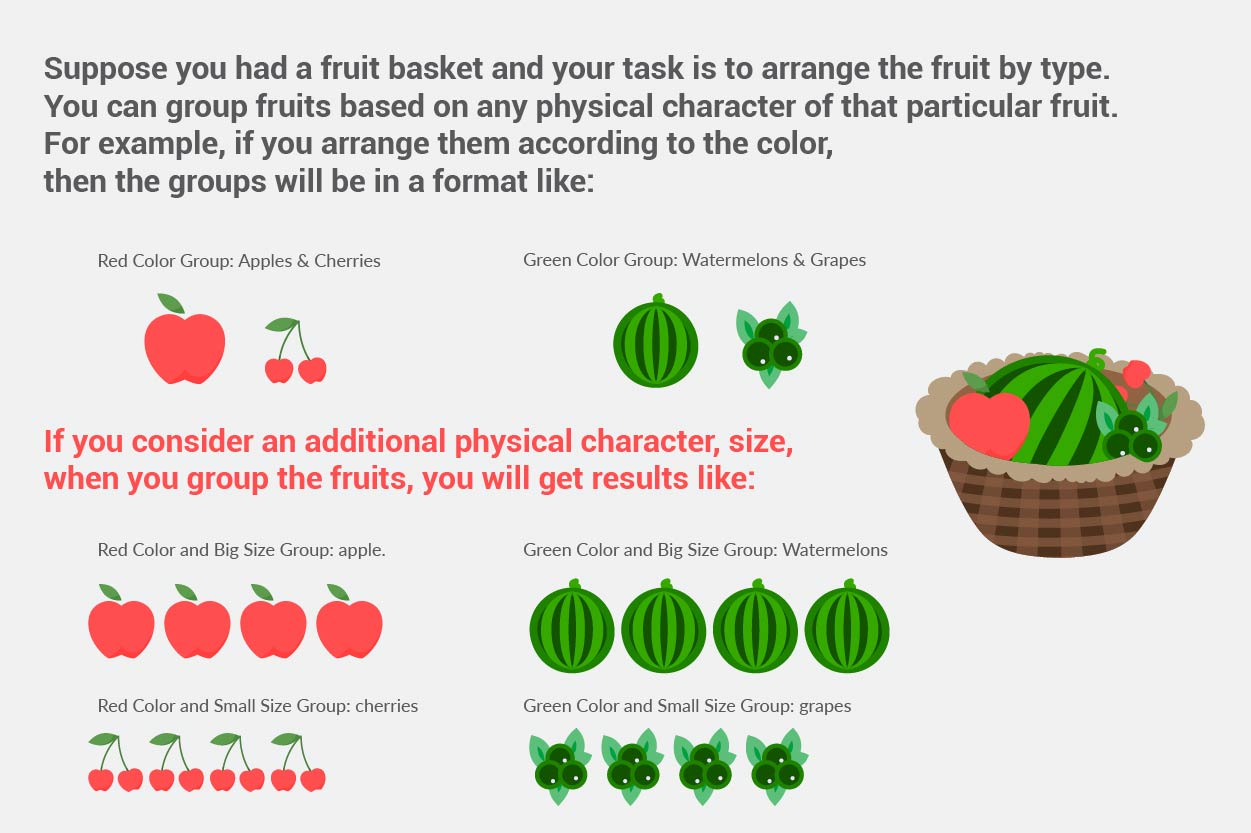
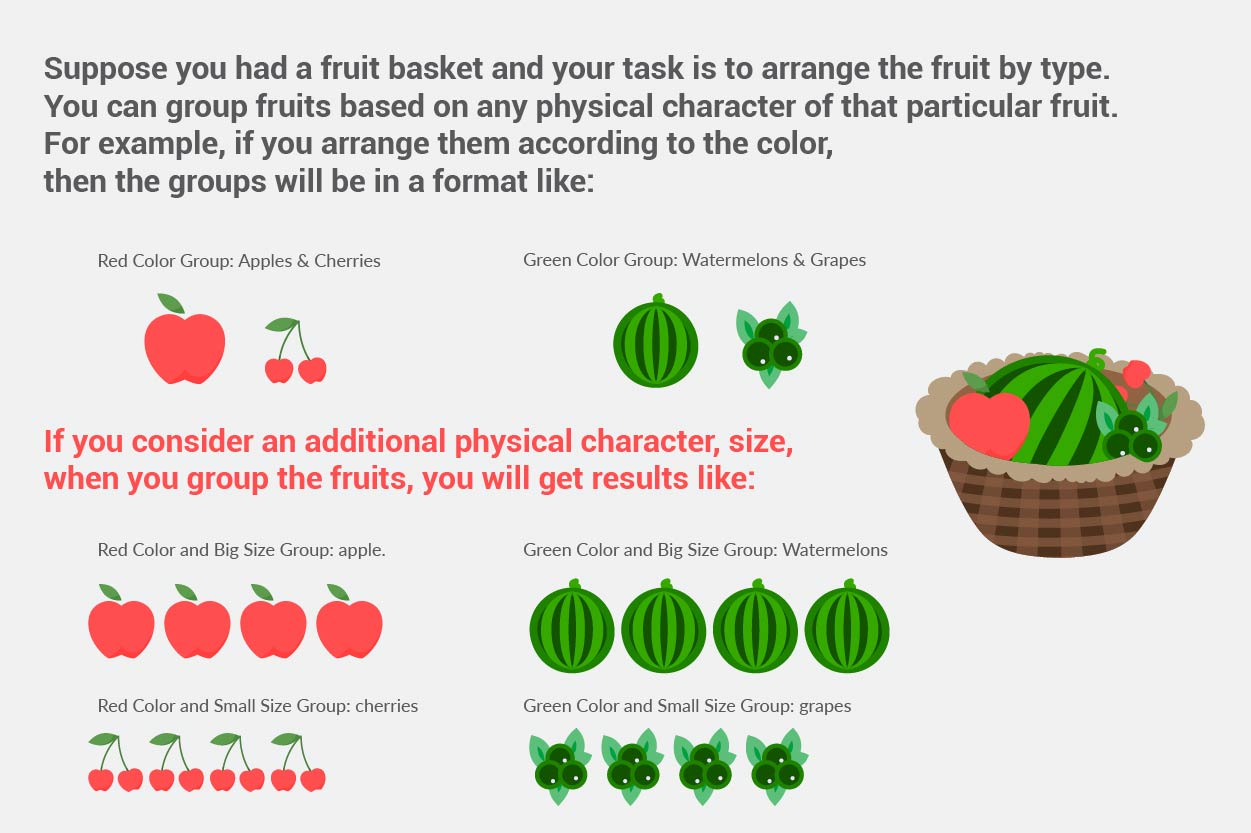
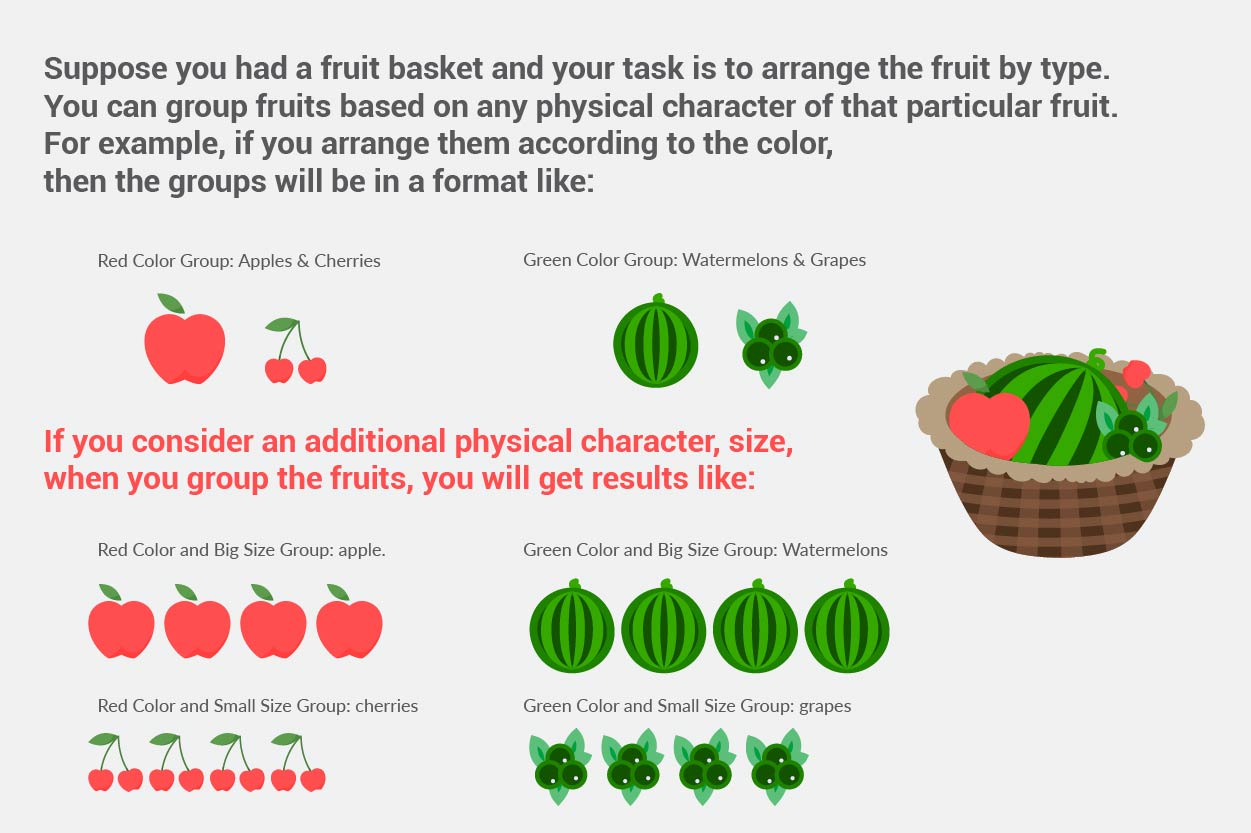
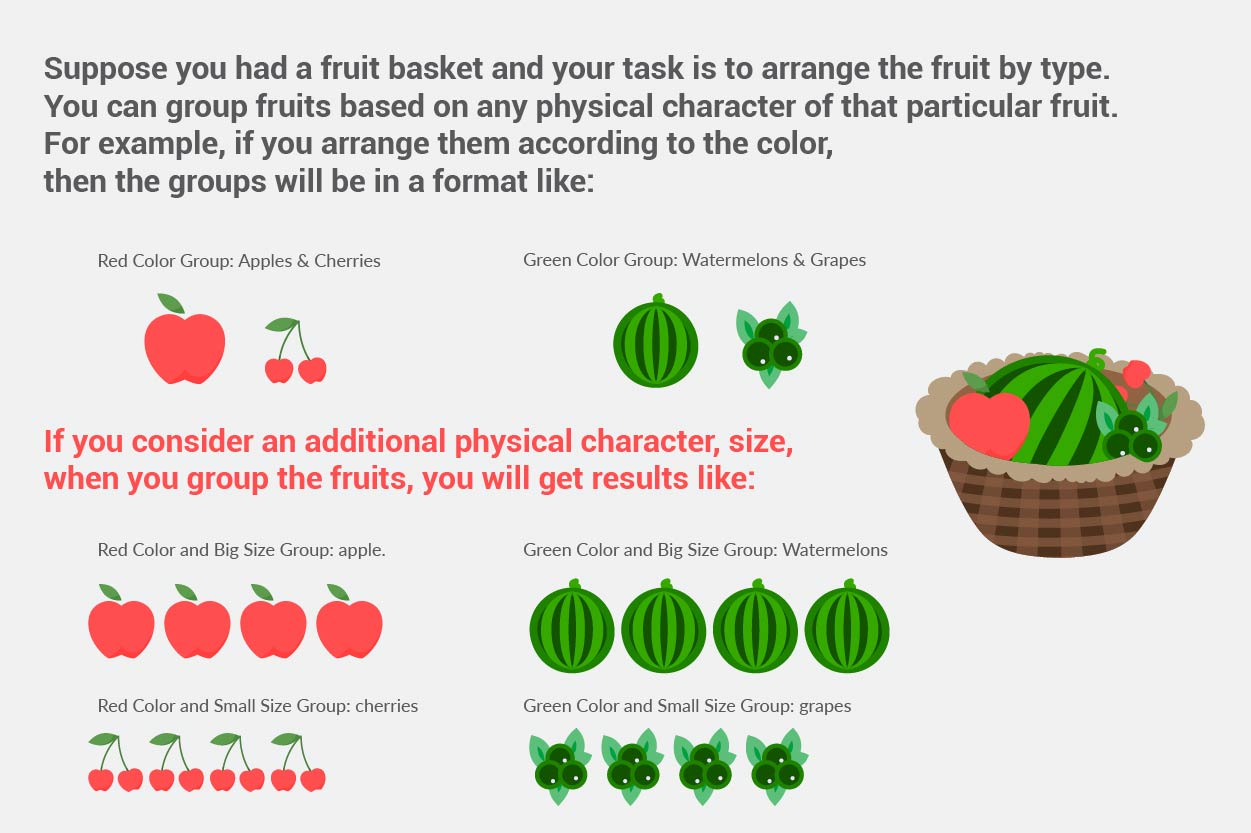
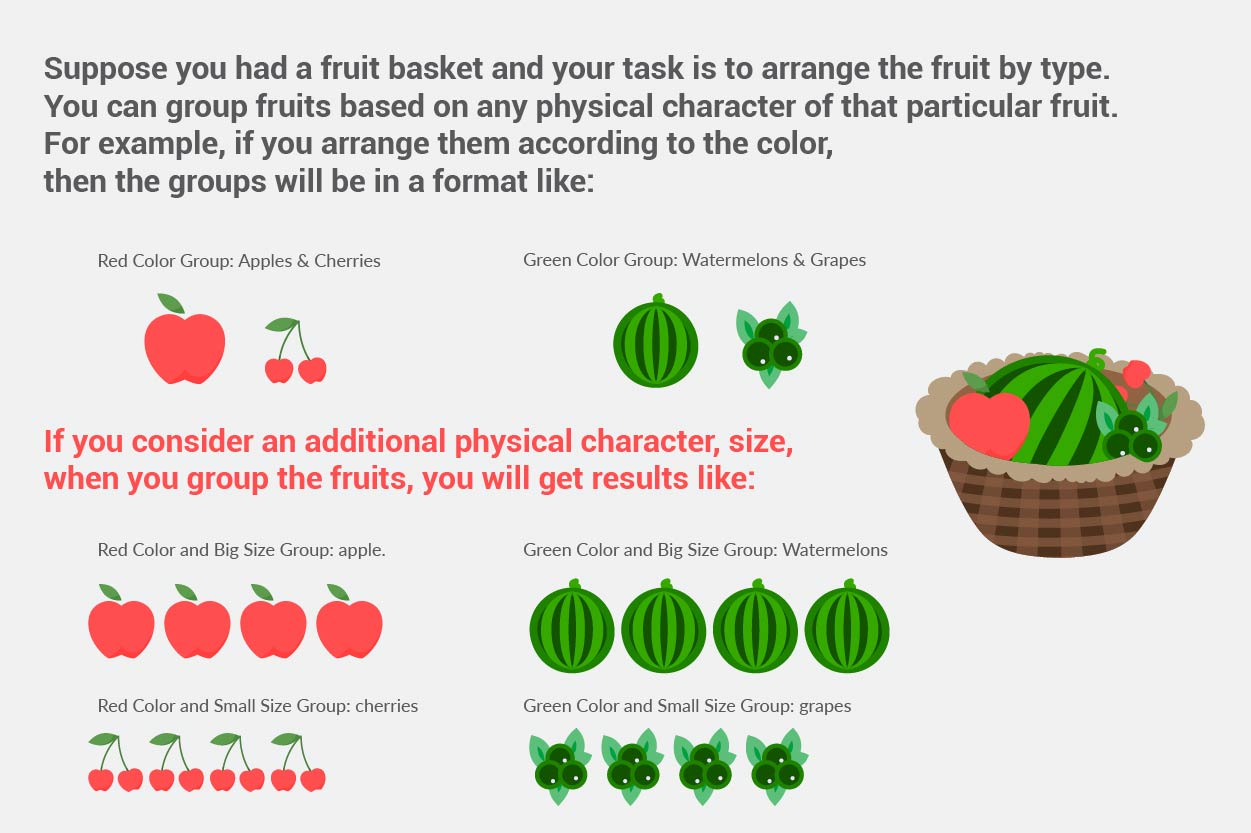
**Clustering**

