МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

ФГБОУ ВО «ВГУ»

Отчет по теме

Кластерный анализ

Задание №15

09.03.02 Информационные системы и технологии

Программная инженерия в информационных системах

Зав. кафедрой_____ С. Д. Махортов, д.ф-м.н. __.__.20__

Обучающийся_____ М.Д. Шеменев, 4 курс, 5.1 группа

Руководитель_____ И.В. Илларионов, доцент

Воронеж 2023

Содержание

Введение

Целью данной работы является проведение кластерного анализа определенного набора данных при помощи метода иерархической кластеризации и метода k-средних. По полученным результатам сделать выводы, отобразив ключевые моменты в отчете.

1 Кластерный анализ

1.1 Отбор необходимых переменных

Перед проведением кластерного анализа необходимо подготовить данные, а именно выбрать необходимые числовые значения. В полученном массиве информации имеются значения «NA». Для анализа подобный формат не подойдет, поэтому мы заменяем все ячейки с данным значением на «0». Так как NA это фактически отсутствие голосов избирателей, то замена их на «0» более чем уместна.

За выполнение перечисленных выше преобразований отвечает блок кода:

```
def read_data(file_path):
    input_data = []
    with open(file_path, "r", encoding="utf8") as f:
        f.readline()
        for line in f.readlines():
            input_data.append([float(x) for x in
line.replace('NA', '0').split(";")[1:]])
    return input_data
```

1.2 Иерархическая кластеризация

Перед выполнением кластеризации нам необходимо выполнить нормализацию данных. Это облегчит сравнение, анализ и обработку данных.

```
def normalization(data):
    scaler = preprocessing.MinMaxScaler().fit(data)
    new_data = scaler.transform(data)
    return new_data
```

Нормализация выполняется с помощью функции MinMaxScaler из библиотеки sklearn. Данный метод позволяет произвести приведение числовых переменных к диапазону от 0 до 1.

Теперь мы можем выполнить кластеризацию и отобразить результат работы функции с помощью дендограммы.

Функция, которая выполняет вычисление кластеров и визуализирует результат:

```
def hierarchical_clustering(normal_data, labels):
    distance_matrix = linkage(normal_data,
method=linkage_method, metric=linkage_metric)
    plt.figure(figsize=(25, 15))
    dendrogram(distance_matrix, leaf_font_size=15,
labels=labels)
    plt.title('Hierarchical Clustering Dendrogram')
    plt.ylabel('Distance')
    plt.show()
```

Для кластеризации был выбран иерархический метод, который представлен функцией linkage. В качестве параметров был передан подготовленный набор данных, а также установлены метрики. Метод «ward» минимизирует дисперсию внутри кластеров и, таким образом, стремится создать более однородные кластеры. И в качестве метода выбора минимального расстояния – Евклидово расстояние «Euclidean». Результат работы метода мы отправляем на визуализацию в функцию dendrogram. Мы получаем следующее Рисунок 1.

