference = preference  print('Лучшая точность AffinityPropagation: ', best\_affinity\_propagation\_score)  print('Лучшее значение параметра damping: ', best\_damping)  print('Лучшее значение параметра preference: ', best\_preference)  #### Вывод  method\_name = ['KMeans', 'MiniBatchKMeans', 'AffinityPropagation']  scores = [best\_kmeans\_score, best\_mini\_batch\_kmeans\_score, best\_affinity\_propagation\_score]  best\_method\_name = method\_name[np.argmax(scores)]  for id, name in enumerate(method\_name):  print(f'Точность метода {name}: {scores[id]}')  print(f'\nИтог: лучшим оказался метод {best\_method\_name}') |