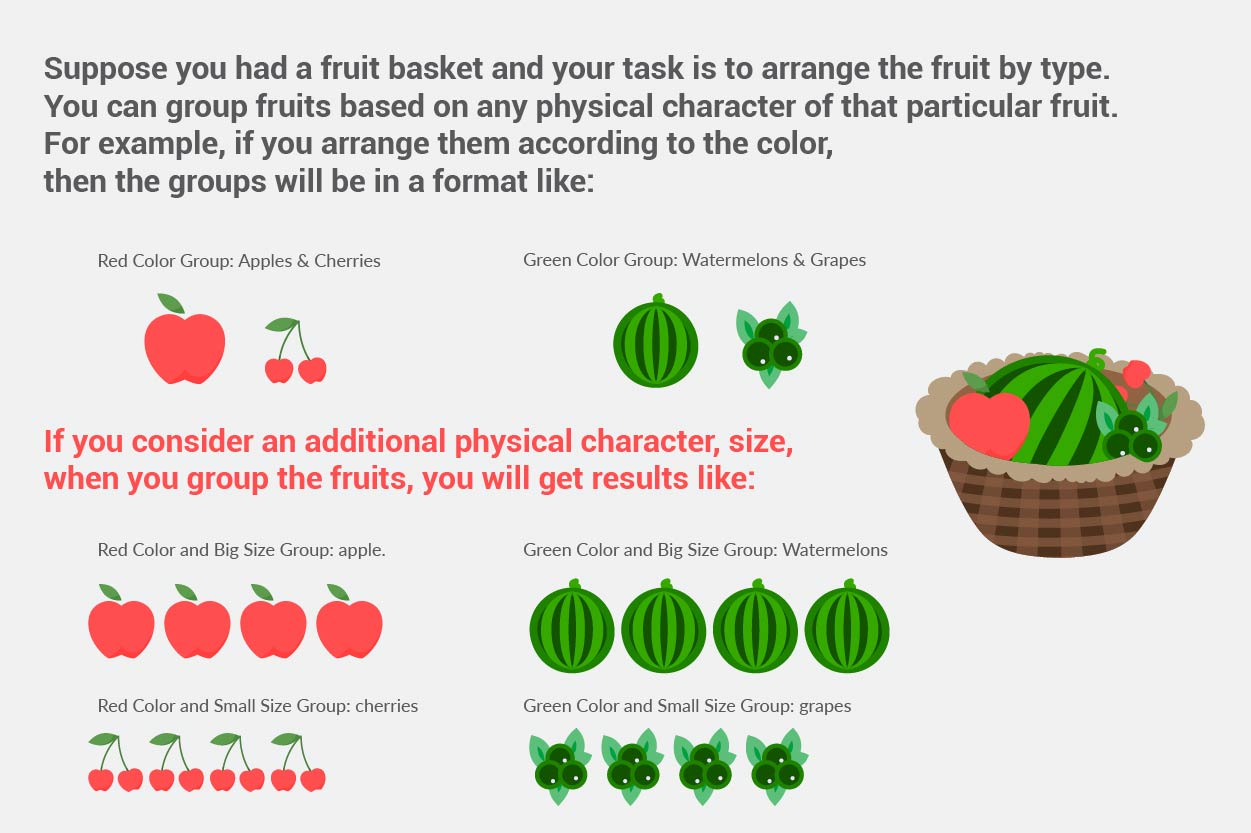
**Clustering** adalahproses pengelompokan sekumpulan objek sedemikian rupa sehingga objek dalam kelompok yang sama (*cluster*) lebih mirip satu sama lain daripada dengan yang ada di kelompok lainnya.



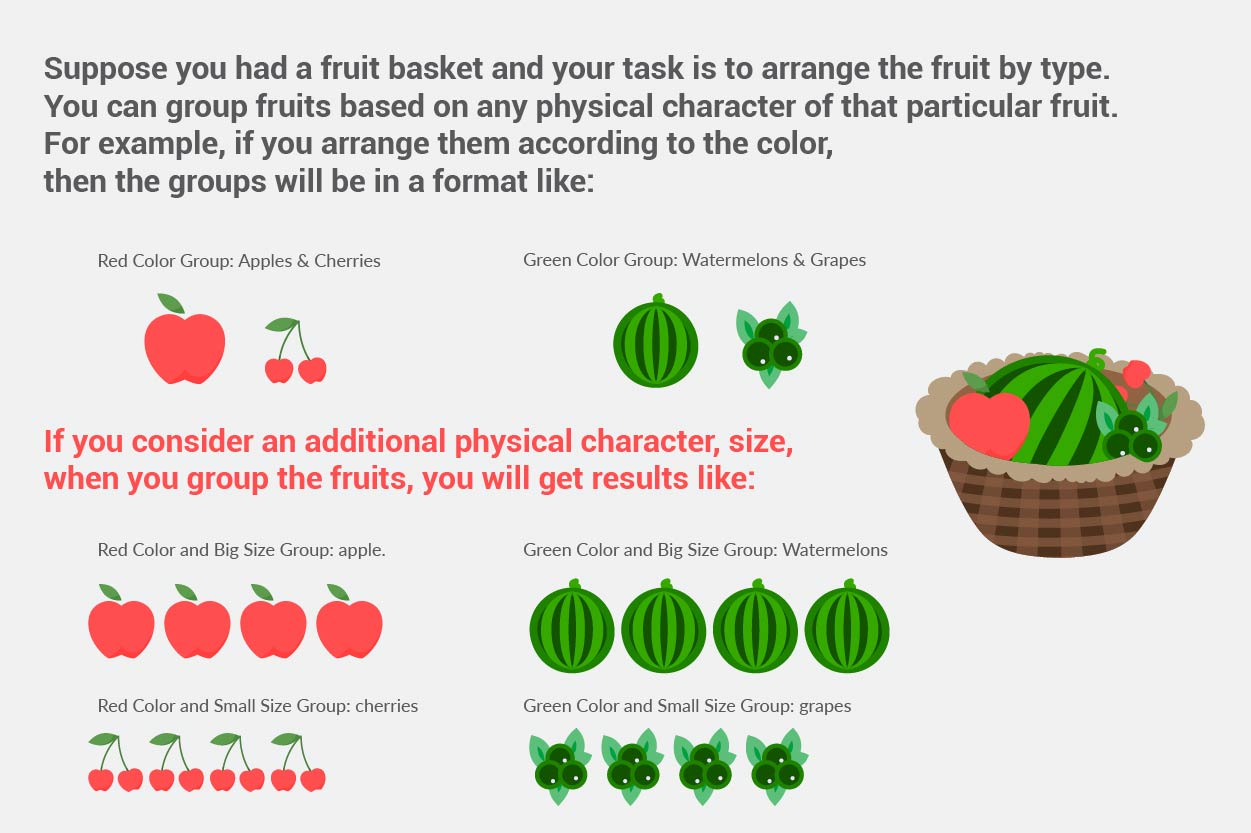
Misal, kita mempunyai sekeranjang buah-buahan yang berisi 4 jenis buah yang berbeda yaitu, buah ceri, apel, semangka, dan anggur.



Jika kita melakukan *clustering* terhadap buah-buahan ini, maka kita bisa membuat 2 *cluster* berdasarkan warna buah tersebut yaitu *cluster* buah berwarna merah dan *cluster* buah berwarna hijau.



Lalu jika kita menambahkan atribut lain yaitu ukuran buah, maka kita dapat membuat 4 *cluster* yaitu, *cluster* buah merah dan besar, *cluster* dan buah merah kecil, *cluster* buah hijau dan besar, *cluster* buah hijau dan kecil.



Jadi, dapat dilihat bahwa anggota *cluster* lebih mirip satu sama lain daripada dengan yang ada di kelompok lainnya. Karena kemiripan antar objek adalah hal yang penting dalam *clustering* maka kita perlu tahu cara untuk mengukur kemiripan antar objek atau yang biasa disebut ***Similarity Measure***.

