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 3](#_Toc146034208)

[1 Кластерный анализ 4](#_Toc146034209)

[1.1 Отбор необходимых переменных 4](#_Toc146034210)

[1.2 Иерархическая кластеризация 4](#_Toc146034211)

[2 Определение числа кластеров 7](#_Toc146034212)

[Заключение 11](#_Toc146034213)

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. 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. 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

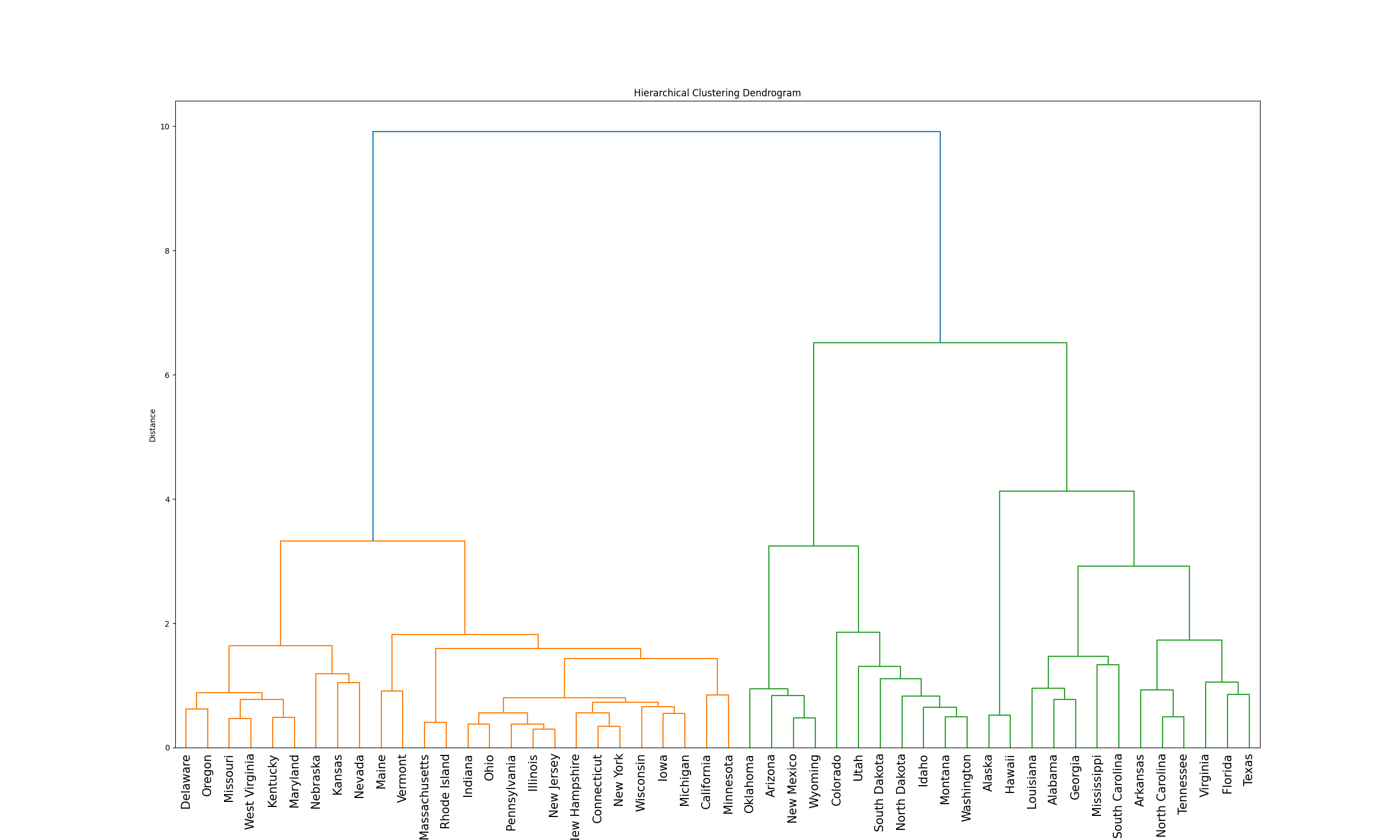
Нормализация выполняется с помощью функции MinMaxScaler из библиотеки sklearn. Данный метод позволяет произвести приведение числовых переменных к диапазону от 0 до 1.

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

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.

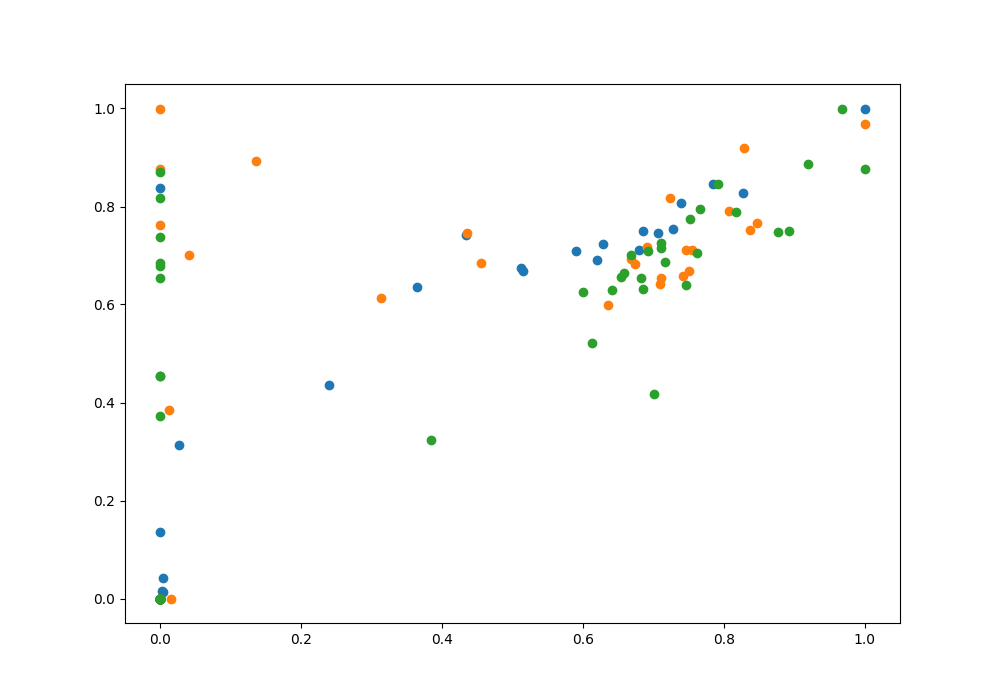


1. Дендрограмма

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.



1. 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.

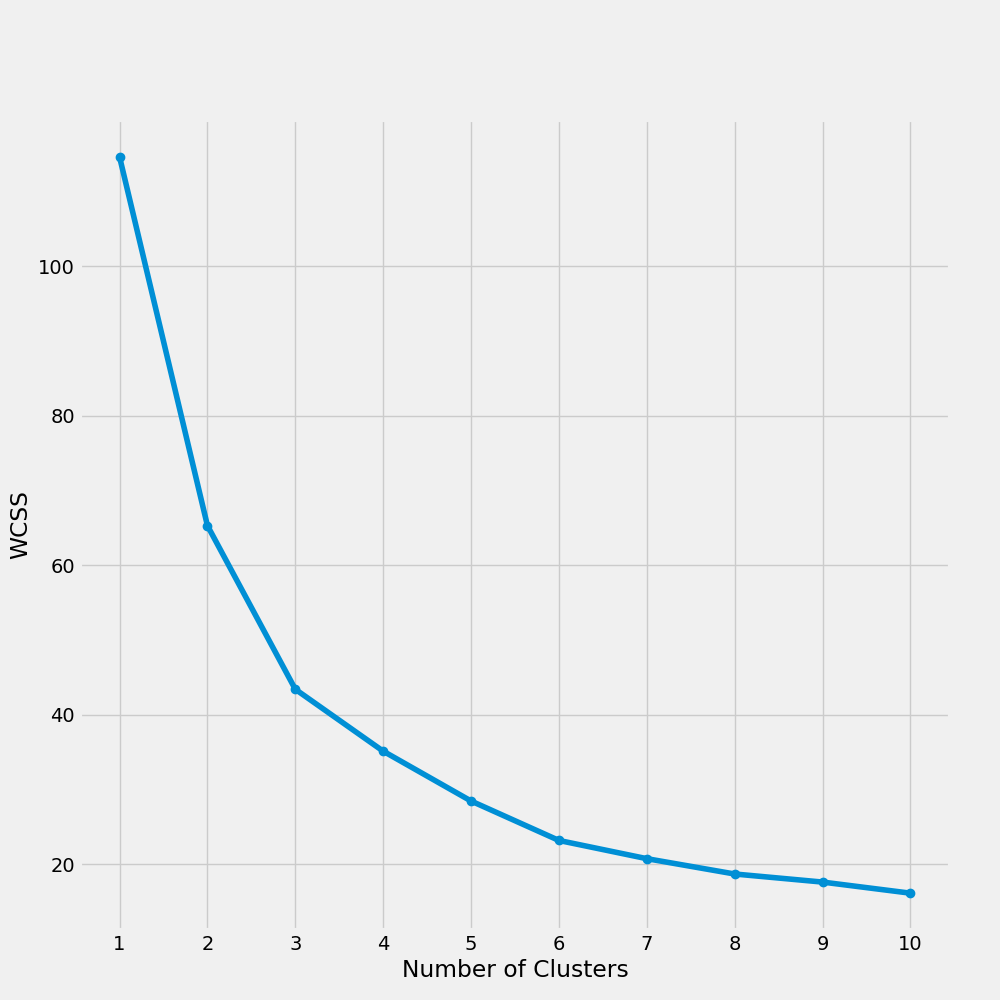


Рисунок 3 - График каменистая осыпь

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.