

LAPORAN TUGAS BESAR TAHAP I

I. Formulasi Masalah

Clustering yang akan dilakukan memiliki tujuan untuk memprediksi apakah pelanggan tertarik untuk membeli kendaraan baru atau tidak berdasarkan data pelanggan di dealer. Dataset yang digunakan merupakan `Kendaraan_train.csv` yang merupakan seluruh data pelanggan dan kendaraan pelanggan yang tersimpan dalam database dealer. Dataset ini diberikan oleh Dosen dari Mata Kuliah Machine Learning sebagai acuan mahasiswa dalam mengerjakan tugas clustering ini.

Dalam tugas ini saya memisahkan bagian eksplorasi data dan bagian clustering menjadi dua file yang berbeda. File `1301184245_Muhammad Raihan Muhith.ipynb` merupakan bagian eksplorasi dan persiapan data, sedangkan file `1301184245_Muhammad Raihan Muhith(clustering).ipynb` merupakan bagian clustering, eksperimen, dan evaluasi. Hal ini saya lakukan agar code-code yang ada pada bagian clustering tidak akan memengaruhi dataset awal, sehingga dataset awal terjaga integritasnya, alasan lainnya agar proses running berjalan dengan cepat. Pada bagian clustering, Dataset yang digunakan adalah `data_model.csv`, data ini merupakan atribut-atribut yang telah saya pilih dan di scale dari dataset awal pada bagian eksplorasi data.

Masalah lain yang dapat diselesaikan menggunakan clustering pada data ini salah satunya adalah mengkategorikan pelanggan yang berhak diberikan pemotongan biaya baik pemotongan biaya beli kendaraan maupun pemotongan biaya pemeliharaan kendaraan dari dealer. Hal ini didasari oleh data dengan kolom lama berlangganan, perusahaan sebaiknya memberikan pemotongan biaya kepada pelanggan dengan lama berlangganan tertentu dan umur tertentu, atau bisa juga kepada pelanggan dengan lama berlangganan tertentu dan jumlah pembayaran premi tertentu.

Hasil clustering data tentu saja akan memberikan dampak positif kepada perusahaan, contohnya perusahaan akan mengetahui kategori pelanggan apa yang kira-kira akan membeli kendaraan baru, sehingga perusahaan cukup memberikan rekomendasi kepada pelanggan-pelanggan tertentu saja.

II. Eksplorasi dan Persiapan Data

Pada bagian ini, mahasiswa diwajibkan untuk mengetahui informasi-informasi dari dataset. Memahami data merupakan hal yang penting untuk dilakukan sebelum melakukan clustering. Terdapat beberapa tahapan yang saya lakukan pada bagian ini.

a. Data Exploration and Understanding

Berikut merupakan gambaran dari dataset yang akan digunakan pada bagian eksplorasi data.

id	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
1	Wanita	30.0	1.0	33.0	1.0	< 1 Tahun	Tidak	28029.0	152.0	97.0	0
2	Pria	48.0	1.0	39.0	0.0	> 2 Tahun	Pernah	25800.0	29.0	158.0	0
3	NaN	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	32733.0	160.0	119.0	0
4	Wanita	58.0	1.0	48.0	0.0	1-2 Tahun	Tidak	2630.0	124.0	63.0	0
5	Pria	50.0	1.0	35.0	0.0	> 2 Tahun	NaN	34857.0	88.0	194.0	0
...
85827	Wanita	23.0	1.0	4.0	1.0	< 1 Tahun	Tidak	25988.0	152.0	217.0	0
85828	Wanita	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	44686.0	152.0	50.0	0
85829	Wanita	23.0	1.0	50.0	1.0	< 1 Tahun	Tidak	49751.0	152.0	226.0	0
85830	Pria	68.0	1.0	7.0	1.0	1-2 Tahun	Tidak	30503.0	124.0	270.0	0
85831	Pria	45.0	1.0	28.0	0.0	1-2 Tahun	Pernah	36480.0	26.0	44.0	0

Data tersebut memiliki keterangan kolom sebagai berikut.