**Similarity Measure**

Berikut ini merupakan beberapa cara untuk mengukur jarak antara dua objek:

1. **Euclidean**

*Euclidean distance* adalah perhitungan jarak secara garis lurus antara dua buah objek. Dua objek memiliki tingkat kemiripan yang lebih tinggi jika jarak antara kedua objek tersebut kecil, sebaliknya, jarak yang besar menunjukan tingkat kemiripan yang lebih rendah. *Euclidean distance* dapat dihitung menggunakan formula di bawah ini:

**d (x, y)**

Jarak antara x dan y

**n**

Jumlah atribut

**x­i**

Hasil dari atribut – i dari objek x

**yi**

Hasil dari atribut – i dari objek y

**Contoh**:

Misalkan kita ingin menghitung *Euclidean distance* antara 2 objek berikut ini:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objek** | **Attribute 1** | **Attribute 2** | **Attribute 3** | **Attribute 4** |
| A | 5 | 3.2 | 1.2 | 0.2 |
| B | 7.7 | 3 | 6.1 | 2.3 |

Maka jarak antara objek A dengan B adalah 5.979.

1. **Cosine**

*Cosine similarity* adalah cara mengukur kemiripan dari dua objek dengan menghitung sudut kosinus antara kedua objek tersebut. Dua vektor dengan orientasi yang sama memiliki kesamaan kosinus 1, dua vektor pada 90° memiliki kesamaan kosinus 0, dan dua vektor yang bertolak belakang memiliki kesamaan kosinus -1.

**cos (x, y)**

*Cosine similarity* antara objek x dan y

**n**

Jumlah atribut dari objek x atau y

**xi**

Nilai dari atribut ke-i dari objek x

**yi**

Nilai dari atribut ke-i dari objek y

Contoh:

|  |  |  |
| --- | --- | --- |
| **Objek** | **X** | **Y** |
| A | -1 | 1 |
| B | -1 | -1 |
| C | 1 | -1 |

Dapat dilihat dari hasil perhitungan kosinus bahwa semakin tinggi hasilnya maka jarak antar objek semakin dekat (A, A) dan (A, B), tetapi semakin rendah hasilnya maka jarak antar objek semakin jauh (A, C).

1. **Jaccard**

Jaccard mengukur kesamaan antara dua himpunan dengan melihat anggota yang sama dan anggota *distinct* diantaranya.

**J (A, B)**

Jarak antara A and B

Jumlah anggota yang sama antara A dan B

Jumlah anggota *distinct* antara A dan B

Contoh:

|  |  |  |  |
| --- | --- | --- | --- |
| **Objek** | **Pekerjaan** | **Menikah** | **Pendidikan** |
| A | Petani | Tidak | SMA |
| B | Dokter | Ya | S1 |
| C | Dokter | Tidak | S1 |

1. = { }

= {Petani, Tidak, SMA, Dokter, Ya, S1}

1. = {Tidak}

= {Petani, Dokter, Tidak, SMA, S1}

1. = {Dokter, S1}

= {Dokter, Ya, Tidak, S1}

Dapat dilihat jika semakin besar hasilnya maka semakin mirip objek tersebut.

**Algoritma Clustering**

Ada beberapa algoritma yang dapat digunakan untuk melakukan *clustering* yaitu:

1. **K-Means**

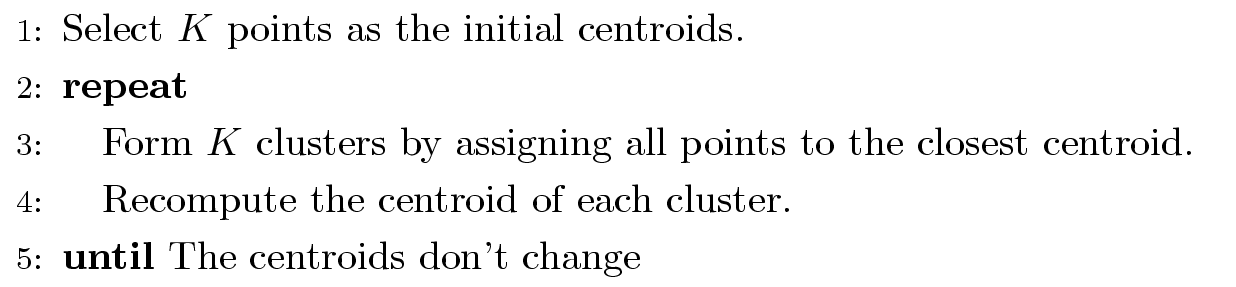
Algoritma *k-means* mendefinisikan ***centroid*** dari sebuah *cluster* sebagai nilai rata-rata dari titik-titik di dalam *cluster*. K-Means akan menggunakan parameter **k** sebagai jumlah *cluster* yang akan dihasilkan dari data. Algoritma K-Means secara acak memilih **k** dari objek-objek dalam dataset **D**, masing-masing objek mewakili sebuah cluster mean atau pusat.

Algoritma K-Means akan melakukan beberapa langkah di bawah ini untuk setiap iterasi:

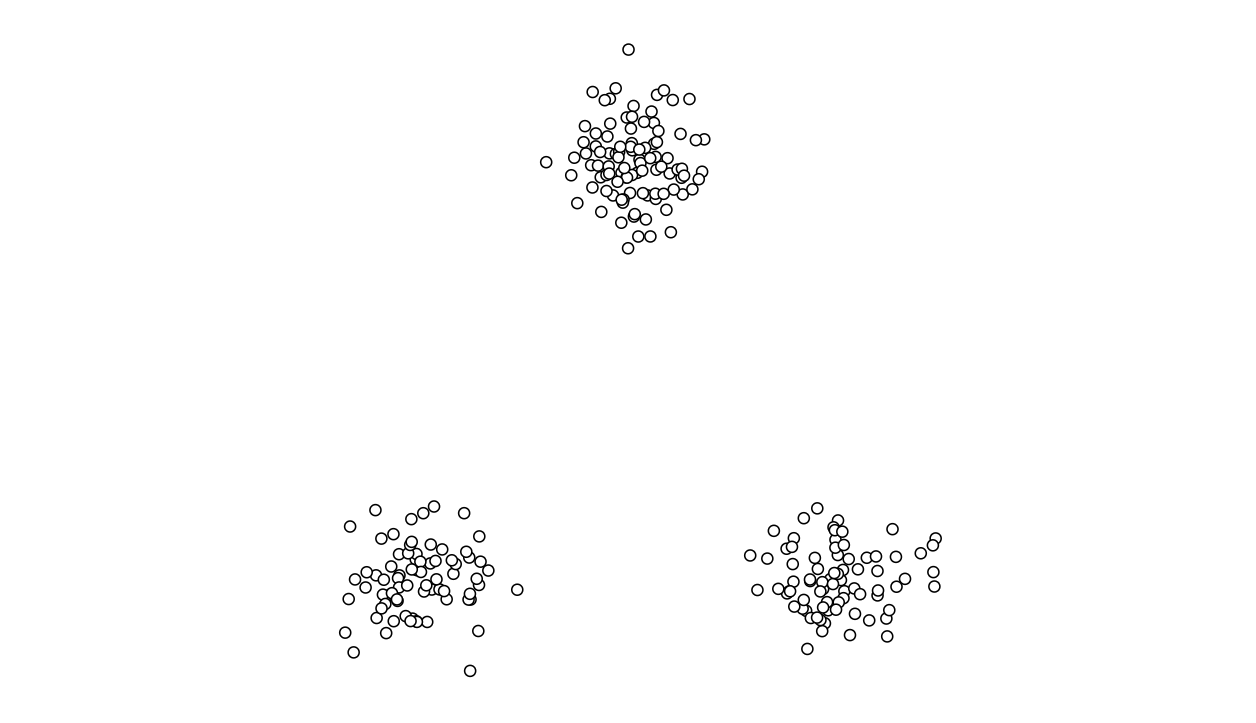
* Tiap objek dimasukan ke *cluster* yang paling mirip, berdasarkan **Euclidean distance** antara objek dan **cluster mean**.
* Tiap *cluster* akan menghitung **mean baru** menggunakan anggota *cluster* dengan menghitung nilai rata-rata setiap atribut dari anggota *cluster* tersebut.
* Bandingkan **cluster mean** terakhir sebelum iterasi saat ini dengan **mean baru** yang dihitung dalam iterasi saat ini. Jika **mean baru** **berubah** dari **cluster mean pada langkah petama** kemudian kembali ke langkah pertama.

Dari langkah algoritma K-Means, ada kasus bahwa **iteration** tidak akan pernah berhenti karena **member** dari setiap ***cluster*** **tidak pernah *convergent*** dalam setiap iterasi, yang menyebabkan ***centroid*** **selalu berubah**.

