LAPORAN OBSERVASI
TUGAS BESAR 01 – CLUSTERING

diajukan untuk memenuhi tugas mata kuliah (CII3C3) Pembelajaran Mesin

oleh:
Otniel Abiezer (NIM 1301180469)

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG
2021

1. FORMULASI MASALAH

Masalah yang akan diselesaikan adalah pada toko dealer mobil. Toko dealer tersebut memiliki data-data pelanggan yang mencakup ID, jenis kelamin, umur, sudah memiliki SIM atau belum, kode daerah, sudah pernah asuransi atau belum, umur kendaraan, kendaran pernah rusak atau tidak, premi, kanal penjualan, lama berlangganan dan tertarik atau tidak.

Untuk itu, dilakukan pengklasteran (pengelompokan) pelanggan-pelanggan tersebut untuk berdasarkan data yang ada. Pengelompokan itu akan dipakai untuk analisis lebih lanjut dari pihak dealer tersebut. Untuk itu dibangun model machine learning K-Means Clustering yang merupakan algoritma Unsupervised Learning yaitu Clustering yang dapat mengelompokkan pelanngan-pelanggan tersebut tanpa memperhatikan ketertarikan atau tidak.

2. EKSPLORASI DAN PERSIAPAN DATA

1.1.Eksplorasi Data

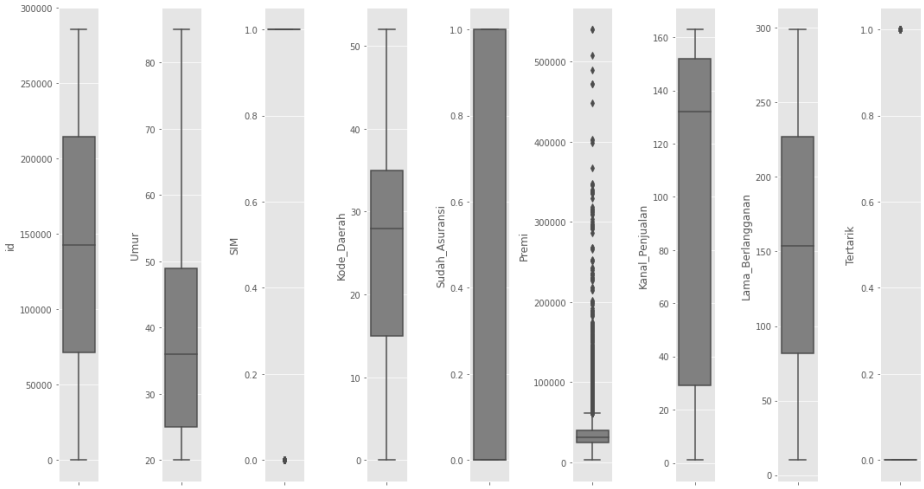
a. Mengetahui ukuran dataset (kendaraan_train.csv), yaitu 285831 x 12

