

LAPORAN TUGAS *K-MEANS*
Diajukan untuk Memenuhi Tugas Pada Mata Kuliah
Machine Learning



Dipersiapkan oleh :

Raden Muhammad Imam - 1301154106

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
2018

1. Deskripsi Kasus

Bangunlah sebuah sistem ***K-Means*** merupakan salah satu algoritma *clustering*. Dimana algoritma ini memiliki tujuan untuk membagi data menjadi beberapa kelompok. Algoritma ini menerima masukan berupa data tanpa label kelas. Hal ini berbeda dengan *supervised learning* yang menerima masukan berupa vektor $(x_1, y_1), (x_2, y_2), \dots, (x_i, y_i)$, dimana x_i merupakan data dari suatu data pelatihan dan y_i merupakan label kelas untuk x_i . Pada algoritma pembelajaran ini, komputer mengelompokkan sendiri data-data yang menjadi masukannya tanpa mengetahui terlebih dulu target kelasnya. Pembelajaran ini termasuk dalam *Unsupervised Learning*. Masukan yang diterima adalah data atau objek dan k buah kelompok (*cluster*) yang diinginkan. Algoritma ini akan mengelompokkan data atau objek ke dalam k buah kelompok tersebut. Pada setiap *cluster* terdapat titik pusat (*centroid*) yang merepresentasikan *cluster* tersebut.

2. Rancangan Metode Yang Digunakan

i. Algoritma

Berdasarkan ketentuan dari tugas 2, untuk membangun sistem ***K-Means Clustering*** yang bertujuan untuk mengelompokkan data ke dalam *cluster*. Algoritma dari ***K-Means Clustering*** sebagai berikut:

- 1) Pilih K buah titik *centroid* secara acak
- 2) Kelompokkan data sehingga terbentuk K buah *cluster* dengan titik *centroid* dari setiap *cluster* merupakan titik *centroid* yang telah dipilih sebelumnya
- 3) Perbaharui nilai titik *centroid*
- 4) Ulangi langkah kedua dan ketiga sampai nilai dari titik *centroid* tidak lagi berubah

Proses pengelompokkan data ke dalam suatu *cluster* dapat dilakukan dengan cara menghitung jarak terdekat dari suatu data ke sebuah titik *centroid*. Perhitungan jarak *Minkowski* dapat digunakan untuk menghitung jarak antar 2 buah data. Rumus untuk menghitung jarak tersebut adalah:

$$d(x_i, x_j) = (|x_{i1} - x_{j1}|^g + |x_{i2} - x_{j2}|^g + \dots + |x_{ip} - x_{jp}|^g)^{1/g}$$

Dimana:

- $g = 1$, untuk menghitung jarak *Manhattan*
- $g = 2$, untuk menghitung jarak *Euclidean*
- $g = \infty$, untuk menghitung jarak *Chebychev*
- x_i, x_j = Dua buah data yang akan dihitung jaraknya
- p = Dimensi dari sebuah data

Pembaharuan suatu titik *centroid* dapat dilakukan dengan rumus berikut:

$$\mu_k = \frac{1}{N_k} \sum_{q=1}^{N_k} x_q$$

Dimana:

- μ_k = Titik centroid dari cluster ke-K
- N_k = Banyaknya data pada cluster ke-K
- x_q = Data ke-q pada cluster ke-K