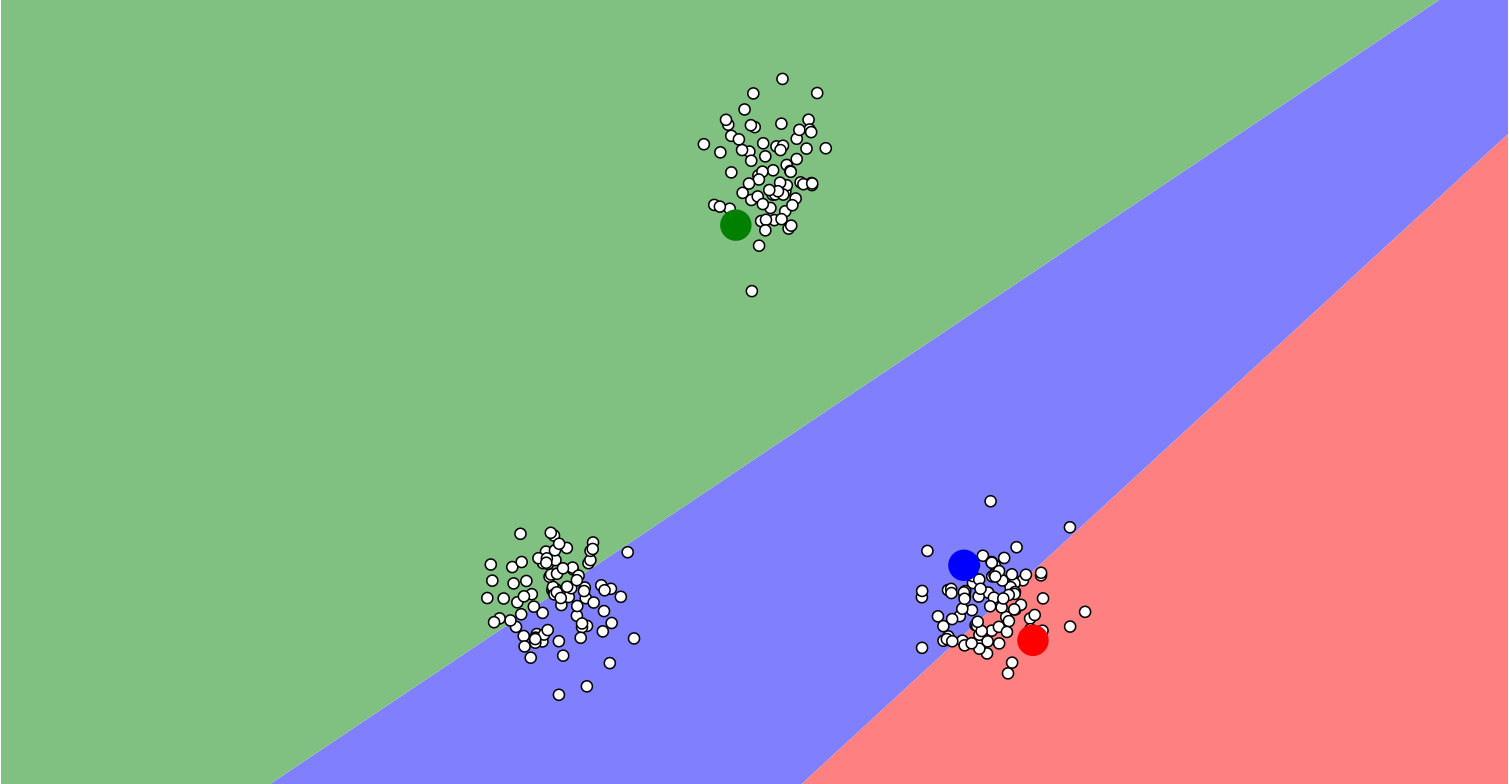
**Algoritma**



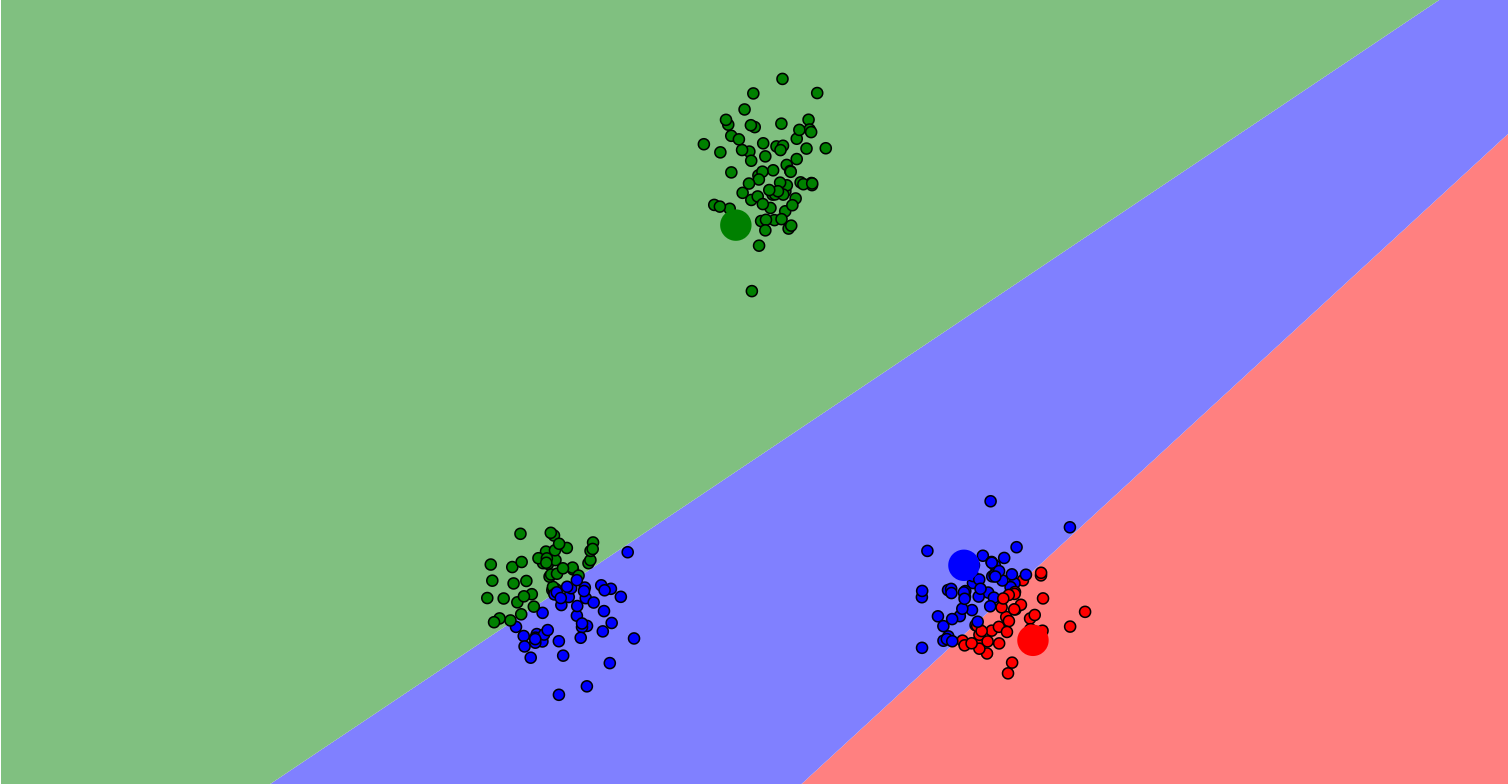
Misal kita mempunyai data sebagai berikut dan ingin membuat 3 *cluster* (**K = 3**).



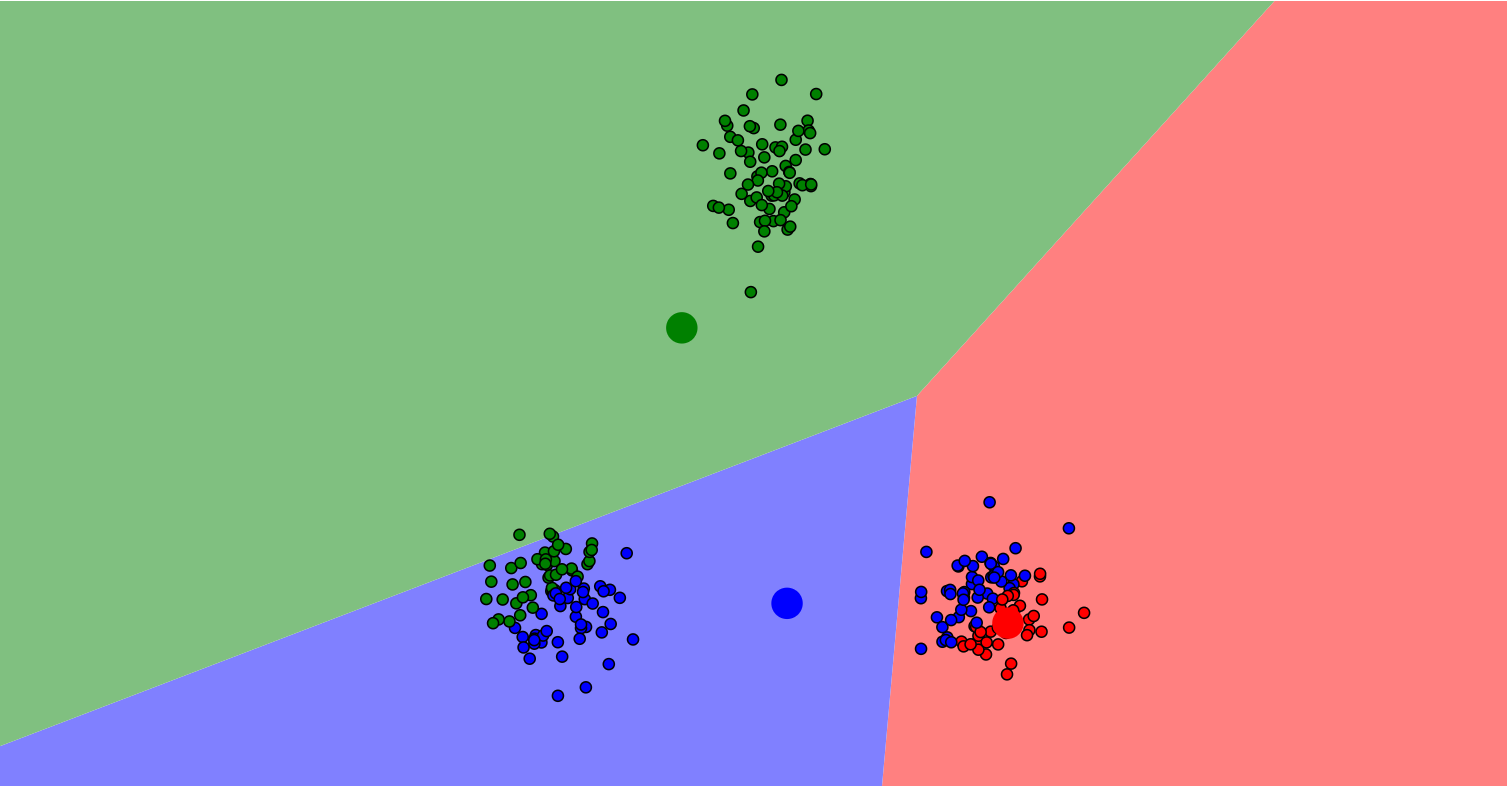
Lalu kita memilih objek sebanyak **k** (3) untuk dijadikan *centroid*. Pada gambar dibawah *centroid* ditandai dengan titik besar dan dibedakan dengan warna.



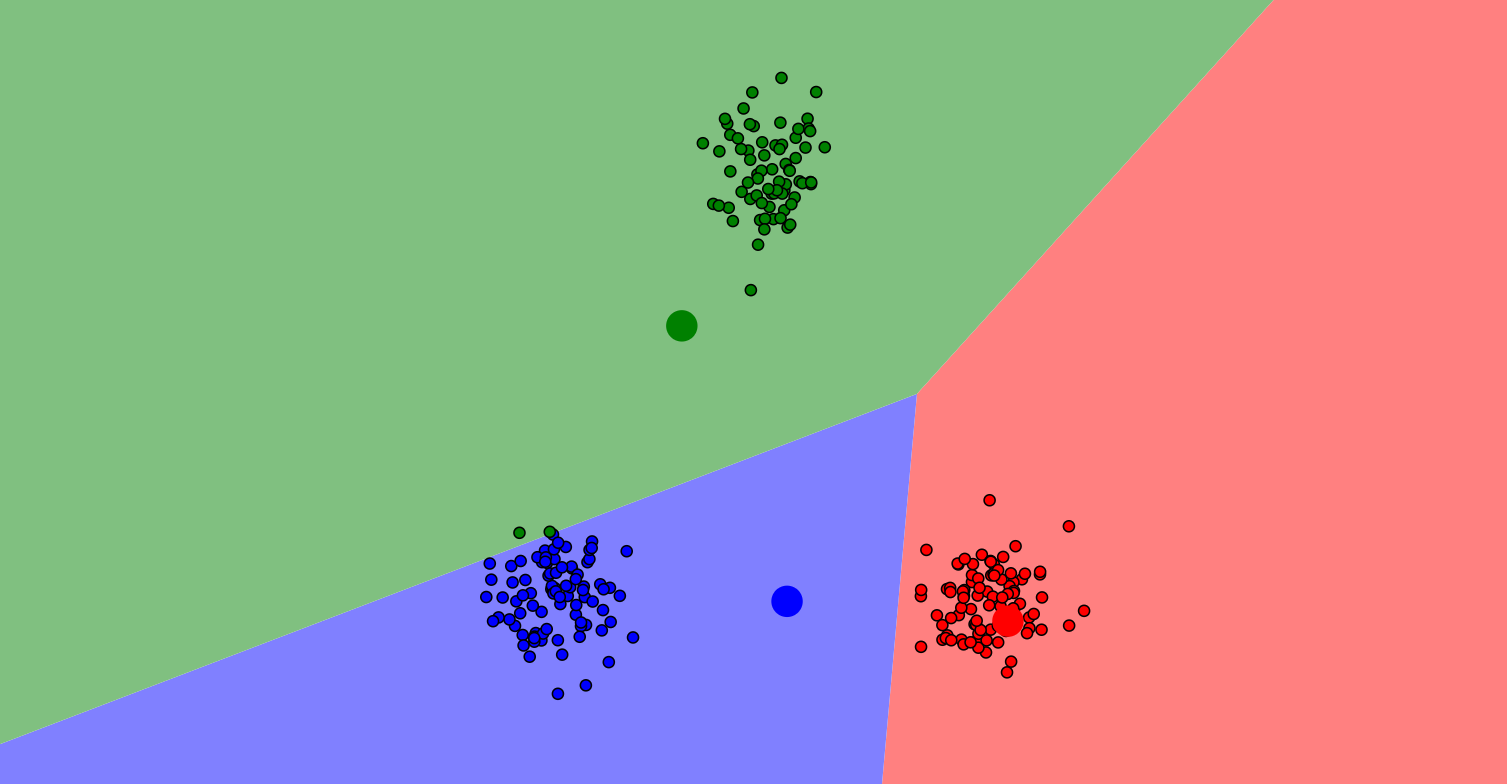
Langkah pertama, tiap objek dimasukan ke *cluster* yang memiliki jarak paling dekat.



