Курсова работа

по Размити множества и приложения

Тема: Размита клъстеризация с “fuzzy c means”

Изготвен от:

Даниел Делчев, фн:25897, Изкуствен интелект, 1 курс

Зимен семестър 2019г

Увод

Клъстеризацията е процес на класификация без учител, при който обучаващите примери не са описани с принадлежността си към дадени понятия. Клъстеризацията има за цел да открие зависимости между обучаващите примери за да открие групи от примери, такива че примерите в една група са сходни, а примерите от различни групи не са сходни. Тези групи се наричат клъстери.

При класическата (crisp) клъстеризация, всеки пример принадлежи на не повече от едно понятие.

При размитата (fuzzy) клъстеризация наричане още soft клъстеризация един пример от обучаващите може да принадлежи към повече от един клъстери със различна степен на принадлежност, като сумата от степените на принадлежност към всеки от клъстерите за даден пример е единица.

Един от най-популярните алгоритми за размита клъстеризация е fuzzy c means, който е модификация на класическия c-means.

Целта на текущия проект е да представи една имплементация на алгоритъма fuzzy-c-means. При имплементацията се е прилагало възможно най-много въздържание от използване на готови библиотеки и имплементации. (Някои езици за програмиране и среди като Python/R/Woflram/Matlab предоставят готова имплементация).

Направени са и някой модификации и допълнения към класическия fuzzy-c-means, които са описани в последствие. Имплементацията работи с примери с числови атрибути, които са представени в .csv файл , с конкретен формат (наличие на заглавия на колоните и първа колона id с идентификатори от 1 до N за примерите).

Имплементираният fuzzy-c-means е прилаган върху следните 3 множества от данни:

1) Данни за измервания на характеристики на червени вина

https://archive.ics.uci.edu/ml/datasets/Wireless+Indoor+Localization

1599 примера с 12 атрибута.

2) Данни за измервания на вид морски охлюви

https://archive.ics.uci.edu/ml/datasets/Abalone

4177 примера с 9 атрибута (от които един категориен е изключен)

3) Измервания на сила на 7 wifi сигнала в затворено помещение

https://archive.ics.uci.edu/ml/datasets/Wireless+Indoor+Localization

2000 примера с 8 атрибута (от които един категориен е изключен)

Имплементацията е на езиците C++ и Python. За визуализация на многомерните данни в двумерни графики са използвани PCA и LDA, от които първия е имплементиран в предоставената реализация (с изключение на пресмятането на собствените вектори и собствените стойности на ковариационната матрица на атрибутите), а вторията LDA е използван наготово от предоставанета имплементация от sklearn на Python. Проекта се компилира с подходящ компилатор , като са предоставени makefile-ове . При разработката е използвана GCC.

Освен условията на задачата от moodle е предоставена още и възможността за съставяне на матрица представяща релация на подобие между обучаващите примери, както и метрика за определяне на степента на размитост на размитата клъстеризация.

Теоретична постановка и използван алгоритъм.

Fuzzy-c-means се нуждае от параметър K, който указва броя клъстери, които да търси.

Ако предположим, че имаме N обучаващи примери и входен параметър от K клъстера, то тогава алгоритъма ползва K на брой функции на принадлежност (по една за всеки клас).

Те не се представят в явен вид ,а вместо това се използва матрица U с размери KxN ,в която в клетка с индекси i и k стои степента на принадлежност на примера с индекс k към клъстера с индекс i .

Тоест матрицата U =[μ_{ik}] с размери KxN, където μ_{ik} е клетката в ред i и колона k, представя изцяло размитата клъстеризация.

Матрицата отговаря на инвариантите , че всяка нейна колона се сумира до единица и сумата не всеки неин ред е положителна и по-малка от N (броя примери).

Освен това в процеса на изчисления се използва и множсество от центрове на клъстерите (те представляват линейни комбинации на примерите от съответния клъстер, тоест един център на клъстер (центроид) е претеглен по съответните стойности на принадлежносрт на примерите, медицентър в n-мерно простстранство).

По начало се инициализира на някъкъв случаен или евристичен принцип или матрицата U, или множеството от центроиди, след което итеративно се обновяват матрицата и центроидите.

При този процес , на всяка итерации се минимизира следната сума:

$$J_m(\underset{\sim}{\mathrm{U}}, \mathbf{v}) = \sum_{k=1}^{n}\sum_{i=1}^{c}(\mu_{ik})^{m'}(d_{ik})^2,$$

,

Където U е матрицата със стойностите на принадлежност, v е множеството от центроиди, m’ е параметър, подаван към алгоритъма, а d_{ik} е разстоянието (евклидово) между примера с индекс k и центроида на клъстера с индекс i .

Тоест сумата минимизира произведението между квадрата от разстоянията на примерите от даден клас до центроида на съответния клас, и степента на принадлежност към съответните класове на съответните примери.

Тъй като намирането на глобален минимум на така описаната оптимизационна функция е изключително трудуемка ,задача , наместо това се използва итеративен процес на обновяване на матрицата U и множеството от центроиди v, което не гарантира намирането на глобален минимум. Обикновено се използва метрика-критерии за край на процеса от итерации. Примери за такива критерии са - праг на разликата на две стойности на оптимизационната функция за две последователни матрици U в процеса на итерация, който трябва да се надскочи за да продължат итерациите, или пък минимална абсолютна разлика между две съответни стойности на клетки в две последователни матрици в процеса на итериране, който трябва да се надскочи за да продължат итерациите.

m’ , който още бележим като m , е входен параметър по-голям от единица, за който няма приет критерии за избор. Обикновено се избира число между 1 и 3 , като при 1 клъстеризацията реално е crisp. При по-високи стойности за m клъстеризацията бива по-размита и схождането към критерии за спиране на итерациите е по-бързо.

Ако v_i е центроида на клъстера с индекс i, той се обновява на всяка итерации по следният начин:

$\mathrm{v}_{ij} = \frac{\sum_{k=1}^{n}\mu_{ik}^{m}\cdot x_{kj}}{\sum_{k=1}^{n}\mu_{ik}^{m}}$, където x_{kj} е j-тата компонента на примера с индекс k,

А v_{ij} е j-тата компонента на центроида на класа i .

Обновяването на матрицата U , става по следният начин - всяка клетка μ_{ik} се обновява спрямо новите центроиди по следния начин:

1) За всеки пример с индекс к, нека I_k е множеството от индекси на клъстери, чиито центроиди за на разстояние 0 от този примерът с индекс к.
2) За всеки пример с индекс к
   ако I_k е празното множество, то тогава обнови μ_{ik} така

$$\mu_{ik} = \frac{1}{\sum_{j=1}^{C}(\frac{d_{ik}}{d_{jk}})^{2/(m-1)}}$$

Ако пък I_k не е празното множество, тогава

ако і принадлежи на I_k , тогава $\mu_{ik} = \frac{1}{|I_k|}$, където $|I_k|$ е кардиналността на I_k

ако пък і не принадлежи на I_k тогава $\mu_{ik} = 0$

С други думи ако даден пример не съвпада с никой центроид, то неговите стойности на принадлежност се преизчисляват по горно описаната формула, ако пък пример съвпада с някои центроиди, то той вече ще принадлежи поравно към техните класове и няма да принадлежи към други класове .

Итеративно се обновяват матрицата и центроидите, докато не се удовлетвори даден критерии за прекратяване. Поначало или центроидите или матрицата се инициализират случайно или посредством евристика (като не се нарушават описаните инварианти за матрицата).

Това е общата постановка на алгоритъма fuzzy-c-means.

Предоставената имплементация е в съглашение с нея, като за начална инициализация се инициализират центроидите чрез евристика, избираща за центроиди обучаващи примери , с вероятност право пропорционална на квадрата от евклидовото разстояние на дадения пример до най-близкия вече избран центроид. Първия от тези центроиди се избира на случаен принцип. В екстремални случай на по-малко уникални обучаващи примери от брой класове, отново се избират центроиди от обучаващите примери на случаен принцип.

След инициализацията на центроидите итеративно се обновяват матрицата и центроидите , докато е вярно , минимална абсолютна разлика между две съответни стойности на клетки в две последователни матрици в процеса на итериране, е по-висока от предварително дефинирана стойност-праг.

Също така, предварително данните са нормирани в z стойности на нормално разпределение покомпонентно, за да не се вземат с по-голяма тежест стойности на компоненти (характеристики) на примерите с големи стойности.

След като е изчислена крайната матрица U. Се изчислява мярка – коефициент на размитост на клъстеризацията

$$F_c(\underset{\sim}{U}) = \frac{\mathrm{tr}(\underset{\sim}{U} * \underset{\sim}{U}^{\mathrm{T}})}{n},$$

, където tr е означение за сумата от главния диагонал на квадратна матрица.

За мярката имаме че $1/c \leq F_c(\underset{\sim}{U}) \leq 1.$ като при стойност на мярката единица, имаме crisp клъстеризация, а при 1/c имаме максимално размита клъстеризация.

Освен това, след намиране на крайната матрица U, намиране още и релация на подобие между примерите, показваща до колко един пример е сходен с друг. Матрицата R представяща релацията получаваме като:

$$\underset{\sim}{R} = \left(\underset{\sim}{U}^{\mathrm{T}} \left(\sum \wedge\right) \underset{\sim}{U}\right) = [r_{kj}],$$

$$r_{kj} = \sum_{i=1}^{c} \mu_{ik} \wedge \mu_{ij},$$

, където $(\sum \wedge)$ означава сума от минимуми.

След всички тези изчисления, предоставената програмна реализация на алгоритъма записва следните файлове:

Файл с резултатите от размитата класификация – csv файл в който има записи състоящи се от идентификатор на пример и степента му на принадлежност към всеки клас.

Файл със crisp резултатите от размитата класификация – csv файл в който има записи състоящи се от идентификатор на пример , клъстер ,към който има най-голяма степен на принадлежност и индекс на центроид, до който е най-близо. (Този файл ни е нужен за графиките.)

Файл със стандартизираните в нормално разпределение изходни данни.

Файл със крайните центроиди стандартизирани в нормално разпределение.

Файл със крайните центроиди (без стандартизиране).

Файл в който е записана мярката за размитост.

Файл с матрицата на подобие между примерите. (при поискване се конвертира до xlsx таблица от програмата за по-лесно преглеждане).

За да се представят графики визуализиращи класификацията , е нужно да намалим броя на измеренията до 3 или в случая до 2.

Има различни стратегии за това, и няма такава при която да не се губи информация.

В случая са използвани 2 стратегии за това за съставяне на 2 вида графики – едната стратегия е PCA (principal component analysis) като методите за това са имплементирани в текущата реализация (с изключение на пресмятането на собствените вектори и собствените стойности на ковариационната матрица на атрибутите, за което е използвани библиотеката JAMA, за да се пресметнат бързо и многонишково).

Най-накратко, стратегията на PCA е да състави нов базис на пространството, и примерите се привеждат в него, като дисперсията на стойностите на първия компонент на примерите е запазена възможно най-много, след това на втория компонент е запазена възможно най-много , но по-малко от първия и т.н.

След прилагане на PCA избираме първите 2 компонента и представяме това пространство на графиките.

За целта записваме 2 файла със стандартизираните входни данни и стандартизираните центроиди в PCA базис.

Другата приложена стратегия е LDA. Linear Discriminant Analysis. За разлика от PCA , LDA не е имплементиран в предоставената програмна реализация, а се използва наготово от sklearn . Най-накратко, LDA намаля пространството, така че различието между класовете да се запази най-много. При него се използва получената клъстеризация (затвърдената crisp клъстеризация) за визуализацията, така, че тя е пристрастна. PCA е безпристрастен към класовете.

Проекта може да се компилира посредством приложените makefile–ове.

В директорията на проекта “make –f makefile_linux” за компилация с GCC .

Или „make –f makefile_windows” за компилация на windows с друг подходящ компилатор (не е тествано).

Така се съставя изпълним файл FKM

Който може да се изпълнява от терминала с подходящи аргументи.

```
Usage: FKM <directory> <inputFilename> <prefix for filenamse> <m value> <epsilon singificance
difference> <max Itererations> <K> <convert> <metric>
<directory> (string)- directory in which input file is located and output files will be
generated
<inputFilename.csv> (string)- name of the input .csv file (in the dir folder)
<prefix> (string)- string to add to all created files in the dir folder
<m-value>  (double)- m parameter for fuzzy K means
<epsilon>  (double)- if there is no difference bigger than epsilon in a mu value of the current
and last matrix, stop
<max Iteratons> (integer)- max iterations, before stopping (regardless of epsilon)
<convert> (t/T/TRUE/true/f/F/FALSE/false)- whether to convert the similarity matrix to XLSX
<K> (integer)- number of clusters, if K=0, then <metric> is used for unsupervised choice of best
K
[metric] - !supplied when K is 0 for unsupervised choice of K! (integer)- the metric to use to
use for choice of K, (used only when <K> = 0)
metrix is one of {1,2,3,4,5,6,7,8} 1=PC+ 2=PE- 3=MPC+ 4=FSI- 5=CHI+ 6=XBI- 7=CSI+ 8=KI-
```

(при извикване с неправилни аргументи се показва горния help)

Примерно

```
./FKM data/wifi wifi.csv myPrefix 2.2 0.05 40 true 4
```

Ще стартира програмата за файла wifi.csv в директорията data/wifi . Генерираните файлове и графики ще имат префикс myPrefix в името. m параметърът ще е 2.2, прага за абсолютна разлика на съответни клетки в две последователни U матрици ще е 0.05, като ако той не се премине до 40 итерации, итерациите ще престанат. Също така ще се преобразува csv файла със релацията на подобие между примери то excel таблица (това преобразуване става сравнително бавно – няколко минути). Броят класове K, които ще се търсят ще е 4.

```
./FKM data/wifi wifi.csv myPrefix 2.2 0.05 40 true 0 5
```

Пък ще изпълни действия подобни на описаните горе, с тази разлика че стойността на K ще се определи автоматично според метриката с номер 5 - `5=CHI+` . Накрая след намиране на най-доброто K според метриката, ще се изпълни алгоритъма за съответното K, и ще се съставят файловете и графиките. „+” срещу метриката означава , че по-голяма стойност е по-добра , а „-„ , че по-малка е по-добра.

Определянето на K става , като за K от 2 до 16 се извършват по 5 пускания на алгоритъма. След всяко пускане на алгоритъма се пресмята съответната метрика. Накрая след 5те пускания тя се осреднява. K-то за което метриката е най-добра (максимална/минимална) се избира като най-добро.

Метриките, от които може да се избира са следните:

1)

The partition coefficient (PC+) evaluates the compactness by using the averaged strength of belongingness of data, and is defined as:

$$PC(K) = \frac{1}{N}\sum_{i=1}^{K}\sum_{j=1}^{N}\mu_{ij}^2$$

2)

The partition entropy (PE-) is formed based upon the logarithmic form of PC , and is defined as:

$$PE(K) = -\frac{1}{N}\sum_{i=1}^{K}\sum_{j=1}^{N}\mu_{ij}\log_2(u_{ij})$$

3)

The modification of PC (MPC+) is designed to reduce the monotonic tendency of PC and PE. The index is defined as:

$$MPC(K) = 1 - \frac{K}{K-1}(1 - PC)$$

4)

he Fukuyama and Sugeno Index (FSI-) is designed to measure the discrepancy between fuzzy compactness and fuzzy separation, i.e.,:

$$FSI(K) = \sum_{i=1}^{K}\sum_{j=1}^{N} u_{ij}^{m} \,||\, x_j - z_i \,||^2 - \sum_{i=1}^{K}\sum_{j=1}^{N} u_{ij}^{m} \,||\, z_i - \bar{z} \,||^2$$

Със z се бележат центроидите, а z с черта отгоре се бележи медицентърът на всички примери.

5)

The Calinski-Harabasz Index (CHI+) is a ratio-type index in which compactness is measured by the distance (W_K) between each within-cluster point to its centroid, and separation is based on the distance (B_K) between each centroid to the global centroid (z), i.e.

$$CHI(K) = \frac{B_K}{K-1} \Big/ \frac{W_K}{N-K},$$

където

$$B_K = \sum_{i=1}^{K} N_i \,||\, z_i - \bar{z} \,||^2, \; W_K = \sum_{i=1}^{K} \sum_{x_j \in C_i} ||\, x_j - z_i \,||^2$$

6)

The Xie and Beni Index (XBI-) is also a ratio-type index, which measures the average within-cluster fuzzy compactness against the minimum between-cluster separation, i.e.,:

$$XBI(K) = \frac{\sum_{i=1}^{K}\sum_{j=1}^{N} \mu_{ij}^{2} \,||\, x_j - z_i \,||^2}{N \cdot \min_{i \neq k}\{ \,||\, z_i - z_k \,||^2\}}$$

7) The SC Index (SCI+) measures the fuzzy compactness/separation ratio of clustering by using the difference between two functions, SC_1 and SC_2, i.e.,:

$$SCI(K) = SC_1(K) - SC_2(K)$$

where SC_1 evaluates the compactness/separation ratio by considering the grades of membership and the original data: the larger the SC_1, the better the clustering:

$$SC_1(K) = \frac{(\frac{1}{K}\sum_{i=1}^{K} ||z_i - \bar{z}||)}{\sum_{i=1}^{K}(\sum_{j=1}^{N} \mu_{ij}^m ||x_j - z_i||^2 / \sum_{j=1}^{N} \mu_{ij})},$$

while SC_2 measures the ratio by using the grades of membership only: the smaller the SC_2, the better the clustering:

$$SC_2(K) = \frac{\sum_{i=1}^{K-1}\sum_{k=i+1}^{K}(\sum_{j=1}^{N}(\min(\mu_{ij}, \mu_{kj})^2)/n_{jk})}{(\sum_{j=1}^{N}\max_{1\leqslant i\leqslant K}\mu_{ij}^2)/(\sum_{j=1}^{N}\max_{1\leqslant i\leqslant K}\mu_{ij})}$$

where $n_{jk} = \sum_{j=1}^{N}\min(\mu_{ij}, \mu_{kj})$.

8)

The Kwon Index (KI-) aims to overcome the shortcoming of XBI that decreases monotonically when the cluster number approaches the actual cluster number of data. Here, a penalty function was introduced to the numerator of XBI, i.e.,:

$$KI(K) = \frac{\sum_{i=1}^{K}\sum_{j=1}^{N}\mu_{ij}^2 ||x_j - z_i||^2 + \frac{1}{K}\sum_{i=1}^{K}||z_i - \bar{z}||^2}{\min_{i\neq k}\{||z_i - z_k||^2\}}$$

Резултати над множествата от данни

За Fuzzy-C-means , освен стойността на параметъра K, определянето на параметъра m е също от значение. Тъй като комбинираното търсене на K и m би било прекалено тежко начинание, за всяко от 3те множества от данни определяме подходящо m „на око” спрямо коефициента на размита клъстеризация за някоя конкретна стойност на K. След това чрез всяка от 8те метриките избираме K , и представяме крайните резултати във вид на графики.

1) Множеството от данни “redwine.csv”

При к = 3

За m = 1.5 получаваме:

Fuzzy Partition Coefficient = 0.530711

1\3 <= 0.530711 <= 1

1 means crisp clustering, 1\3 means complete ambiguity.

За m=2 получаваме:

Fuzzy Partition Coefficient = 0.345091

1\3 <= 0.345091 <= 1

1 means crisp clustering, 1\3 means complete ambiguity

За m = 1.65 получаваме:

Fuzzy Partition Coefficient = 0.433095

1\3 <= 0.433095 <= 1

1 means crisp clustering, 1\3 means complete ambiguity

За m =1.4 получаваме:

Fuzzy Partition Coefficient = 0.61534

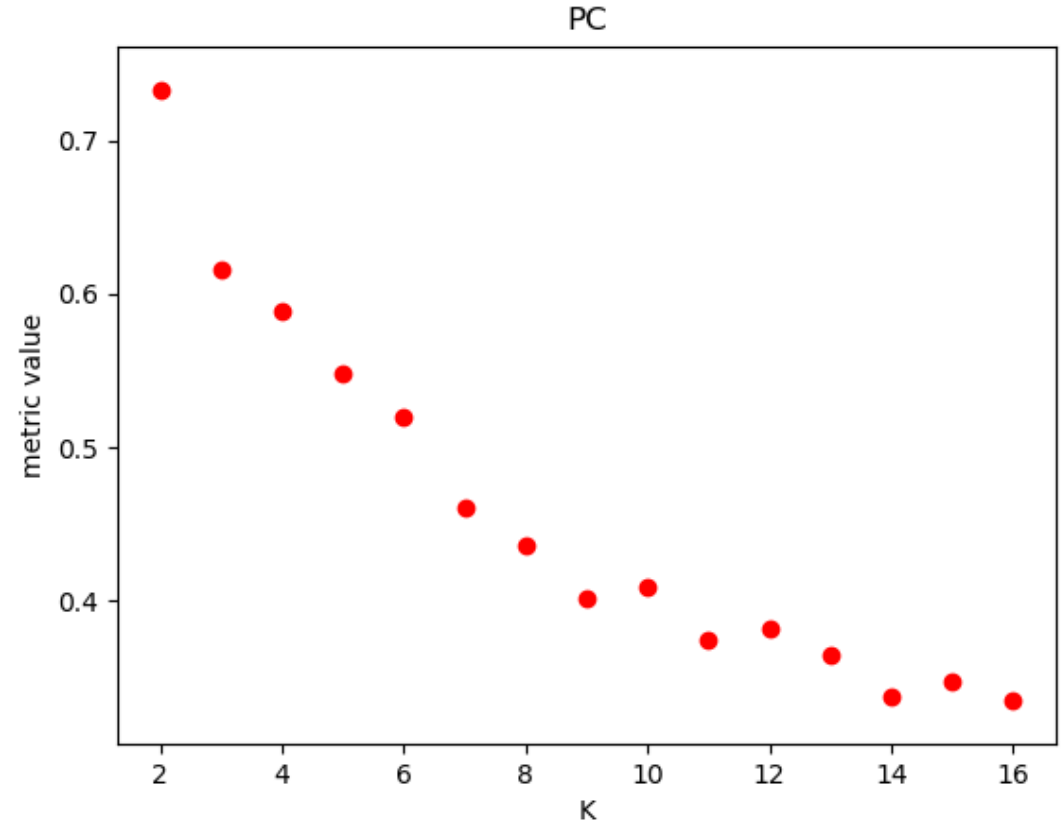
1\3 <= 0.61534 <= 1

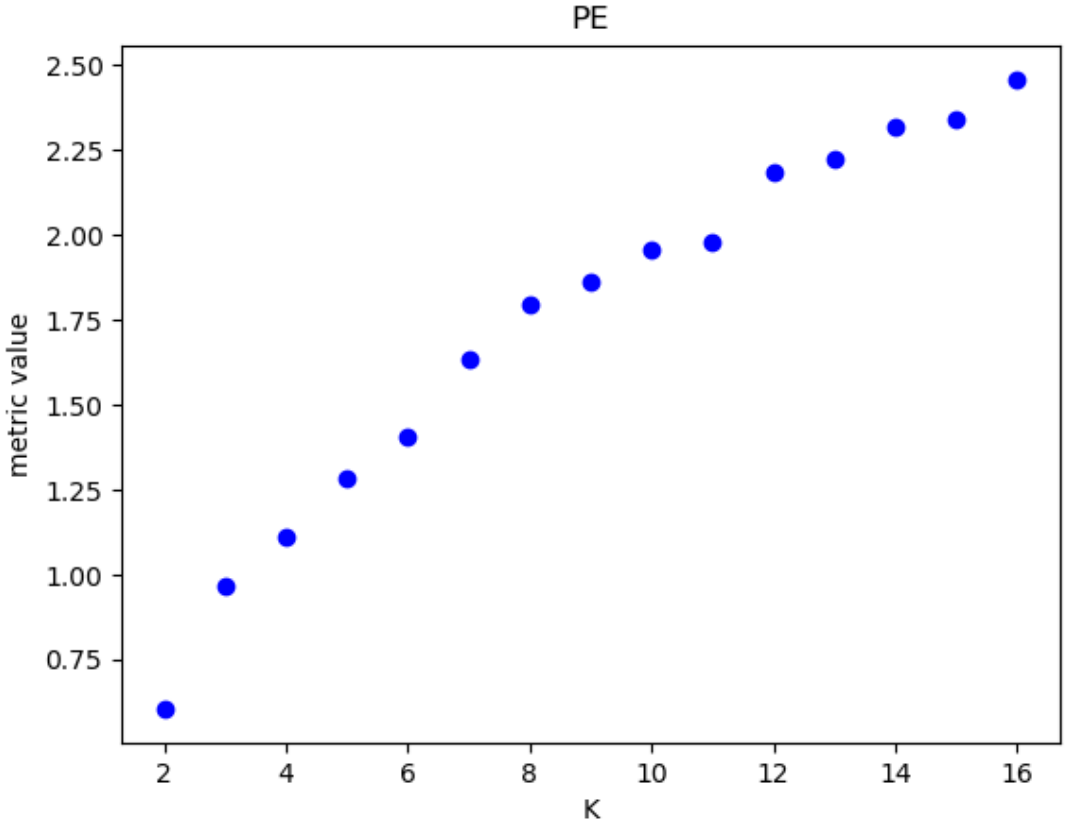
1 means crisp clustering, 1\3 means complete ambiguity

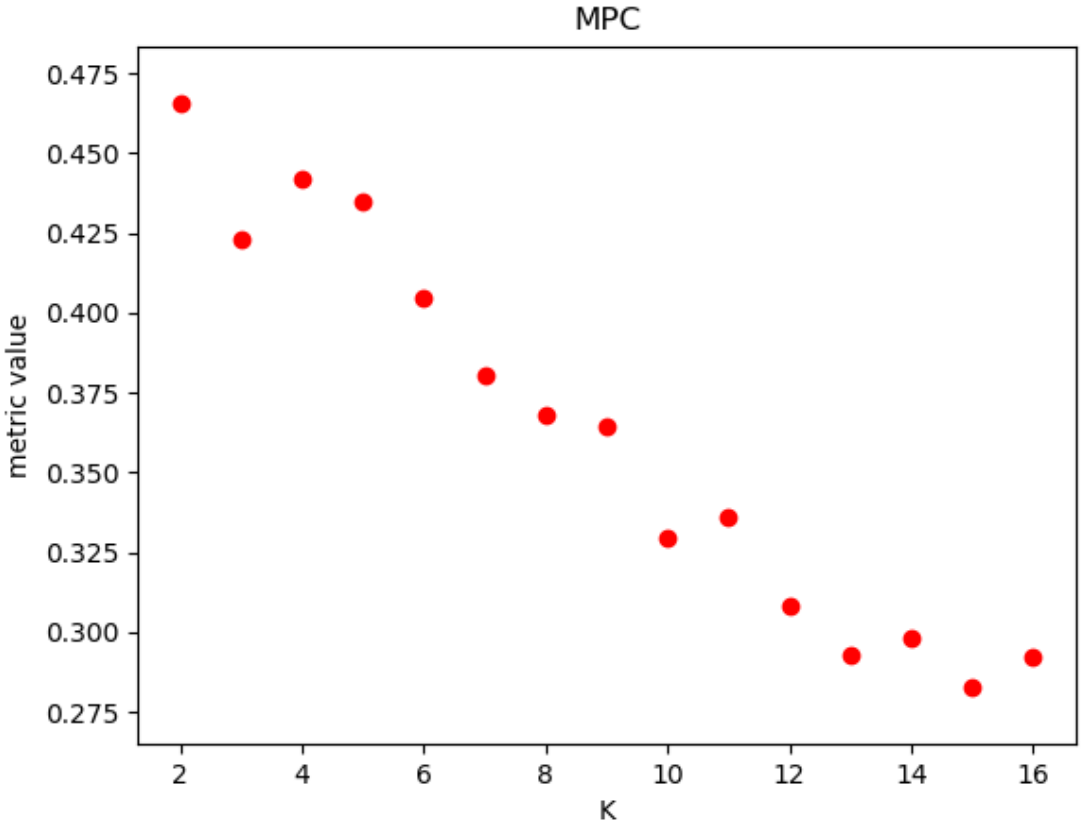
Спираме се на m=1.4 и прилагаме алгоритъма с автоматично определяне на К за 8те метрики със праг 0.001 или 100 итерации

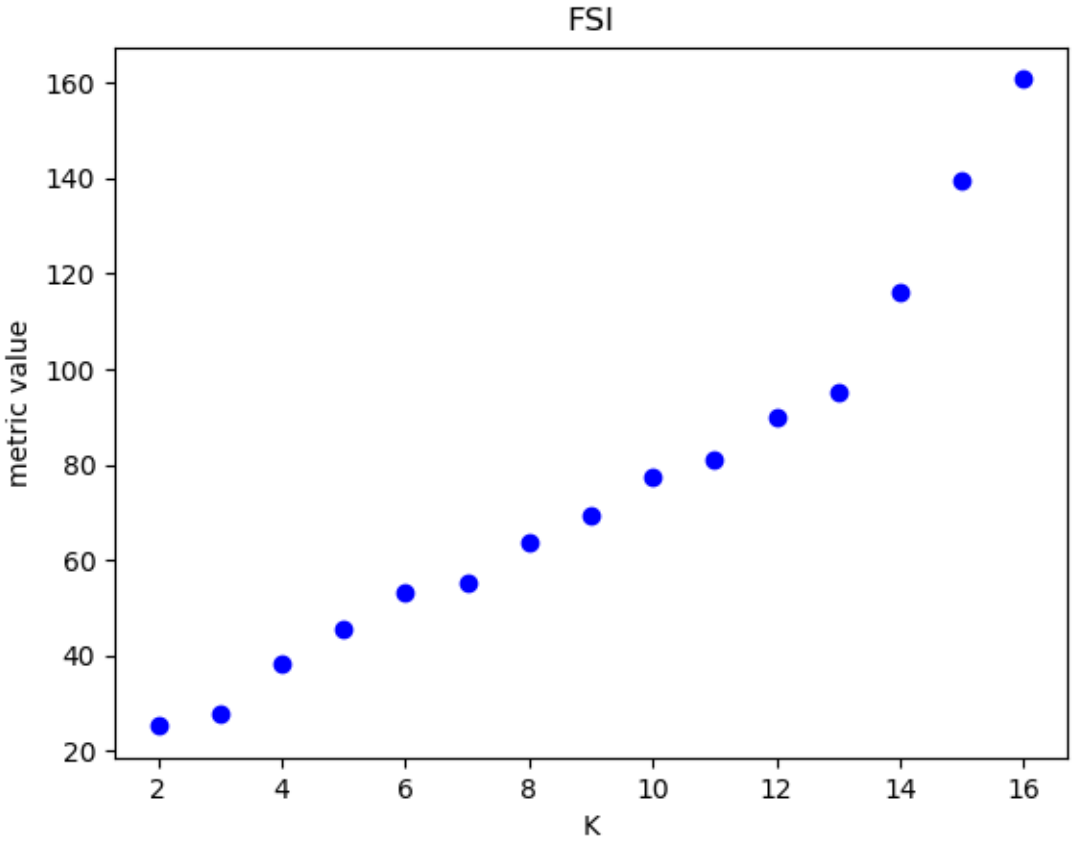
1=PC+ 2=PE- 3=MPC+ 4=FSI- 5=CHI+ 6=XBI- 7=CSI+ 8=KI-

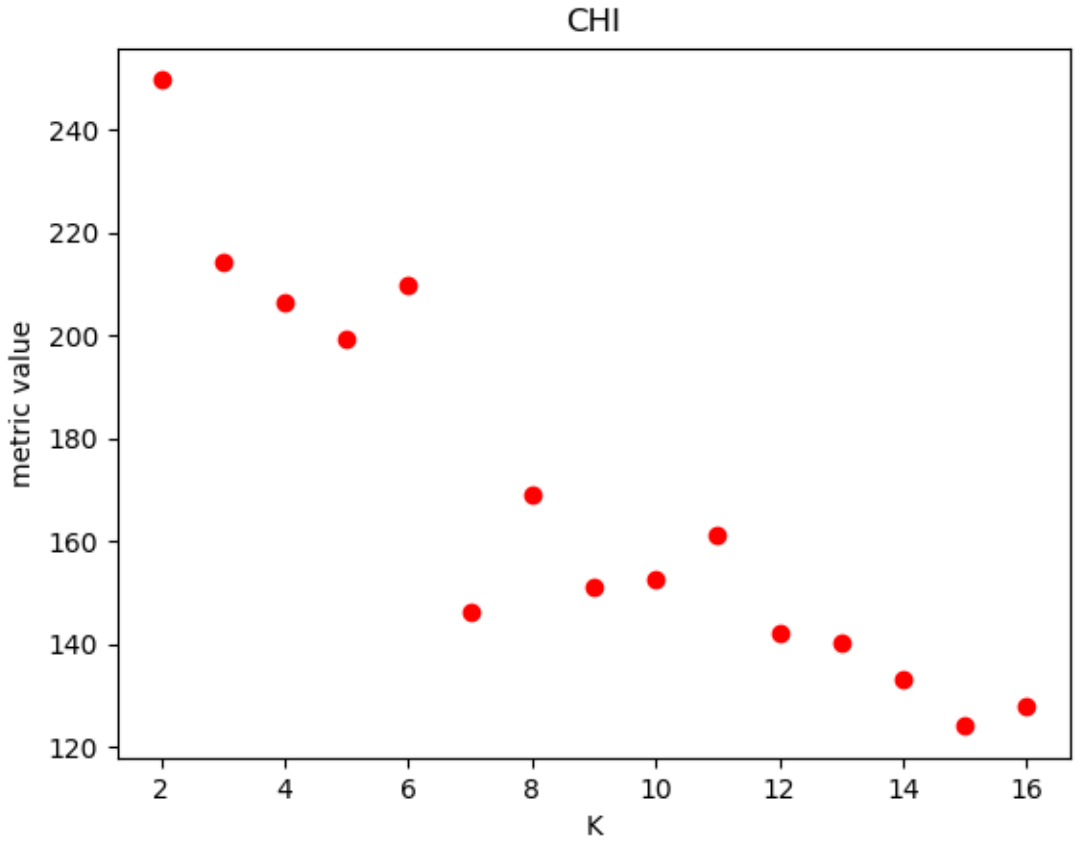
Графиките за метриките , за които търсим максимум са в червено, а тези за които търсим минимум са в синьо.

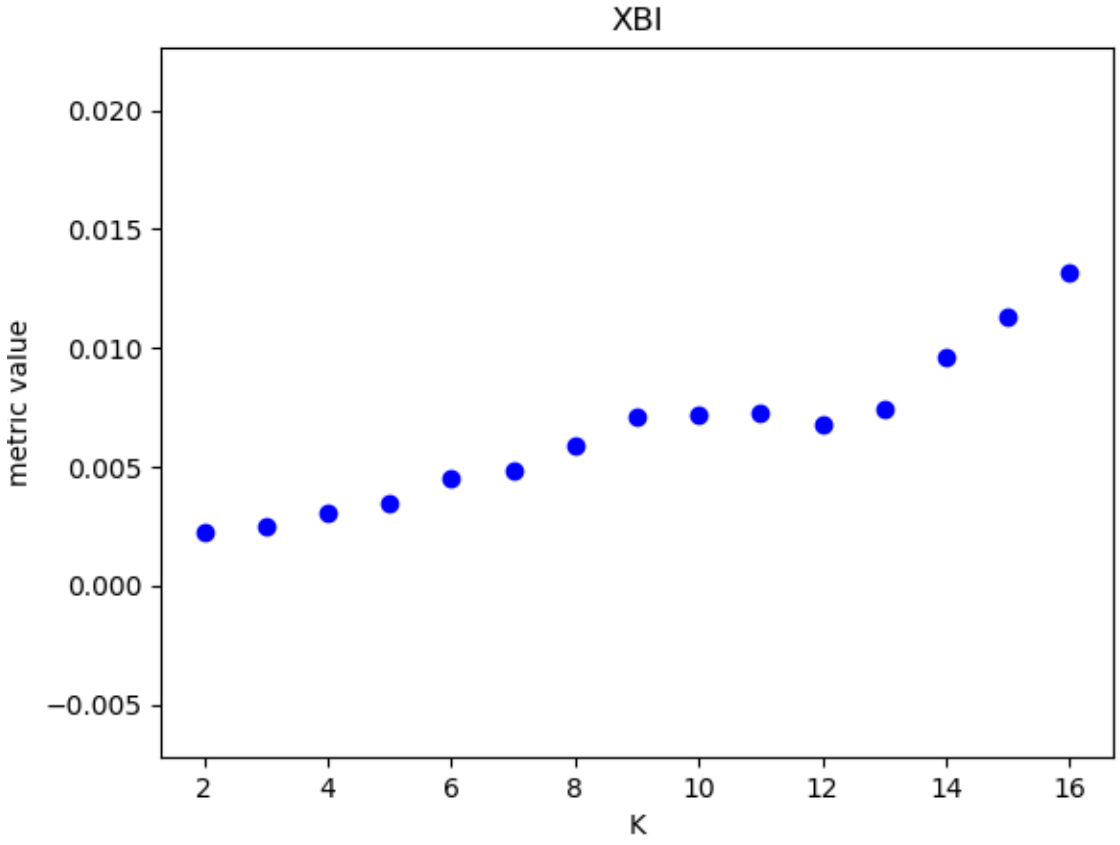

PC
metric value
0.7
0.6
0.5
0.4
2
4
6
8
10
12
14
16
K

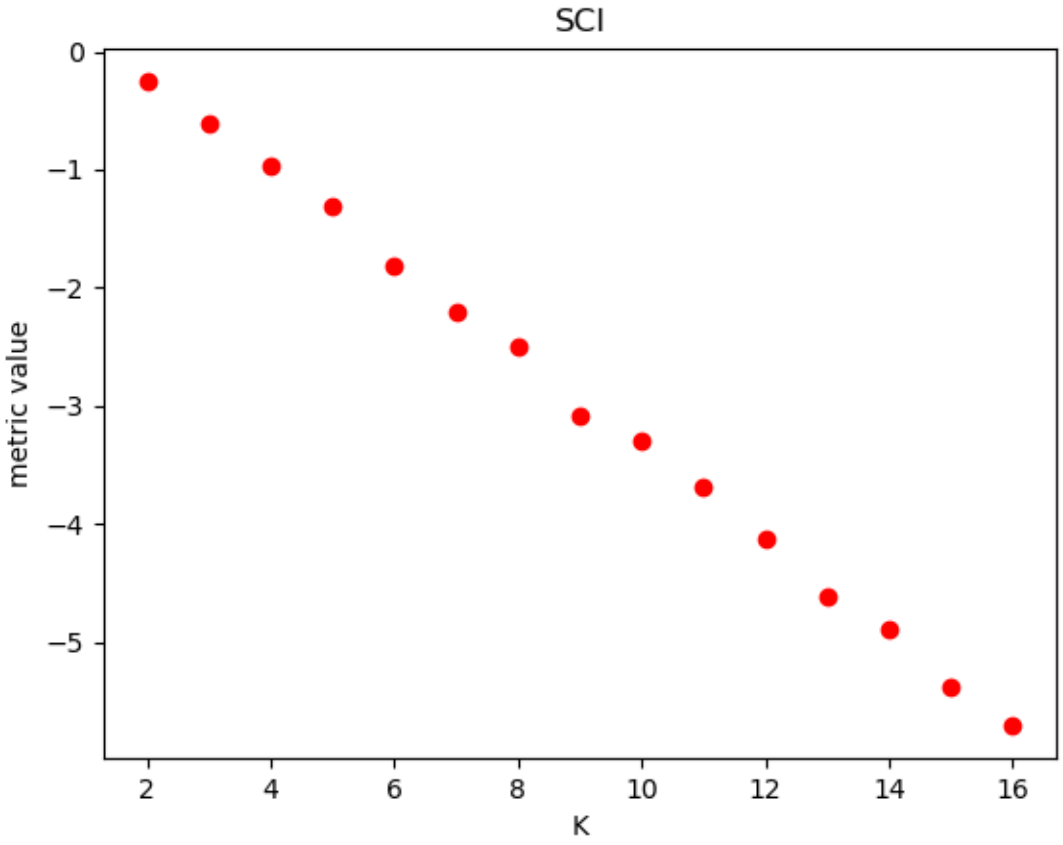

PE
metric value
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.75
2
4
6
8
10
12
14
16
K

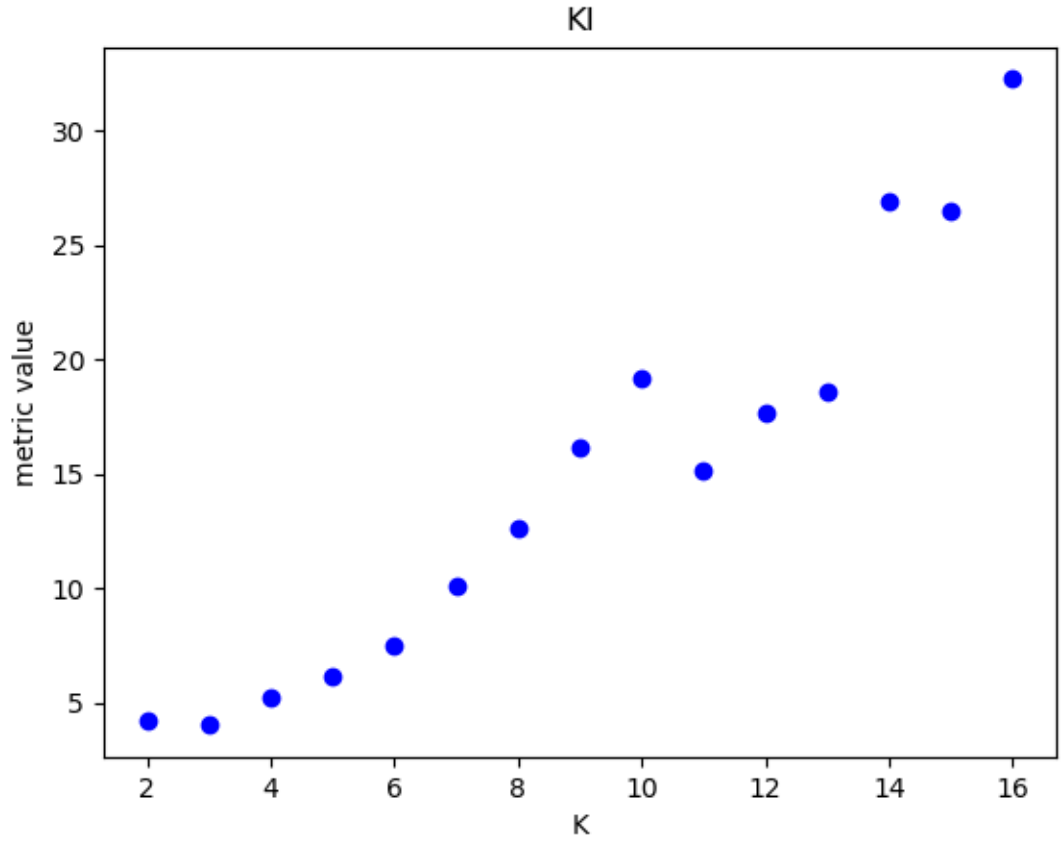

MPC
metric value
0.475
0.450
0.425
0.400
0.375
0.350
0.325
0.300
0.275
2
4
6
8
10
12
14
16
K


FSI
metric value
160
140
120
100
80
60
40
20
2
4
6
8
10
12
14
16
K


CHI
metric value
240
220
200
180
160
140
120
2
4
6
8
10
12
14
16
K

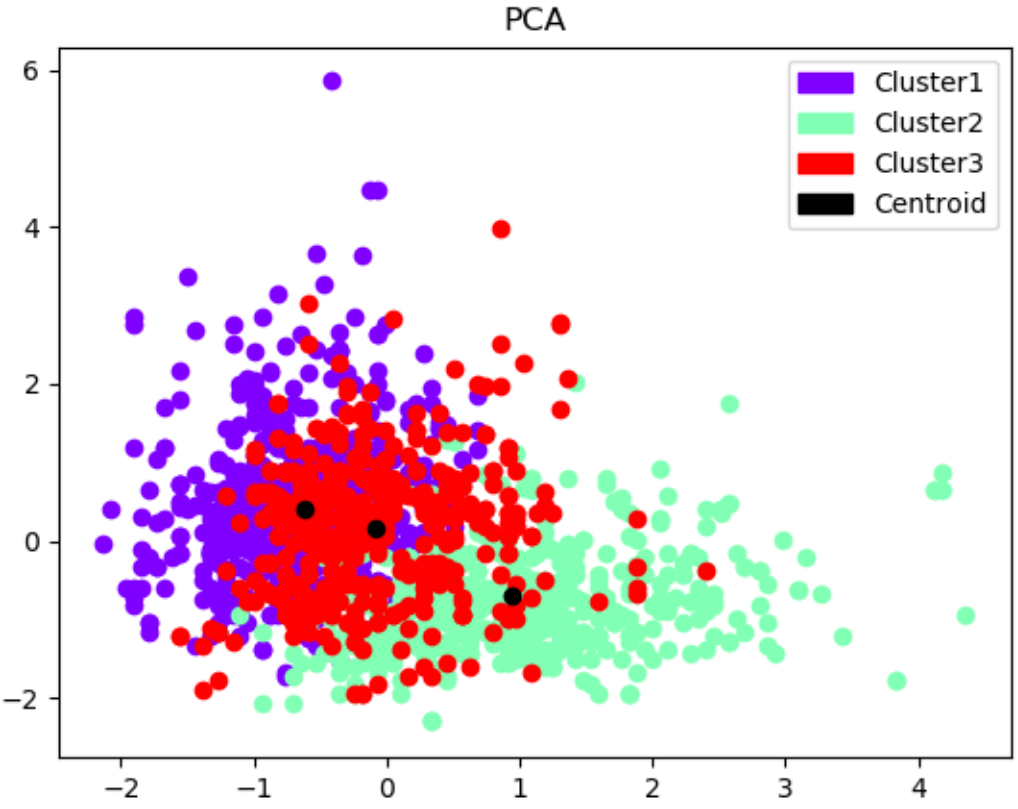

XBI
metric value
0.020
0.015
0.010
0.005
0.000
−0.005
2
4
6
8
10
12
14
16
K


SCI
metric value
K
0
−1
−2
−3
−4
−5
2
4
6
8
10
12
14
16


KI
metric value
K
30
25
20
15
10
5
2
4
6
8
10
12
14
16

Освен метриката KI, която намира 3 класа за оптимални, всички други определят 2 за оптимални.

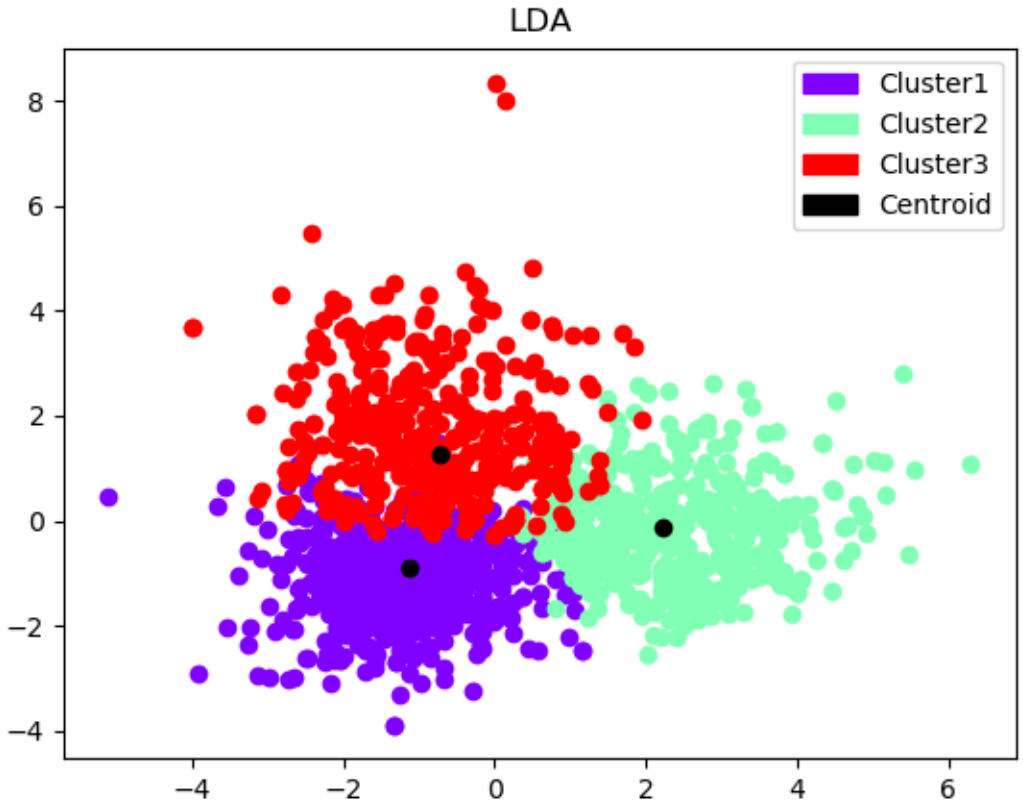
При K=3 за същите параметри имаме:



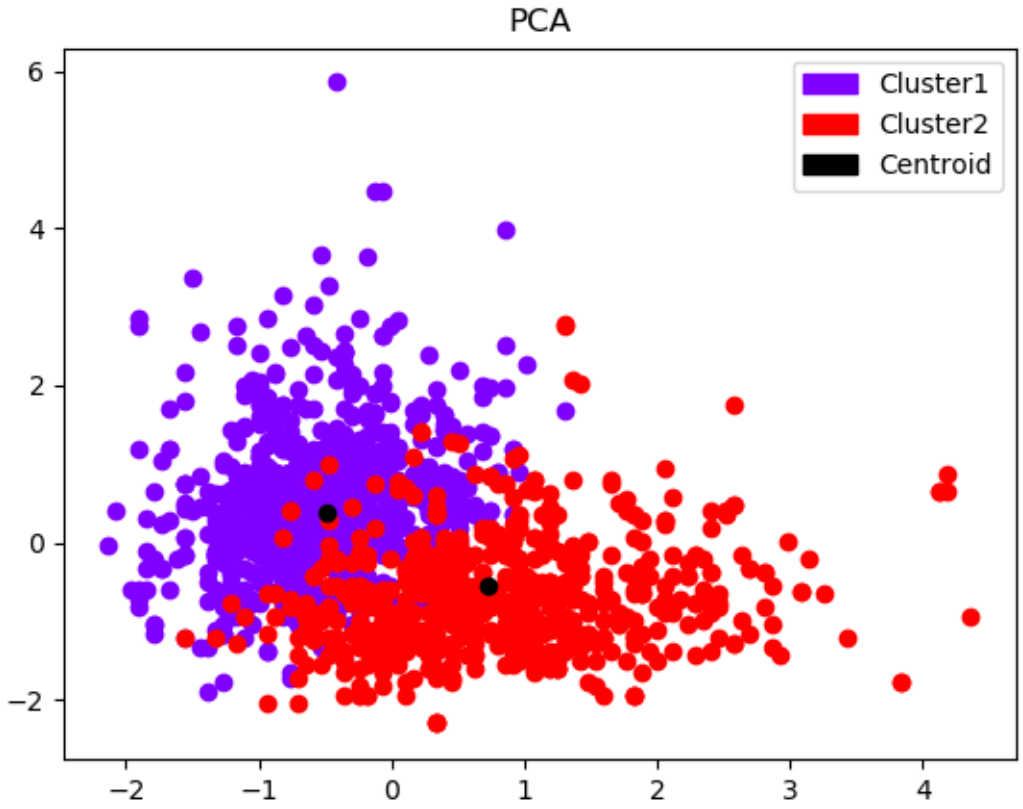
Fuzzy Partition Coefficient = 0.615336

1\3 <= 0.615336 <= 1

1 means crisp clustering, 1\3 means complete ambiguity



За K=2 за същите параметри имаме:



Fuzzy Partition Coefficient = 0.732762

1\2 <= 0.732762 <= 1

1 means crisp clustering, 1\2 means complete ambiguity

LDA графика за К=2 не е възможна.

2) Множеството от данни “wifi.csv”
   За К=4

За m=2.4 получаваме

Fuzzy Partition Coefficient = 0.344399

1\4 <= 0.344399 <= 1

1 means crisp clustering, 1\4 means complete ambiguity

За m=2.1 получаваме

Fuzzy Partition Coefficient = 0.41292

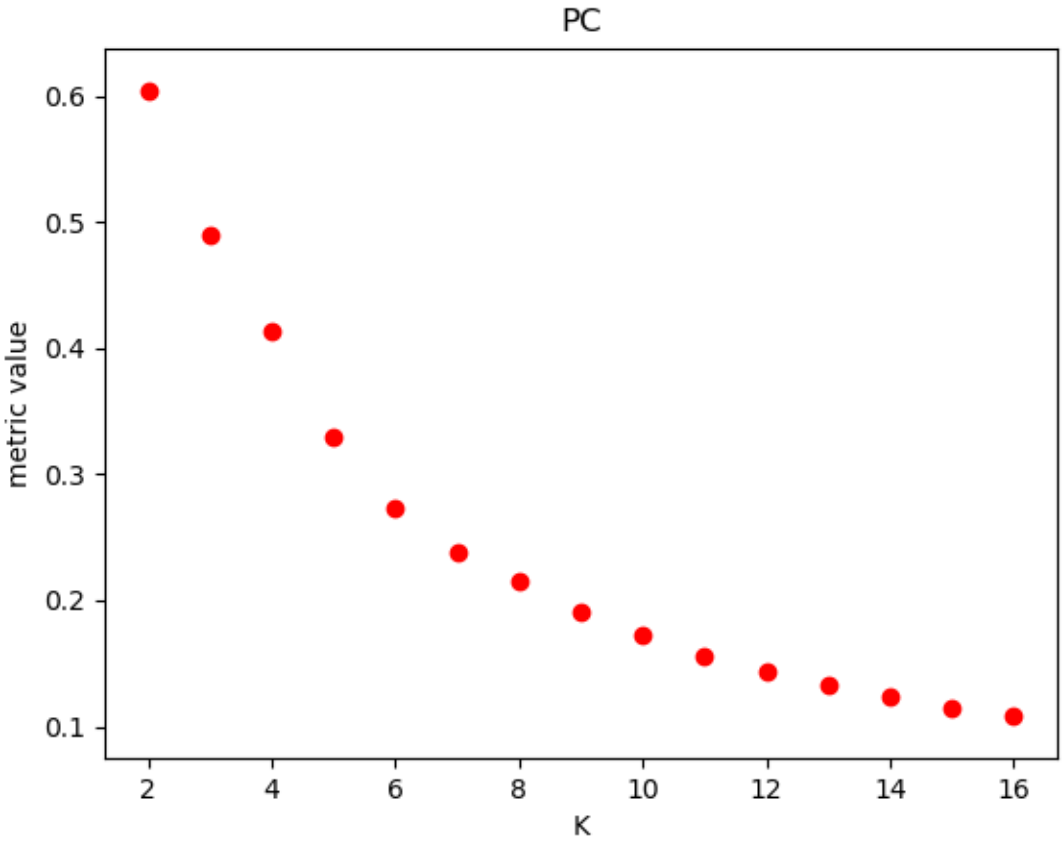
1\4 <= 0.41292 <= 1

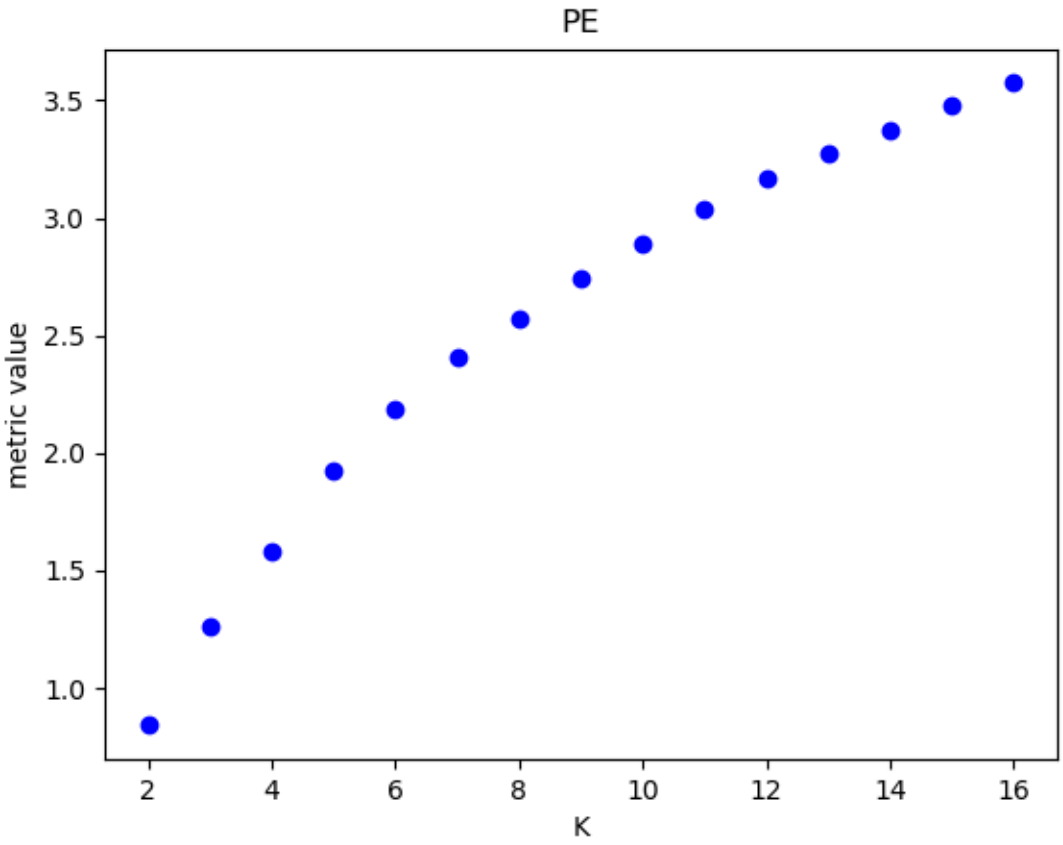
1 means crisp clustering, 1\4 means complete ambiguity

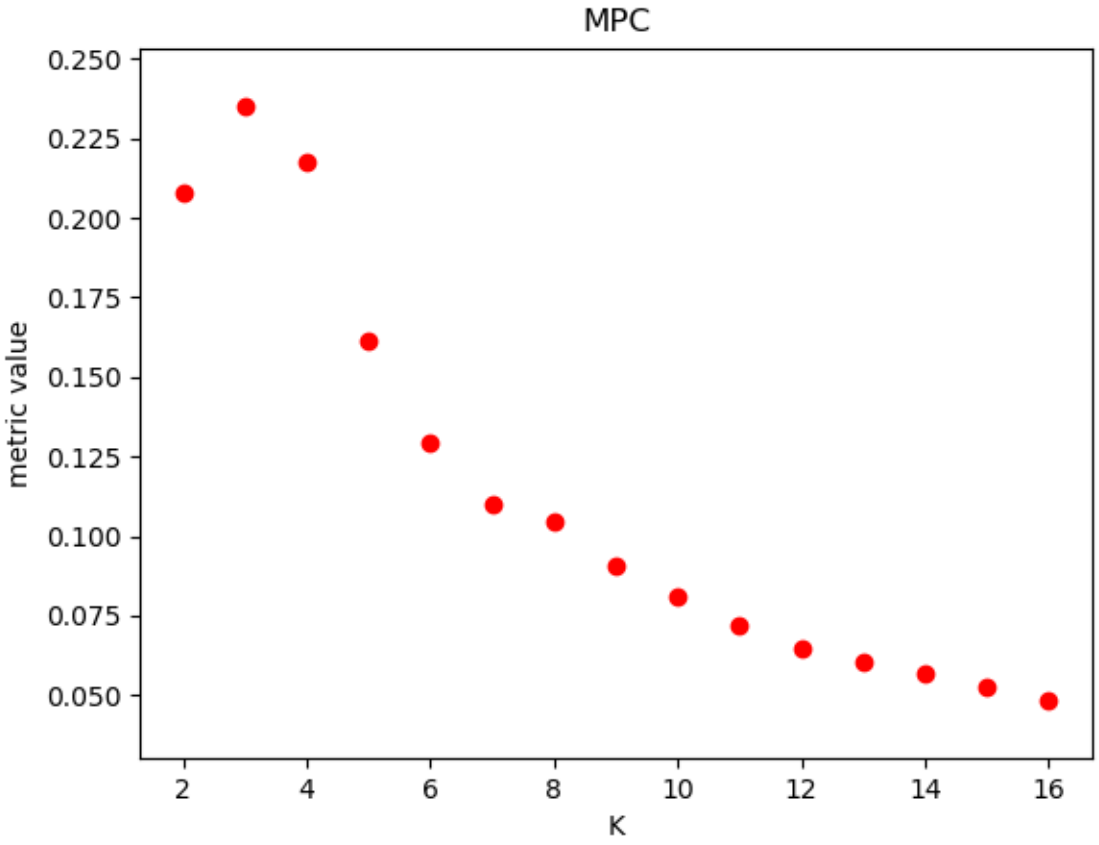
Избираме m=2.1 и прилагаме алгоритъма с автоматично определяне на К за 8те метрики със праг 0.01 или 100 итерации

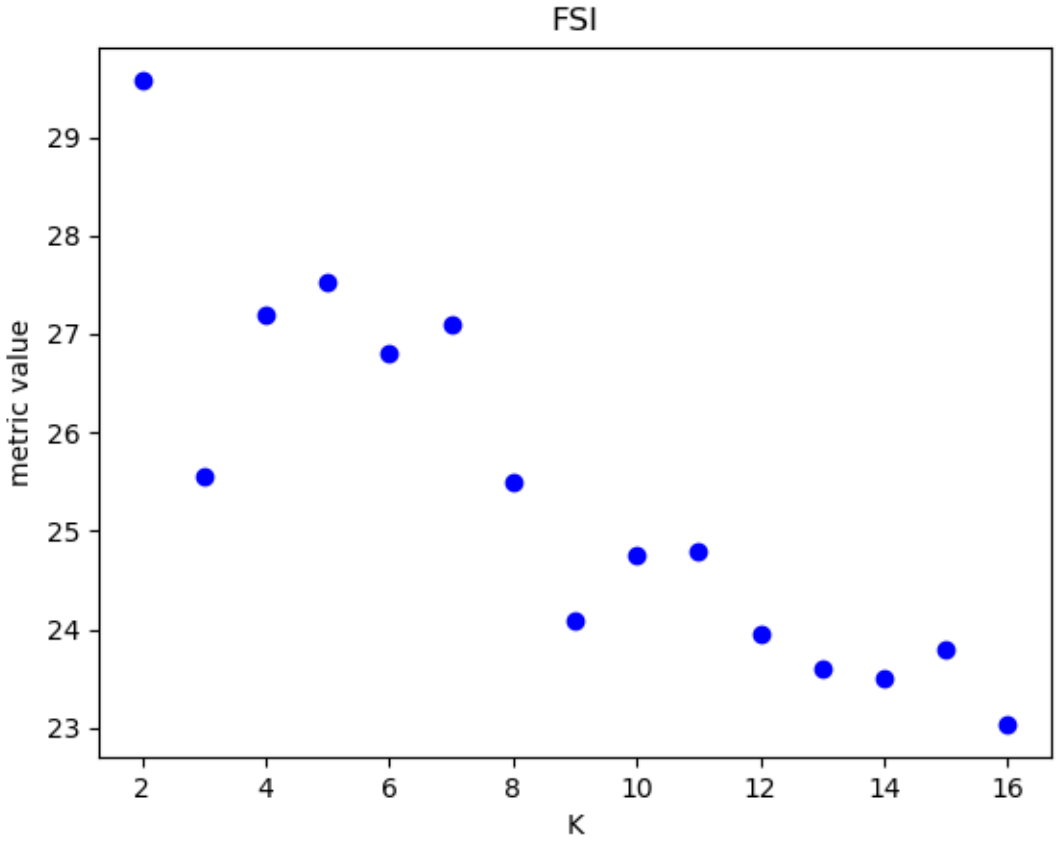
1=PC+ 2=PE- 3=MPC+ 4=FSI- 5=CHI+ 6=XBI- 7=CSI+ 8=KI-

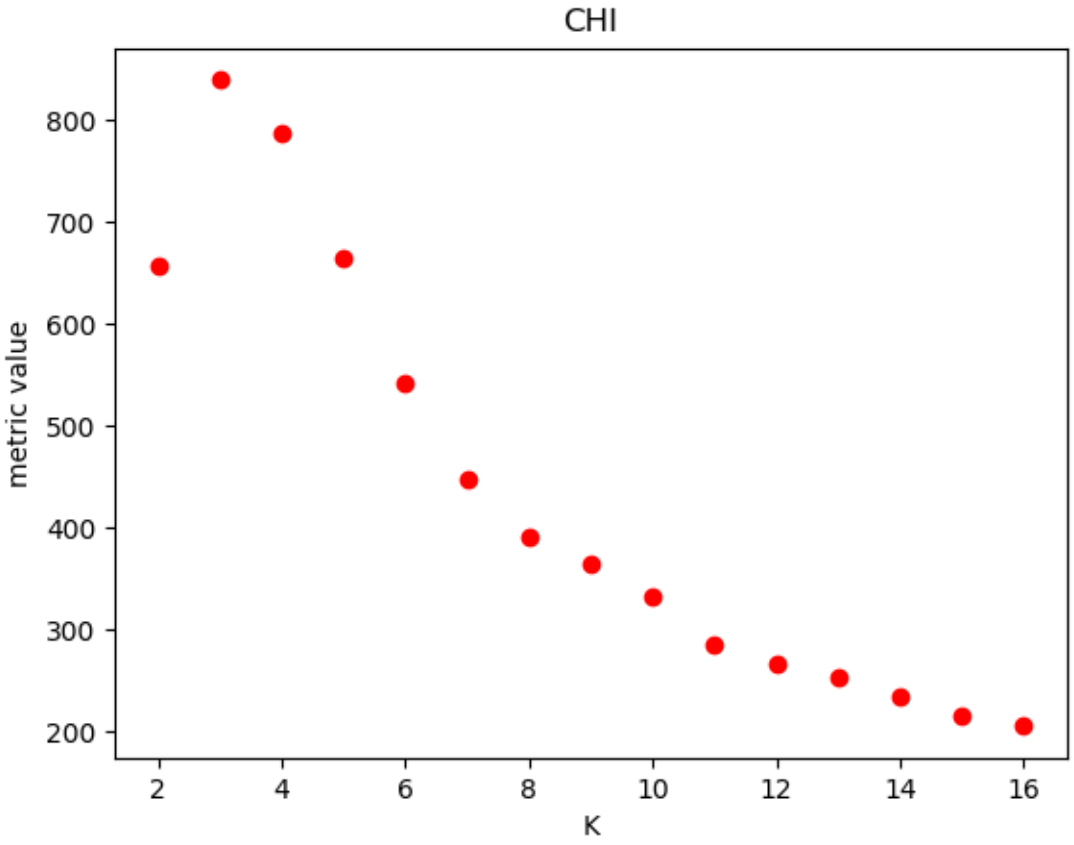
Графиките за метриките , за които търсим максимум са в червено, а тези за които търсим минимум са в синьо.