ii. Source Code

Pada kali ini menggunakan Python 3.6 dengan Spyder sebagai IDE

```
Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\yaden\Downloads\1301154106_K-Means\1301154106_K-Means.py
temp.py 1301154106_K-Means.py
1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4 from sklearn.cluster import KMeans
5
6 #Membuat Array Untuk Data
7 test1 = []
8 test2 = []
9 labtest = []
10
11 train1 = []
12 train2 = []
13 labtrain = []
14
15 #Rumus Euclidean Distance (Mencari Jarak Nilai Dua titik)
16 def euclead ( a , b , c , d ):
17     return math.sqrt ( pow ( b - a , 2 ) + pow ( d - c , 2 ) )
18
19 #Load Data Uji
20 file = open ( 'TestsetTugas2.txt' , 'r' )
21 for line in file:
22     a = line.split()
23     #Menambah Elemen Pada List
24     test1.append ( float ( a [ 0 ] ) )
25     test2.append ( float ( a [ 1 ] ) )
26     labtest.append ( int ( 0 ) )
27
28 #Load Data Latih
29 file = open ( 'TrainsetTugas2.txt' , 'r' )
30 for line in file:
31     a = line.split()
32     #Menambah Elemen Pada List
33     train1.append ( float ( a [ 0 ] ) )
34     train2.append ( float ( a [ 1 ] ) )
35     labtrain.append ( int ( 0 ) )
```

```
Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\yaden\Downloads\1301154106_K-Means\1301154106_K-Means.py
temp.py 1301154106_K-Means.py
36
37 #Mencari Nilai Tengah Setiap Kelas
38 Kelas1X = train1 [ 82 ]
39 Kelas1Y = train2 [ 82 ]
40 Kelas2X = train1 [ 18 ]
41 Kelas2Y = train2 [ 18 ]
42 Kelas3X = train1 [ 581 ]
43 Kelas3Y = train2 [ 581 ]
44 Kelas4X = train1 [ 245 ]
45 Kelas4Y = train2 [ 245 ]
46 Kelas5X = train1 [ 457 ]
47 Kelas5Y = train2 [ 457 ]
48 Kelas6X = train1 [ 357 ]
49 Kelas6Y = train2 [ 357 ]
50 Kelas7X = train1 [ 303 ]
51 Kelas7Y = train2 [ 303 ]
52
53 #Loop Untuk Menentukan Masuk Ke Dalam Kelas Mana
54 for i in range ( 0 , train1.__len__() ):
55     Hasilkelas1 = euclead ( Kelas1X , train1 [ i ] , Kelas1Y , train2 [ i ] )
56     Hasilkelas2 = euclead ( Kelas2X , train1 [ i ] , Kelas2Y , train2 [ i ] )
57     Hasilkelas3 = euclead ( Kelas3X , train1 [ i ] , Kelas3Y , train2 [ i ] )
58     Hasilkelas4 = euclead ( Kelas4X , train1 [ i ] , Kelas4Y , train2 [ i ] )
59     Hasilkelas5 = euclead ( Kelas5X , train1 [ i ] , Kelas5Y , train2 [ i ] )
60     Hasilkelas6 = euclead ( Kelas6X , train1 [ i ] , Kelas6Y , train2 [ i ] )
61     Hasilkelas7 = euclead ( Kelas7X , train1 [ i ] , Kelas7Y , train2 [ i ] )
