МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Системы искусственного интеллекта**

**Отчет по выполнению лабораторной работы №5**

Выполнил

студент группы ИВТАПбд-31

Гаязов Т. Р.

Проверил:

преподаватель

Хайруллин И. Д.

Ульяновск

2024

**Цель работы**

1. Произвести масштабирование признаков (scaling).
2. С использованием библиотеки [scikit-learn](http://scikit-learn.org/stable/) 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).

**Ход работы**

1. Импорт и подготовка данных

Импортированы необходимые библиотеки: pandas, matplotlib и модули из scikit-learn. Данные загружены из файла movement\_libras\_1.data.

Масштабируем все столбцы кроме последнего (целевой класс) с использованием StandardScaler, что нормализовало различные признаки и улучшило качество кластеризации.

2. Применение алгоритмов кластеризации

В работе использованы три алгоритма кластеризации:

* KMeans:
  + Преимущества: Быстрота работы на больших наборах данных.
  + Недостатки: Требуется заранее знать количество кластеров.
* Affinity Propagation:
  + Преимущества: Не требуется заранее знать количество кластеров.
  + Недостатки: Высокая вычислительная сложность.
* Mean Shift:
  + Преимущества: Автоматически определяет количество кластеров.
  + Недостатки: Высокая вычислительная сложность.

3. Обучение и оценка алгоритмов

Для каждого алгоритма модель обучалась на масштабированных данных. Были получены метки кластеров и рассчитана метрика качества кластеризации — Silhouette Score. Эта метрика помогает определить, насколько хорошо объекты кластеризованы: значение ближе к 1 указывает на хорошую кластеризацию, ближе к 0 — на нахождение объекта на границе между кластерами, отрицательное значение свидетельствует о неправильном присвоении объекта к кластеру.

4. Визуализация результатов

Для каждого алгоритма были построены графики, демонстрирующие распределение данных по кластерам. Объекты данных отображались в виде точек на графиках, окрашенных в зависимости от принадлежности к определенному кластеру.

5. Результаты и анализ

* KMeans: Показал среднее значение Silhouette Score.
* Affinity Propagation: Показал высокую вычислительную сложность и средний Silhouette Score.
* Mean Shift: Показал высокую вычислительную сложность и значение Silhouette Score, сравнимое с другими алгоритмами.