b. Melihat banyaknya missing value dan tipe data setiap kolom

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285831 entries, 0 to 285830
Data columns (total 12 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   id                 285831 non-null  int64
 1   Jenis_Kelamin      271391 non-null  object
 2   Umur               271617 non-null  float64
 3   SIM                271427 non-null  float64
 4   Kode_Daerah        271525 non-null  float64
 5   Sudah_Asuransi     271602 non-null  float64
 6   Umur_Kendaraan     271556 non-null  object
 7   Kendaraan_Rusak    271643 non-null  object
 8   Premi              271262 non-null  float64
 9   Kanal_Penjualan    271532 non-null  float64
 10  Lama_Berlangganan  271839 non-null  float64
 11  Tertarik           285831 non-null  int64
dtypes: float64(7), int64(2), object(3)
memory usage: 26.2+ MB
```

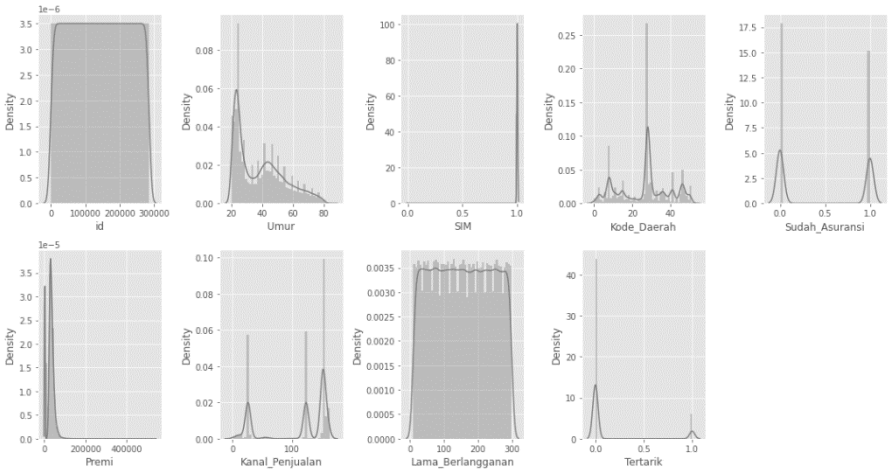
c. Melakukan pembagian kolom yang menjadi kategorikal (object) dan numerikal (int64, float64). Untuk kategorikal adalah Jenis_Kelamin, Umur_Kendaraan, Kendaraan_Rusak. Sisanya adalah numerikal.

d. Memvisualsasikan box plot untuk melihat outlier



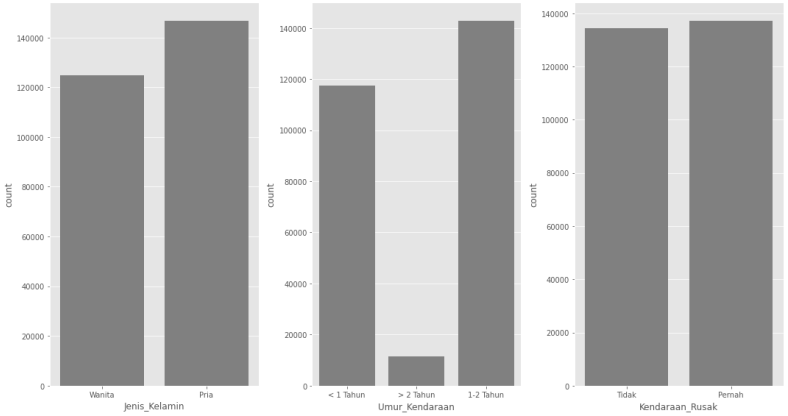
Selain Premi, tidak ada outlier pada masing-masing kolom.

e. Memvisualisasikan persebaran data



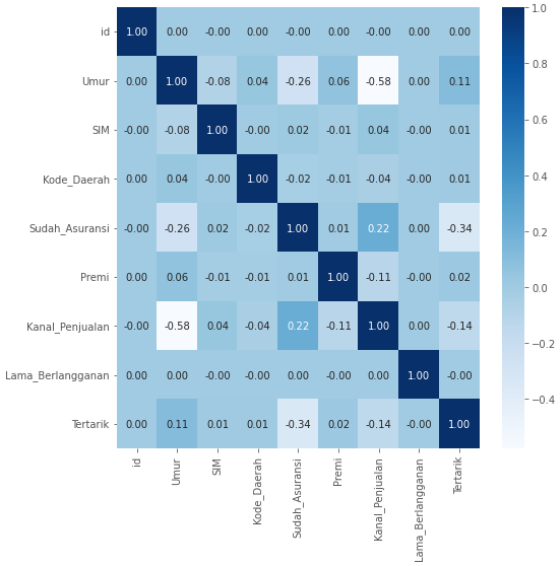
Untuk kolom SIM data imbalance sehingga tidak akan terlalu dipakai ke depannya, dan untuk Premi data skew (berat sebelah) serta skala angkanya terlalu besar. Selain itu, sudah cukup baik.

f. Visulasasi data kategorikal untuk melihat kemunculan



Banyak kategori hanya berkisar dari 2 sampai 3

g. Melihat korelasi