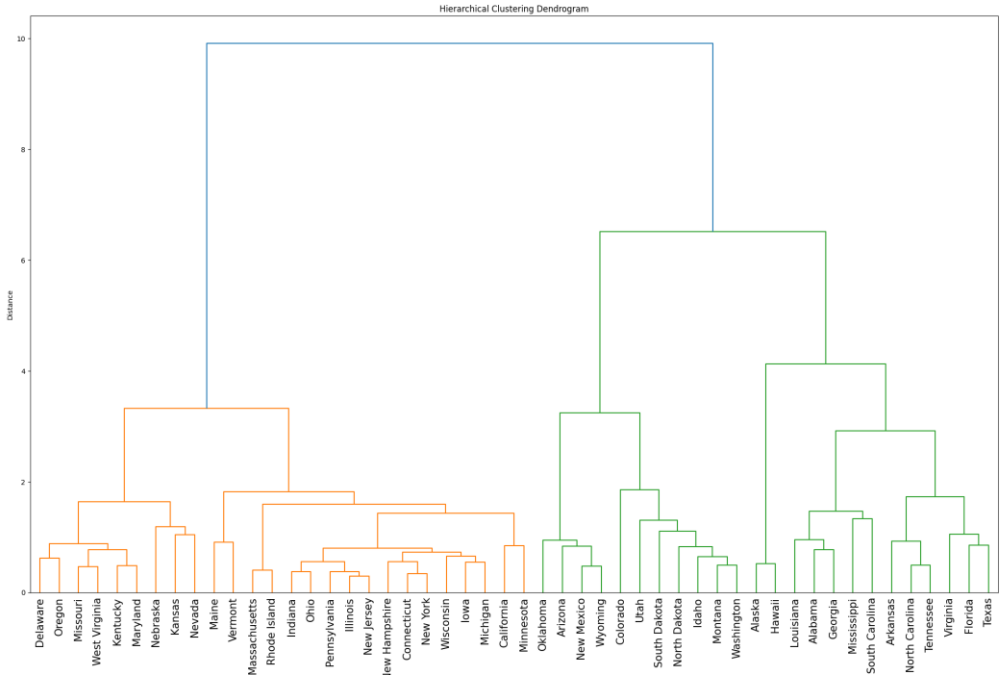

Рисунок 1 - Дендрограмма

Так же полученный результат мы можем визуализировать с помощью шкалирования набора данных. Метод, выполняющий данные действие:

```
def data_set_scaling(data):
    plt.figure(figsize=(10, 7))
    for i in range(1, 4):
        plt.scatter(data[:, i - 1], data[:, i])
    plt.show()
```

Результатом работы метода представлен на Рисунке 2.

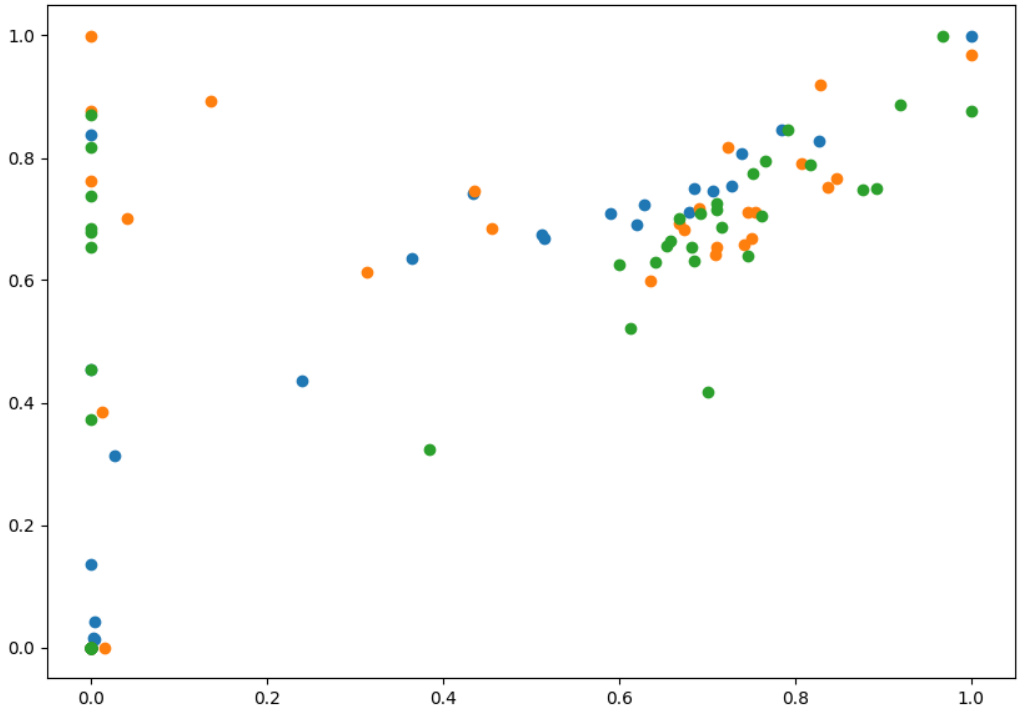

Рисунок 2 - Многомерное шкалирование

2 Определение числа кластеров

Чтобы проверить результат прошлых пунктов мы выполним кластеризацию при помощи метода k-средних. Метод к-средних используется, чтобы «прикинуть» количество кластеров и проверить наличие неучтенных данных и связей в наборах. Чтобы наглядно продемонстрировать результат мы построим график "локоть".

Метод, который выполняет вычисление и построение графика:

```
def elbow_method(new_data):
    WCSS = []
    for k in range(1, 11):
        kmeans = KMeans(n_clusters=k, init="k-means++",
n_init=10, max_iter=400, random_state=0)
        kmeans.fit(new_data)
        WCSS.append(kmeans.inertia_)
    plt.style.use("fivethirtyeight")
    plt.figure(figsize=(10, 10))
    plt.plot(range(1, 11), WCSS, marker='o')
    plt.xticks(range(1, 11))
    plt.xlabel("Number of Clusters")
    plt.ylabel("WCSS")
    plt.show()
```

Для подбора оптимального числа кластеров для исходного набора данных, нам необходимо проанализировать результат метода KMeans с фиксированным числом кластеров, а именно начиная от одного и в плоть до десятого включительно. В результате мы получим график, который представлен на Рисунке 3.