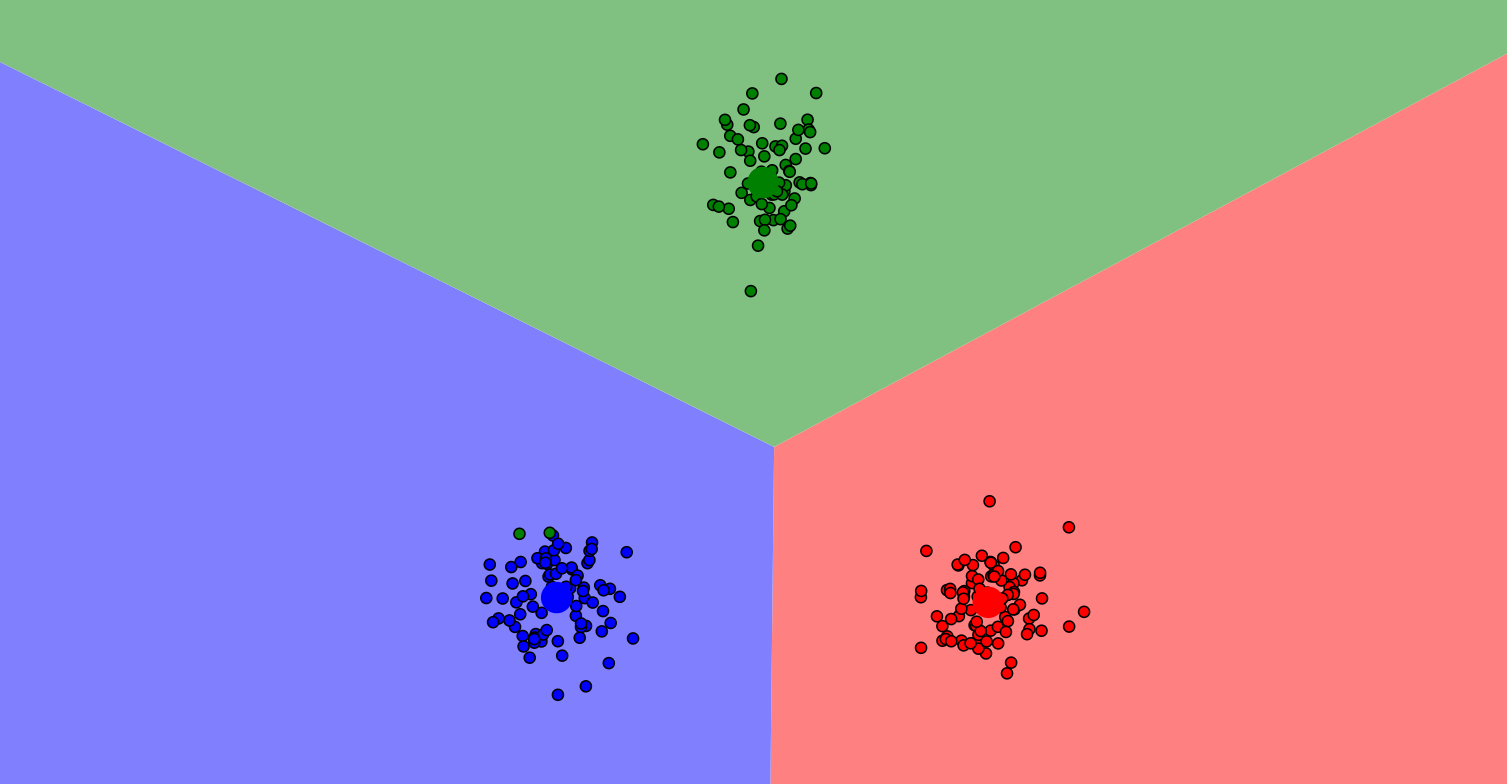
Langkah kedua, tiap *cluster* akan mengupdate *centroid* menggunakan anggota *cluster* dengan menghitung nilai rata-rata setiap atribut dari anggota *cluster* tersebut. Jika *centroid* berubah kembali ke langkah pertama. Dapat dilihat bahwa *centroid* bergeser dan ada objek yang berpindah *cluster*.



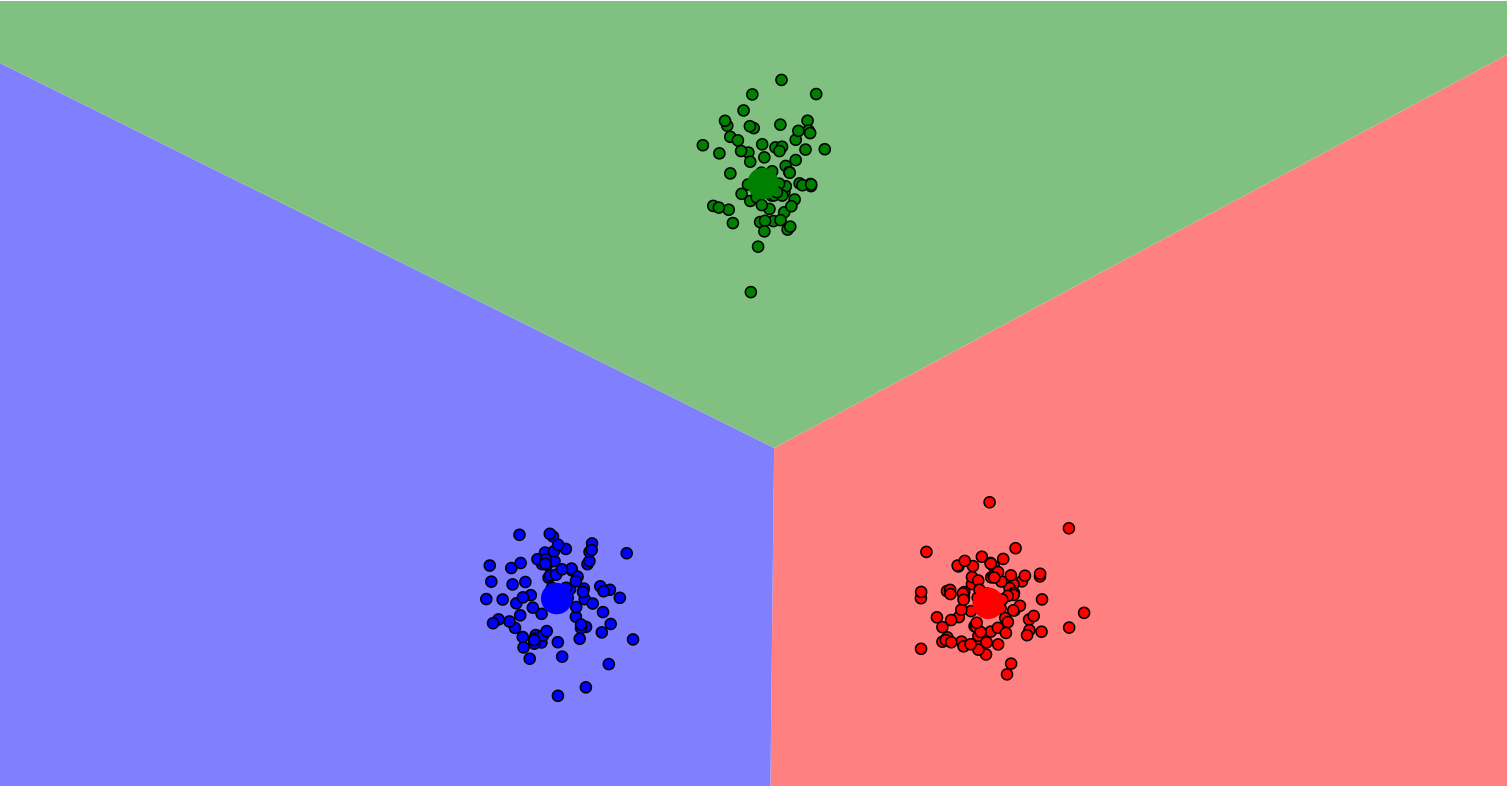
Langkah ketiga, tiap objek dimasukan ke *cluster* yang memiliki jarak paling dekat.



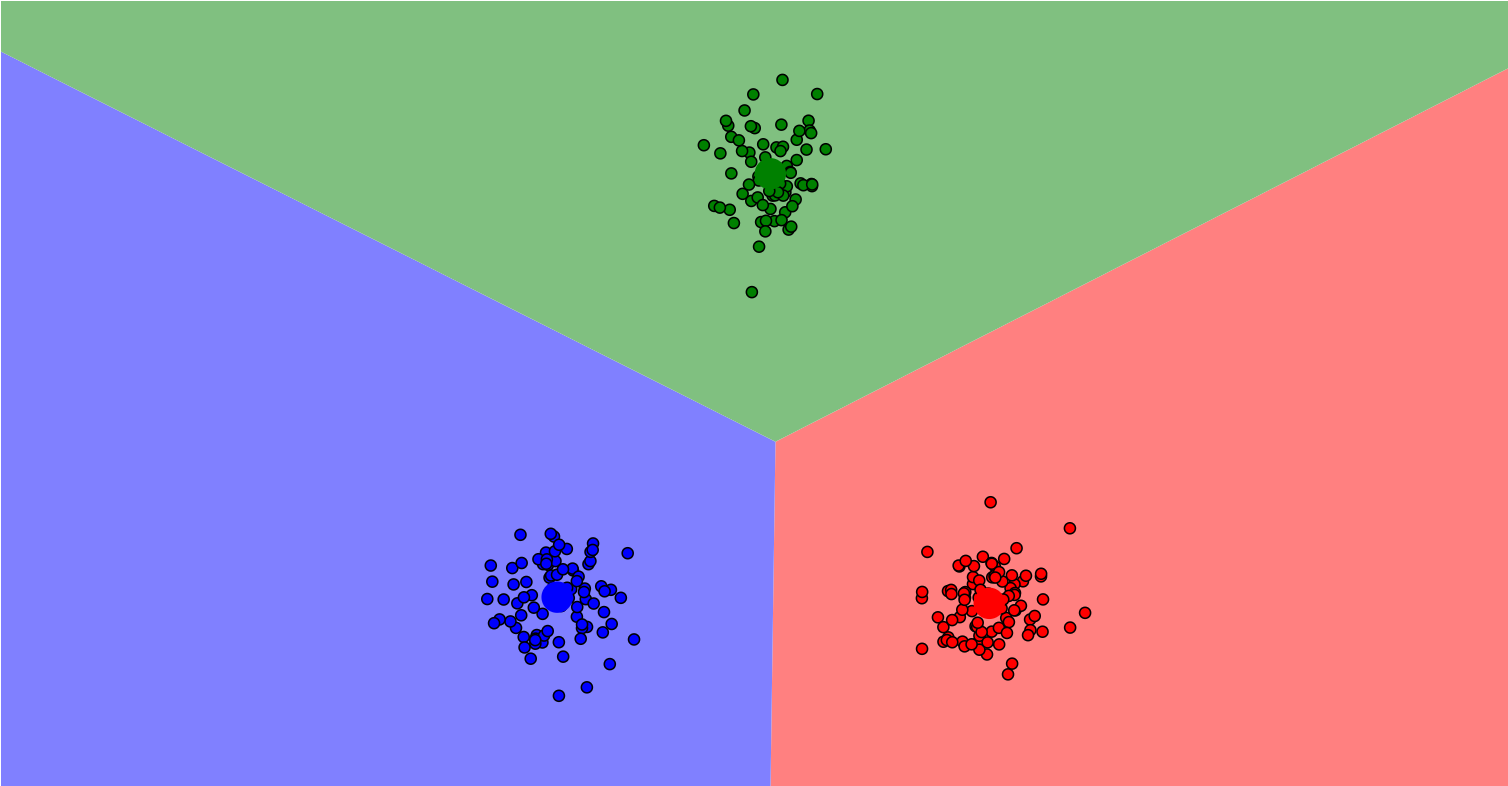
Langkah keempat, tiap *cluster* akan mengupdate *centroid* menggunakan anggota *cluster* dengan menghitung nilai rata-rata setiap atribut dari anggota *cluster* tersebut. Jika *centroid* berubah kembali ke langkah pertama.



Langkah kelima, tiap objek dimasukan ke *cluster* yang memiliki jarak paling dekat.



Langkah keenam, tiap *cluster* akan mengupdate *centroid* menggunakan anggota *cluster* dengan menghitung nilai rata-rata setiap atribut dari anggota *cluster* tersebut. Karena *centroid* lama sama dengan *centroid* baru, maka algoritmanya berhenti disini dan 3 cluster telah terbentuk.



**Contoh**

Sampel dari data Iris:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **attr1** | **attr2** | **attr3** | **attr4** |
| id\_36 | 5.0 | 3.2 | 1.2 | 0.2 |
| id\_46 | 4.8 | 3.0 | 1.4 | 0.3 |
| id\_86 | 6.0 | 3.4 | 4.5 | 1.6 |
| id\_96 | 5.7 | 3.0 | 4.2 | 1.2 |
| id\_136 | 7.7 | 3.0 | 6.1 | 2.3 |

* Langkah 1

Tentukan **K** berapa banyak *cluster* yang akan dihasilkan dan pilih secara acak **K** objek sebagai ***initial centroid***. Dalam hal ini, kita mendefinisikan **k = 3** dan memilih objek **k** menjadi *centroid* awal.