**Тестирование**

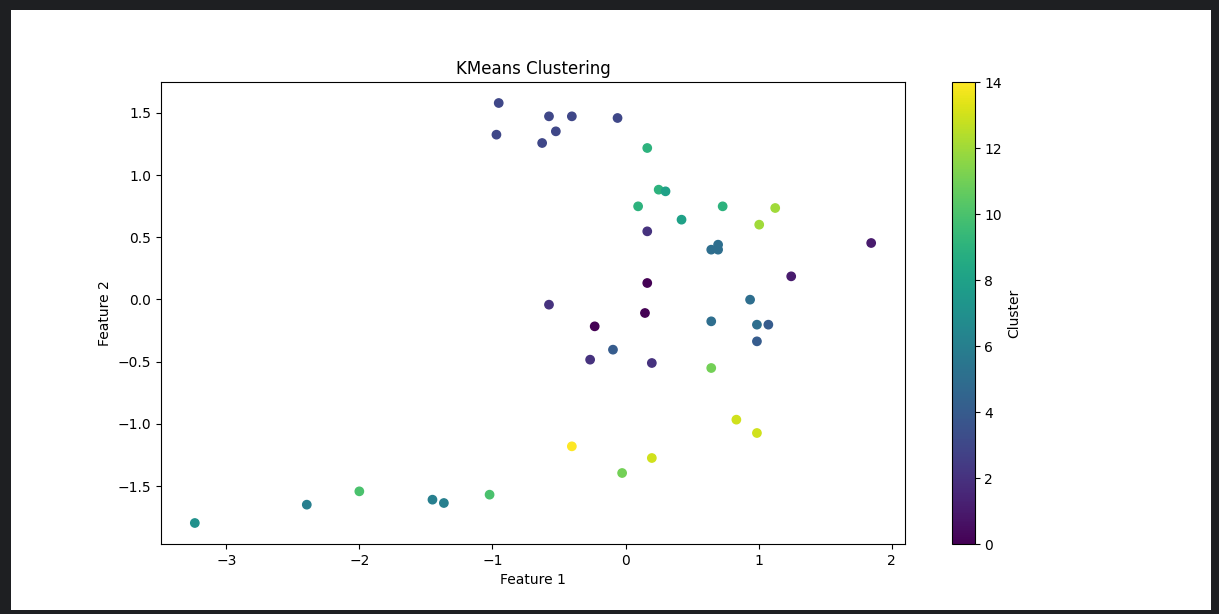


Рис.1. Кластеризация KMeans

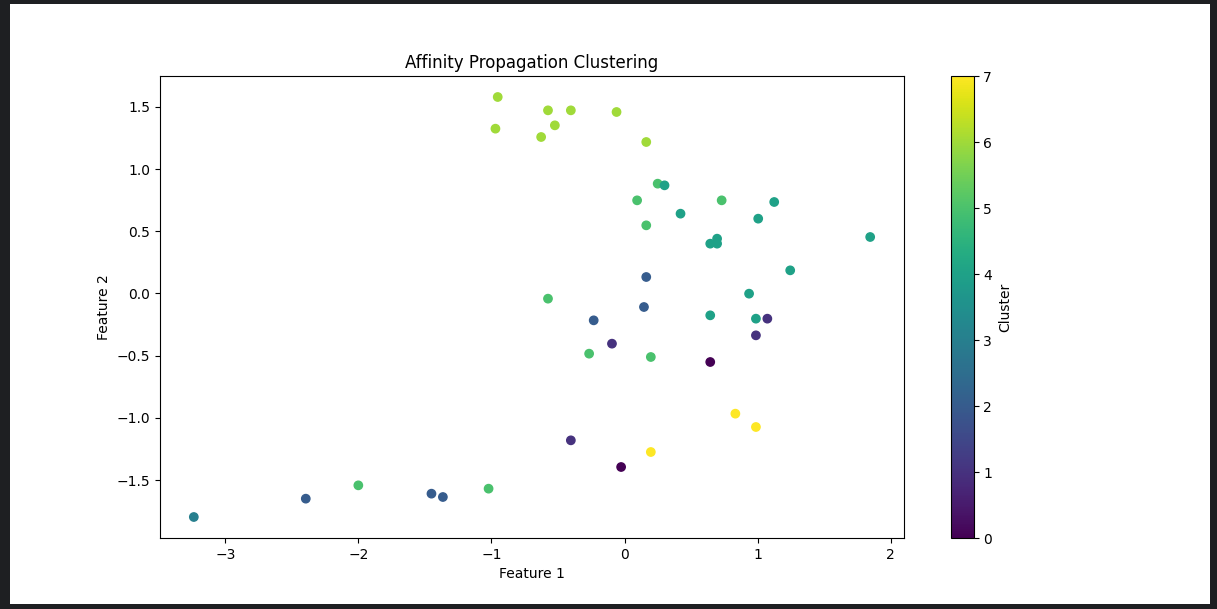


Рис.2. Кластеризация Affinity Propagation

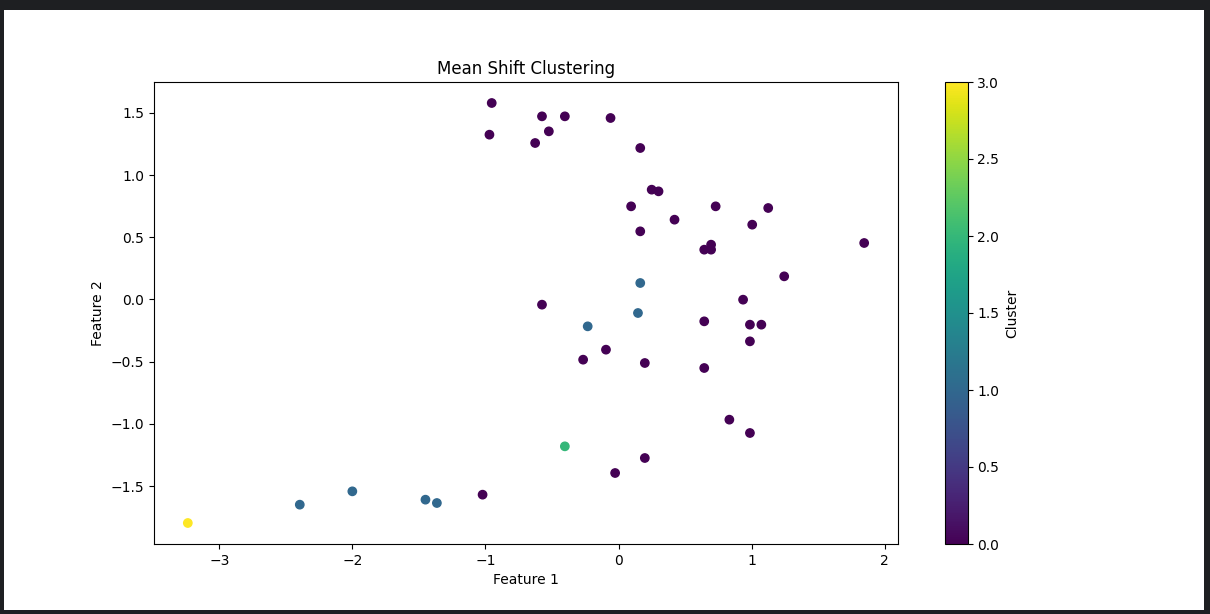


Рис.3. Кластеризация Mean Shift

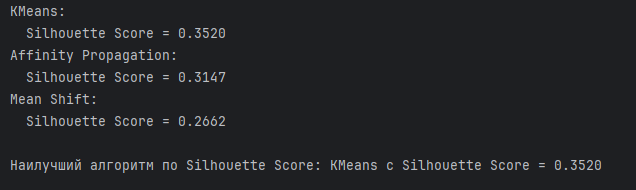


Рис.4 Точность вычислений

**Вывод**

В ходе выполнения данной лабораторной работы были реализованы алгоритмы кластеризации данных, такие как KMeans, Mean Shift и AffinityPropagation, с целью разделения исходной выборки на классы. Затем были проведены эксперименты с целью определения наилучшего алгоритма кластеризации и оптимальных параметров для этого алгоритма.

**Список литературы**

1. <https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html#sphx-glr-auto-examples-linear-model-plot-ols-py>
2. <https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html#sphx-glr-auto-examples-model-selection-plot-underfitting-overfitting-py>
3. <https://scikit-learn.org/stable/modules/linear_model.html>

**Приложение**

|  |
| --- |
| import pandas as pd import matplotlib.pyplot as plt from sklearn.preprocessing import StandardScaler from sklearn.cluster import KMeans, AffinityPropagation, MeanShift from sklearn.metrics import silhouette\_score, calinski\_harabasz\_score, davies\_bouldin\_score  # Загрузка данных file\_path = './data/movement\_libras\_1.data' df = pd.read\_csv(file\_path, header=None)  # Масштабирование признаков scaler = StandardScaler() scaled\_features = scaler.fit\_transform(df.iloc[:, :-1]) # Масштабируем все столбцы кроме последнего (целевой класс)   # Метод локтя для определения оптимального числа кластеров def plot\_elbow\_method(scaled\_features):  distortions = []  K = range(1, 