Nama Kolom	Keterangan
id	Id dari pelanggan
Jenis_Kelamin	Jenis kelamin dari pelanggan
Umur	Umur Pelanggan
SIM	0 : Tidak punya SIM, 1 : Punya SIM
Kode_Daerah	Kode area tempat tinggal pelanggan
Sudah_Asuransi	1 : Pelanggan sudah memiliki asuransi kendaraan, 0 : Pelanggan belum memiliki asuransi kendaraan
Umur_Kendaraan	Umur kendaraan pelanggan
Kendaraan_Rusak	Apakah Kendaraan pelanggan pernah rusak atau tidak
Premi	Jumlah premi yang harus dibayarkan per tahun.

Kanal_Penjualan	Kode kanal untuk menghubungi pelanggan (email, telpon, dll)
Lama_Berlangganan	Sudah berapa lama pelanggan menjadi klien perusahaan
Tertarik (diisi pada Tahap II tugas besar machine learning)	1 : Pelanggan tertarik untuk membeli kendaraan baru, 0 : Pelanggan tidak tertarik untuk membeli kendaraan baru

Selanjutnya kita cek ada berapa baris dan kolom dari data yang kita punya.

```
print('Number of Rows :',len(df.axes[0]))
print('NUmber of Columns :',len(df.axes[1]))
```

```
Number of Rows : 285831
NUmber of Columns : 12
```

Setelah itu kita identifikasi mana kolom dengan variable categorical, karena umumnya variable categorical memiliki value bertipe data string, sehingga akan terjadi eror ketika akan dilakukan operasi matematika atau fungsi agregasi table. Kita tahu bahwa kolom yang merupakan varibel categorical dengan tipe data kolom tersebut string adalah kolom Jenis_Kelamin, Umur_Kendaraan, dan Kendaraan_Rusak. Disini saya ubah(*mapping*) isi kolom tersebut berdasarkan kategori sebagai berikut.

Nama Kolom	Awal Isi Kolom	Setelah di ubah
Jenis Kelamin	Pria	0
	Wanita	1
Umur_Kendaraan	< 1 Tahun	0
	1-2 Tahun	1
	> 2 Tahun	2
Kendaraan_Rusak	Tidak	0
	Pernah	1

Maka hasil dari mapping adalah sebagai berikut.

Jenis_Kelamin	Umur_Kendaraan	Kendaraan_Rusak
1.0	0.0	0.0
0.0	2.0	1.0
NaN	0.0	0.0
1.0	1.0	0.0
0.0	2.0	NaN
...
1.0	0.0	0.0
1.0	0.0	0.0
1.0	0.0	0.0
0.0	1.0	0.0
0.0	1.0	1.0

Cek tipe data seluruh kolom.

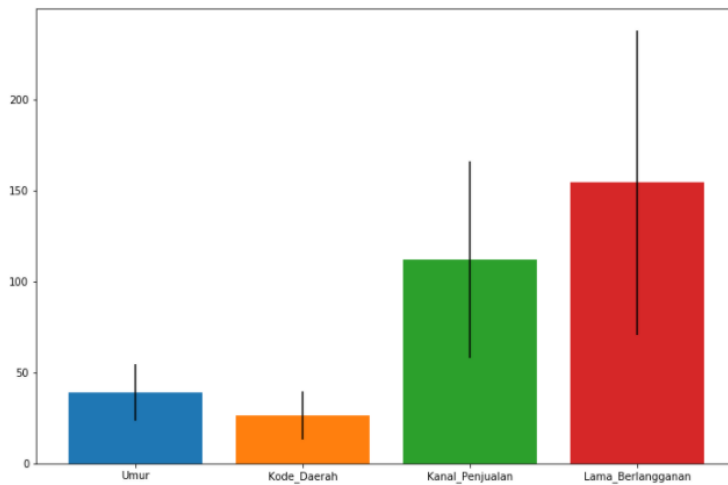
```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285831 entries, 0 to 285830
Data columns (total 12 columns):
id                285831 non-null int64
Jenis_Kelamin     271391 non-null float64
Umur              271617 non-null float64
SIM               271427 non-null float64
Kode_Daerah       271525 non-null float64
Sudah_Asuransi    271602 non-null float64
Umur_Kendaraan    271556 non-null float64
Kendaraan_Rusak   271643 non-null float64
Premi             271262 non-null float64
Kanal_Penjualan   271532 non-null float64
Lama_Berlangganan 271839 non-null float64
Tertarik         285831 non-null int64
dtypes: float64(10), int64(2)
memory usage: 26.2 MB
```