* id\_36 => C1
* id\_86 => C2
* id\_136 => C3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cluster** | **attr1** | **attr2** | **attr3** | **attr4** |
| C1 | 5.0 | 3.2 | 1.2 | 0.2 |
| C2 | 6.0 | 3.4 | 4.5 | 1.6 |
| C3 | 7.7 | 3.0 | 6.1 | 2.3 |

* Langkah 2
  + Hitung jarak dari masing-masing objek ke setiap *centroid* *cluster* menggunakan **Euclidean Distance**. **Distance function** yang diwarnai adalah jarak terdekat antara objek dan setiap *centroid* *cluster*. Persamaan berikut digunakan untuk menghitung jarak berdasarkan **Euclidean Distance**.

0

0

* + Tetapkan setiap objek ke kelompok yang memiliki jarak terdekat ke objek.

Cluster-1 (C1): {id\_36, id\_46}

Cluster-2 (C2): {id\_86, id\_96}

Cluster-3 (C3): {id\_136}

* + Hitung *centroid* baru untuk setiap *cluster* berdasarkan anggotanya. Untuk menghitung *centroid* baru setiap *cluster*, hitung nilai rata-rata untuk setiap atribut dari anggota *cluster*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cluster** | **attr1** | **attr2** | **attr3** | **attr4** |
| C1 | = (5.0 + 4.8) / 2  = 4.9 | = (3.2 + 3.0) / 2  = 3.1 | = (1.2 + 1. 4) / 2  = 1.3 | = (0.2 + 0.3) / 2  = 0.25 |
| C2 | = (6.0 + 5.7) / 2  = 5.85 | = (3.4 + 3.0) / 2  = 3.2 | = (4.5 + 4.2) / 2  = 4.35 | = (1.6 + 1.2) / 2  = 1.4 |
| C3 | = 7.7 | = 3.0 | = 6.1 | = 2.3 |

* + Bandingkan mean baru dengan mean lama

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Old Mean** | | | | |
| **Cluster** | **attr1** | **attr2** | **attr3** | **attr4** |
| C1 | 5.0 | 3.2 | 1.2 | 0.2 |
| C2 | 6.0 | 3.4 | 4.5 | 1.6 |
| C3 | 7.7 | 3.0 | 6.1 | 2.3 |
| **New Mean** | | | | |
| **Cluster** | **attr1** | **attr2** | **attr3** | **attr4** |
| C1 | 4.9 | 3.1 | 1.3 | 0.25 |
| C2 | 5.85 | 3.2 | 4.35 | 1.4 |
| C3 | 7.7 | 3.0 | 6.1 | 2.3 |

Karena *centroid* baru masih berubah dari *centroid* sebelumnya maka kita akan melakukan iterasi lagi untuk menentukan *centroid* baru.

* Langkah 3
  + Hitung jarak dari masing-masing objek ke setiap *centroid* *cluster* menggunakan **Euclidean Distance**. **Distance function** yang diwarnai adalah jarak terdekat antara objek dan setiap *centroid* *cluster*.

0

0

* + Tetapkan setiap objek ke kelompok yang memiliki jarak terdekat ke objek.

Cluster-1 (c1): {id\_36, id\_46}

Cluster-2 (c2): {id\_86, id\_96}

Cluster-3 (c3): {id\_136}

* + Hitung *centroid* baru untuk setiap *cluster* berdasarkan anggotanya. Untuk menghitung *centroid* baru setiap *cluster*, hitung nilai rata-rata untuk setiap atribut dari anggota *cluster*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cluster** | **attr1** | **attr2** | **attr3** | **attr4** |
| C1 | = (5.0 + 4.8) / 2  = 4.9 | = (3.2 + 3.0) / 2  = 3.1 | = (1.2 + 1. 4) / 2  = 1.3 | = (0.2 + 0.3) / 2  = 0.25 |
| C2 | = (6.0 + 5.7) / 2  = 5.85 | = (3.4 + 3.0) / 2  = 3.2 | = (4.5 + 4.2) / 2  = 4.35 | = (1.6 + 1.2) / 2  = 1.4 |
| C3 | = 7.7 | = 3.0 | = 6.1 | = 2.3 |

* + Bandingkan mean baru dengan mean lama

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Old Mean** | | | | |
| **Cluster** | **attr1** | **attr2** | **attr3** | **attr4** |
| C1 | 4.9 | 3.1 | 1.3 | 0.25 |
| C2 | 5.85 | 3.2 | 4.35 | 1.4 |
| C3 | 7.7 | 3.0 | 6.1 | 2.3 |
| **New Mean** | | | | |
| **Cluster** | **attr1** | **attr2** | **attr3** | **attr4** |
| C1 | 4.9 | 3.1 | 1.3 | 0.25 |
| C2 | 5.85 | 3.2 | 4.35 | 1.4 |
| C3 | 7.7 | 3.0 | 6.1 | 2.3 |

Karena *centroid* baru tidak berubah dari *centroid* sebelumnya maka iterasi akan dihentikan dan *centroid* baru akan menjadi pusat dari masing-masing kelas.

* Langkah 4

Hasil *clustering*

**Centroid Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cluster** | **attr1** | **attr2** | **attr3** | **attr4** |
| C1 | 4.9 | 3.1 | 1.3 | 0.25 |
| C2 | 5.85 | 3.2 | 4.35 | 1.4 |
| C3 | 7.7 | 3.0 | 6.1 | 2.3 |

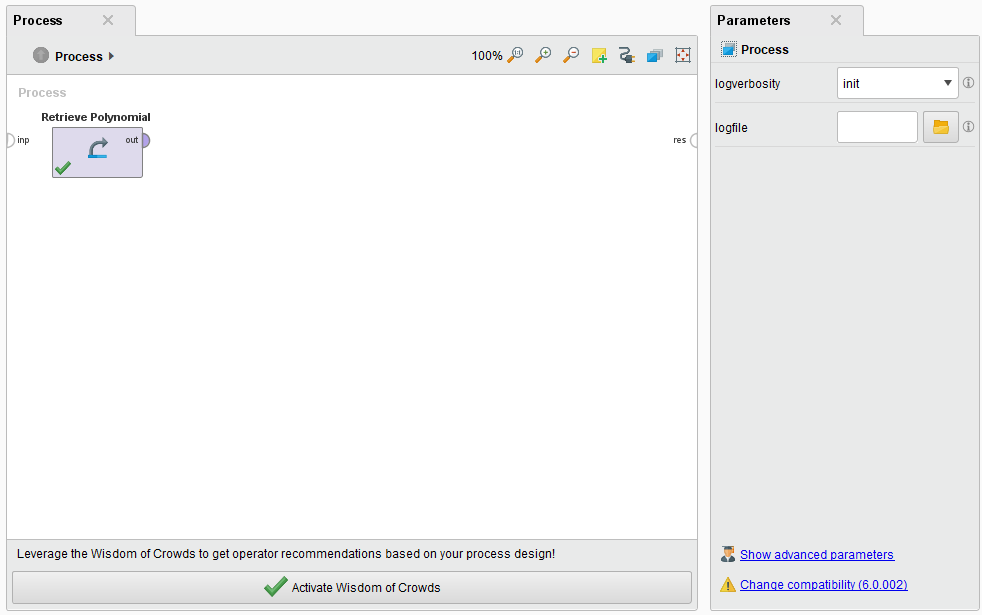
**Clustered Set**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **attr1** | | **attr2** | | **attr3** | **attr4** | **Cluster** |
| id\_36 | 5.0 | | 3.2 | | 1.2 | 0.2 | C1 |
| id\_46 | 4.8 | 3.0 | | 1.4 | | 0.3 | C1 |
| id\_86 | 6.0 | 3.4 | | 4.5 | | 1.6 | C2 |
| id\_96 | 5.7 | 3.0 | | 4.2 | | 1.2 | C2 |
| id\_136 | 7.7 | 3.0 | | 6.1 | | 2.3 | C3 |

**Practicum in RapidMiner**

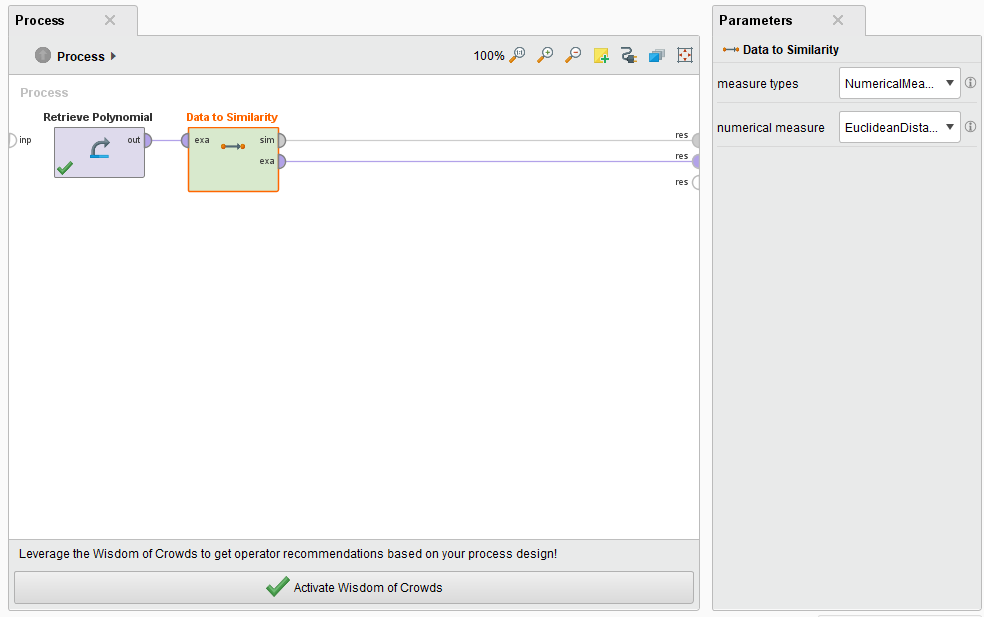
**Similarity Measure**

* Langkah 1

Gunakan **Polynomial Data** dalam folder sampel. 

* Langkah 2

Gunakan operator **Data to Similarity** untuk mengukur kesamaan antar data. Pada contoh ini kita akan menggunakan algoritma *Euclidean Distance* untuk mengukur kesamaan antar data pada data Polynomial.