21)  for k in K:  kmean\_model = KMeans(n\_clusters=k, random\_state=42)  kmean\_model.fit(scaled\_features)  distortions.append(kmean\_model.inertia\_)   plt.figure(figsize=(12, 6))  plt.plot(K, distortions, 'bx-')  plt.xlabel('Количество кластеров')  plt.ylabel('Искажение')  plt.title('Метод локтя для оптимального числа кластеров')  plt.show()   plot\_elbow\_method(scaled\_features)  algorithms = {  "KMeans": KMeans(n\_clusters=15, random\_state=42), # Указываем 15 кластеров согласно числу классов в данных  "Affinity Propagation": AffinityPropagation(),  "Mean Shift": MeanShift() # метод сдвига среднего. }  for name, algorithm in algorithms.items():  model = algorithm.fit(scaled\_features)  labels = model.labels\_  silhouette\_avg = silhouette\_score(scaled\_features, labels)    results.append((name, silhouette\_avg))  print(f"{name}:")  print(f" Silhouette Score = {silhouette\_avg:.4f}") #метрика оценки качества кластеризации, которая помогает определить, насколько хорошо объекты кластеризованы.  # ближе к 1, хорошо кластеризованы, ближе к 0, указ на нахождение объекта на границе м-ду кластерами, если отриц. то объект неправильно присвоен к кластеру  # Визуализация кластеров (используем первые два признака для 2D визуализации)  plt.figure(figsize=(12, 6))  plt.scatter(scaled\_features[:, 0], scaled\_features[:, 1], c=labels, cmap='viridis')  plt.title(f"{name} Clustering")  plt.xlabel('Feature 1')  plt.ylabel('Feature 2')  plt.colorbar(label='Cluster')  plt.show()  # Определение наилучшего алгоритма по silhouette score best\_algorithm = max(results, key=lambda x: x[1]) print(f"\nНаилучший алгоритм по Silhouette Score: {best\_algorithm[0]} с Silhouette Score = {best\_algorithm[1]:.4f}") |