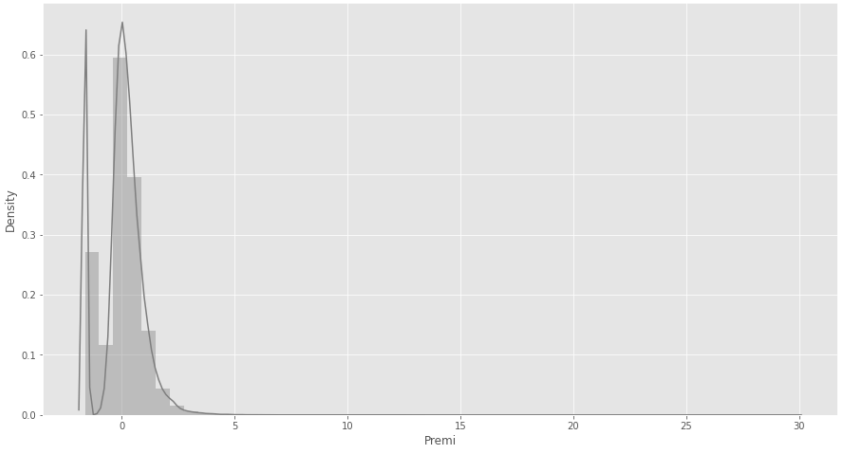
1.2.Persiapan Data

a. Menangani Missing Value

Melakukan drop (hapus) baris yang memiliki missing value. Setelah melakukan drop, banyaknya data menjadi 59.849% dari data asli.

b. Scaling Standarisasi

Dengan standarisasi pada kolom Premi agar angkanya tidak terlalu besar



c. Categorical Encoding

Mengubah 3 kolom tipe kategorikal menjadi numerikal, agar bisa dilakukan pembelajaran. Karena kategorinya tidak terlalu banyak (2 sampai 3 masing-masing), maka bisa dengan mendefinisikan angka untuk setiap kategori sebagai berikut:

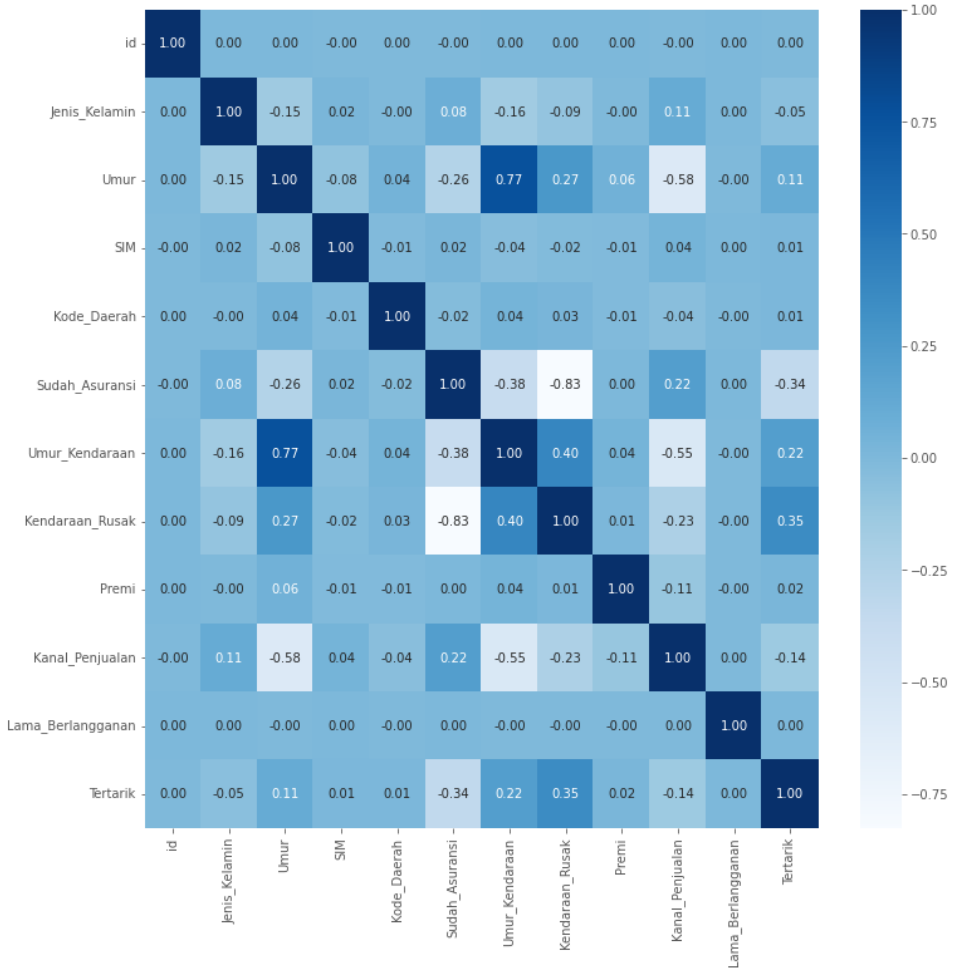
Jenis_Kelamin : 0 untuk laki-laki dan 1 untuk perempuan
Kendaraan_Rusak : 0 untuk tidak dan 1 untuk pernah
Umur_Kendaraan : 0 untuk < 1 tahun, 0.5 untuk 1 – 2 Tahun, 1 untuk > 2 Tahun

Sehingga tipe data semua kolom menjadi seperti ini

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 171068 entries, 0 to 285830
Data columns (total 12 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   id                 171068 non-null  int64
 1   Jenis_Kelamin      171068 non-null  float64
 2   Umur               171068 non-null  float64
 3   SIM                171068 non-null  float64
 4   Kode_Daerah        171068 non-null  float64
 5   Sudah_Asuransi     171068 non-null  float64
 6   Umur_Kendaraan     171068 non-null  float64
 7   Kendaraan_Rusak    171068 non-null  float64
 8   Premi              171068 non-null  float64
 9   Kanal_Penjualan    171068 non-null  float64
 10  Lama_Berlangganan  171068 non-null  float64
 11  Tertarik           171068 non-null  int64
dtypes: float64(10), int64(2)
memory usage: 17.0 MB
```

