LAPORAN TUGAS

**K-MEANS CLUSTERING**

**Mata Kuliah:**

**MACHINE LEARNING**

**Genap 2019/2020**

**Dosen Pengampu:**

**Dr., INDAH AGUSTIEN SIRADJUDDIN, S.Kom., M.Kom**

**Oleh :**

**[Miftahul Choir](https://new.edmodo.com/user/108671307/)**

**[160411100055](https://new.edmodo.com/user/108671307/)**

**[Dennis Thandy Nur Cahyono](https://new.edmodo.com/user/21129755/)**

**[160411100079](https://new.edmodo.com/user/21129755/)**

**[Nurrachmad Widyanto](https://new.edmodo.com/user/24931618/)**

**[160411100179](https://new.edmodo.com/user/24931618/)**

**[Muhammad Yusqi Alfan Thoriq](https://new.edmodo.com/user/108909389/)**

**[160411100187](https://new.edmodo.com/user/108909389/)**

****

**PROGRAM STUDI INFORMATIKA**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

**BANGKALAN**

**2019**

1. **Deskripsi Dataset**

Dataset yang digunakan pada metode K-Means Clustering ini adalah data pembeli di Mal Supermarket yang diambil dari

<https://www.kaggle.com/vjchoudhary7/customer-segmentation-tutorial-in-python>yang memiliki 5 fitur atau data tentang pelanggan, seperti ID Pelanggan, usia, jenis kelamin, pendapatan tahunan, dan skor pengeluaran. Skor Pengeluaran adalah nilai yang ditetapkan untuk pelanggan berdasarkan parameter yang ditentukan seperti perilaku pelanggan dan data pembelian. Dengan tujuan agar memahami pelanggan[Target Pelanggan] sehingga hasil perhitungan dapat diberikan kepada tim pemasaran dan merencanakan strategi yang sesuai dengan hasil yang ada.

1. **Penjelasan Kode Program**
2. **Kode Scikit**

|  |
| --- |
| from sklearn.cluster import KMeans  import numpy as np  import loadCSV  from sklearn.model\_selection import train\_test\_split  datasets = loadCSV.LoadCSV('Mall\_Customers.csv', ',')  data = datasets[:,:]  data\_training, data\_test = train\_test\_split(data, test\_size=0.30, random\_state=0)  est = KMeans(2)  est.fit(data)  y\_kmeans = est.predict(data)  centroid = est.cluster\_centers\_  label = est.labels\_  c0 = []  c1 = []  print("Centroid = ",centroid)  print("Label = ",label)  for i in range(len(data)):  if (label[i] == 0):  c0.append(data[i])  else:  c1.append(data[i])  C0 = np.array(c0)  C1 = np.array(c1)  a=0  b=0  s=0  for i in range(len(C0[1:])):  a+=(C0[0]-C0[1:][i])\*\*2  for i in range(len(C1)):  b+=(C0[0]-C1[i])\*\*2  a=a\*\*0.5  a=np.average(a)  b=b\*\*0.5  b=np.average(b)  if a < b:  s = 1-(a/b)  elif a > b:  s = (b/a)-1  else:  s=0  print("Accuracy pada a[0] = ",s) |

1. **Penjelasan Bagian Kode Program**

|  |
| --- |
| datasets = loadCSV.LoadCSV('Mall\_Customers.csv', ',')  data = datasets[:,:]  data\_training, data\_test = train\_test\_split(data, test\_size=0.30, random\_state=0) |

Membagi data menjadi data training dan data testing

|  |
| --- |
| est.fit(data)#Membuat model KMeans  y\_kmeans = est.predict(data) #Memprediksi model berdasarkan data  centroid = est.cluster\_centers\_ #Memperoleh nilai centroid  label = est.labels\_ #Memperoleh class/label |

|  |
| --- |
| for i in range(len(data)):  if (label[i] == 0):  c0.append(data[i])  else:  c1.append(data[i]) |