62     minimum = min ( Hasilkelas1 , Hasilkelas2 , Hasilkelas3 , Hasilkelas4 , Hasilkelas5 , Hasilkelas6 , Hasilkelas7 )
63
64     if ( minimum == Hasilkelas1 ):
65         Kelas1X = 0.5 * Kelas1X + 0.5 * train1 [ i ]
66         Kelas1Y = 0.5 * Kelas1Y + 0.5 * train2 [ i ]
67         labtrain [ i ] = 1
68     elif ( minimum == Hasilkelas2 ):
69         Kelas2X = 0.5 * Kelas2X + 0.5 * train1 [ i ]
70         Kelas2Y = 0.5 * Kelas2Y + 0.5 * train2 [ i ]
71         labtrain [ i ] = 2
```

```
Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\raden\Downloads\1301154106_K-Means\1301154106_K-Means.py
temp.py 1301154106_K-Means.py
72 elif ( minimum == Hasilkelas3 ):
73     Kelas3X = 0.5 * Kelas3X + 0.5 * train1 [ i ]
74     Kelas3Y = 0.5 * Kelas3Y + 0.5 * train2 [ i ]
75     labtrain [ i ] = 3
76 elif ( minimum == Hasilkelas4 ):
77     Kelas4X = 0.5 * Kelas4X + 0.5 * train1 [ i ]
78     Kelas4Y = 0.5 * Kelas4Y + 0.5 * train2 [ i ]
79     labtrain [ i ] = 4
80 elif ( minimum == Hasilkelas5 ):
81     Kelas5X = 0.5 * Kelas5X + 0.5 * train1 [ i ]
82     Kelas5Y = 0.5 * Kelas5Y + 0.5 * train2 [ i ]
83     labtrain [ i ] = 5
84 elif ( minimum == Hasilkelas6 ):
85     Kelas6X = 0.5 * Kelas6X + 0.5 * train1 [ i ]
86     Kelas6Y = 0.5 * Kelas6Y + 0.5 * train2 [ i ]
87     labtrain [ i ] = 6
88 elif ( minimum == Hasilkelas7 ):
89     Kelas7X = 0.5 * Kelas7X + 0.5 * train1 [ i ]
90     Kelas7Y = 0.5 * Kelas7Y + 0.5 * train2 [ i ]
91     labtrain [ i ] = 7
92
93 #Perulangan Untuk Menentukan Data Test Masuk Ke Kelas Mana
94 for i in range(0, test1.__len__()):
95     Hasilkelas1 = euclead ( Kelas1X , test1 [ i ] , Kelas1Y , test2 [ i ] )
96     Hasilkelas2 = euclead ( Kelas2X , test1 [ i ] , Kelas2Y , test2 [ i ] )
97     Hasilkelas3 = euclead ( Kelas3X , test1 [ i ] , Kelas3Y , test2 [ i ] )
98     Hasilkelas4 = euclead ( Kelas4X , test1 [ i ] , Kelas4Y , test2 [ i ] )
99     Hasilkelas5 = euclead ( Kelas5X , test1 [ i ] , Kelas5Y , test2 [ i ] )
100    Hasilkelas6 = euclead ( Kelas6X , test1 [ i ] , Kelas6Y , test2 [ i ] )
101    Hasilkelas7 = euclead ( Kelas7X , test1 [ i ] , Kelas7Y , test2 [ i ] )
102    minimum = min ( Hasilkelas1 , Hasilkelas2 , Hasilkelas3 , Hasilkelas4 , Hasilkelas5 , Hasilkelas6 , Hasilkelas7 )
103
104    if ( minimum == Hasilkelas1 ):
105        Kelas1X = 0.5 * Kelas1X + 0.5 * test1 [ i ]
106        Kelas1Y = 0.5 * Kelas1Y + 0.5 * test2 [ i ]
107        labtest [ i ] = 1
```

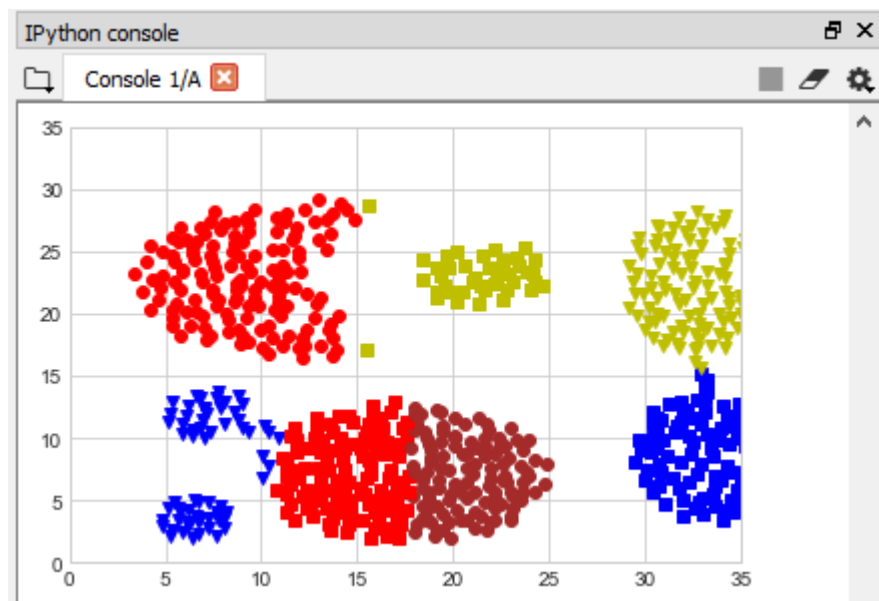
```
Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\raden\Downloads\1301154106_K-Means\1301154106_K-Means.py
temp.py 1301154106_K-Means.py
108 elif ( minimum == Hasilkelas2 ):
109     Kelas2X = 0.5 * Kelas2X + 0.5 * test1 [ i ]
110     Kelas2Y = 0.5 * Kelas2Y + 0.5 * test2 [ i ]
111     labtest [ i ] = 2
112 elif ( minimum == Hasilkelas3 ):
113     Kelas3X = 0.5 * Kelas3X + 0.5 * test1 [ i ]
114     Kelas3Y = 0.5 * Kelas3Y + 0.5 * test2 [ i ]
115     labtest [ i ] = 3
116 elif ( minimum == Hasilkelas4 ):
117     Kelas4X = 0.5 * Kelas4X + 0.5 * test1 [ i ]
118     Kelas4Y = 0.5 * Kelas4Y + 0.5 * test2 [ i ]
119     labtest [ i ] = 4
120 elif ( minimum == Hasilkelas5 ):
121     Kelas5X = 0.5 * Kelas5X + 0.5 * test1 [ i ]
122     Kelas5Y = 0.5 * Kelas5Y + 0.5 * test2 [ i ]
123     labtest [ i ] = 5
124 elif ( minimum == Hasilkelas6 ):
125     Kelas6X = 0.5 * Kelas6X + 0.5 * test1 [ i ]
126     Kelas6Y = 0.5 * Kelas6Y + 0.5 * test2 [ i ]
127     labtest [ i ] = 6
128 elif ( minimum == Hasilkelas7 ):
129     Kelas7X = 0.5 * Kelas7X + 0.5 * test1 [ i ]
130     Kelas7Y = 0.5 * Kelas7Y + 0.5 * test2 [ i ]
131     labtest [ i ] = 7
132
133 for i in range ( 0 , train1.__len__() ):
134     print ( labtrain [ i ] )
135 for i in range ( 0 , test1.__len__() ):
136     print ( labtest [ i ] )
137
138 plt.style.use ( 'seaborn-whitegrid' )
139
140 #Pewarnaan Untuk Masing-Masing Cluster Untuk Membedakannya
141 x = np.array ( list ( zip ( train1 , train2 ) ) ).reshape ( len ( train1 ) , 2 )
142 #Menentukan Warna
143 colors = ( 'b' , 'y' , 'r' , 'brown' , 'b' , 'y' , 'r' )
```

```
Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\yaden\Downloads\1301154106_K-Means\1301154106_K-Means.py
temp.py 1301154106_K-Means.py
138 plt.style.use ( 'seaborn-whitegrid' )
139
140 #Pewarnaan Untuk Masing-Masing Cluster Untuk Membedakannya
141 x = np.array ( list ( zip ( train1 , train2 ) ) ).reshape ( len ( train1 ) , 2 )
142 #Menentukan Warna
143 colors = ( 'b' , 'y' , 'r' , 'brown' , 'b' , 'y' , 'r' )
144 #Menentukan Bentuk
145 markers = ( 'v' , 'v' , 'o' , 'o' , 's' , 's' , 's' )
146
147 #Menggunakan K = 7
148 K = 7
149 Model = KMeans ( n_clusters = K ).fit(x)
150 plt.plot()
151 for i, l in enumerate(Model.labels_):
152     plt.plot ( train1 [ i ] , train2 [ i ] , color = colors [ l ] , marker = markers [ l ] , ls = 'None' )
153     plt.xlim ( [ 0 , 35 ] )
154     plt.ylim ( [ 0 , 35 ] )
155 plt.show()
156
157 x = np.array(list(zip(test1,test2))).reshape(len(test1),2)
158 colors = ( 'r' , 'g' , 'b' , 'cyan' , 'r' , 'g' , 'b' )
159 markers = ( 'v' , 'v' , 'o' , 'o' , 's' , 's' , 's' )
160
161 K = 7
162 Model = KMeans(n_clusters=K).fit(x)
163 plt.plot()
164
165 #Membuat Folder txt Untuk Menampung Hasil Akurasi
166 thefile = open ( 'Akurasi.txt' , 'w' )
167 for item in labtest:
168     thefile.write ( "%s\n" % item )
169
170 for i, l in enumerate(Model.labels_):
171     plt.plot ( test1 [ i ] , test2 [ i ] , color = colors [ l ] , marker = markers [ l ] , ls = 'None' )
172     plt.xlim ( [ 0 , 35 ] )
173     plt.ylim ( [ 0 , 35 ] )
174 plt.show()
```