Catatan:

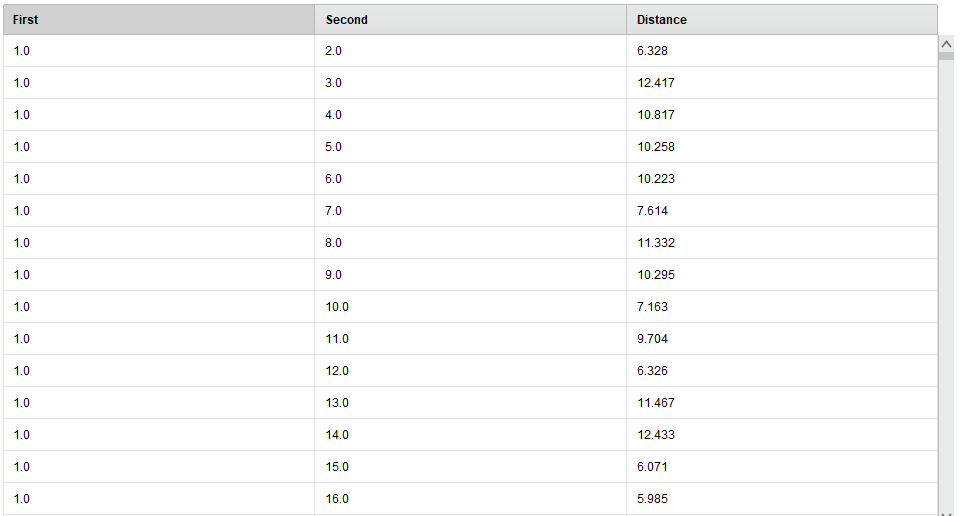
**Measure Types**

Tipe data yang ingin di hitung

**Measure**

Algoritma yang ingin dipakai untuk menghitung

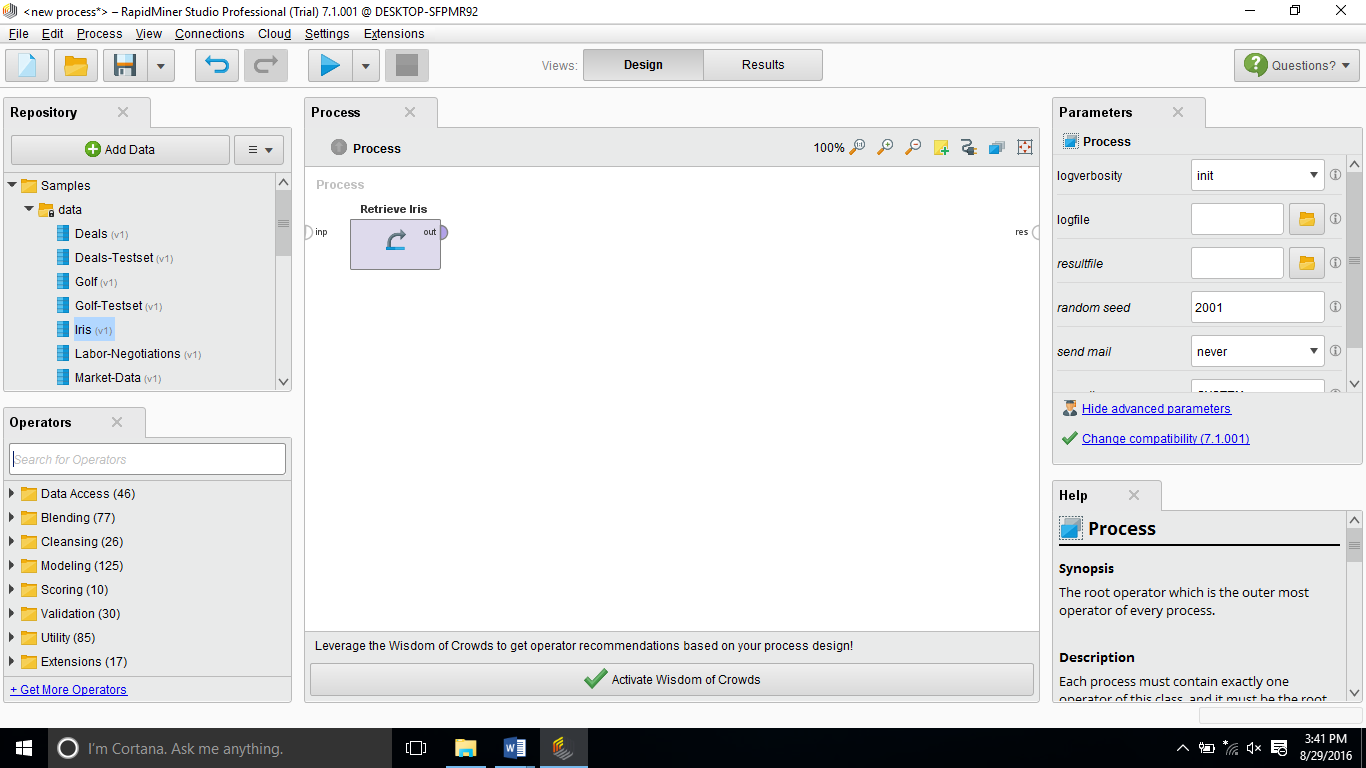
Hasil:



**K-Means**

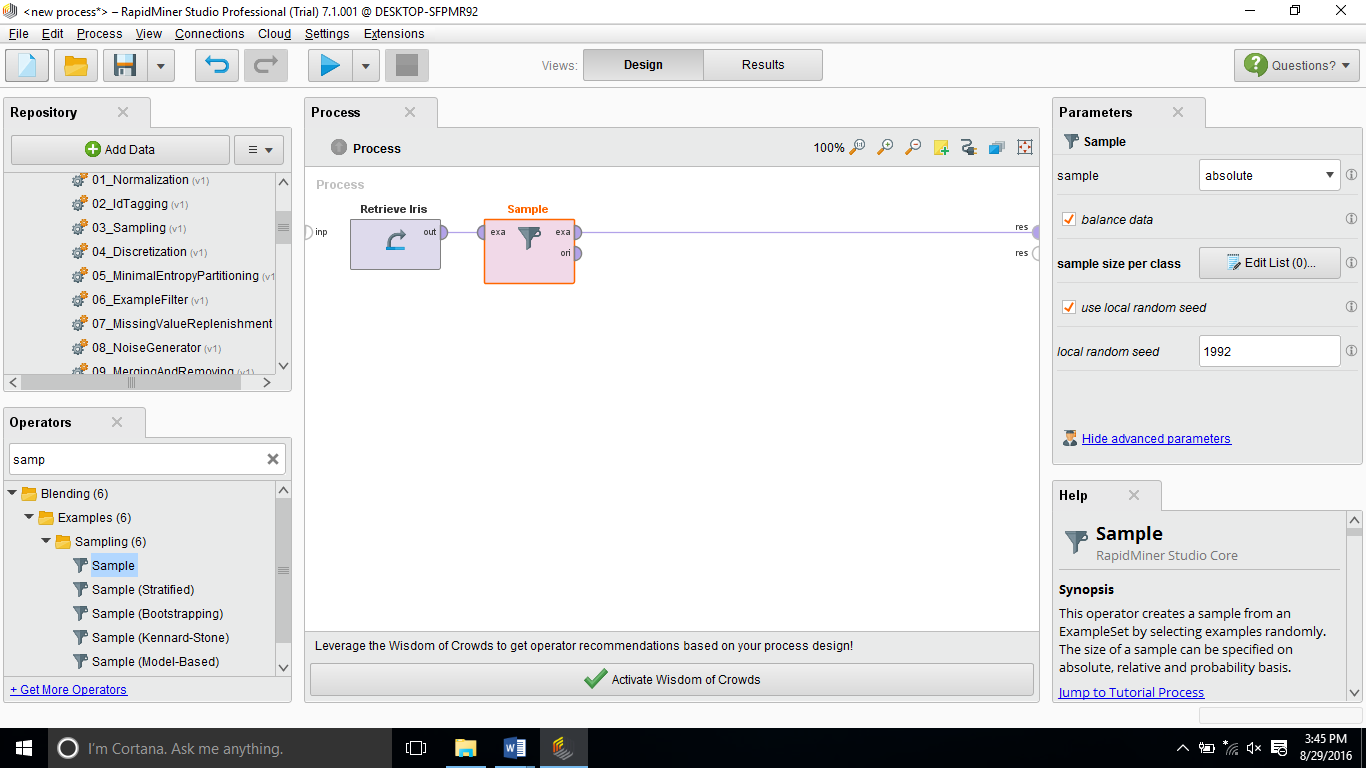
* Langkah 1

Gunakan **Iris Data** dalam folder sampel.



* Langkah 2

Gunakan operator **Sample** untuk memilih data secara acak berdasarkan parameter.



Catatan:

**Sample**

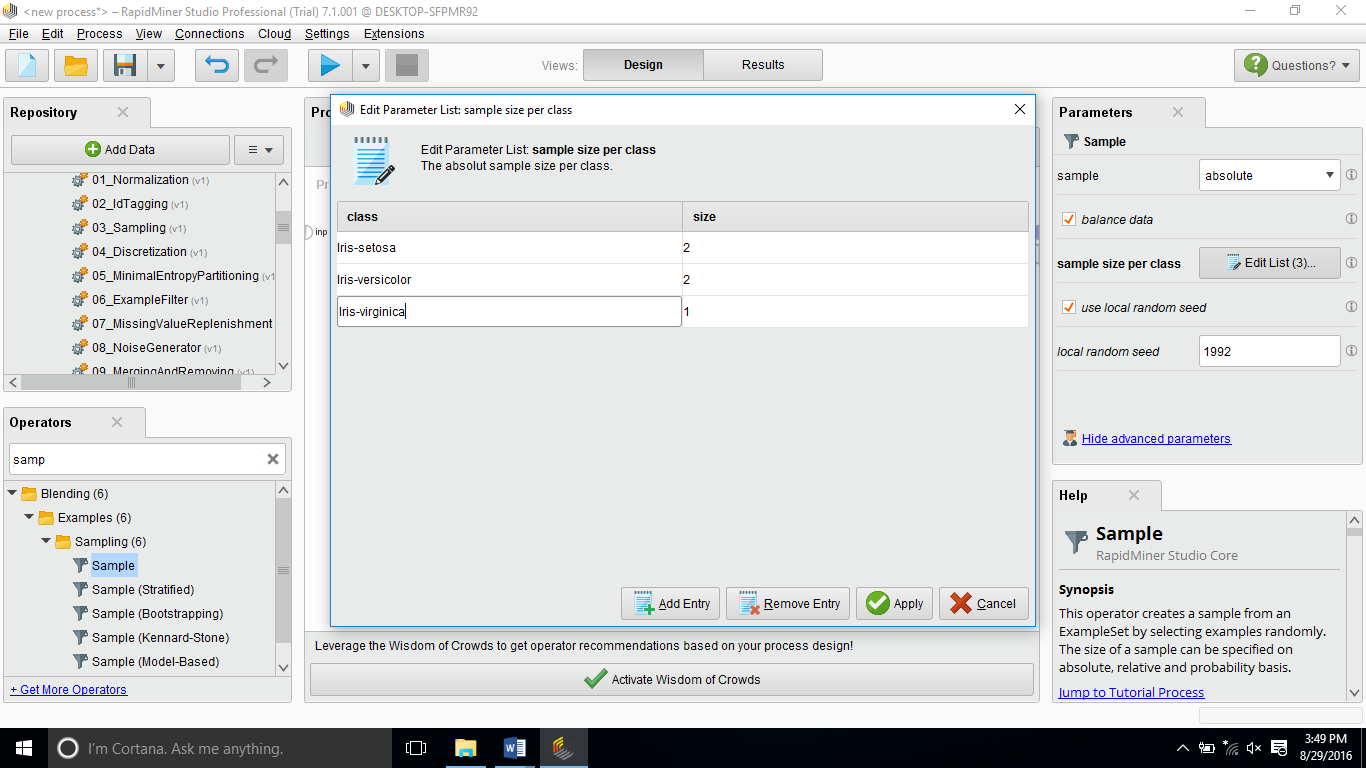
Menentukan berapa jumlah data yang ditentukan.

**Balance data**

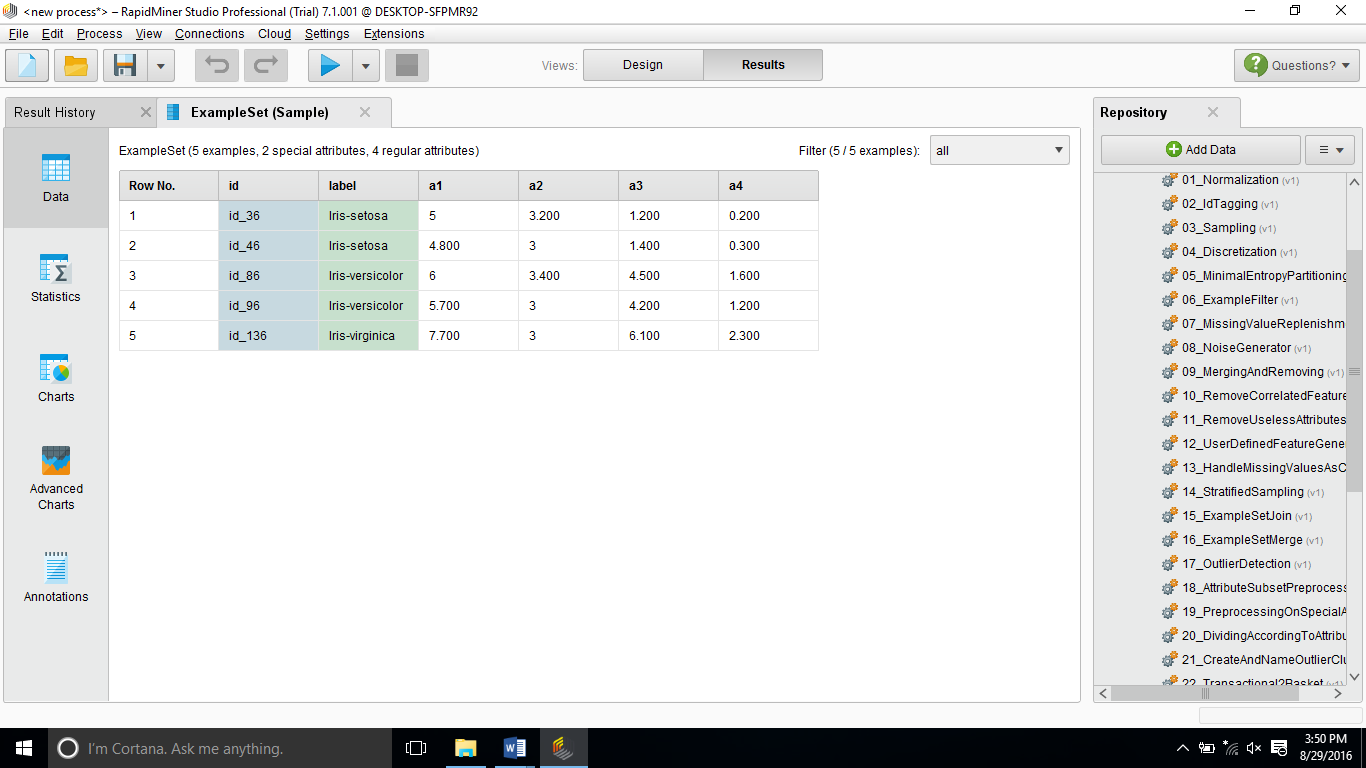
Dapatkan data saldo untuk setiap kelas.