Looping untuk memisahkan data berdasarkan cluster 1 dan 0

|  |
| --- |
| for i in range(len(C0[1:])):  a+=(C0[0]-C0[1:][i])\*\*2 |

Menghitung jarak antara data dengan data pada cluster terdekat (rumus ecludian distance)

|  |
| --- |
| for i in range(len(C1)):  b+=(C0[0]-C1[i])\*\*2 |

Mengitung jarak antara data dengan data pada cluster terjauh

1. **Kode Manual Incremental Learning**

|  |
| --- |
| import numpy as np  import csv  from sklearn.model\_selection import train\_test\_split  import math  import random  def openCsv(file, delimiterSymbol):  data = []  with open(file, newline='') as csvfile:  spamreader = csv.reader(csvfile, delimiter = delimiterSymbol, quotechar='|')  for row in spamreader:  data.append(row)  return np.array(data[1:], dtype=float)  def centroid(dataset, numOfClass):  fitur = []  centroid = []  for i in range(dataset.shape[1]):  fitur.append(dataset[:,i])  for i in range(numOfClass):  temp = []  for j in range(len(fitur)):  temp.append(random.choice(fitur[j]))  centroid.append(temp)  return centroid  def euclidean(data, centroid):  result = []  distance = []  for i in range(len(centroid)):  result.append((data[i] - centroid[i]) \*\* 2)  return sum(result) \*\* 0.5  def kmeans(data, centroid, epoch):  for g in range(epoch):  cluster = [ [] for i in range(len(centroid)) ] #list untuk menampung hasil cluster  for i in range(len(dataset)):  distance = []  for j in range(len(centroid)):  distance.append(euclidean(dataset[i], centroid[j]))  min = np.argmin(distance)  for l in range(len(centroid)):  if min == l:  cluster[l].append(dataset[i])  cluster = np.array(cluster)  for h in range(len(centroid)):  centroid[h] = np.average(cluster[h], axis=0) return centroid, cluster  def predic(data, centroid):  cluster = [ [] for i in range(len(centroid)) ]  for i in range(len(data)):  distance = []  for j in range(len(centroid)):  distance.append(euclidean(dataset[i], centroid[j]))  min = np.argmin(distance)  if min == j:  cluster[j].append(dataset[i])  return np.array(cluster)  def silhoutteScore(cluster):  result = 0  distance = []  for i in range(len(cluster[0])):  for j in range(i, len(cluster[0])-1):  distance.append(  np.linalg.norm(cluster[0][i] - cluster[0][j+1])  )  a = np.average(distance)  anotherDistance = []  for i in range(len(cluster[0])):  temp = []  for j in range(len(cluster[1])):  temp.append(  np.linalg.norm(cluster[0][i] - cluster[1][j]))  anotherDistance.append(np.average(temp))  b = np.min(anotherDistance)  result = ( b - a) / np.maximum(a, b)  print(result)  dataset = openCsv('Mall\_Customers.csv', ',')  centroid = centroid(dataset, 2)  newCenter, cluster = kmeans(dataset, centroid, 10)  predict = predic(dataset, newCenter)  silhoutteScore(cluster)  print(predict) |

1. **Penjelasan Bagian Kode Program**

|  |
| --- |
| def openCsv(file, delimiterSymbol):  data = []  with open(file, newline='') as csvfile:  spamreader = csv.reader(csvfile, delimiter = delimiterSymbol, quotechar='|')  for row in spamreader:  data.append(row)  return np.array(data[1:], dtype=float) |

Bagian fungsi diatas bernama *LoadCSV()* yang berfungsi untuk mengambil dataset dari file csv kemudian dimasukkan ke dalam array dan menjadi nilai kembalian.

|  |
| --- |
| def centroid(dataset, numOfClass):  fitur = []  centroid = []  for i in range(dataset.shape[1]):  fitur.append(dataset[:,i])  for i in range(numOfClass):  temp = []  for j in range(len(fitur)):  temp.append(random.choice(fitur[j])) #menentukan centroid awal dengan random  centroid.append(temp)  return centroid |

Bagian fungsi diatas bernama *centroid()* yang berfungsi untuk menentukan nilai titik tengah setiap cluster

|  |
| --- |
| def euclidean(data, centroid):  result = []  distance = []  for i in range(len(centroid)):  result.append((data[i] - centroid[i]) \*\* 2)  return sum(result) \*\* 0.5 |

Bagian fungsi diatas bernama *euclidean*() yang berfungsi untuk menghitung jarak setiap data dengan centroid.

|  |
| --- |
| def kmeans(data, centroid, epoch):  for g in range(epoch):  cluster = [ [] for i in range(len(centroid)) ]  for i in range(len(dataset)):  distance = []  for j in range(len(centroid)):  distance.append(euclidean(dataset[i], centroid[j]))  min = np.argmin(distance)  for l in range(len(centroid)):  if min == l:  cluster[l].append(dataset[i])  cluster = np.array(cluster)  for h in range(len(centroid)):  centroid[h] = np.average(cluster[h], axis=0) return centroid, cluster |

