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 15

Выполнила

Студентка группы ИВТАСбд-31

Микка Е. И.

Ульяновск, 2024

# Общее задание на лабораторную работу

В данной лабораторной работе необходимо:

1. Произвести масштабирование признаков.
2. Разделить исходную выборку на классы.
3. Провести эксперименты для определения наилучшего алгоритма кластеризации и наилучших его параметров.

# Ход выполнения с описанием программной реализации

Задание 1.

Код извлекает признаки и целевое значение из набора данных, преобразует данные в массивы NumPy для удобства работы, удаляет первый столбец признаков, так как он содержит нечисловые значения, и инициализирует StandardScaler для стандартизации признаков.

X = wholesale\_customers.data.features  
Y = wholesale\_customers.data.targets  
  
X = X.values  
X = X[:, 1:]  
Y = Y.values.ravel()  
  
scaler = StandardScaler()  
  
X = scaler.fit\_transform(X)

Задание 2.

Этот код выполняет кластеризацию данных с использованием трех различных алгоритмов и визуализирует результаты.

KMeans разделяет данные на четыре кластера, обучаясь на наборе данных X и сравнивая полученные метки с исходными метками Y. AffinityPropagation автоматически определяет количество кластеров, обучаясь на данных X и оценивая сходство с метками Y. AgglomerativeClustering группирует данные в четыре кластера, используя иерархический подход, и также оценивает сходство с исходными метками Y.

Визуализация проводится по двум выбранным признакам.

kmeans = KMeans(n\_clusters=4)  
kmeans.fit(X)  
kmeans\_score = normalized\_mutual\_info\_score(Y, kmeans.labels\_)  
  
plt.scatter(X[:, 2], X[:, 4], c=kmeans.labels\_, s=10)  
plt.title('Алгоритм кластеризации - Kmeans')  
plt.show()  
   
affinity\_propagation = AffinityPropagation()  
affinity\_propagation.fit(X)  
affinity\_propagation\_score = normalized\_mutual\_info\_score(Y, affinity\_propagation.labels\_)  
  
plt.scatter(X[:, 2], X[:, 4], c=affinity\_propagation.labels\_, s=10)  
plt.title('Алгоритм кластеризации - AffinityPropagatio')  
plt.show()  
  
agglomerative = AgglomerativeClustering(n\_clusters=4)  
agglomerative.fit(X)  
agglomerative\_score = normalized\_mutual\_info\_score(Y, agglomerative.labels\_)  
  
plt.scatter(X[:, 2], X[:, 4], c=agglomerative.labels\_, s=10)  
plt.title('Алгоритм кластеризации - AgglomerativeClustering')  
plt.show()

Задание 3.

Этот код оценивает и сравнивает эффективность трех методов кластеризации на наборе данных X с истинными метками Y.

KMeans

Создает матрицу acc для хранения оценок точности. Использует KMeans с различным количеством кластеров (n\_clusters) и разным количеством инициализаций (n\_init). Находит наилучшую точность и параметры, при которых она была достигнута.

AffinityPropagation

Проводит аналогичную оценку для Affinity Propagation, изменяя параметры damping (затухание) и preference (предпочтение). Определяет наилучшие параметры для максимальной точности.

AgglomerativeClustering

Оценивает агломеративную кластеризацию с разным количеством кластеров и различными типами связей (linkage). Также находит наилучшие параметры для максимальной точности.

Код выводит наилучшие результаты для каждого метода, что позволяет выбрать оптимальный алгоритм и его параметры для кластеризации данных.  
acc = np.zeros((8, 20))  
for cl in range(8):  
 for init in range(20):  
 kmeans = KMeans(n\_clusters=cl+1, n\_init=init+1)  
 kmeans.fit(X)  
 acc[cl][init] = normalized\_mutual\_info\_score(Y, kmeans.labels\_)  
  
best\_acc = np.max(acc)  
best\_index = np.argmax(acc)  
best\_cl, best\_init = np.unravel\_index(best\_index, acc.shape)  
  
print(f"KMeans\n"  
 f"Наилучшая точность - {best\_acc}\n"  
 f"Наилучший параметр n\_clusters - {best\_cl+1}\n"  
 f"Наилучший параметр n\_init - {best\_init+1}\n")  
  
par\_damping = [0.5, 0.6, 0.7, 0.8, 0.9]  
par\_preference = np.linspace(-50, 50, 20)  
acc = np.zeros((5, 20))  
for dm in range(5):  
 for pr in range(20):  
 affinity\_propagation = AffinityPropagation(damping=par\_damping[dm], preference=par\_preference[pr])  
 affinity\_propagation.fit(X)  
 acc[dm][pr] = normalized\_mutual\_info\_score(Y, affinity\_propagation.labels\_)  
  
best\_acc = np.max(acc)  
best\_index = np.argmax(acc)  
best\_dm, best\_pr = np.unravel\_index(best\_index, acc.shape)  
  
print(f"AffinityPropagatio\n"  
 f"Наилучшая точность - {best\_acc}\n"  
 f"Наилучший параметр damping - {par\_damping[best\_dm]}\n"  
 f"Наилучший параметр preference - {par\_preference[best\_pr]}\n")  
  
par\_linkage = ['ward', 'average', 'complete', 'single']  
acc = np.zeros((8, 4))  
for cl in range(8):  
 for la in range(4):  
 agglomerative = AgglomerativeClustering(n\_clusters=cl+1, linkage=par\_linkage[la])  
 agglomerative.fit(X)  
 acc[cl][la] = normalized\_mutual\_info\_score(Y, agglomerative.labels\_)  
  
best\_acc = np.max(acc)  
best\_index = np.argmax(acc)  
best\_cl, best\_la = np.unravel\_index(best\_index, acc.shape)  
  
print(f"AgglomerativeClustering\n"  
 f"Наилучшая точность - {best\_acc}\n"  
 f"Наилучший параметр n\_clusters - {best\_cl+1}\n"  
 f"Наилучший параметр linkage - {par\_linkage[best\_la]}\n")

# 

**Тестирование**

График 1. Исходная выборка



График 2. Алгоритм кластеризации - Kmeans

****

График 3. Алгоритм кластеризации - AffinityPropagatio



График 4. Алгоритм кластеризации - AgglomerativeClustering



Вывод в консоли