Semua tipe data sekarang sudah bertipe numerik. Jadi tidak perlu ada lagi atribut yang perlu diubah tipe datanya.

Analisa-analisa yang dilakukan terhadap dataset :

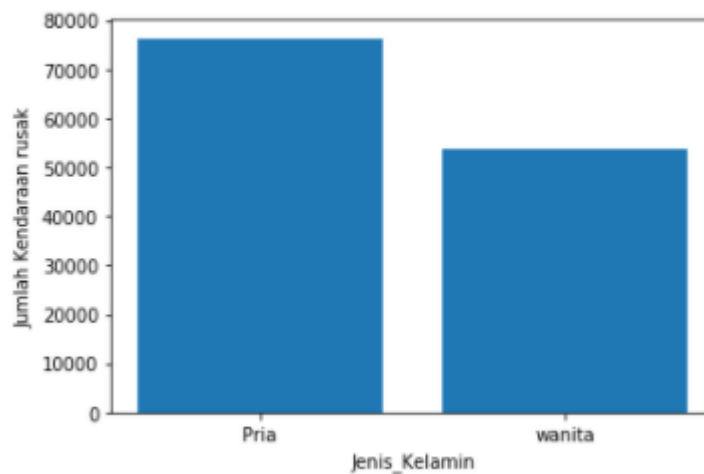
- Analisa distribusi data kolom Umur, Kode_Daerah, Kanal_Penjualan, dan Lama_Berlangganan.



Gambar diatas menunjukkan bahwa bar chart merepresentasikan central tendency berupa nilai mean dari data dengan kolom Umur, Kode_Daerah, Kanal_Penjualan, dan Lama_Berlangganan. sedangkan garis hitam menunjukkan standar deviasi dari data tersebut. Lama_Berlangganan memiliki sebaran data yang paling lebar sedangkan Umur memiliki sebaran data yang paling kecil, artinya distribusi Umur terpusat di satu titik tidak begitu menyebar.

- ii. Analisa perbandingan jumlah kendaraan rusak berdasarkan jenis kelamin.

Grafik Perbandingan Jumlah Kendaraan Rusak Berdasarkan Jenis Kelamin



Dengan visualisasi heatmap, kita dapat mengetahui nilai-nilai korelasi antar atribut. Dari gambar diatas, nilai korelasi antar atribut yang paling menjauhi angka 0 adalah korelasi antara Sudah_asuransi dengan Kendaraan_Rusak dan Umur dengan Umur_Kendaraan.

Setelah dilakukannya Analisa-analisa pada atribut-atribut dataset, tentukan atribut apa saja yang perlu kita cluster. Disini saya memilih atribut **Umur** dan **Umur_Kendaraan**. Karena kedua atribut tersebut memiliki nilai korelasi yang tinggi, dan salah satu dari kedua atribut tersebut merupakan tipe variabel numerik atau kuantitatif.

Setelah menentukan atribut apa saja yang ingin dilakukan clustering, cek apakah dari kedua atribut memiliki *missing value* dan data pencilan, jika iya, tentukan apa yang akan kita lakukan kepada *missing value* dan data pencilan tersebut.

- Cek Missing Value

```
: df[['Umur', 'Umur_Kendaraan']].isna().sum()
: Umur          14214
: Umur_Kendaraan 14275
: dtype: int64
```

Dari atribut umur, terdapat 14214 jumlah missing value, dan dari atribut Umur_Kendaraan terdapat 14275 jumlah missing value.

- Cek Data Pencilan



Hasil visualisasi boxplot dari kedua atribut menggambarkan bahwa atribut Umur dan Umur_Kendaraan tidak memiliki data pencilan.

b. Data Cleansing

Pada bagian data understanding, kita sudah menentukan atribut apa saja yang ingin di cluster, dan kita juga sudah mengetahui atribut-atribut tersebut memiliki *missing value* dan data pencilan atau tidak.

i. Handling *missing value* atribut Umur

Untuk menjaga kestabilan distribusi dari data atribut Umur, maka semua *missing value* pada data atribut umur saya isi dengan nilai rata-ratanya. Untuk rata-rata data atribut umur itu sendiri bernilai 38.8, sehingga angka tersebut akan dibulatkan, menjadi 39. Artinya seluruh *missing value* akan diisi dengan nilai 39.

```
: df.Umur.mean()
: 38.84433595835312

: #Handling Missing Value Umur
df.Umur.fillna(round(df.Umur.mean()),inplace=True)
```

ii. Handling *missing value* atribut Umur_Kendaraan

```
: df_1 = df.Umur_Kendaraan[df.Umur_Kendaraan == 0]
df_2 = df.Umur_Kendaraan[df.Umur_Kendaraan == 1]
df_3 = df.Umur_Kendaraan[df.Umur_Kendaraan == 2]
print('Umur Kendaraan < 1 tahun :',len(df_1))
print('Umur Kendaraan 1-2 tahun :',len(df_2))
print('Umur Kendaraan > 2 tahun :',len(df_3))

Umur Kendaraan < 1 tahun : 117378
Umur Kendaraan 1-2 tahun : 142761
Umur Kendaraan > 2 tahun : 11417
```

Dari gambar diatas, kita tahu bahwa rata-rata umur kendaraan dari pelanggan adalah 1-2 tahun, maka dari itu saya mengisi *missing value* dari data atribut Umur_Kendaraan dengan angka 1, karena angka 1 adalah hasil *mapping* dari nilai '1-2 tahun'.

```
#Handling Missing value Umur_Kendaraan
df.Umur_Kendaraan.fillna(1, inplace = True)
```



Karena atribut Umur dan Umur_Kendaraan tidak memiliki data pencilan, maka tidak perlu ada handling data pencilan.

Setelah data-data dari kedua atribut sudah bersih, buat dataframe baru yang hanya memiliki data dari atribut Umur dan Umur_Kendaraan, dengan kata lain drop seluruh kolom kecuali kolom Umur dan Umur_Kendaraan, karena kita sudah tidak membutuhkannya lagi.

c. Feature Engineering

Pada bagian ini, seluruh data dari kedua atribut akan dilakukan scalling. Scalling merupakan sebuah metode untuk mentransformasikan nilai data menjadi nilai dalam range tertentu. Disini saya menggunakan scalling menggunakan cara MinMax Normalization. MinMax Normalization akan mentransformasikan data kedalam range 0 – 1. Berikut merupakan cara perhitungan MinMax Normalization.

$$X_{normalized} = \frac{(X - X_{minimum})}{(X_{maximum} - X_{minimum})}$$

Berikut merupakan data dari kedua atribut yang telah discale.

	Umur	Umur_Kendaraan
0	0.153846	0.0
1	0.430769	1.0
2	0.015385	0.0
3	0.584615	0.5
4	0.461538	1.0
...
285826	0.046154	0.0
285827	0.015385	0.0
285828	0.046154	0.0
285829	0.738462	0.5
285830	0.384615	0.5

285831 rows × 2 columns

III. Pemodelan

Hasil dari scaling saya export kedalam data csv yang baru dan diberi nama 'data_model.csv'. model clustering yang saya buat adalah Clustering Menggunakan KMeans dengan metode k-means++. Berikut merupakan algoritma dari metode clustering.

Langkah 1: Tentukan berapa banyak *cluster* k dari dataset yang akan dibagi.

Langkah 2: Tetapkan secara acak data k menjadi pusat awal lokasi klaster.

Langkah 3: Untuk masing-masing data, temukan pusat *cluster* terdekat. Dengan demikian berarti masing-masing pusat *cluster* memiliki sebuah subset dari dataset, sehingga mewakili bagian dari dataset. Oleh karena itu, telah terbentuk *cluster* k: C1, C2, C3, ..., Ck .

Langkah 4: Untuk masing-masing *cluster* k, temukan pusat luasan klaster, dan perbarui lokasi dari masing-masing pusat *cluster* ke nilai baru dari pusat luasan.

Langkah 5: Ulangi langkah ke-3 dan ke-5 hingga data-data pada tiap *cluster* menjadi terpusat atau selesai.

Library pembantu yang saya gunakan adalah *pandas*, *matplotlib.pyplot*, *numpy*, dan *random*. Didalam file kodingan clustering, terdapat fungsi KMeans2 yang merupakan pemodelan clustering menggunakan KMeans. Berikut merupakan penjelasan fungsi KMeans2.

a. Step 1 & 2

```
import random as rd

def KMeans2(k,data):
    #Step 1 = tentukan jumlah cluster(disini melalui parameter)

    diff = 1 #random assign difference, untuk ngecek perbedaan centroid awal dan baru
    j=0

    #Step 2 = inisiasi random centroids menggunakan random uniform
    centroids = []
    for i in range(k):
        centroid_x = np.random.uniform(min(data.iloc[:,0]),max(data.iloc[:,0]))
        centroid_y = np.random.uniform(max(data.iloc[:,1]),max(data.iloc[:,1]))
        centroids.append([centroid_x,centroid_y])
```

Gambar diatas merupakan bagian step 1 dan step 2 pada algoritma KMeans, yaitu menentukan banyaknya cluster dan menginisiasi centroid secara random. Untuk step 2 saya menginisiasikan centroid secara random menggunakan *random.uniform*, yaitu mencari nilai random untuk dijadikan titik koordinat centroids, dengan batas bawah dan batas atas secara berturut-turut adalah nilai terkecil dan terbesar dari kedua data.

b. Step 3

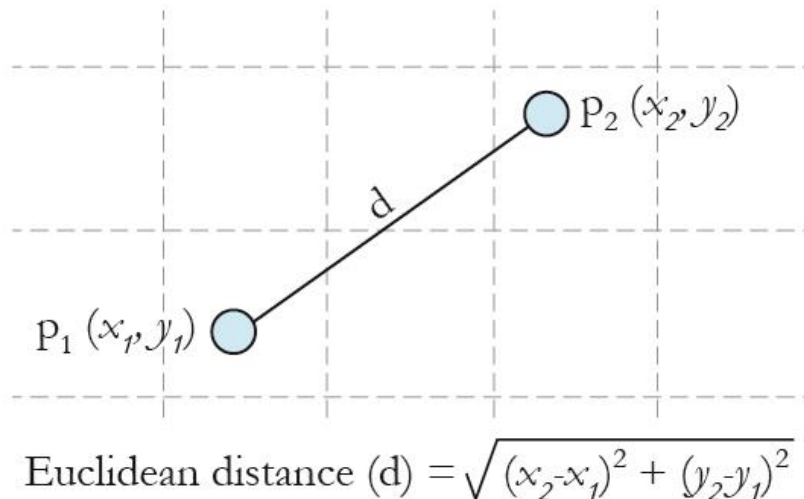
```
#step 3 = mengelompokkan points ke centroid terdekat
#menghitung jarak setiap point ke setiap centroids menggunakan euclidean
dataD=data
i=1
for idx_centroids in centroids:
    Euclidean=[]
    for index2,row_d in dataD.iterrows():
        X=(idx_centroids[0]-row_d['Umur'])**2
        Y=(idx_centroids[1]-row_d["Umur_Kendaraan"])**2
        ed=np.sqrt(X+Y)
        Euclidean.append(ed)
    data[i]=Euclidean
    i+=1

label=[]

#mengelompokkan point berdasarkan jarak terdekat points dengan centroid
for index,row in data.iterrows():
    min_distance=row[1]
    pos=1
    for i in range(k):
        if row[i+1] < min_distance:
            min_distance = row[i+1]
            pos=i+1
    label.append(pos)

#menambahkan kolom Labels untuk mengkategorikan kelompok
data["Labels"]=label
```

Gambar diatas merupakan bagian step 3 pada algoritma KMeans, yaitu mengelompokkan setiap data points ke centroid terdekat. Setiap data points dihitung jaraknya dengan setiap centroid proses perhitungan menggunakan Euclidean, berikut merupakan gambaran perhitungan Euclidean distance.



Sumber : <https://wirasetiawan29.files.wordpress.com/2015/04/euclidean-distance-graphic.jpg>

lalu setelah setiap data points mengetahui jaraknya dengan ke setiap centroids, tiap-tiap points mencari jarak terdekat mereka dengan setiap centroid, dan berikan label points tersebut untuk memberikan informasi bahwa points tersebut masuk kedalam kelompok centroids ke berapa.

c. Step 4

```
#step 4 = cari centroids baru
#cari centroid baru dari rata-rata posisi di tiap cluster
centroids_new = data.groupby(["Labels"]).mean()[["Umur", "Umur_Kendaraan"]].values.tolist()
```

Gambar diatas merupakan bagian step 4 pada algoritma KMeans, yaitu mencari centroids baru dari jarak rata-rata tiap kelompok.

d. Step 5

```
while(diff!=0): #ketika centroids sebelumnya dan centroid baru tidak sama, lakukan looping

    if j == 0:
        diff=1
        j=j+1
    else:
        for c1,c2 in zip(centroids,centroids_new):
            #menghitung jarak tiap centroid awal dan baru
            diff = np.sqrt(sum(np.square(np.array(c1)-np.array(c2))))
```

Gambar diatas merupakan bagian step 4 pada algoritma KMeans, yaitu ulangi step 3 dan 4 jika posisi centroid belum terpusat atau masih terdapat perbedaan antara letak centroid awal dan baru.

Fungsi ini mengembalikan sebuah dataframe baru dengan kolom tambahan yaitu kolom jarak antar centroid, dan Labels. Kolom Labels merupakan kolom yang menandakan data point tersebut masuk kedalam kelompok mana. Berikut merupakan hasil tranformasi dataframe sebelum dan sesudah dilakukannya KMeans dengan jumlah $k = 3$.

Sebelum :

	Umur	Umur_Kendaraan
0	0.153846	0.0
1	0.430769	1.0
2	0.015385	0.0
3	0.584615	0.5
4	0.461538	1.0
...
285826	0.046154	0.0
285827	0.015385	0.0
285828	0.046154	0.0
285829	0.738462	0.5
285830	0.384615	0.5

Sesudah :

	Umur	Umur_Kendaraan	1	2	3	Labels
99472	0.061538	0.0	0.818516	0.020283	0.577885	2
121244	0.107692	0.5	0.544869	0.500669	0.211163	3
61214	0.338462	0.5	0.317503	0.562018	0.027048	3
217003	0.061538	0.0	0.818516	0.020283	0.577885	2
131233	0.015385	0.0	0.852188	0.066437	0.599803	2
...
63300	0.476923	0.5	0.185171	0.637264	0.159815	3
99605	0.061538	0.0	0.818516	0.020283	0.577885	2
131968	0.015385	0.0	0.852188	0.066437	0.599803	2
79033	0.815385	0.5	0.181944	0.887758	0.497600	1
265970	0.338462	0.5	0.317503	0.562018	0.027048	3

Alasan saya memilih pemodelan KMeans Clustering adalah karena metode ini memiliki algoritma yang paling sederhana dibanding semua algoritma-algoritma clustering lainnya, sehingga akan mempermudah proses pembuatannya.

IV. Evaluasi

Pada bagian ini Saya memilih metode evaluasi Sum Square Error. Berikut merupakan cara perhitungan SSE.

$$\sum_{j=1}^k \sum_{x_i \in c_j} \|c_j - x_i\|^2$$

Keterangan :

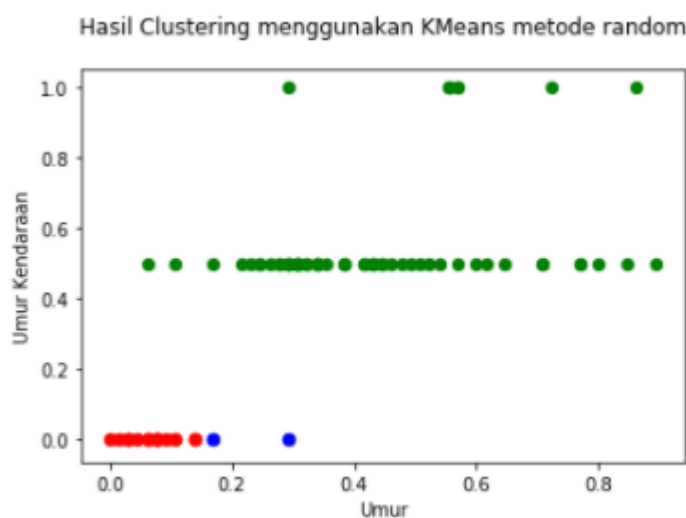
- k = jumlah kluster
- C_j = posisi centroid ke j
- X_i = posisi data ke i
- $\|c_j - x_i\|$ = jarak data ke i dengan centroid ke j .

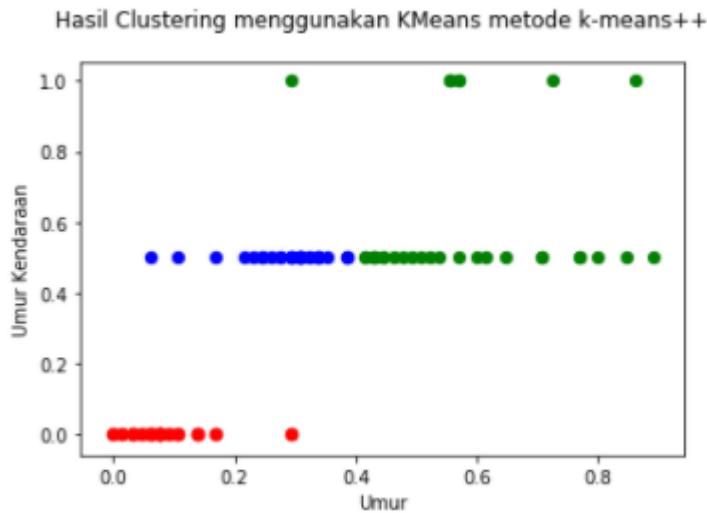
Hasil cluster yang terbentuk dari metode K-means ini sangatlah tergantung pada inisiasi nilai pusat awal cluster yang diberikan. Maka dari itu saya ingin menghasilkan cluster yang lebih baik yaitu dengan menerapkan Sum Of Squared Error (SSE) untuk membantu K-Means Clustering dalam menentukan jumlah cluster yang paling optimum.

V. Eksperimen

Terdapat 2 metode KMeans yang sering digunakan dalam melakukan clustering, yaitu metode 'random' dan metode 'k-means++'. Perbedaan dari kedua metode tersebut terletak pada step 2 algoritma KMeans, yaitu inisiasi random centroid sebanyak k . cara metode 'random' dalam menginisiasikan random centroid adalah mencari centroids secara random dari point, sedangkan metode 'k-means++' menginisiasikan random centroid dengan cara mencari titik koordinat dari random.uniform dengan batas bawah nilai minimum dan batas atas nilai maksimum dari kedua atribut.

Dalam melakukan eksperimen, saya membuat 2 metode KMeans, yaitu metode 'random' dan 'uniform'. Berikut perbedaan hasil clustering dengan menggunakan kedua metode tersebut.





Dapat dilihat dari kedua gambar diatas, hasil pengelompokkan menggunakan metode k-means++ terlihat lebih baik dibanding hasil pengelompokkan menggunakan metode random. Hal ini didasari oleh peluang letak inisiasi centroid. peluang letak centroid-centroid awal berada di suatu titik yang berdekatan pada metode random lebih besar daripada metode k-means++. Karena pada metode random, centroid dicari secara random melalui point-point, sehingga letak awal hanya akan berada di titik koordinat y 0,0.5, dan 1.

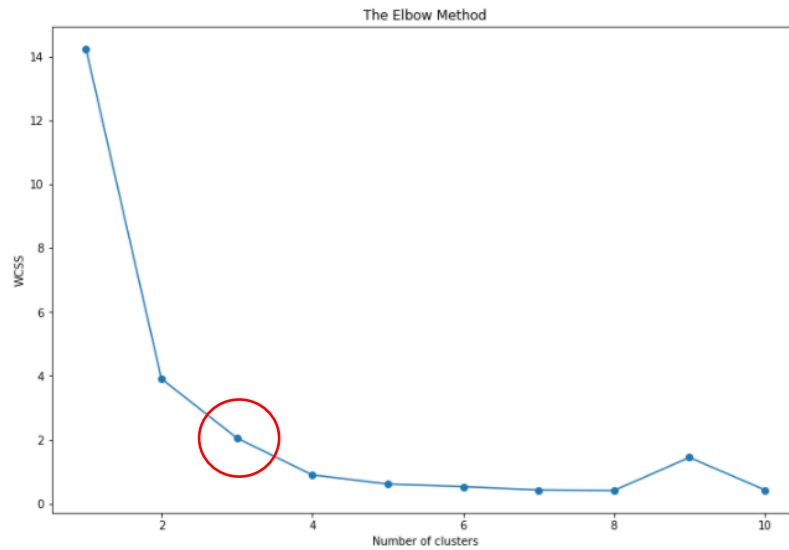
VI. Kesimpulan

Setelah dilakukannya clustering menggunakan metode KMeans ‘k-means++’, kita dapat mengetahui kategori-kategori pelanggan yang berlangganan dengan Dealer. Kelompok pelanggan pertama adalah pelanggan dengan angka umur yang rendah dan umur kendaraannya < 1 tahun, lalu ada kelompok kedua yaitu pelanggan dengan angka umur yang rendah dan umur kendaraannya 1-2 tahun, lalu kelompok pelanggan terakhir adalah pelanggan dengan angka umur yang tinggi dengan umur kendaraan 1-2 tahun dan > 2 tahun.

Untuk menyelesaikan masalah pada bagian Formulasi Masalah, dari Kelompok-kelompok tersebut, kita perlu identifikasi kelompok mana yang mungkin lebih tertarik dalam membeli kendaraan baru.

a. Hasil Evaluasi

Hasil evaluasi merupakan grafik yang menunjukkan perubahan nilai SSE di setiap jumlah k pada clustering menggunakan KMeans.



Grafik line diatas menunjukkan bahwa semakin kecil nilai k maka semakin kecil juga nilai SSE nya. Nilai k optimum didapatkan dari curved line yang terdapat pada grafik. Dari gambar diatas, $k = 3$ yang menempati curved line grafik, maka dapat kita simpulkan jumlah pengelompokkan yang optimum adalah 3.

Saran dari saya adalah untuk melakukan clustering sebaiknya dilakukan kepada variabel-variabel kuantitatif, agar sebaran data clusteringnya luas. Namun pada dataset ini atribut yang merupakan variabel kuantitatif tidak memiliki nilai korelasi antar atribut yang cukup untuk dilakukannya clustering.