Bagian fungsi diatas bernama *kmeans()* yang berfungsi menentukan setiap data termasuk dalam cluster yang memiliki jarak terpendek dengan centroid

|  |
| --- |
| def predic(data, centroid):  cluster = [ [] for i in range(len(centroid)) ]  for i in range(len(data)):  distance = []  for j in range(len(centroid)):  distance.append(euclidean(dataset[i], centroid[j]))  min = np.argmin(distance)  if min == j:  cluster[j].append(dataset[i])  return np.array(cluster) |

Bagian fungsi diatas bernama *predict()* yang berfungsi memprediksi data akan termasuk cluster yang mana dengan centroid yang baru.

|  |
| --- |
| def silhoutteScore(cluster):  result = 0  distance = []  for i in range(len(cluster[0])):  for j in range(i, len(cluster[0])-1):  distance.append(  # euclidean(cluster[0][i], cluster[0][j+1])  np.linalg.norm(cluster[0][i] - cluster[0][j+1])  )  a = np.average(distance)  anotherDistance = []  for i in range(len(cluster[0])):  temp = []  for j in range(len(cluster[1])):  temp.append(  # euclidean(cluster[0][i], cluster[1][j])  np.linalg.norm(cluster[0][i] - cluster[1][j])  )  anotherDistance.append(np.average(temp))  b = np.min(anotherDistance)  result = ( b - a) / np.maximum(a, b)  print(result) |

Bagian fungsi diatas bernama silhoutteScore() yang berfungsi untuk menghitung akurasi dari ketepatan prediksi

1. **Hasil Uji Coba dan Perthitungan Manual**
2. **Hasil Uji Coba**
3. **Kode Scikit**

Dari hasil uji coba menggunakan program manual diperoleh hasil score silhoutte -0.064568. dan hasil uji coba menggunkan model scikit memperoleh score silhoutte -0.09599402598571949 pada akurasi a[0]

1. **Perhitungan Manual**

**Dataset**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 19 | 15 | 39 |
| 1 | 21 | 15 | 81 |
| 0 | 20 | 16 | 6 |
| 0 | 23 | 16 | 77 |
| 0 | 31 | 17 | 40 |
| 0 | 22 | 17 | 76 |
| 0 | 35 | 18 | 6 |
| 0 | 23 | 18 | 94 |
| 1 | 64 | 19 | 3 |
| 0 | 30 | 19 | 72 |
| 1 | 67 | 19 | 14 |
| 0 | 35 | 19 | 99 |
| 0 | 58 | 20 | 15 |
| 0 | 24 | 20 | 77 |
| 1 | 37 | 20 | 13 |
| 1 | 22 | 20 | 79 |
| 0 | 35 | 21 | 35 |
| 1 | 20 | 21 | 66 |
| 1 | 52 | 23 | 29 |
| 0 | 35 | 23 | 98 |

Sampel data yang diambil untuk dihitung menggunakan excel adalah 20 data dari 200 total jumlah data.

**Centeroid 1**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 93 | 14 | 4 |

Centeroid pertama pada kluster 1 didefinisikan sebagai *random*

**Centeroid 2**

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 48 | 32 | 68 |

Centeroid pertama pada kluster 2 didefinisikan sebagai *random*.

**Euclidean data dengan centeroid 1**

|  |
| --- |
| 81,86574375 |
| 105,4229577 |
| 73,06161783 |
| 101,1632344 |
| 71,76350047 |
| 101,1681768 |
| 58,18075283 |
| 114,0920681 |
| 29,44486373 |
| 92,83856957 |
| 28,3019434 |
| 111,4226189 |
| 37,18870796 |
| 100,6329966 |
| 57,03507693 |
| 103,4504712 |
| 66,14378278 |
| 96,03124492 |
| 48,856934 |
| 110,8241851 |

Menghitung jarak antar data dengan setiap centeroid

**Euclidean data dengan centeroid 2**

|  |
| --- |
| 44,40720662 |
| 34,46737588 |
| 69,88562084 |
| 31,01612484 |
| 36,02776707 |
| 31,06444913 |
| 64,87680633 |
| 38,69108424 |
| 68,19824045 |
| 22,56102835 |
| 58,71115737 |
| 36,04164258 |
| 55,25395913 |
| 28,3019434 |
| 57,36723804 |
| 30,69201851 |
| 37,13488926 |
| 30,16620626 |
| 40,23679908 |
| 33,91164992 |

Menghitung jarak antar data dengan setiap centeroid

**Pelabelan terhadap Data**

|  |
| --- |
| 1 |
| 1 |
| 1 |
| 1 |
| 1 |
| 1 |
| 0 |
| 1 |
| 0 |
| 1 |
| 0 |
| 1 |
| 0 |
| 1 |
| 0 |
| 1 |
| 1 |
| 1 |
| 1 |
| 1 |

Label diperoleh dari jarak terkecil diantara jarak data pada centeroid pertama dan data pada centeroid kedua. Berlabel 1 jika jarak antara data dan centeroid 2 lebih kecil, dan berlabel 0 jika jarak antara data dan centeroid 1 lebih kecil.

**Kluster 1**

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 35 | 18 | 6 |
| 1 | 64 | 19 | 3 |
| 1 | 67 | 19 | 14 |
| 0 | 58 | 20 | 15 |
| 1 | 37 | 20 | 13 |

Diperoleh data pada tabel diatas yang tergolong pada kluster 1

**Kluster 2**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 19 | 15 | 39 |
| 1 | 21 | 15 | 81 |
| 0 | 20 | 16 | 6 |
| 0 | 23 | 16 | 77 |
| 0 | 31 | 17 | 40 |
| 0 | 22 | 17 | 76 |
| 0 | 23 | 18 | 94 |
| 0 | 30 | 19 | 72 |
| 0 | 35 | 19 | 99 |
| 0 | 24 | 20 | 77 |
| 1 | 22 | 20 | 79 |
| 0 | 35 | 21 | 35 |
| 1 | 20 | 21 | 66 |
| 1 | 52 | 23 | 29 |
| 0 | 35 | 23 | 98 |

Diperoleh data pada table diatas yang tergolong pada kluster 2

Data yang telah terkluster tersebut dicari nilai tengah untuk nilai centeroid yang baru

Proses diatas adalah proses epoch pertama, proses diatas akan diulang sampai epoch yang telah ditentukan, namun pada laporan ini hanya melakukan proses sampai epoch ke 3. pada proses epoch ke 3 diperoleh centroid seperti tabel dibawah.

**Centeroid 1**

|  |  |  |  |
| --- | --- | --- | --- |
| 0,5 | 46 | 19,5 | 15,125 |

**Centeroid 2**

|  |  |  |  |
| --- | --- | --- | --- |
| 0,333333333 | 25,41666667 | 18,33333333 | 74,83333333 |

Dengan centeroid tersebut diperoleh label sebagai berikut

|  |
| --- |
| 0 |
| 1 |
| 0 |
| 1 |
| 0 |
| 1 |
| 0 |
| 1 |
| 0 |
| 1 |
| 0 |
| 1 |
| 0 |
| 1 |
| 0 |
| 1 |
| 0 |
| 1 |
| 0 |
| 1 |

Data yang terkluster pada kelompok 1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 19 | 15 | 39 |
| 0 | 20 | 16 | 6 |
| 0 | 31 | 17 | 40 |
| 0 | 35 | 18 | 6 |
| 1 | 64 | 19 | 3 |
| 1 | 67 | 19 | 14 |
| 0 | 58 | 20 | 15 |
| 1 | 37 | 20 | 13 |
| 0 | 35 | 21 | 35 |
| 1 | 52 | 23 | 29 |

Data yang terkluster pada kelompok 2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 21 | 15 | 81 |
| 0 | 23 | 16 | 77 |
| 0 | 22 | 17 | 76 |
| 0 | 23 | 18 | 94 |
| 0 | 30 | 19 | 72 |
| 0 | 35 | 19 | 99 |
| 0 | 24 | 20 | 77 |
| 1 | 22 | 20 | 79 |
| 1 | 20 | 21 | 66 |
| 0 | 35 | 23 | 98 |

Akurasi yang diperoleh dengan menggunakan metode silhoutte pada perhitungan manual excel yaitu -0,820264592

1. **Pembagian Tugas**
2. Miftahul Choir 160411100055 (Menghitung manual pada excel)
3. Dennis Thandy Nur Cahyono 160411100079 (Program Manual Kmeans)
4. Nurrachmad Widyanto 160411100179 (Laporan)
5. Muhammad Yusqi Alfan Thoriq 160411100187 (Program Kmeans Scikit, dan akurasi manual)