* Langkah 3

Klik ukuran sampel per parameter kelas di operator sampel dan tetapkan jumlah data yang Anda inginkan.

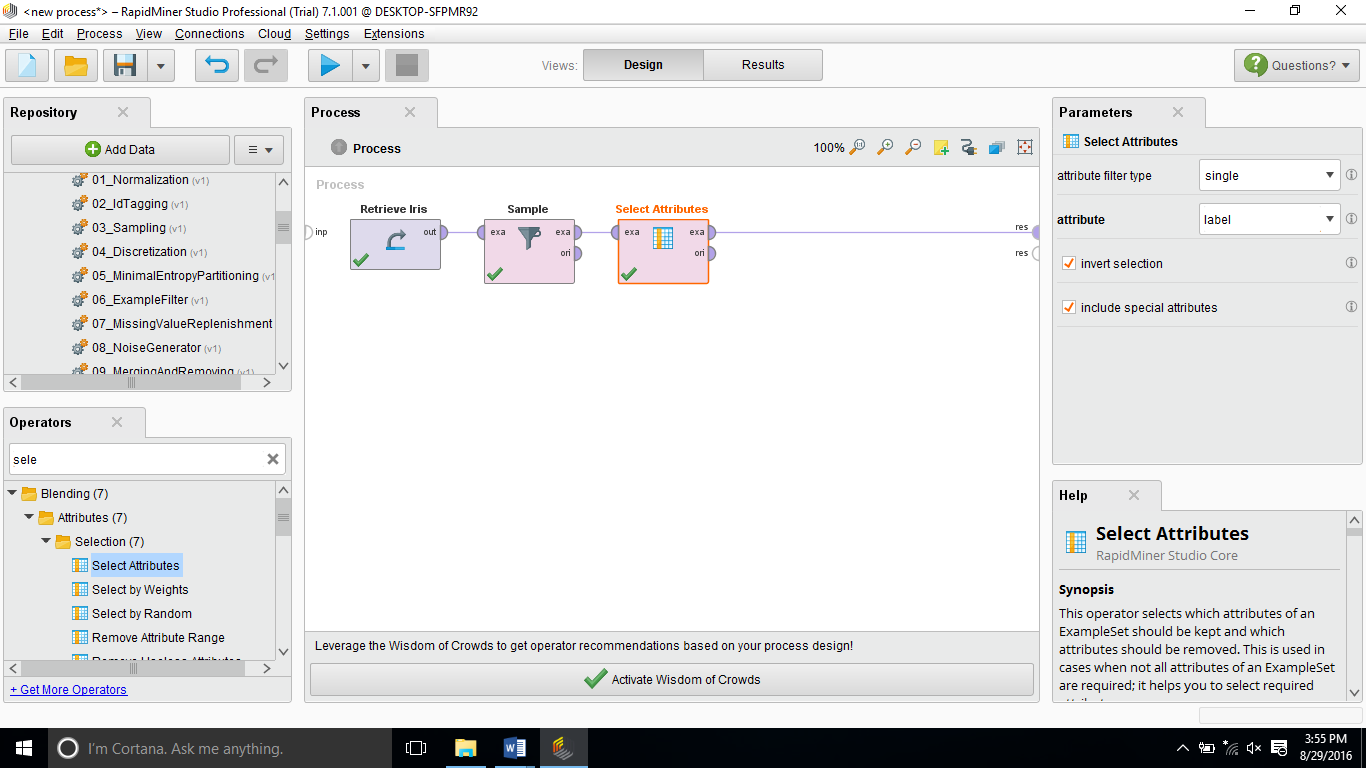


Hasil



* Langkah 4

Gunakan operator **Select Attribute** untuk memilih semua atribut kecuali label.



Catatan

**Attribute Filter Type**

Jenis filter untuk memilih atribut data.

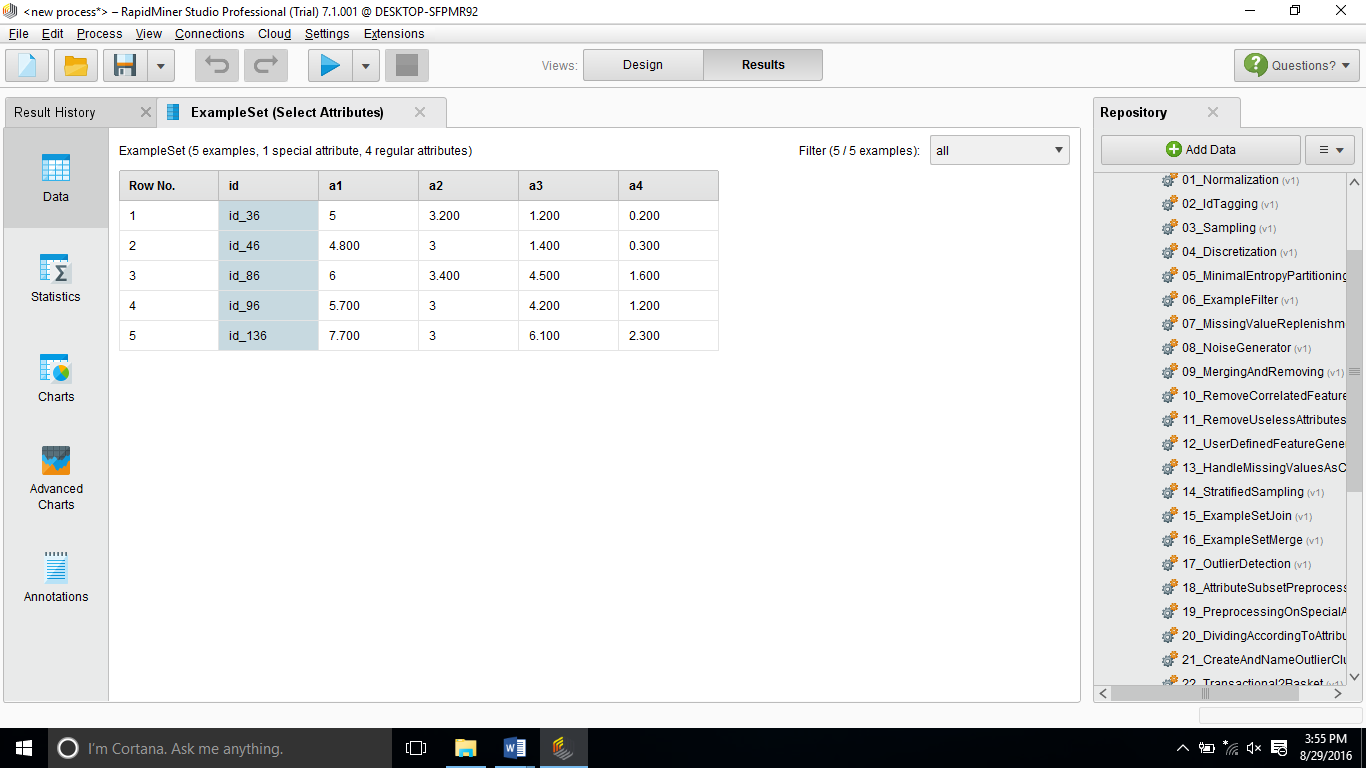
**Attribute**

Nama atribut untuk memilih.

**Invert selection**

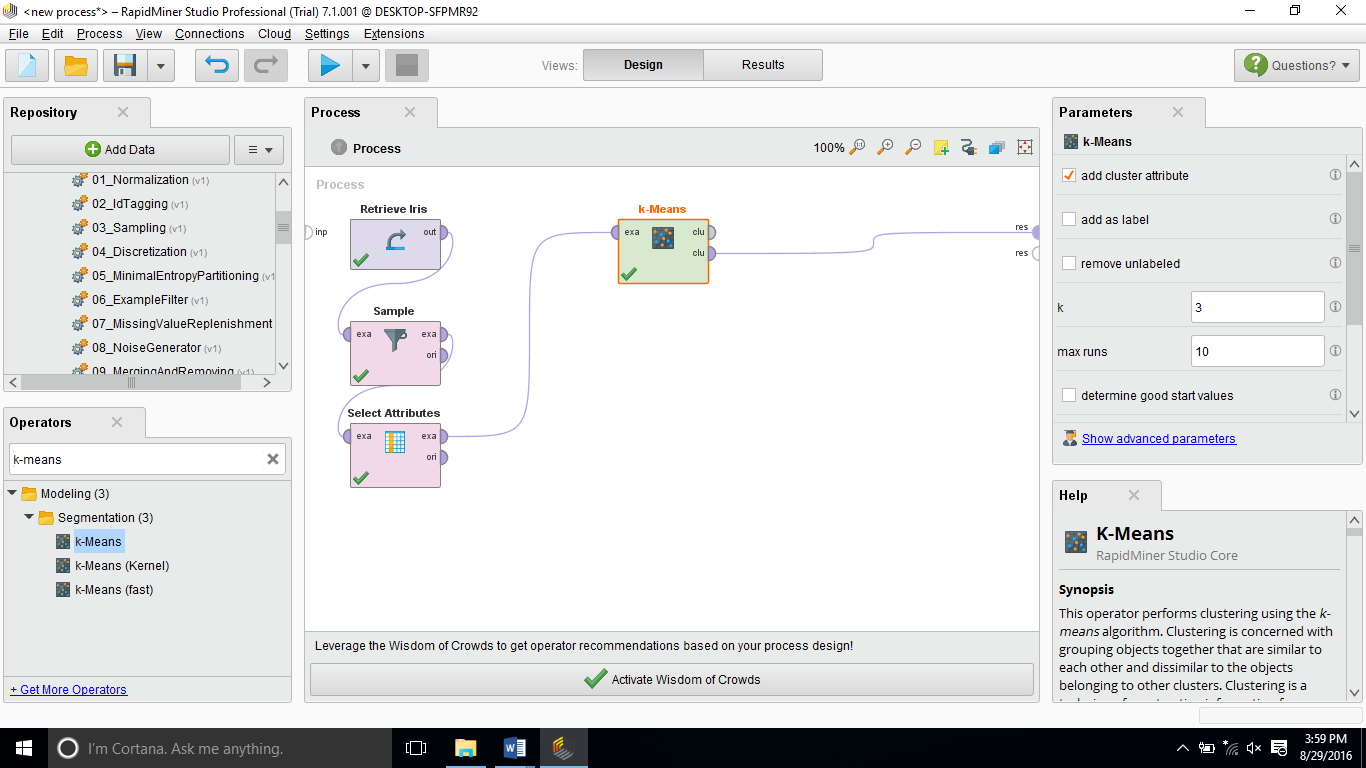
Untuk membalikkan seleksi.

Hasil:



* Langkah 5

Gunakan operator **K-Means** untuk mengelompokkan data menggunakan algoritma k-Means.



Catatan

**K**

Berapa banyak *cluster* yang akan dihasilkan.

Hasil:

