Restauracja “Pod Złotymi Łukami”

Dane

Dane, którymi zajmujemy się tutaj zajmujemy to obserwacje reprezentujące dania. W ramach posprzątania danych usunąłem dane nieliczbowe i znormalizowałem je (poszedłem odrobinę na łatwiznę i wszystkie zmapowałem do przedziału [0, 1])

```
Python 3.7.4 (default, Apr  3 2021, 01:30:40)
In[2]: runfile('/Users/petroniuss/Studies/MachineLearning_1/lab4/main.py', wdir='/Users/petroniuss/Studies/MachineLearning_1/lab4')
+----+------------+----------------------------------+---------------+------------+--------------------+-------------+-----------------------------
|    | Category   | Item                             | Serving Size  |   Calories |   Calories from Fat |   Total Fat |   Total Fat (% Daily Value)
|----+------------+----------------------------------+---------------+------------+--------------------+-------------+-----------------------------
|  0 | Breakfast  | Egg McMuffin                     | 4.8 oz (136 g) |        300 |                120 |          13 |                          20
|  1 | Breakfast  | Egg White Delight                | 4.8 oz (135 g) |        250 |                 70 |           8 |                          12
|  2 | Breakfast  | Sausage McMuffin                 | 3.9 oz (111 g) |        370 |                200 |          23 |                          35
|  3 | Breakfast  | Sausage McMuffin with Egg        | 5.7 oz (161 g) |        450 |                250 |          28 |                          43
|  4 | Breakfast  | Sausage McMuffin with Egg Whites | 5.7 oz (161 g) |        400 |                210 |          23 |                          35
+----+------------+----------------------------------+---------------+------------+--------------------+-------------+-----------------------------

+----+----------------+------------+--------------------+-------------+-----------------------------+----------------+-----------------------------
|    |   Serving Size |   Calories |   Calories from Fat |   Total Fat |   Total Fat (% Daily Value) |   Saturated Fat |   Saturated Fat (% Daily Valu
|----+----------------+------------+--------------------+-------------+-----------------------------+----------------+-----------------------------
|  0 |       0.122581 |   0.159574 |           0.113208 |   0.110169  |                  0.10989    |           0.25 |                       0.245(
|  1 |       0.122581 |   0.132979 |           0.0660377 |   0.0677966 |                  0.0659341 |           0.15 |                       0.147(
|  2 |       0.0935484 |   0.196809 |           0.188679 |   0.194915  |                  0.192308  |           0.4  |                       0.411'
|  3 |       0.151613 |   0.239362 |           0.235849 |   0.237288  |                  0.236264  |           0.5  |                       0.509(
|  4 |       0.151613 |   0.212766 |           0.198113 |   0.194915  |                  0.192308  |           0.4  |                       0.411'
+----+----------------+------------+--------------------+-------------+-----------------------------+----------------+-----------------------------
1.79.57532507539418 1.219352500912387
```

Na tak przygotowanych danych wykonywałem dalszą część zadania.

Szacowanie jak “dobra” jest klasteryzacja

Metryka

Wybrałem indeks Daveisa-Bouldina - który bierze pod uwagę stosunki rozrzucenia danych wewnątrz klastra do separacji pomiędzy klastrami (krótko: im bardziej ściśnięte dane w klastrze i im dalej od innych klastrów tym lepiej).