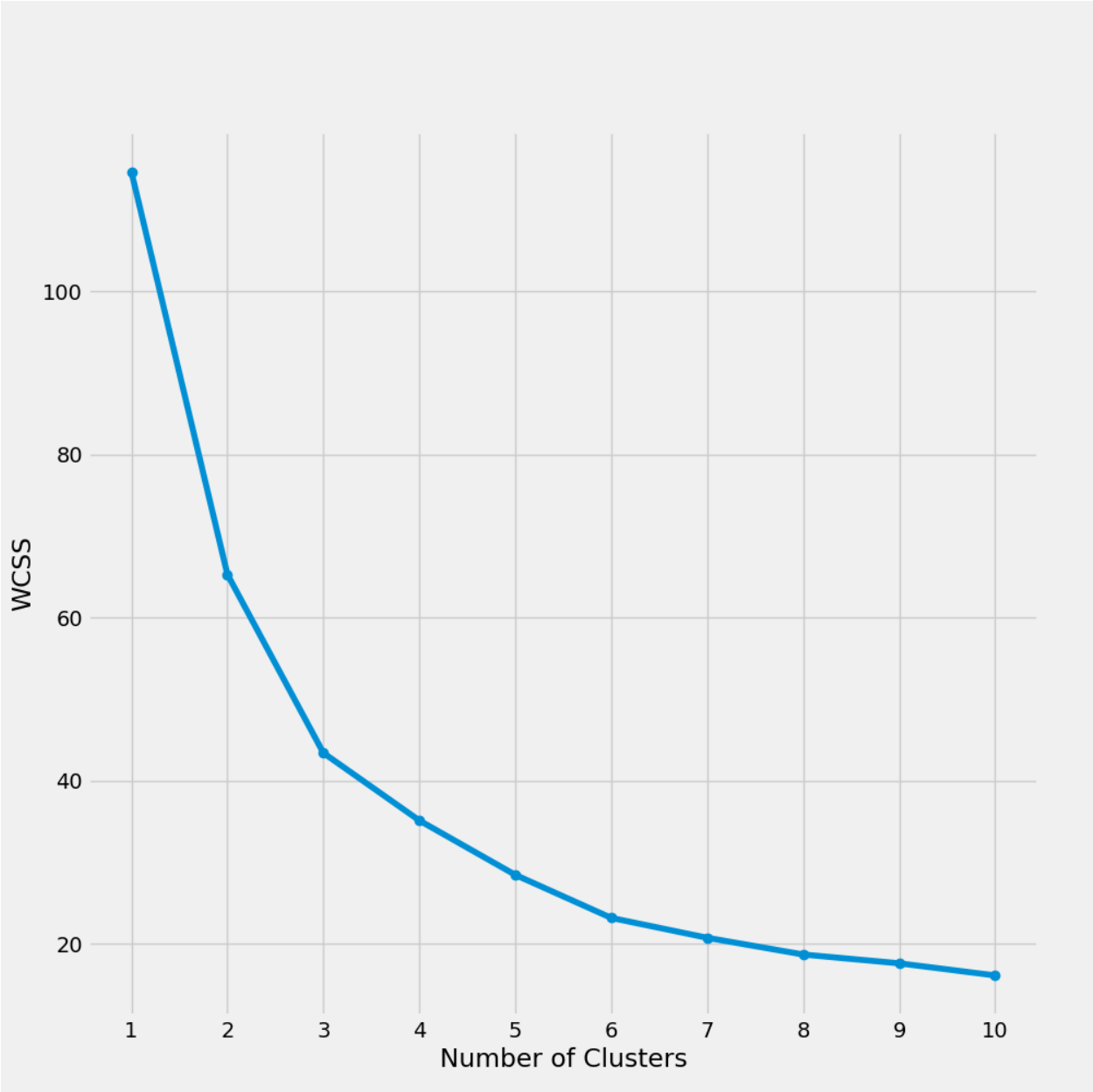

Рисунок 3 - График каменистая осыпь

Заключение

Проанализировав результаты, представленные на графике, можем сказать, что самое оптимальное количество кластеров — это 3.

В результате данной работы мы провели кластерный анализ определенного набора данных при помощи метода иерархической кластеризации и метода k-средних. И описали полученные результаты в виде отчета.