main

June 12, 2024

1 Data Mining Homework 2: K-means++ & K-medoids

This notebook contains the complete implementation of K-means++ and K-medoids algorithms without using any additional packages.

1.1 Contents

- **Section 1**: My System information
- **Section 2**: Running results of each algorithm with runtime (displayed by Jupyter Notebook)
- **Section 3**: Illustrations and plots of 2D datasets (Aggregation, D31, R15)
- **Section 4**: Part Two of the homework, where you can run the cells yourself and see the results. **Note:** Do not run the system information cell.

1.2 Instructions

- To run all algorithms at the same time (without plots), you can execute the `main.py` file or use Jupyter Notebook.
- Project requirements are listed in `requirements.txt`. You can install them using `pip install -r requirements.txt` (it is recommended to use a virtual environment).
- **Important:** When running the Jupyter Notebook, avoid running the system information cell.

1.3 Runtime

The combined runtime for all algorithms (when running `main.py`) on my system is:

```
python main.py  1.78s user 0.27s system 99% cpu 2.057 total
```

[1]:
```
!neofetch --disable title underline host uptime packages shell resolution de wm␣
↪wm_theme theme icons terminal term_font disk battery local_ip public_ip song␣
↪users
```

```
OS: Manjaro Linux x86_64
Host: BDZ-WXX9 M1010
Kernel: 6.8.12-3-MANJARO
Terminal: code
CPU: 11th Gen Intel i5-1135G7 (8) @ 4.200GHz
GPU: Intel TigerLake-LP GT2 [Iris Xe Graphics]
Memory: 4668MiB / 7741MiB
```

[2]:
```
import os
from src.dataReader import *
from src.algorithms import *
```

[3]:
```
"-----------------Aggregation.data-------------------------"
filePath = os.path.join("data", "Aggregation.data")
data = read_aggregation(filePath)
k = data[data.shape[1] - 1].nunique()
result, purity = run_kmeans_multiple_times(data, k)
print("K-means of Aggregation is done !\n \tnumber of iteration: ", result[2])
```

```
print("\tpurity: ",purity)
result, purity = run_kmedian_multiple_times(data, k)
print("K-median of Aggregation is done !\n \tnumber of iteration: ", result[2])
print("\tpurity: ",purity)
```

```
K-means of Aggregation is done !
        number of iteration:  9
        purity:  0.8972081218274112
K-median of Aggregation is done !
        number of iteration:  12
        purity:  0.932741116751269
```

[4]:
```
"-----------------D31.data-------------------------"
filePath = os.path.join("data", "D31.data")
data = read_D31(filePath)
k = data[data.shape[1] - 1].nunique()
result, purity = run_kmeans_multiple_times(data, k)
print("K-means of D31 is done !\n \tnumber of iteration: ", result[2])
print("\tpurity: ",purity)
result, purity = run_kmedian_multiple_times(data, k)
print("K-median of D31 is done !\n \tnumber of iteration: ", result[2])
print("\tpurity: ",purity)
```

```
K-means of D31 is done !
        number of iteration:  10
        purity:  0.9125806451612903
K-median of D31 is done !
        number of iteration:  5
        purity:  0.9441935483870968
```

[5]:
```
"-----------------R15.data-------------------------"
filePath = os.path.join("data", "R15.data")
data = read_R15(filePath)
k = data[data.shape[1] - 1].nunique()
result, purity = run_kmeans_multiple_times(data, k)
print("K-means of R15 is done !\n \tnumber of iteration: ", result[2])
print("\tpurity: ",purity)
result, purity = run_kmedian_multiple_times(data, k)
print("K-median of R15 is done !\n \tnumber of iteration: ", result[2])
print("\tpurity: ",purity)
```

```
K-means of R15 is done !
        number of iteration:  3
        purity:  0.9966666666666667
K-median of R15 is done !
        number of iteration:  2
        purity:  0.9966666666666667
```