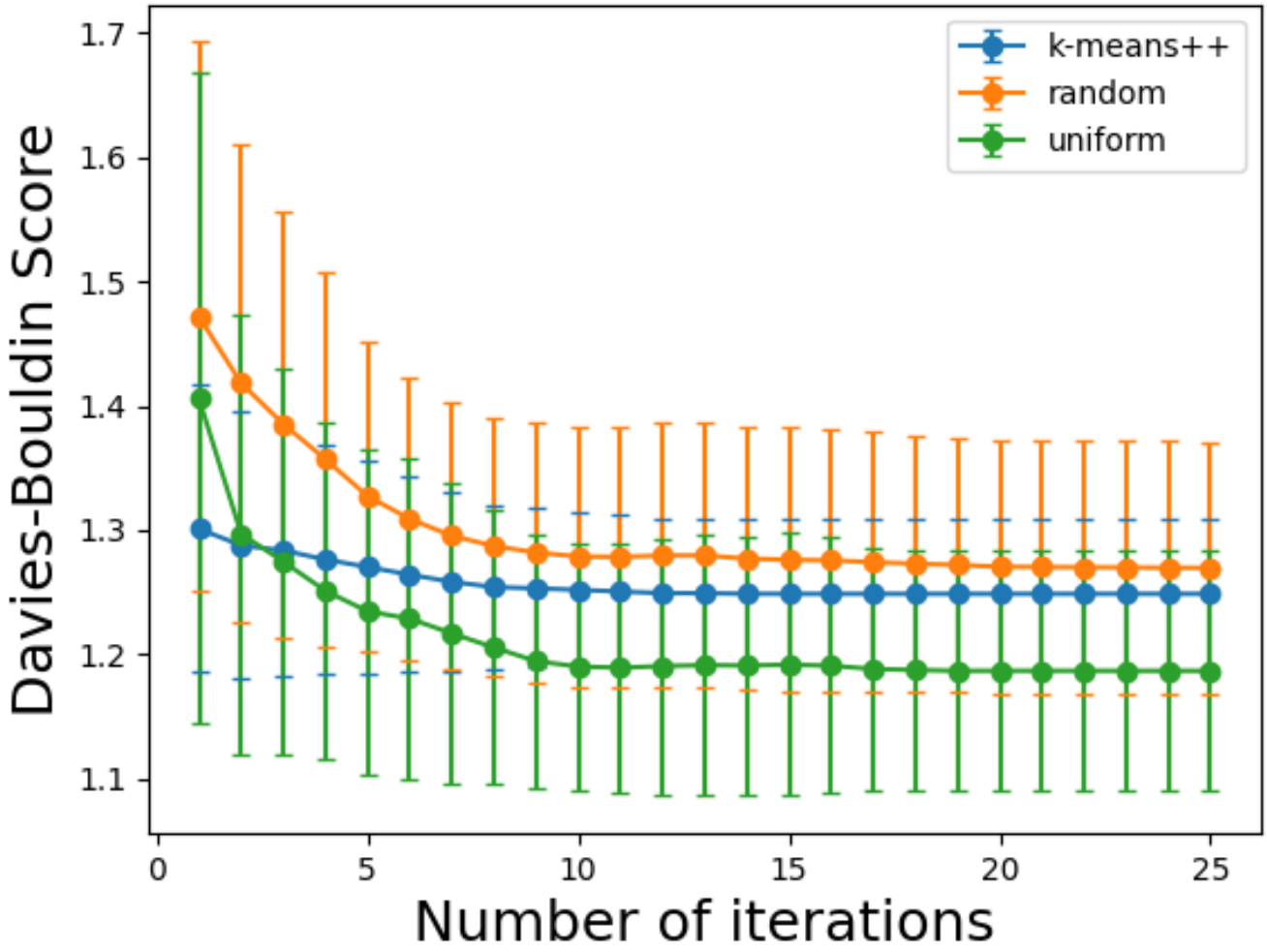
Im niższa wartość tym lepsza jest klasteryzacja.

Obserwacja działania algorytmu dla kolejnych iteracji

Można zobaczyć jak inicjalizacja ma się do zbieżności, a także do osiągane rozwiązania (jako, że łatwo wpaść w minima lokalne). Widać, że inicjalizacja k-means++ zazwyczaj zaczynała od lepszego rozwiązania (to podejście cechuje też najniższa wariancja) ale prowadziło do gorszego rozwiązania.

Najlepiej radziło sobie losowanie każdego parametru z rozkładem jednostajnym - daje to dużą wariancję ale jest to pożądane bo dajemy sobie większą szansę na znalezienie lepszego rozwiązania.

Wyniki mierzyłem dla k = 5 i powtarzałem 50-krotnie.

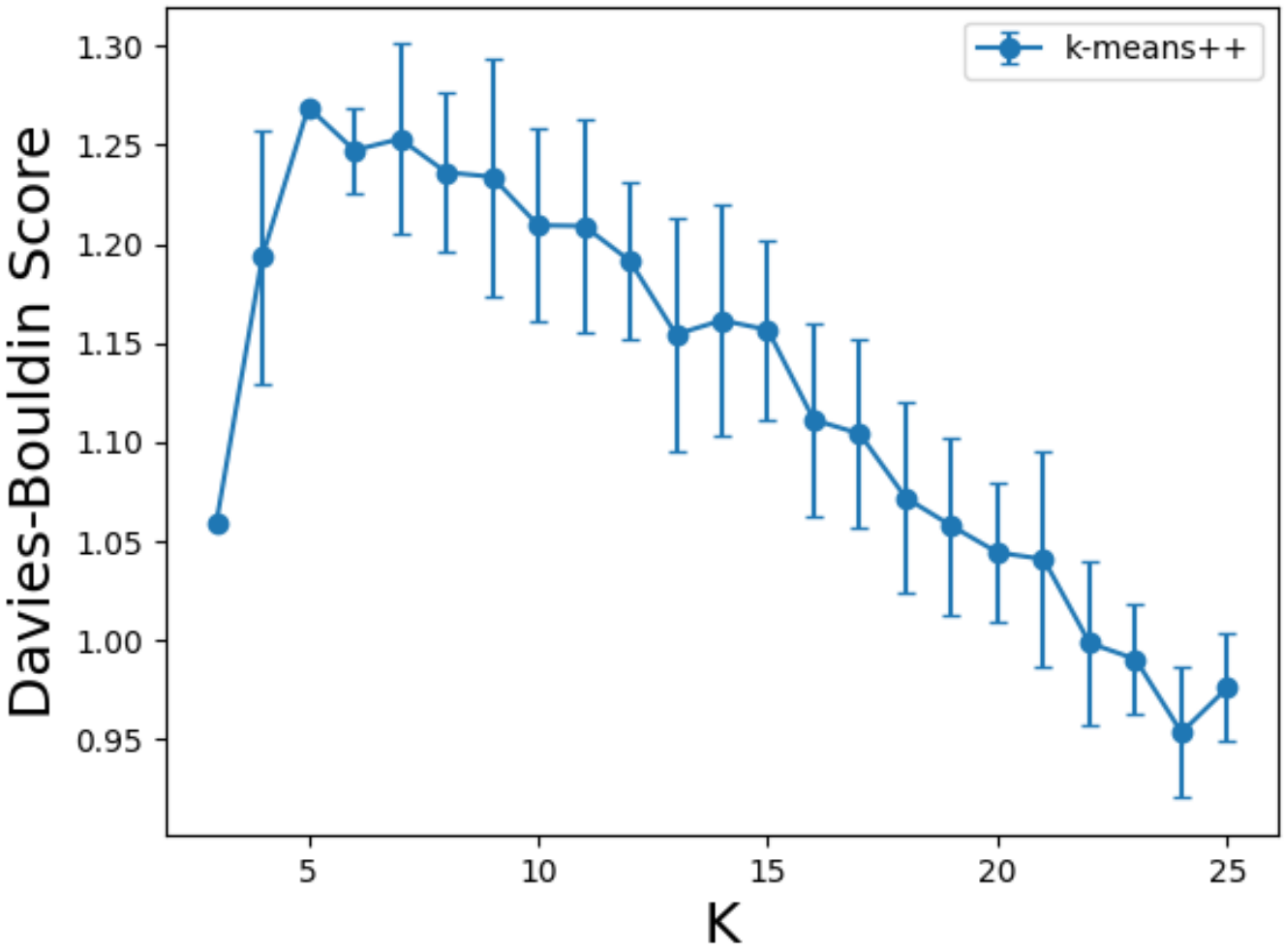


Początkowo myślałem, że ciężko to będzie zrobić bez kopiowania kawałka implementacji SKLearna ale po dłuższej analizie kodu źródłowego okazało się, że można po prostu za każdym razem podawać poprzednio uzyskane środki klastrów jeśli tylko ustawimy maksymalną ilość iteracji na 1.

```python
while True:
    kmeans.fit(X)

    labels = kmeans.labels_
    centroids = kmeans.cluster_centers_

    if np.array_equal(state, labels):
        break

    kmeans.init = centroids
    state = labels

    inertia = kmeans.inertia_
    score = sklearn.metrics.davies_bouldin_score(X, labels)
    i += 1

    ys.append(score)
    print(i, inertia, score)
```

Wybranie najlepszego K

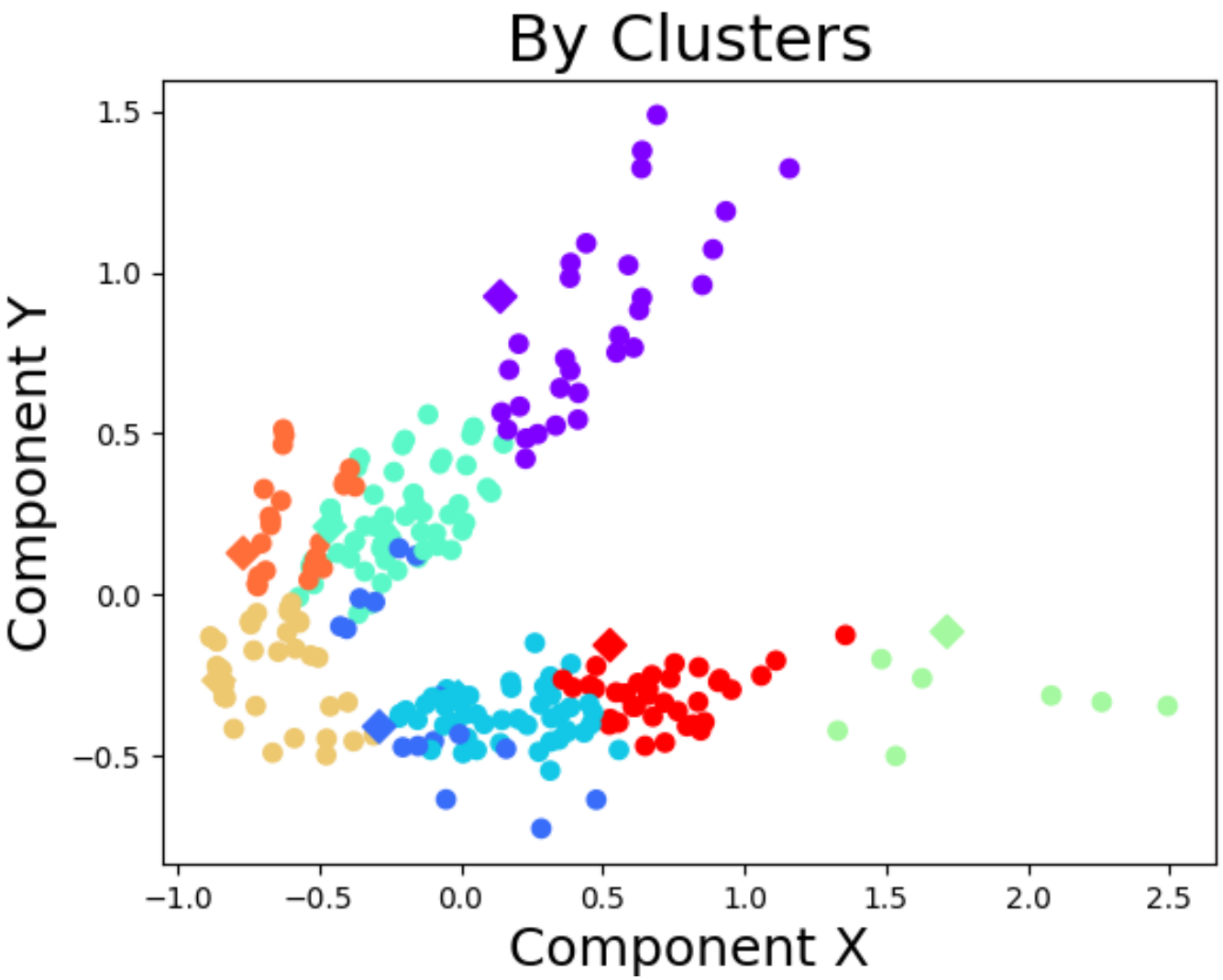
Tak wyglądały wartości metryki - wynik jest zastanawiający bo wygląda na to, że nie ma minimum i tak duża ilość klastrów nie ma zbytnio sensu - co wskazuje, że być może zbyt pośpiesznie znormalizowałem te dane do [0, 1] i być może zostawiłem jakieś kolumny, które psują wyniki. Prawdę powiedziawszy, sam nie wiem.

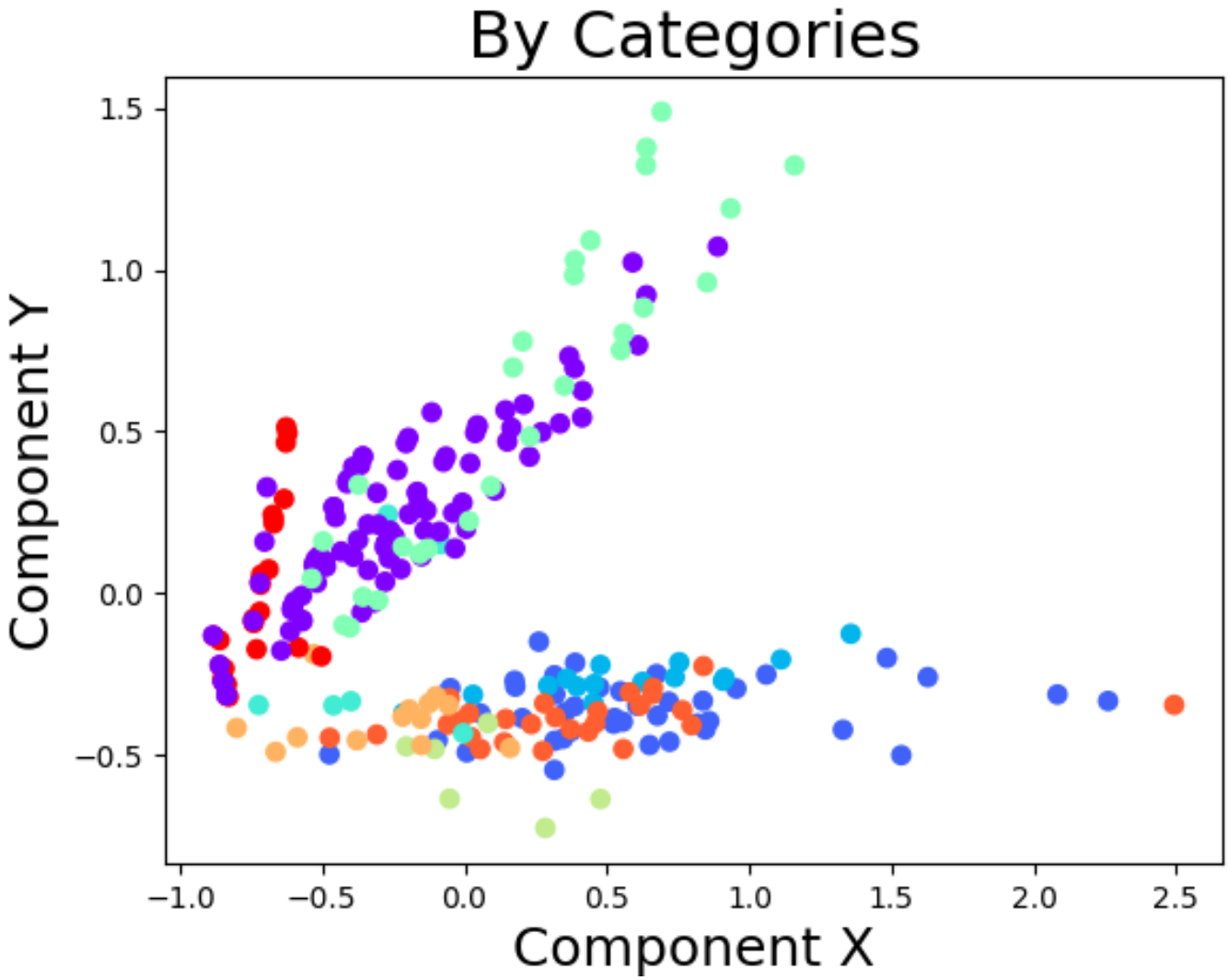


Wizualizacja rezultatów z wykorzystaniem PCA

Odniosłem wrażenie, że jakoś wyników jest wręcz odwrotna do tego, co dostałem w poprzednim punkcie - najlepsze (wizualnie) wyniki odniosłem dla k równego 8 czyli tam gdzie przypadało max według indeksu Daviesa-Bouldina.

Dane faktycznie są zgrupowane w klastry nawet po wykonaniu PCA. Widać, że grupowanie po kategoriach jest bardzo gruboziarniste i nie oddaje, tego jak dane (co do wartości) faktycznie się rozkładają.


By Clusters
Component Y
Component X
1.5
1.0
0.5
0.0
−0.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5



Myślę, że można było uzyskać znacznie lepsze wyniki wkładając więcej wysiłku w czyszczenie danych.