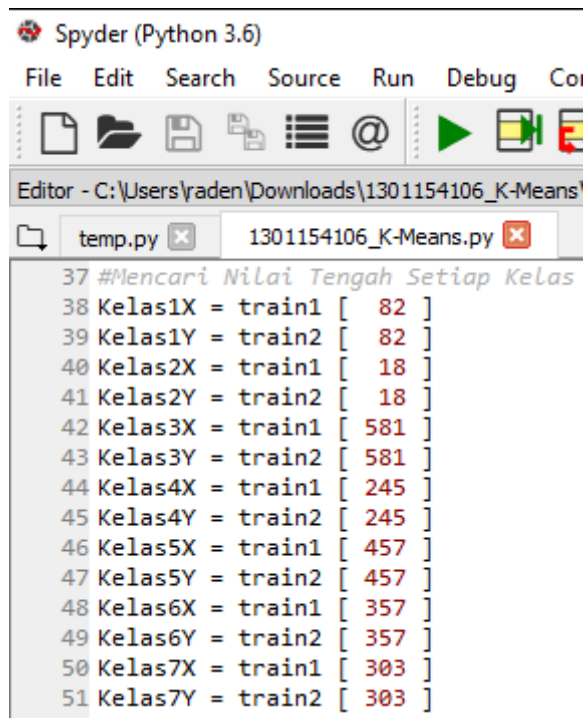
iii. Desain

a. Data Train

Dengan pembuatan grafik penyebaran pertama yaitu terhadap data train dengan jumlah 688 data dan didapatkan grafik seperti ini:



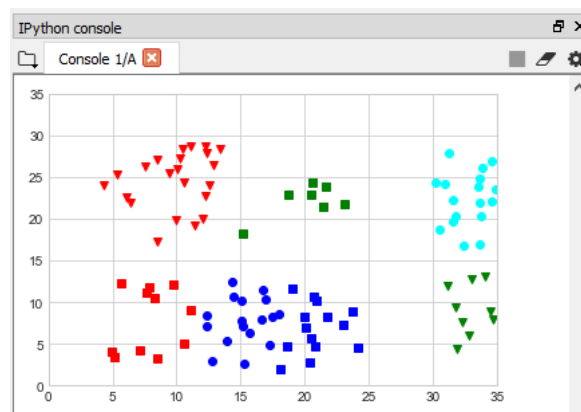
Dimana grafik ini sebagai acuan untuk menentukan *centroid* dengan melihat titik tengah masing-masing *cluster*, dengan asumsi akan memiliki 7 *cluster* dengan warna dan bentuk yang berbeda. Sehingga terdapat titik tengah dari masing-masing *cluster* dimana direpresentasikan dalam bentuk indeks data ke dalam data train yang menunjukkan koordinat dari *centroid* itu sebagai berikut:



```
37 #Mencari Nilai Tengah Setiap Kelas
38 Kelas1X = train1 [ 82 ]
39 Kelas1Y = train2 [ 82 ]
40 Kelas2X = train1 [ 18 ]
41 Kelas2Y = train2 [ 18 ]
42 Kelas3X = train1 [ 581 ]
43 Kelas3Y = train2 [ 581 ]
44 Kelas4X = train1 [ 245 ]
45 Kelas4Y = train2 [ 245 ]
46 Kelas5X = train1 [ 457 ]
47 Kelas5Y = train2 [ 457 ]
48 Kelas6X = train1 [ 357 ]
49 Kelas6Y = train2 [ 357 ]
50 Kelas7X = train1 [ 303 ]
51 Kelas7Y = train2 [ 303 ]
```

b. Data Test

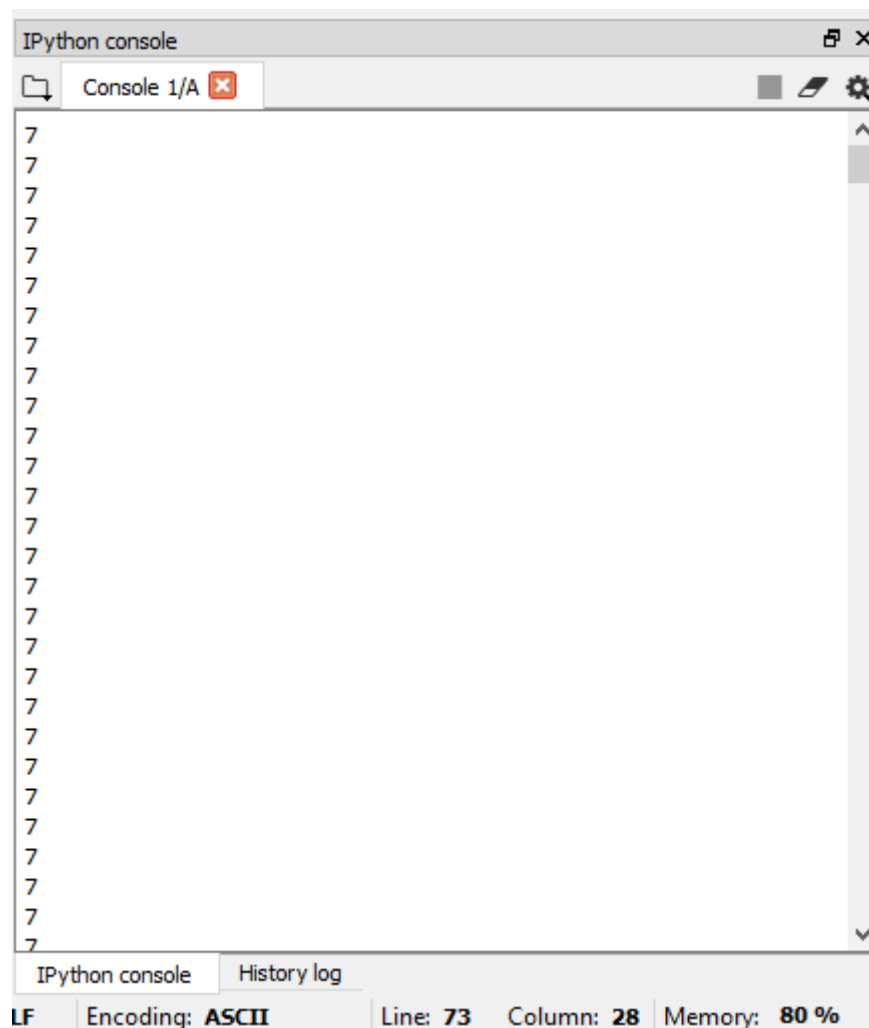
Setelah didapatkan titik centroid dari data train maka selanjutnya akan dilakukan pembuatan grafik terhadap data test berdasarkan dari hasil data train dengan jumlah 100 data. Sehingga didapatkan grafik penyebaran data test seperti berikut:



Hasil dari grafik ini akan dijadikan acuan untuk melakukan *clustering* terhadap 100 data pada data test. Hasil dari clustering dapat di lihat pada file akurasi.txt

3. Screenshot Output Program

Program mengoutputkan hasil dari data train ke data test. Berikut merupakan hasil output program ketika program dijalankan:



The screenshot shows an IPython console window titled "IPython console". The main area displays a list of 20 "7"s, one on each line. The window has a tab labeled "Console 1/A" and a "History log" button at the bottom. The status bar at the bottom indicates "LF", "Encoding: ASCII", "Line: 73", "Column: 28", and "Memory: 80 %".

4. Kesimpulan

Dalam penggunaan *K-Means Clustering* dapat disimpulkan percobaan klasterisasi ini dari hasil data yang ada pada materi sebelumnya. Dan dari percobaan sistem *K-Means Clustering* ini adalah pengelompokan data menjadi ke beberapa kelas (*cluster*) akan sangat bergantung dari penentuan titik *centroid* atau tengah dimana semakin bagus asumsi peletakan *centroid* maka akan semakin bagus juga hasil pengelompokannya.