[6]:
```
"-----------------glass.data-------------------------"
filePath = os.path.join("data", "glass.data")
data = read_glass(filePath)
k = data[data.shape[1] - 1].nunique()
result, purity = run_kmeans_multiple_times(data, k)
print("K-means of glass is done !\n \tnumber of iteration: ", result[2])
print("\tpurity: ",purity)
result, purity = run_kmedian_multiple_times(data, k)
print("K-median of glass is done !\n \tnumber of iteration: ", result[2])
print("\tpurity: ",purity)
```

```
K-means of glass is done !
        number of iteration:  4
        purity:  0.5327102803738317
K-median of glass is done !
        number of iteration:  8
        purity:  0.5093457943925234
```

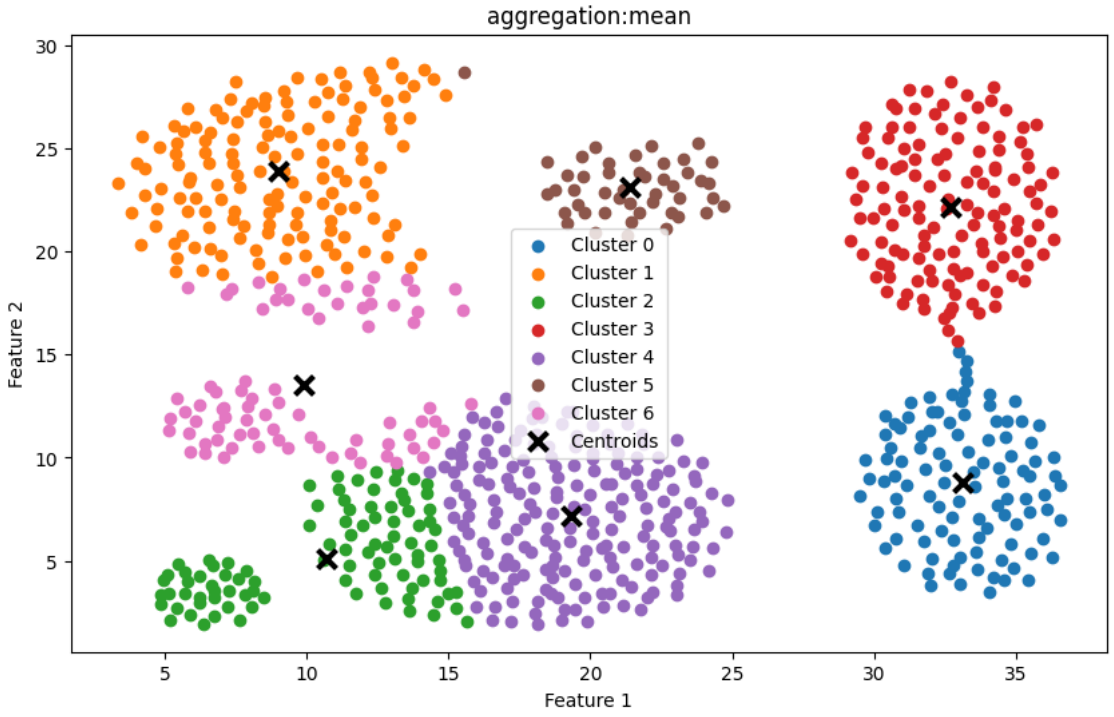
[7]:
```
"-----------------iris.data-------------------------"
filePath = os.path.join("data", "iris.data")
data = read_iris(filePath)
k = data[data.shape[1] - 1].nunique()
result, purity = run_kmeans_multiple_times(data, k)
print("K-means of iris is done !\n \tnumber of iteration: ", result[2])
print("\tpurity: ",purity)
result, purity = run_kmedian_multiple_times(data, k)
print("K-median of iris is done !\n \tnumber of iteration: ", result[2])
print("\tpurity: ",purity)
```

```
K-means of iris is done !
        number of iteration:  10
        purity:  0.8866666666666667
K-median of iris is done !
        number of iteration:  3
        purity:  0.9
```

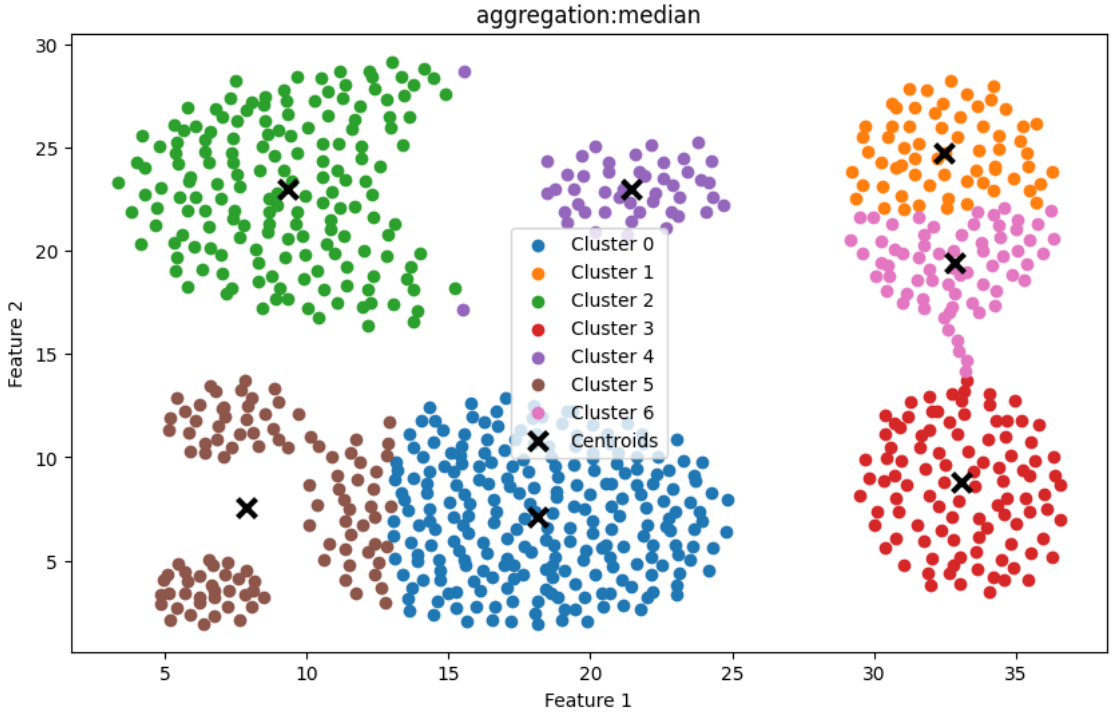
[8]:
```
"Drawing time !!"
from src.plot import draw_clusters
```

[9]:
```
"-----------------Aggregation.data-------------------------"
filePath = os.path.join("data", "Aggregation.data")
data = read_aggregation(filePath)
k = data[data.shape[1] - 1].nunique()
```

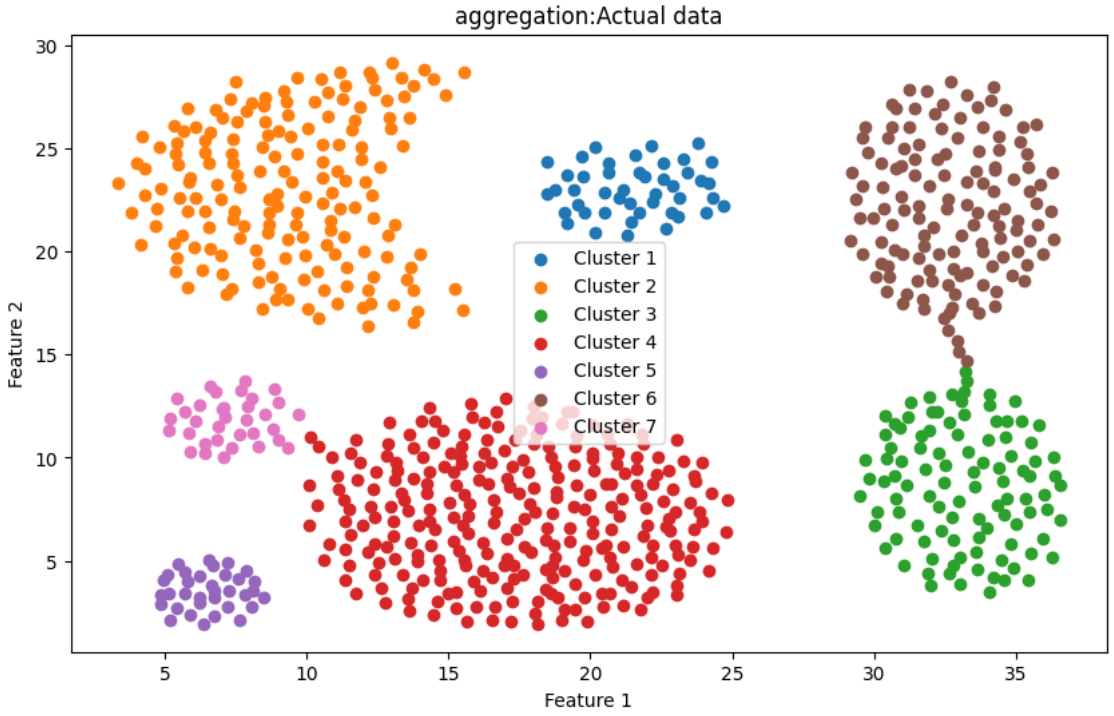
[10]:
```
result, purity = run_kmeans_multiple_times(data, k)
draw_clusters(result[0], result[1], "aggregation", "mean")
```



[11]:
```
result, purity = run_kmedian_multiple_times(data, k)
draw_clusters(result[0], result[1], "aggregation", "median")
```

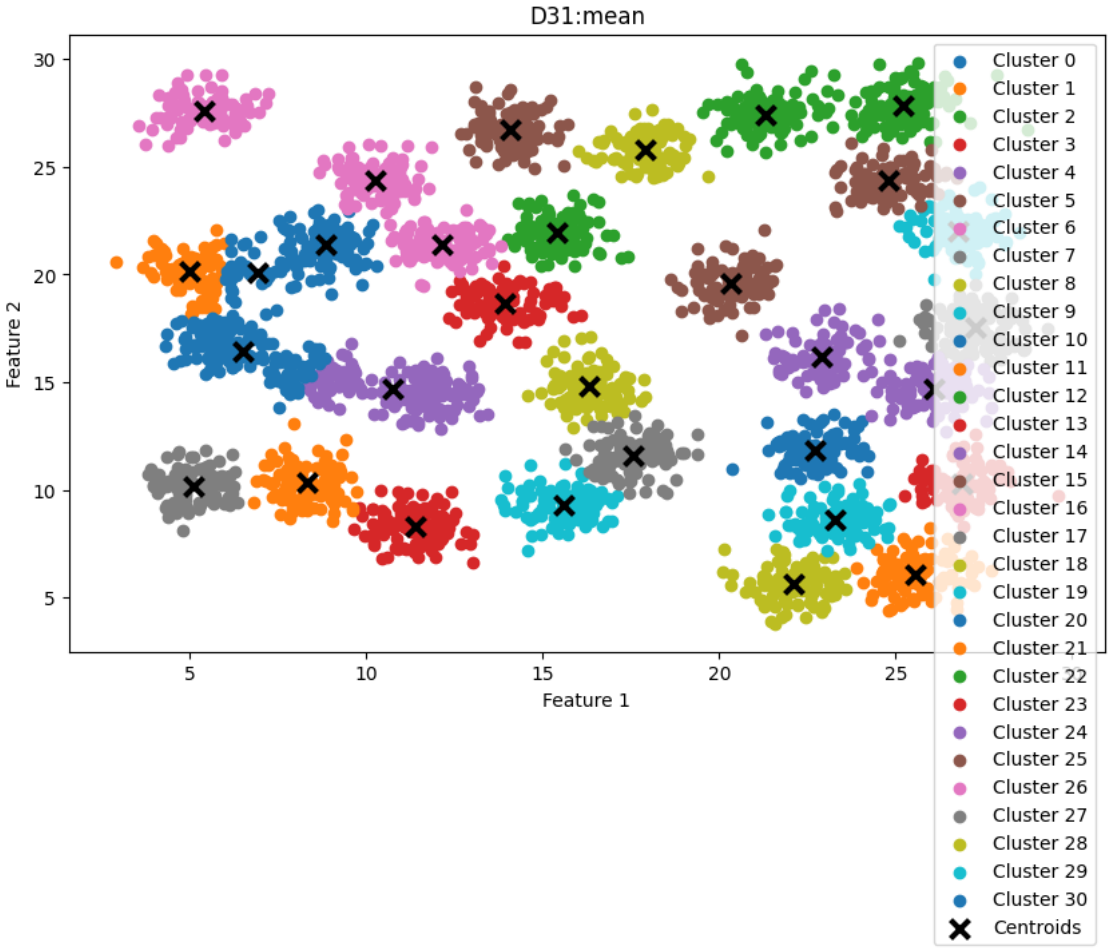


[12]:
```
draw_clusters(data,None, "aggregation", "Actual data")
```

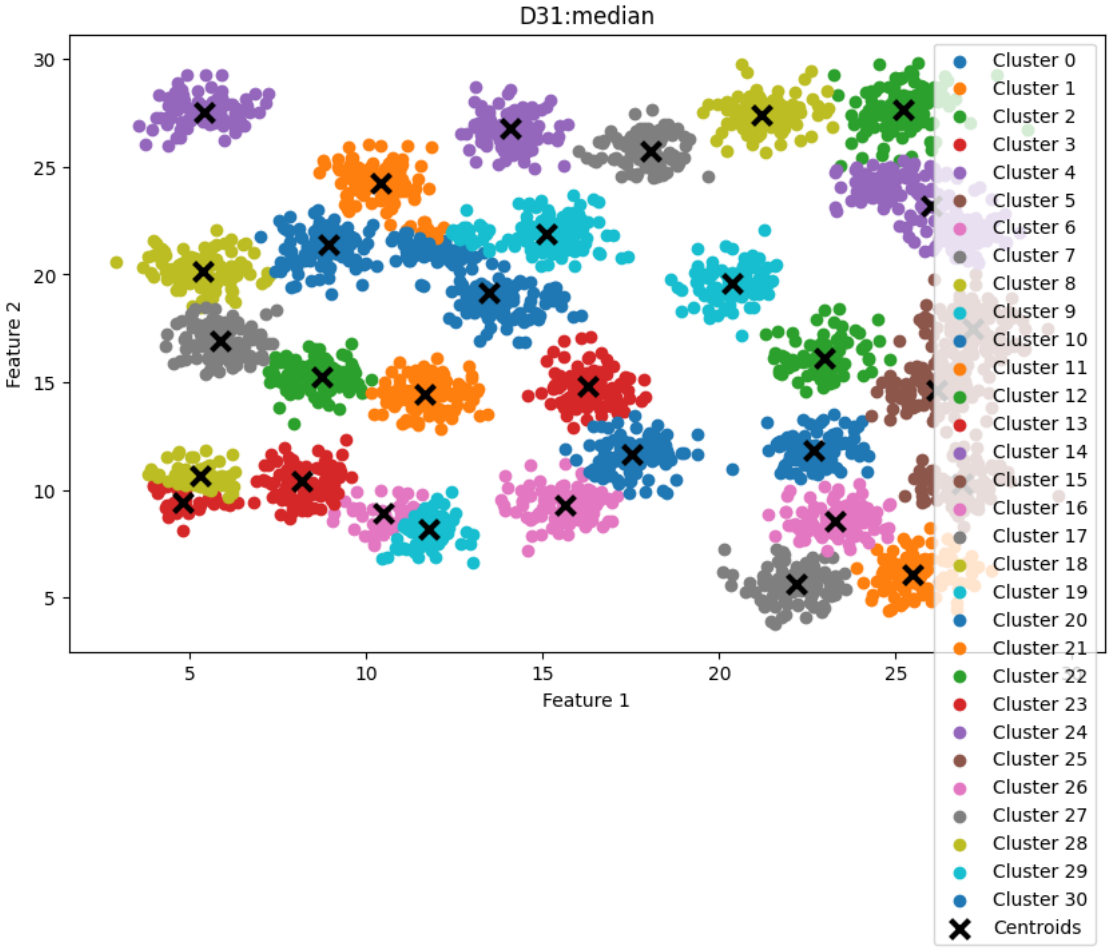


[13]:
```
"-----------------D31.data-------------------------"
filePath = os.path.join("data", "D31.data")
data = read_D31(filePath)
k = data[data.shape[1] - 1].nunique()
```

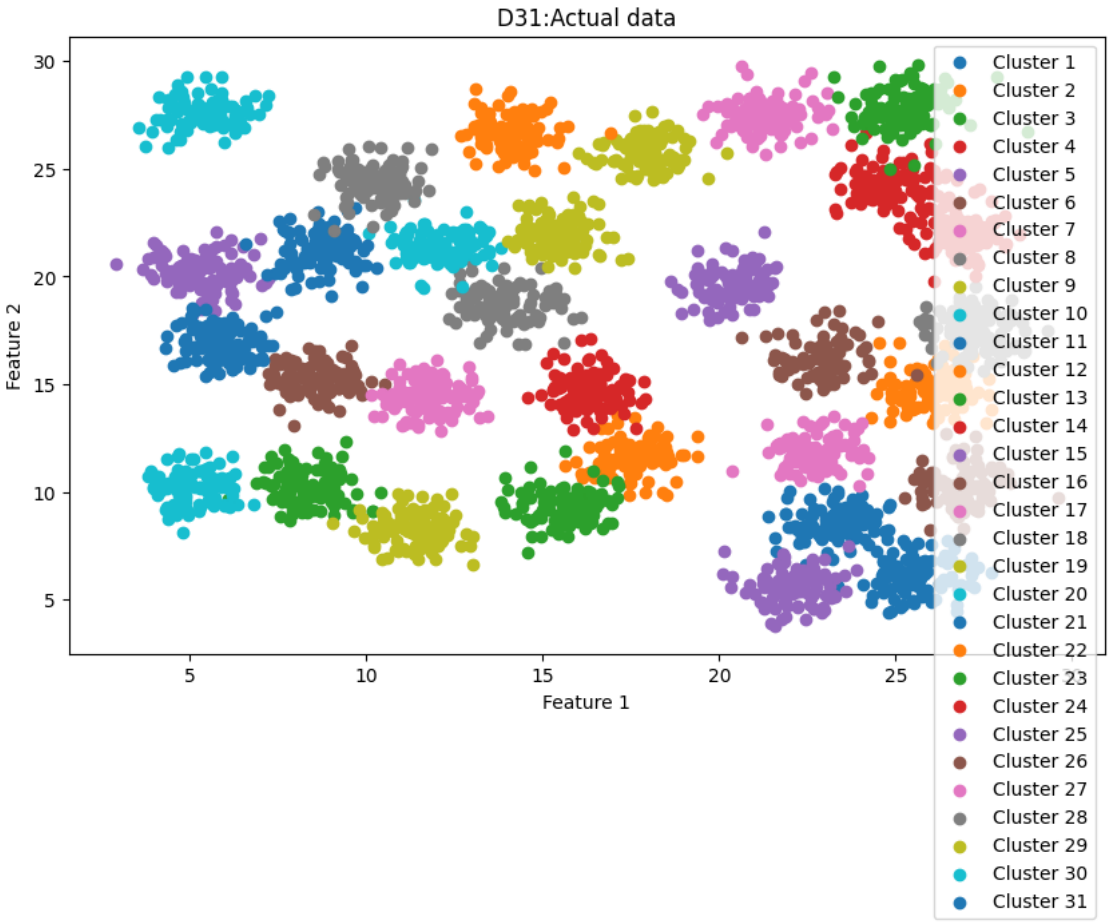
[14]:
```
result, purity = run_kmeans_multiple_times(data, k)
draw_clusters(result[0], result[1], "D31", "mean")
```



[15]:
```
result, purity = run_kmedian_multiple_times(data, k)
draw_clusters(result[0], result[1], "D31", "median")
```

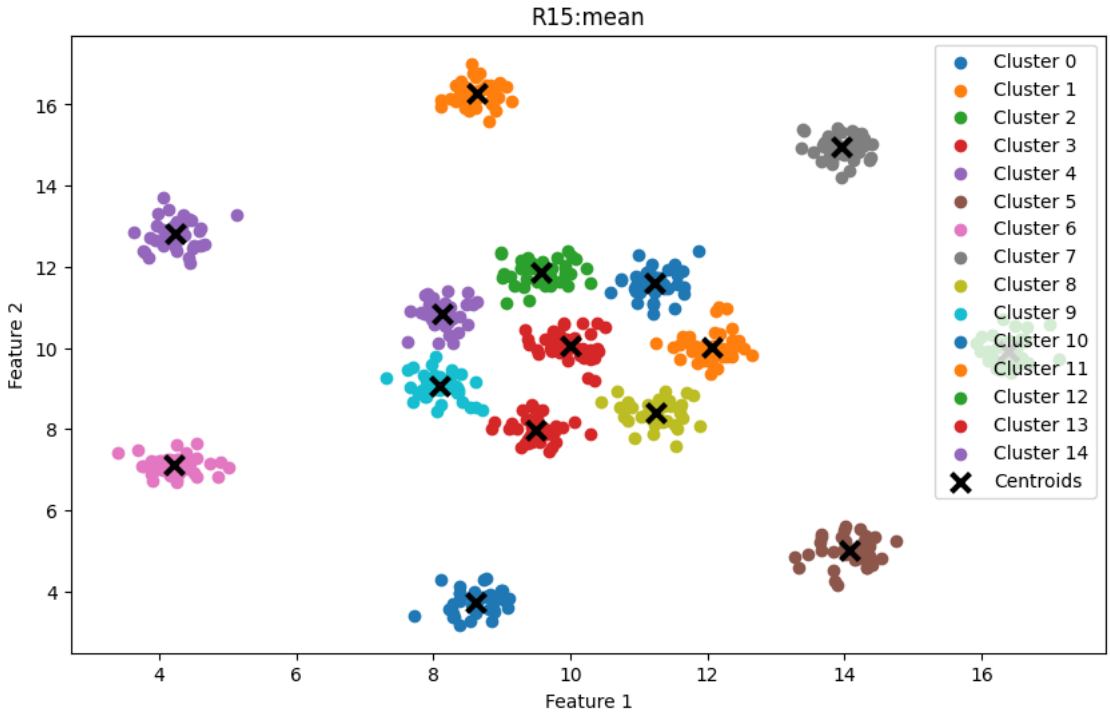


[16]:
```
draw_clusters(data,None, "D31", "Actual data")
```

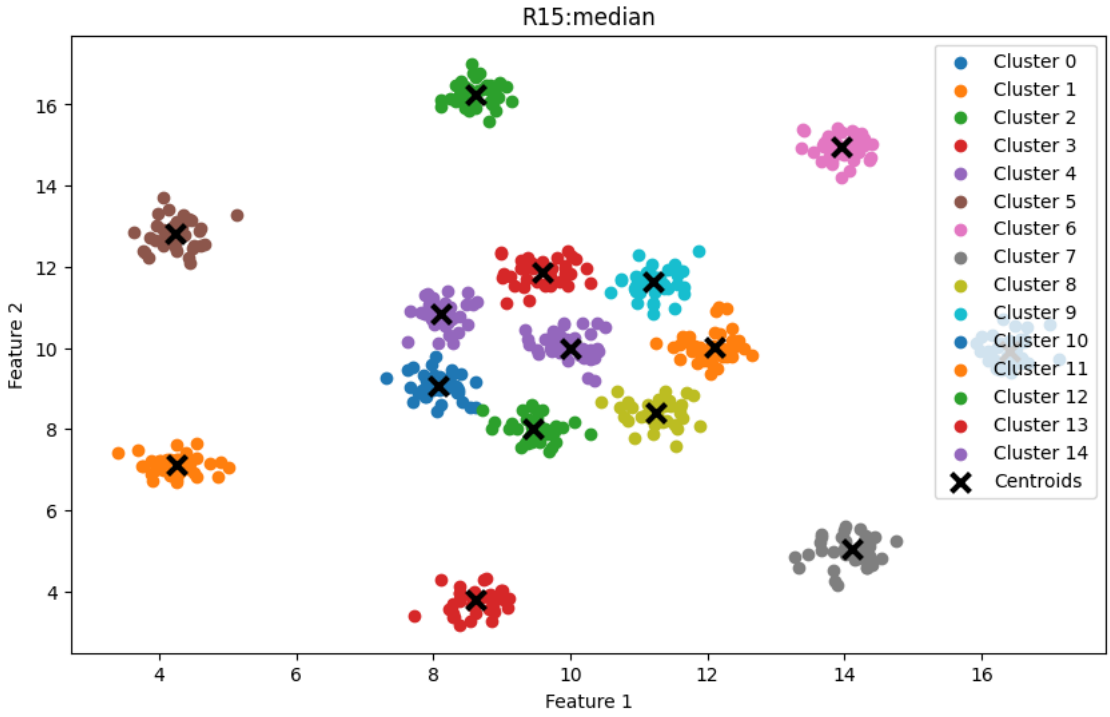


[17]:
```
"-----------------R15.data-------------------------"
filePath = os.path.join("data", "R15.data")
data = read_R15(filePath)
k = data[data.shape[1] - 1].nunique()
```

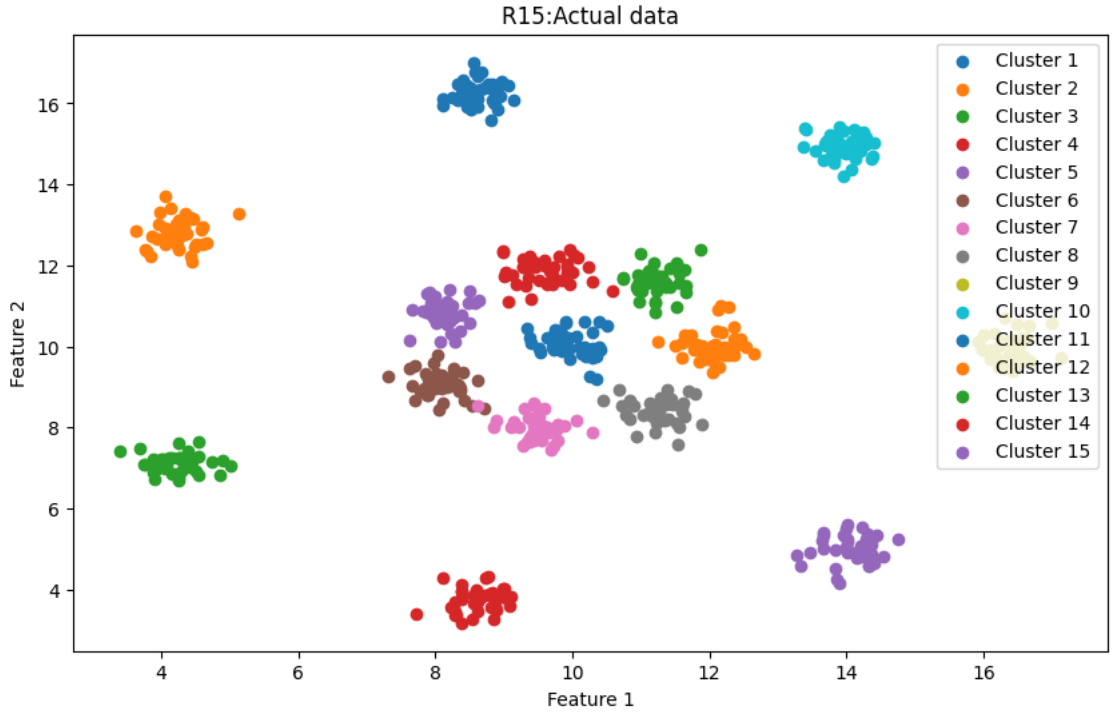
[18]:
```
result, purity = run_kmeans_multiple_times(data, k)
draw_clusters(result[0], result[1], "R15", "mean")
```



[19]:
```
result, purity = run_kmedian_multiple_times(data, k)
draw_clusters(result[0], result[1], "R15", "median")
```



[20]:
```
draw_clusters(data,None, "R15", "Actual data")
```



[21]:
```
"--------------------Home Work Part2----------------------"
from src.algorithms import k_means
import numpy as np
import pandas as pd
data = {
    0: np.array([1, 1.5, 3, 5, 3.5, 4.5, 3.5]),
    1: np.array([1, 2, 4, 7, 5, 5, 4.5])
}
centroids = {
        0: np.array([1, 7]),
        1: np.array([1, 5])
    }

data = pd.DataFrame(data)
centroids = pd.DataFrame(centroids)
k = 2
result= k_means(data, 2, centroids, max_iterations=2,lable = False)
print("\nPart 2 of Home Work")
print("result is:\n", result[0])
```

```
Part 2 of Home Work
result is:
      0    1  cluster
0  1.0  1.0        0
1  1.5  2.0        0
2  3.0  4.0        1
3  5.0  7.0        1
4  3.5  5.0        1
5  4.5  5.0        1
6  3.5  4.5        1
```