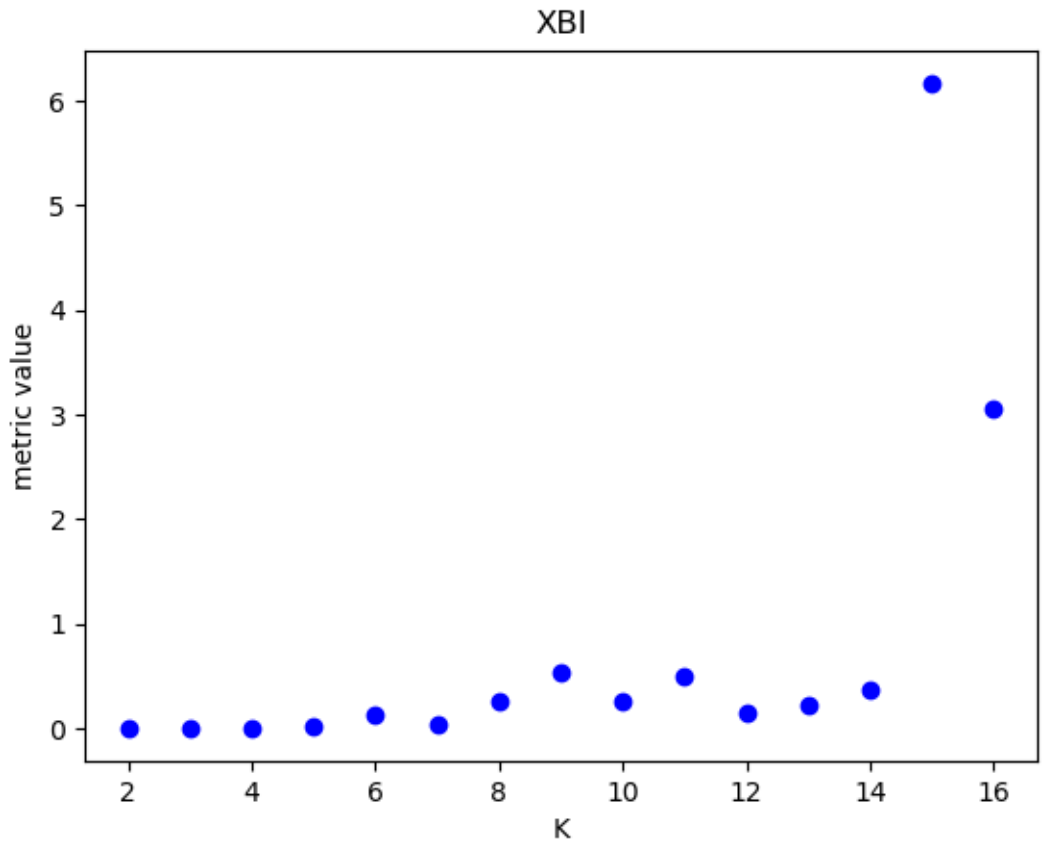

PC
metric value
0.6
0.5
0.4
0.3
0.2
0.1
2
4
6
8
10
12
14
16
K


PE
metric value
3.5
3.0
2.5
2.0
1.5
1.0
2
4
6
8
10
12
14
16
K

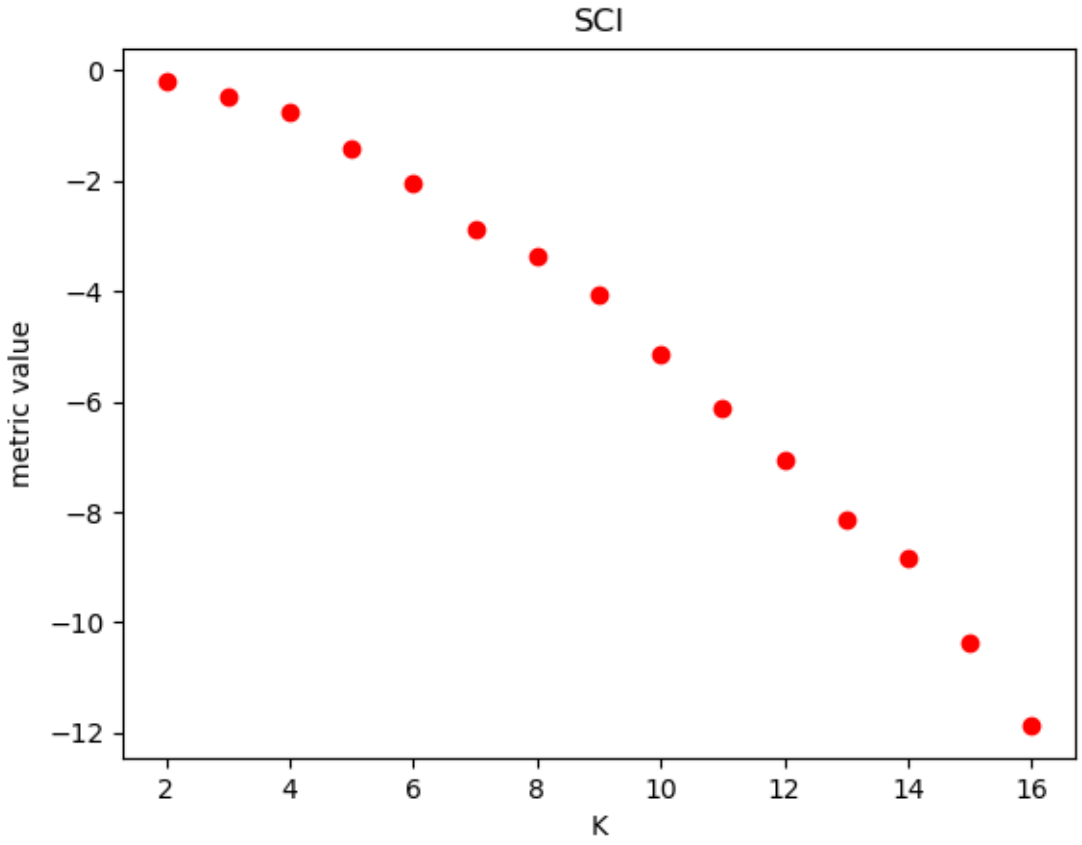

MPC
metric value
0.250
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
2
4
6
8
10
12
14
16
K

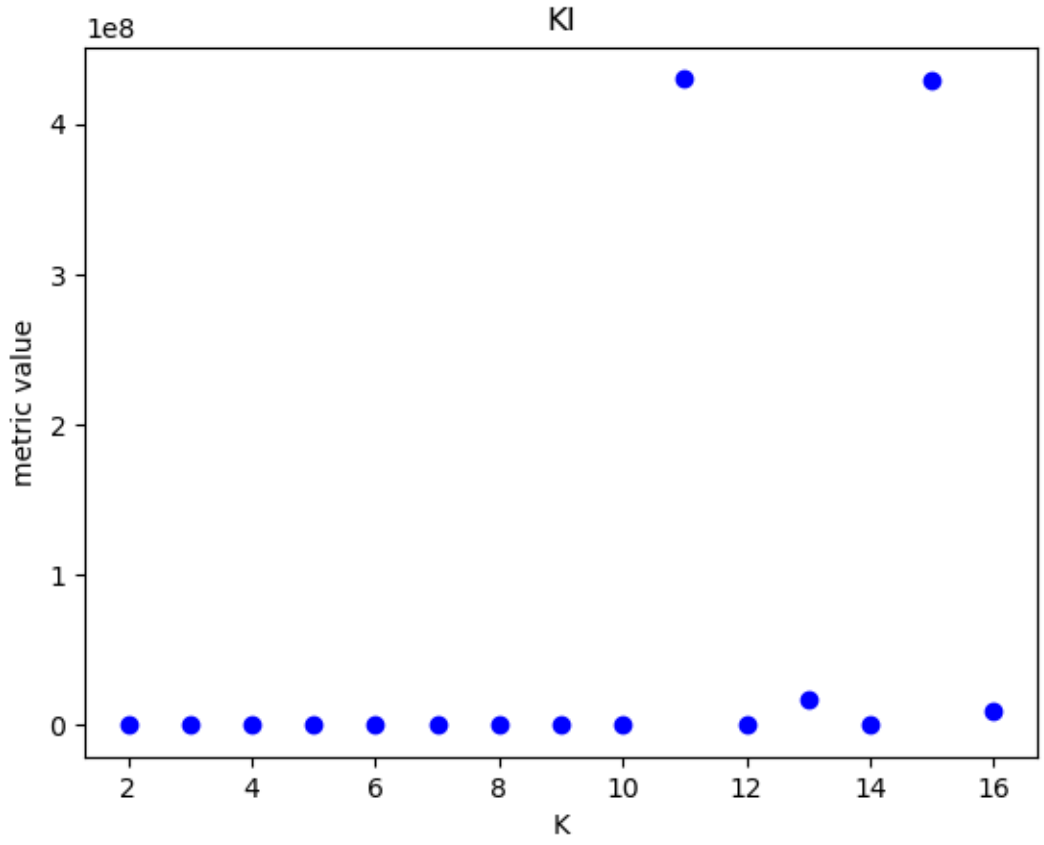

FSI
metric value
29
28
27
26
25
24
23
2
4
6
8
10
12
14
16
K





K= 2,XBI=0.00299123 ; K=3,XBI= 0.00283335 ; **K=4,XBI=0.00279017** ; K=5,XBI= 0.0255515





K=2,KI=6.6608; K=3,KI=5.09566, **K=4,KI=4.54971**; K=5,KI=285.574; K=6, KI=940.15

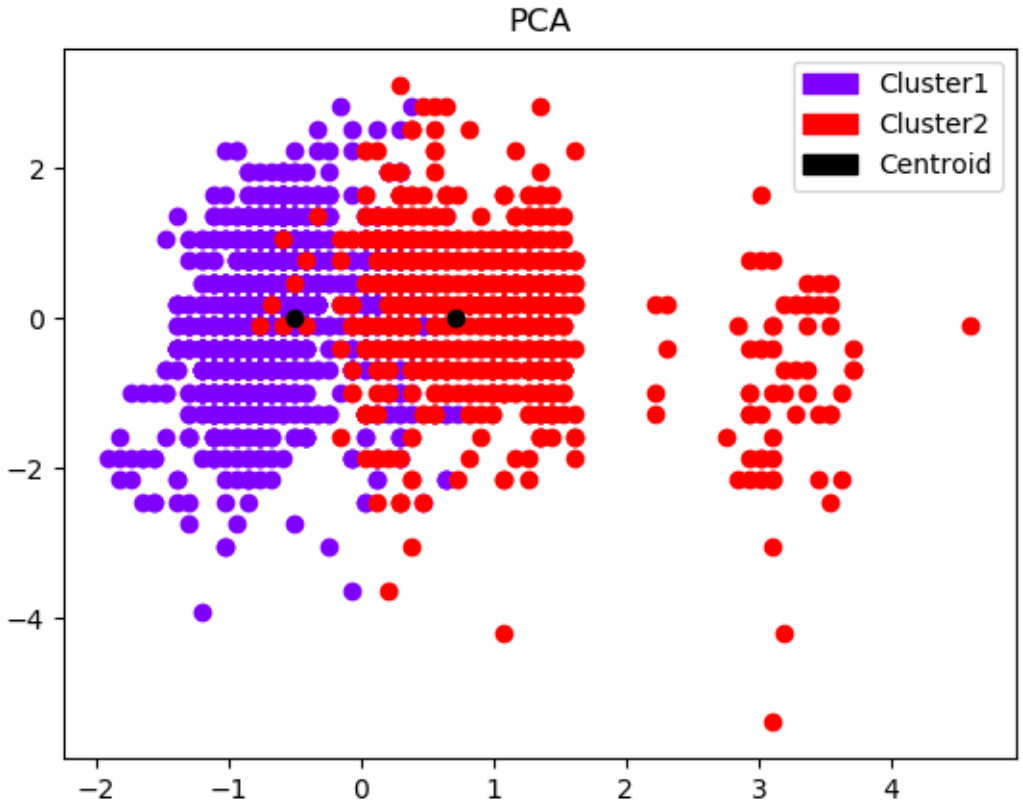
Метриките PC,PE,SCI определят за най-добро K=2 .

Метриките MPC,CHI определят за най-добро K=3 .

Метриките XBI,KI определят за най-добро K=4 . (толкова са и етикетите за това множество от данни (като етикетите не участват в класификацията))

Метриките и FSI пък определят възможно най-голямо K=16

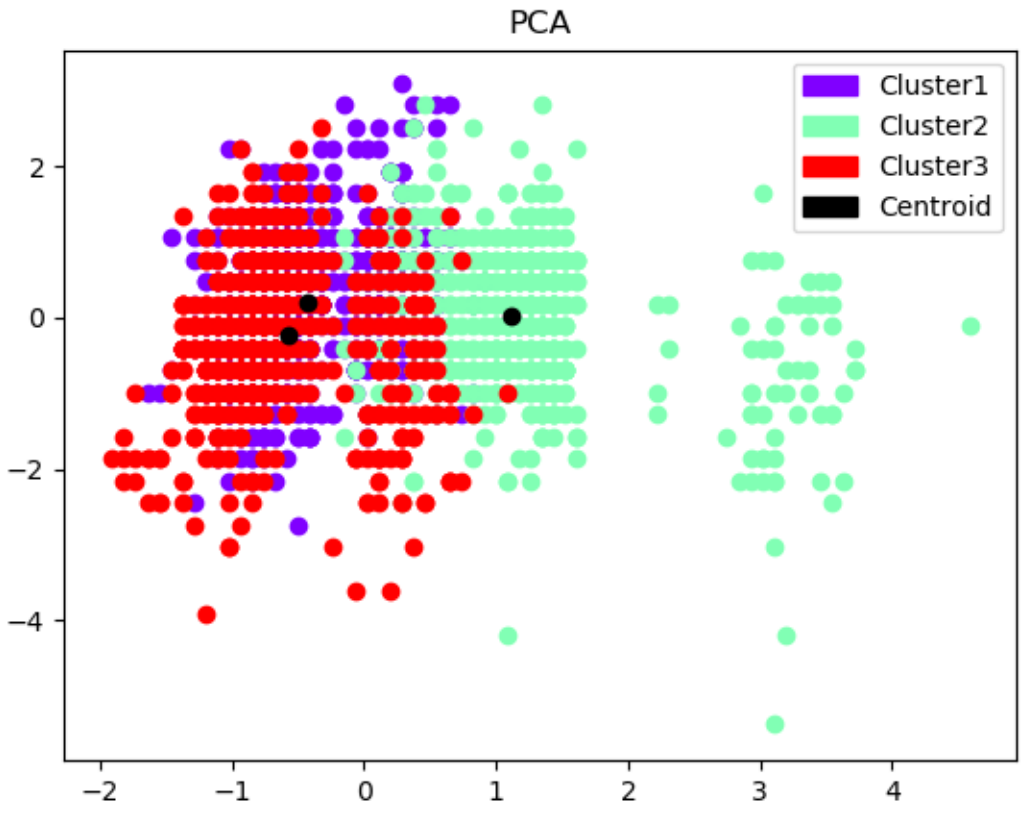
За K=2 при същите мерки, получаваме:

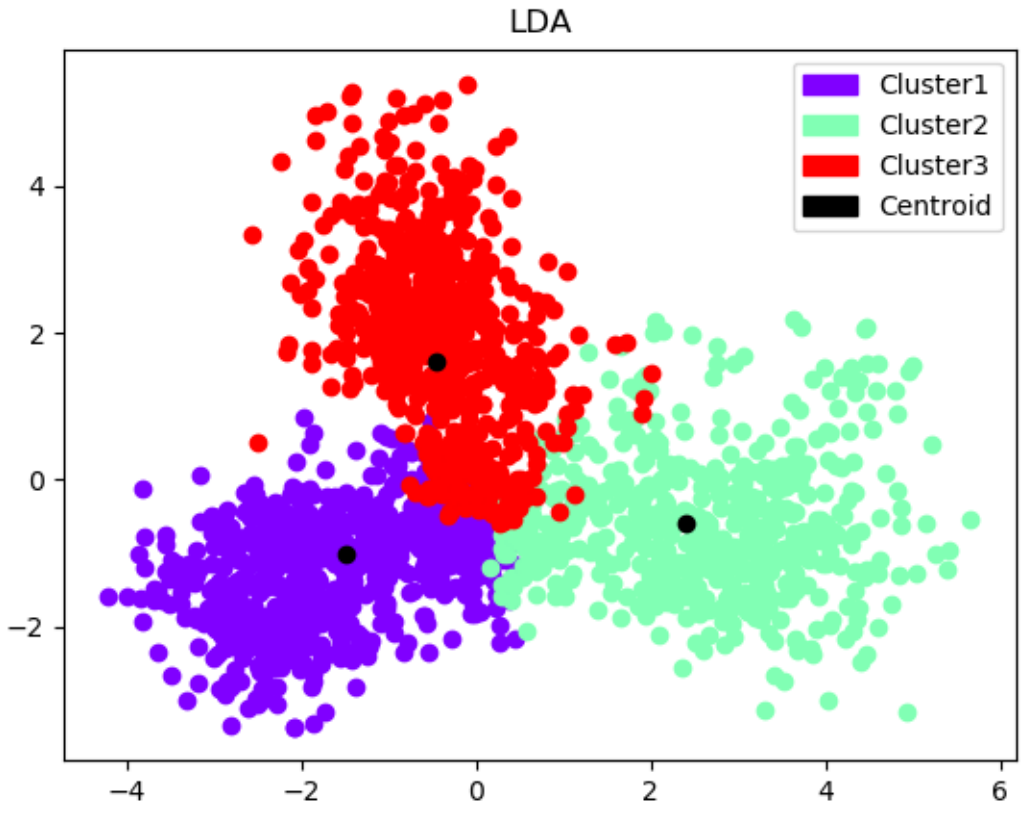


LDA графика за K=2 не е възможна ,

Fuzzy Partition Coefficient = 0.602228
1\2 <= 0.602228 <= 1
1 means crisp clustering, 1\2 means complete ambiguity

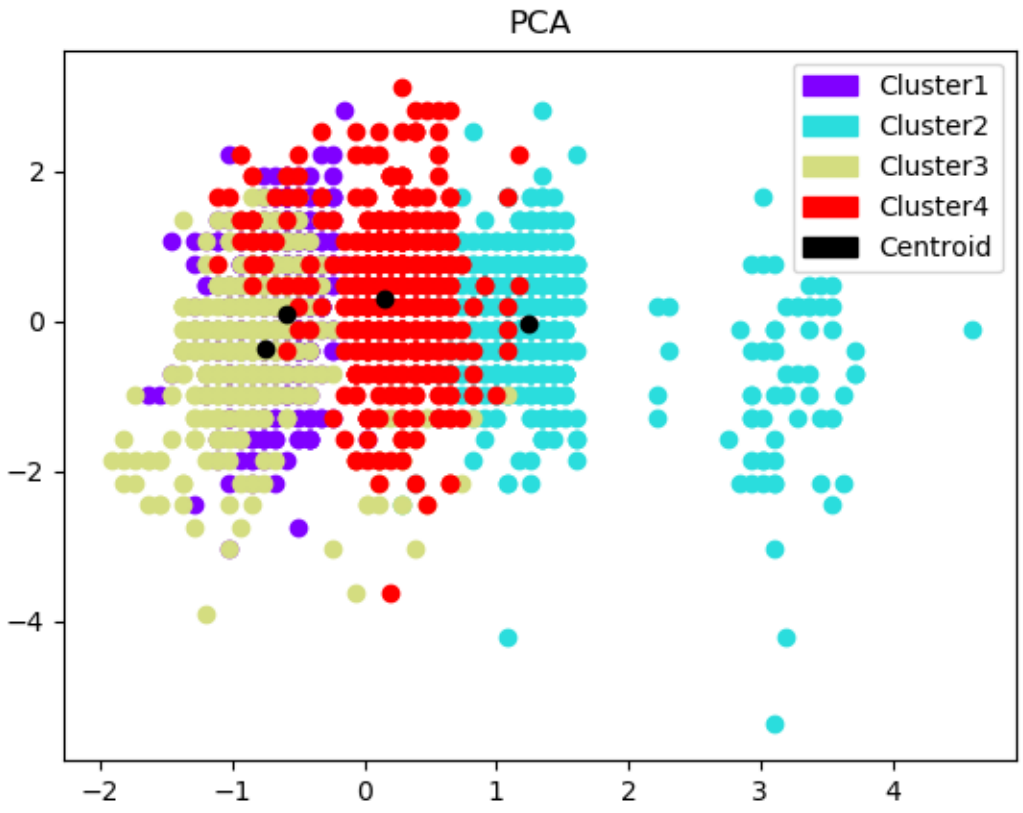
За K=3 при същите мерки, получаваме:

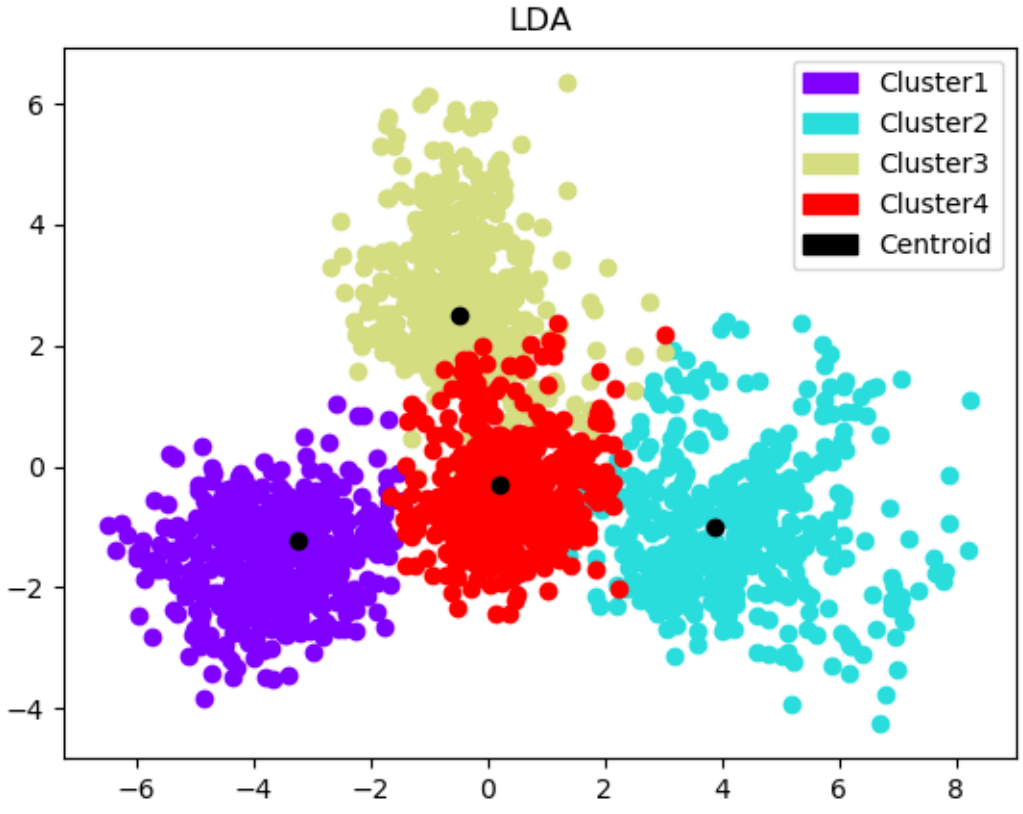




Fuzzy Partition Coefficient = 0.489782
1\3 <= 0.489782 <= 1
1 means crisp clustering, 1\3 means complete ambiguity

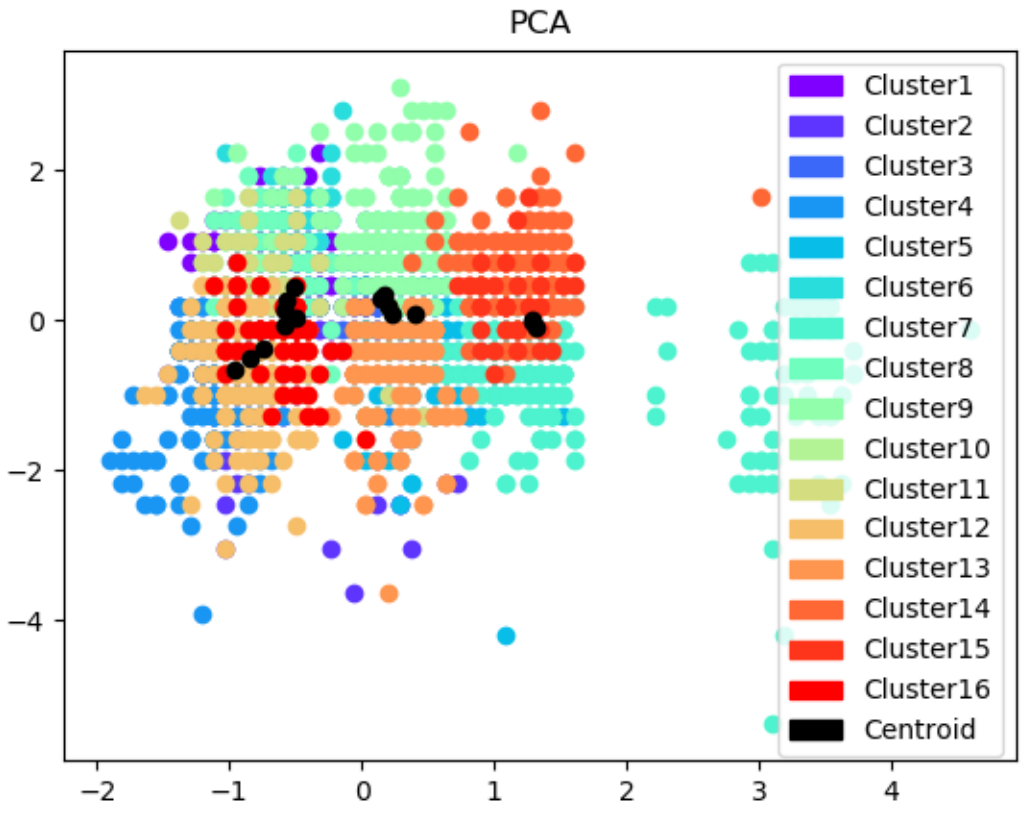
За K=4 при същите мерки, получаваме:

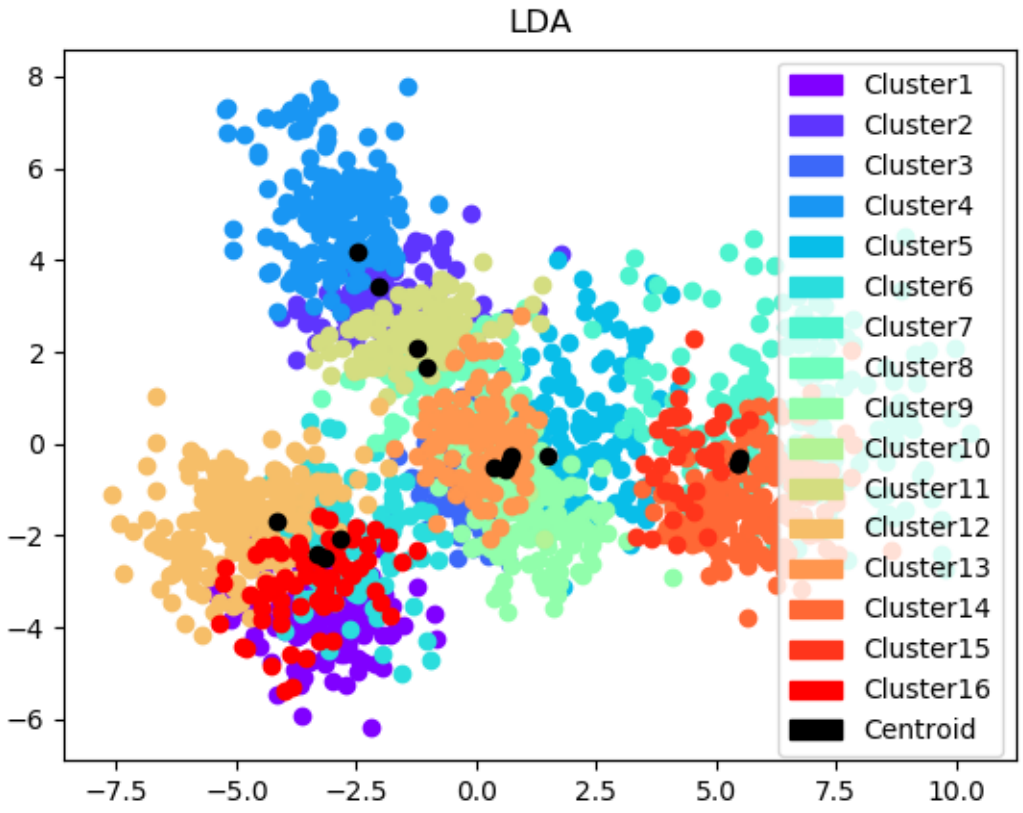




Fuzzy Partition Coefficient = 0.413405
1\4 <= 0.413405 <= 1
1 means crisp clustering, 1\4 means complete ambiguity

За K=16 при същите мерки, получаваме:





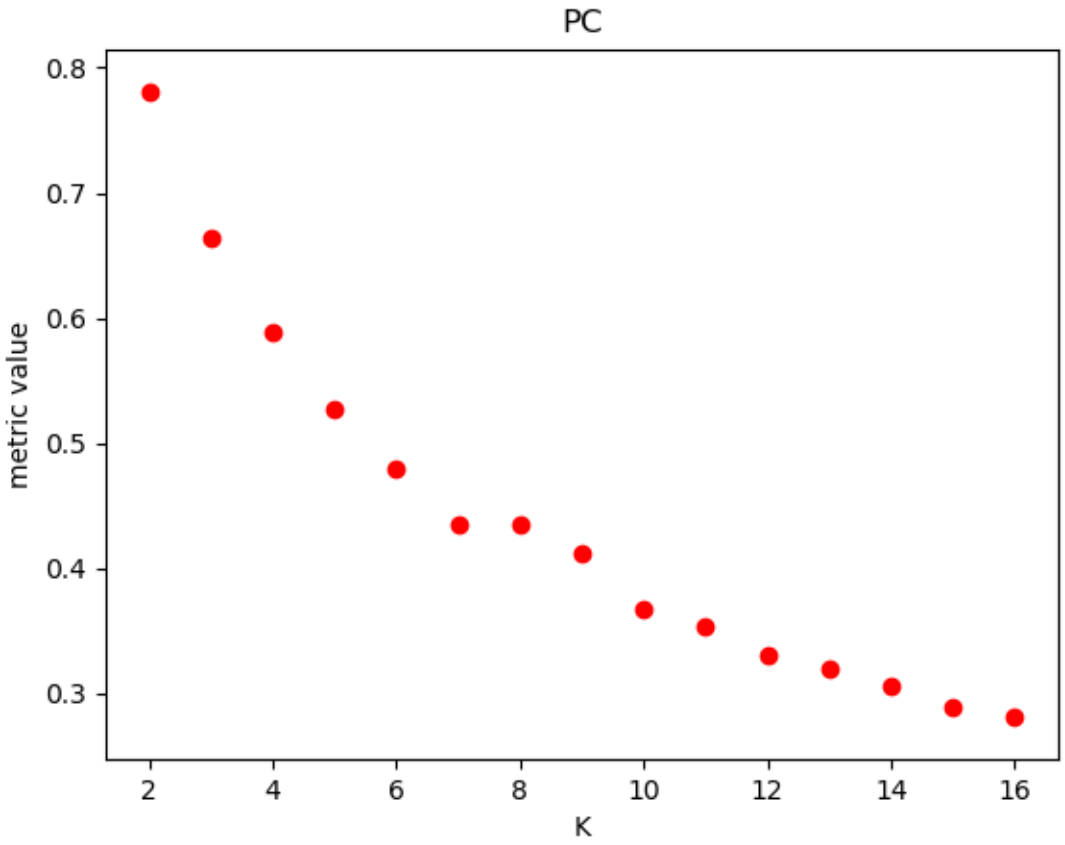
Fuzzy Partition Coefficient = 0.107674
1\16 <= 0.107674 <= 1
1 means crisp clustering, 1\16 means complete ambiguity

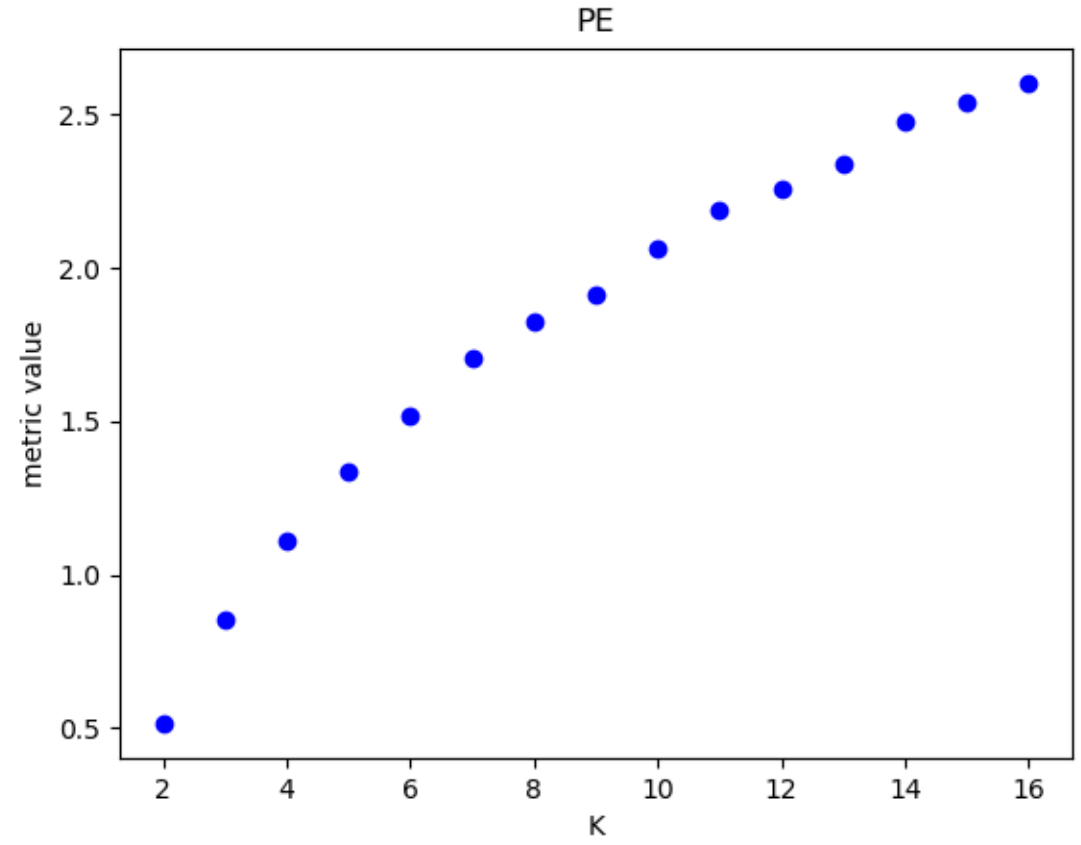
1) Множеството от данни “snails.csv”
   За К=4
   За m=2 имаме
   Fuzzy Partition Coefficient = 0.588425
   1\4 <= 0.588425 <= 1
   1 means crisp clustering, 1\4 means complete ambiguity

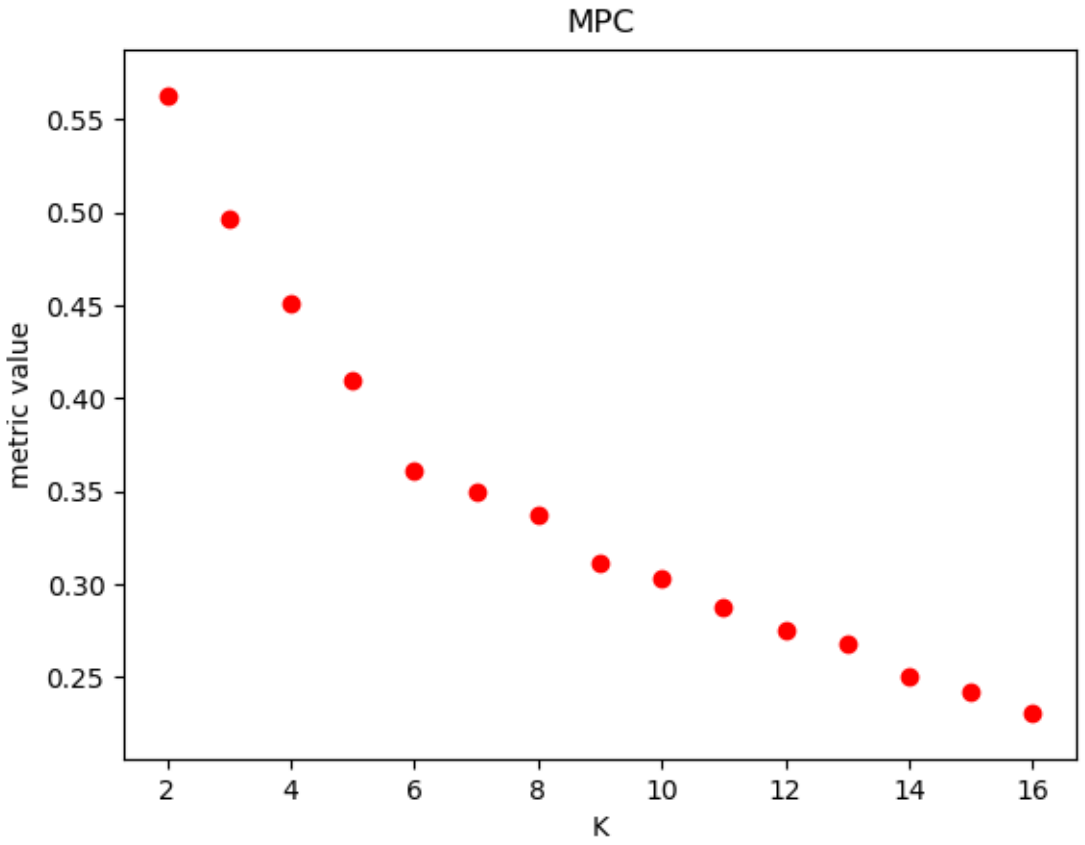
Избираме m=2 и прилагаме алгоритъма с автоматично определяне на К за 8те метрики със праг 0.001 или 100 итерации

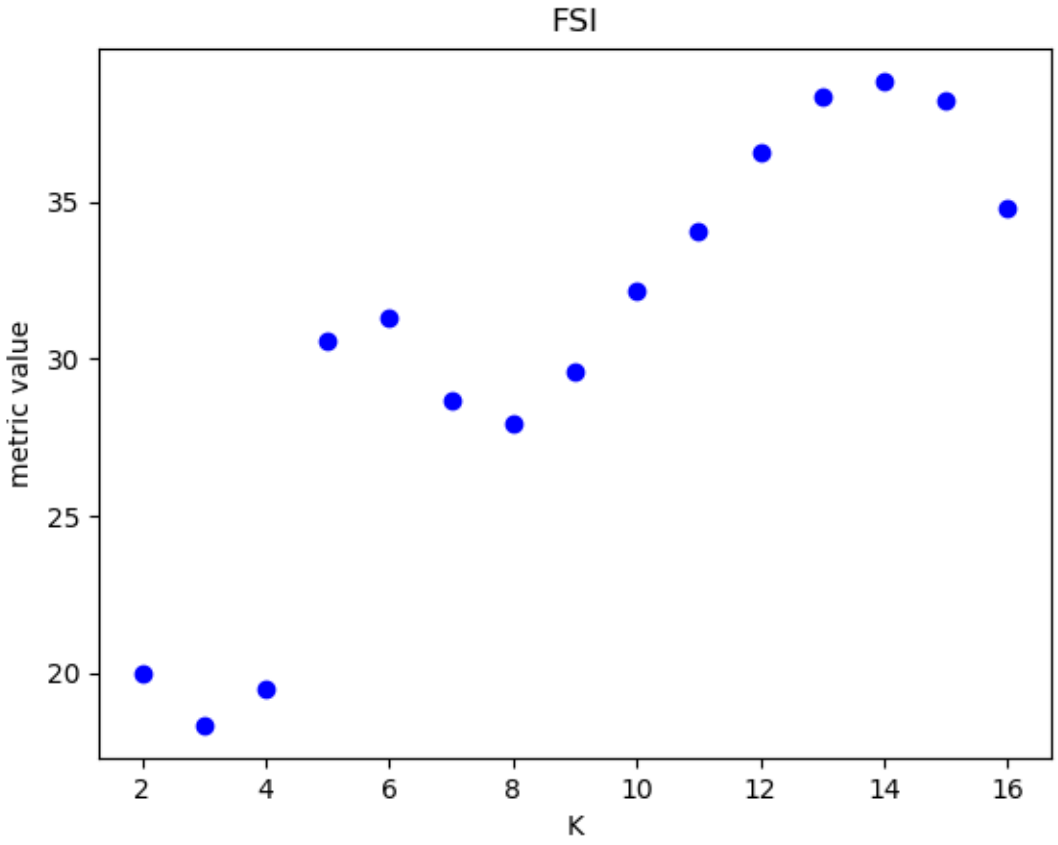
`1=PC+ 2=PE- 3=MPC+ 4=FSI- 5=CHI+ 6=XBI- 7=CSI+ 8=KI-`

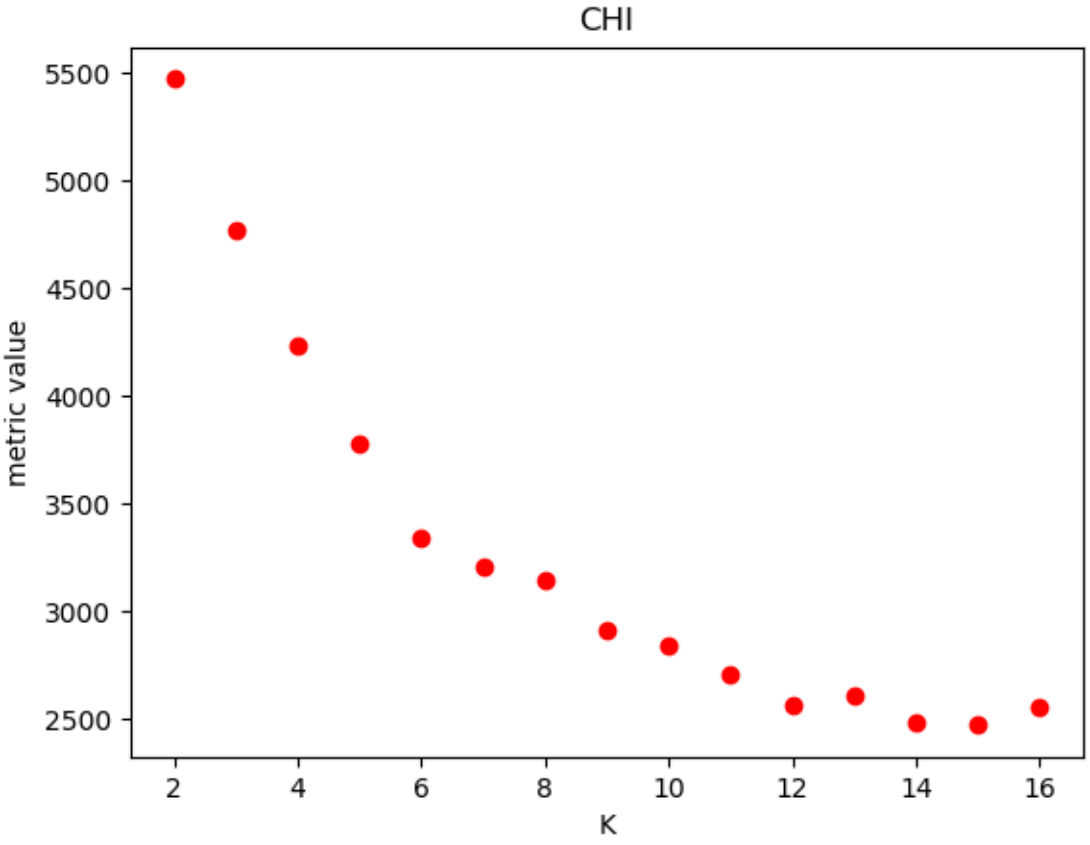
Графиките за метриките , за които търсим максимум са в червено, а тези за които търсим минимум са в синьо.

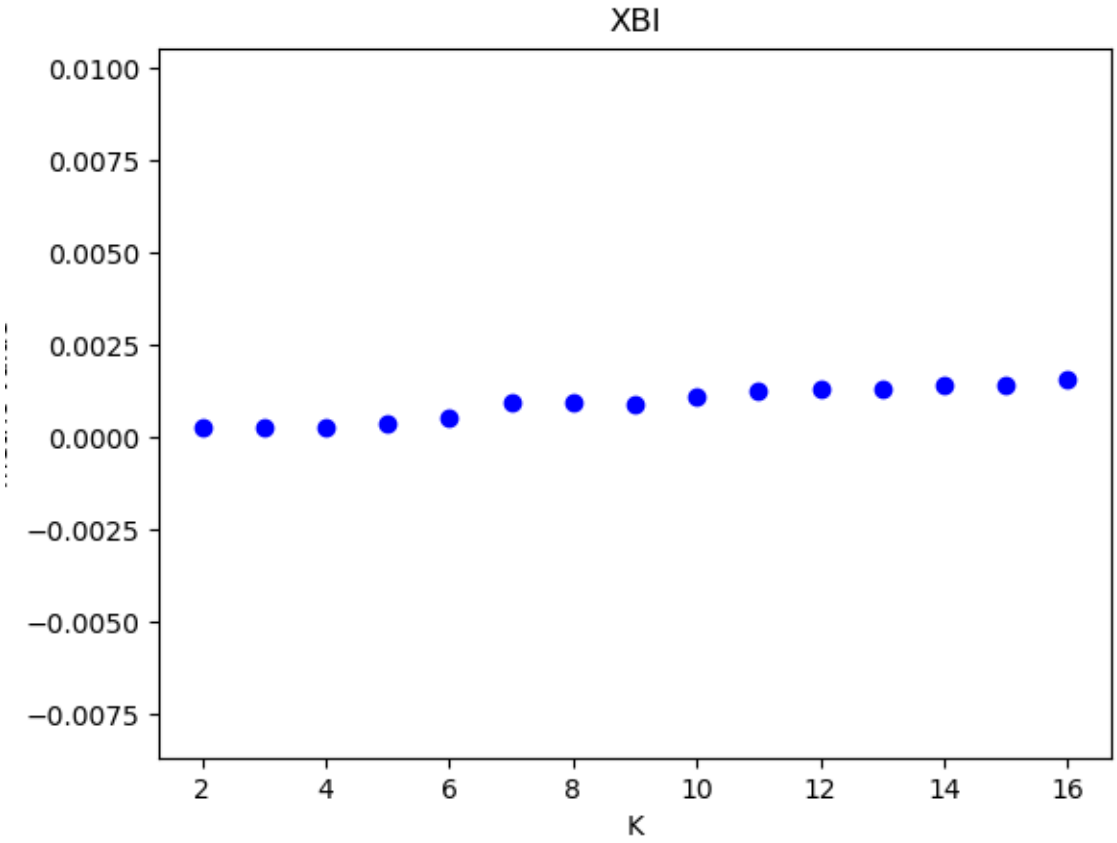



PE
metric value
2.5
2.0
1.5
1.0
0.5
2
4
6
8
10
12
14
16
K

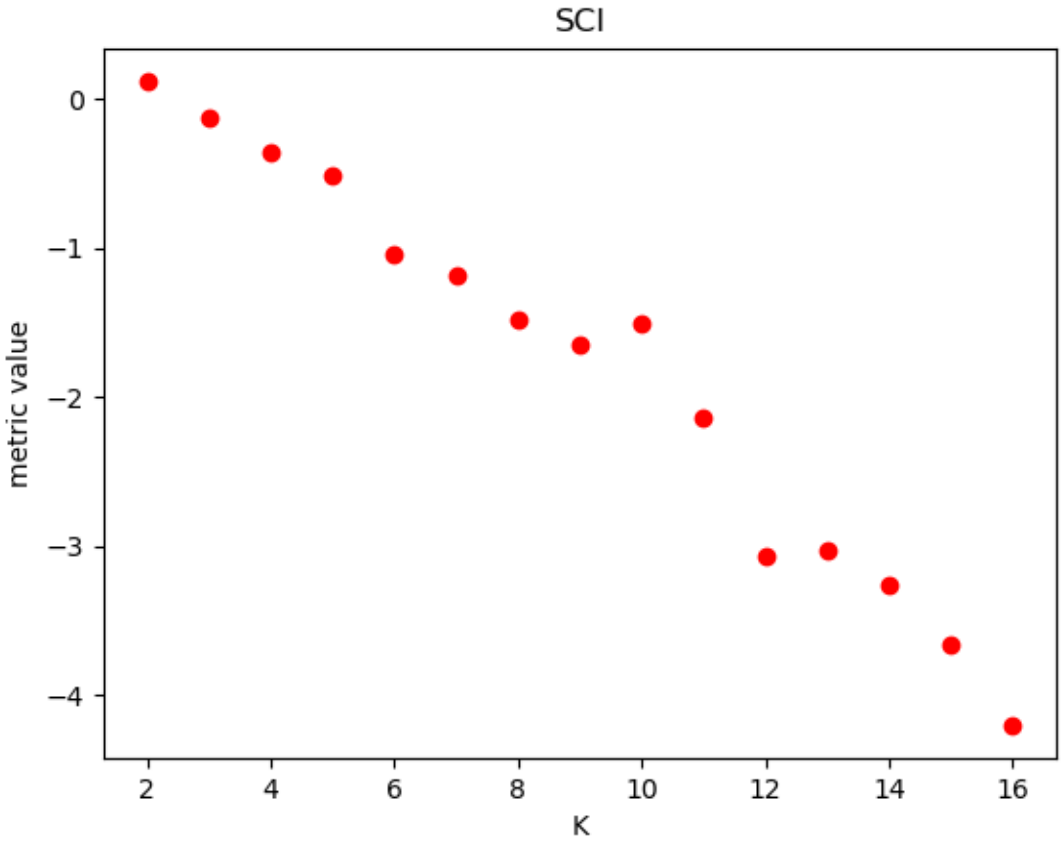

MPC
metric value
0.55
0.50
0.45
0.40
0.35
0.30
0.25
2
4
6
8
10
12
14
16
K

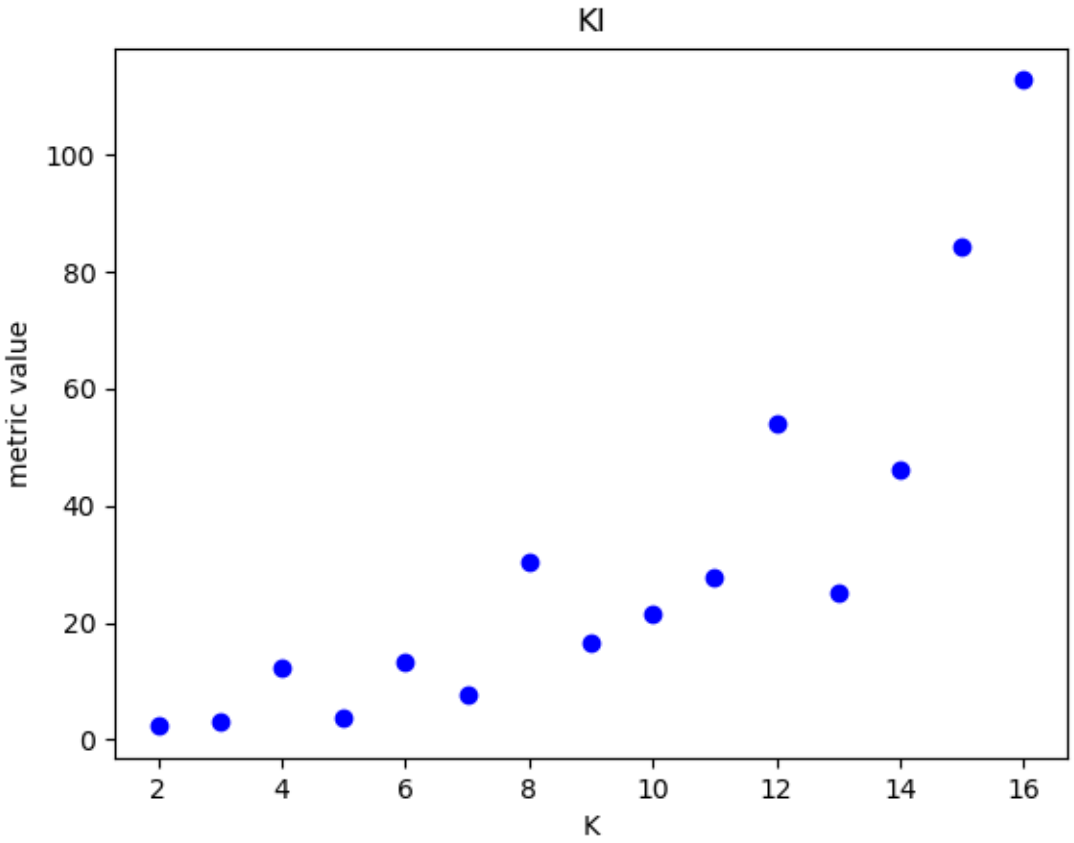

FSI
metric value
35
30
25
20
2
4
6
8
10
12
14
16
K


CHI
metric value
5500
5000
4500
4000
3500
3000
2500
2
4
6
8
10
12
14
16
K



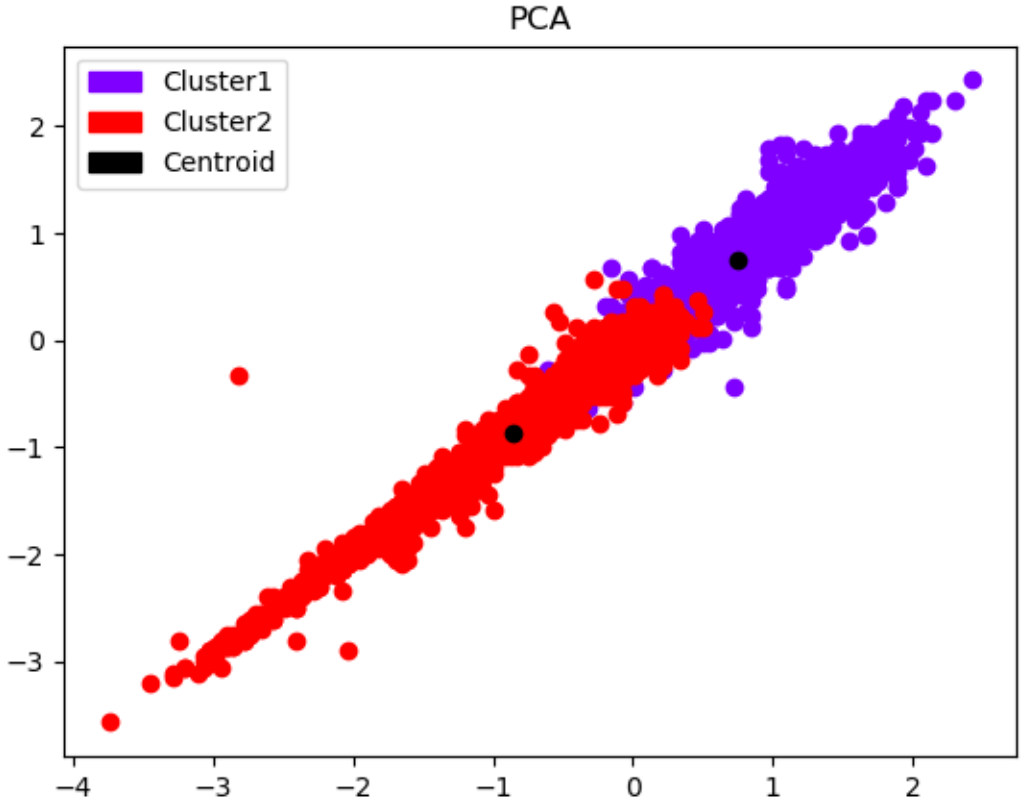
K=2, XBI= 0.000292016; K=3, XBI= 0.00029915 ; K=4, XBI=0.000294888





K=2, KI=2.29871 ; K=3, KI=3.16115
Всички метриките освен FSI определят за най-добро K=2. FSI определя за най =-добор K=3.

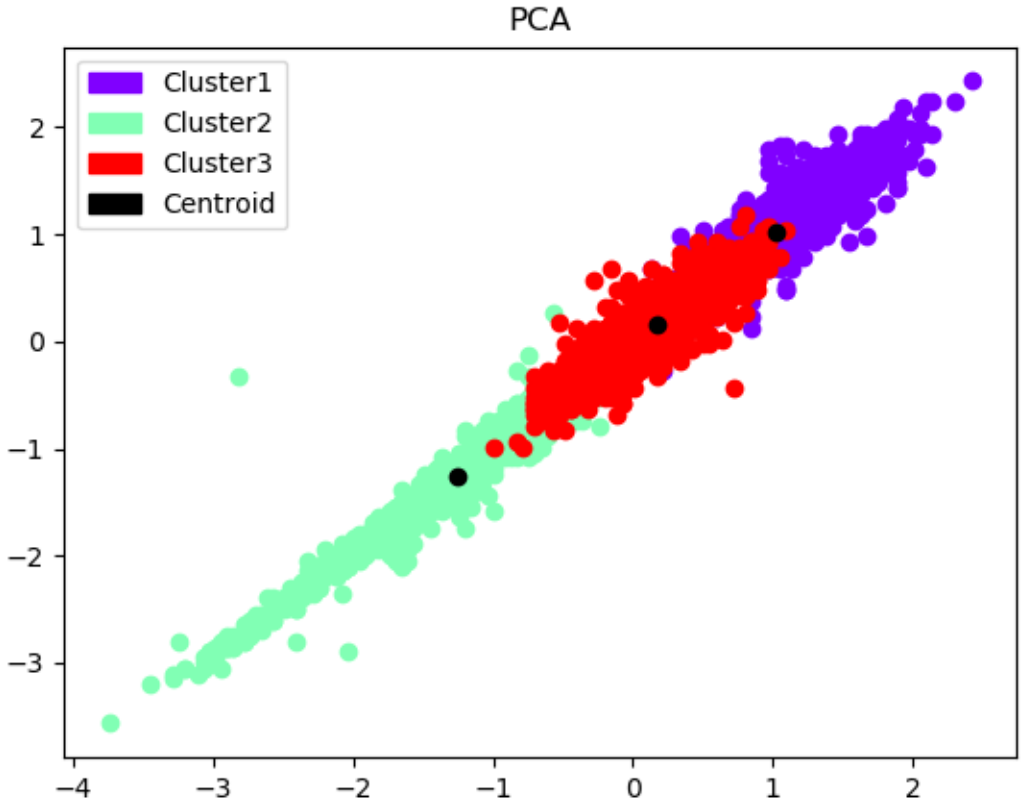
При същите параметри за K=2 имаме:

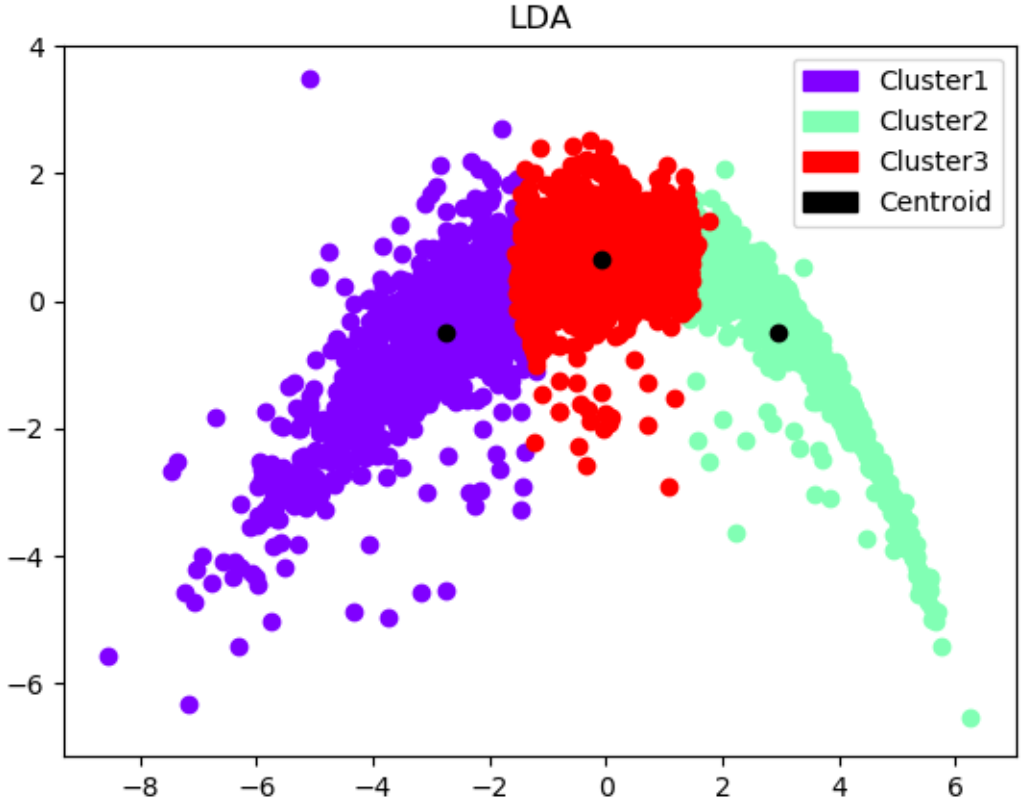


LDA графика за 2 класа не е възможна.

Fuzzy Partition Coefficient = 0.781271
1\2 <= 0.781271 <= 1
1 means crisp clustering, 1\2 means complete ambiguity

При К=3, при същите параметри получаваме:





Fuzzy Partition Coefficient = 0.664297
1\3 <= 0.664297 <= 1
1 means crisp clustering, 1\3 means complete ambiguity

Заключение

Представеният алгоритъм за размита клъстеризация fuzzy-c-means бе подробно представен, конкретната имплементация разяснена и бяха показани резултати за автоматично определяне на параметър K спрямо 8 метрики спрямо 3 различни множества данни.

Както и обикновеният crip C-means , и fuzzy-C-means намира хиперсфери в n-мерно пространство и не е подходящ в случай, когато пространството от примери, не може да бъде добре разделено на класове от такива хипер-сфери.

Въпреки това, алгоритъма е един от най-популярните за размита клъстеризация и често се пробва като пръв базов модел в практически задачи.

Някой подобрения към текущата имплементация биха били да се позволи работа с по-свободно форматирани входни данни, да се работи и с категориини атрибути, да се предостави възможност за определяне автоматично едновременно на m и на K, също така може да бъде паралелизиран за ускорение.

Библиография

[fuzzy logic with engineering applications 3rd edition] (стр. 332-362)

[A tutorial in principal component analysis]

[Kmeans++]

[LDA in sklearn]

[Performance evaluation of Cluster Validity Indices]