d. Feature Selection

Dengan menggunakan heatmap lagi untuk mencari korelasi setelah preprocessing di atas



Korelasi terbesar ada pada Kendaraan_Rusak dan Sudah_Asuransi (-0.83), untuk kedua terbesar ada pada Umur_Kendaraan dan Umur (0.77). Selain itu, Jenis_Kelamin juga bisa dimasukkan karena masih memiliki korelasi walaupun tidak sebesar sebelumnya. Untuk Tertarik tidak dimasukan (karena tidak memperhatikan kolom tertarik) dan juga Kanal_Penjualan tidak dimasukan karena hanya berisi kode untuk menghubungi pelanggan, walaupun memiliki korelasi cukup tetapi tidak akan menyebabkan sebab-akibat.

Sehingga kolom yang dipakai adalah Id (untuk identifikasi), Jenis_Kelamin, Umur, Sudah_Asuransi, Umur_Kendaraan, Kendaraan_Rusak.

e. Scaling Normalisasi

Karena semua kolom di atas memiliki range nilai dari 0-1, untuk itu kolom Umur dilakukan normalisasi agar nilainya juga sama dari 0-1. Sehingga nilai-nilai kolom menjadi seperti berikut

	id	Jenis_Kelamin	Umur	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak
count	171068.000000	171068.000000	171068.000000	171068.000000	171068.000000	171068.000000
mean	142794.020729	0.459338	0.289510	0.459794	0.303999	0.504443
std	82491.716355	0.498345	0.238949	0.498382	0.283813	0.499982
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	71350.750000	0.000000	0.076923	0.000000	0.000000	0.000000
50%	142768.000000	0.000000	0.246154	0.000000	0.500000	1.000000
75%	214070.250000	1.000000	0.446154	1.000000	0.500000	1.000000
max	285831.000000	1.000000	1.000000	1.000000	1.000000	1.000000

f. Hasil preprocessing final

Berikut 5 baris teratas dari file yang telah dilakukan proses persiapan (preprocessing)

	id	Jenis_Kelamin	Umur	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak
0	1	1.0	0.153846	1.0	0.0	0.0
1	2	0.0	0.430769	0.0	1.0	1.0
2	4	1.0	0.584615	0.0	0.5	0.0
3	6	0.0	0.015385	1.0	0.0	0.0
4	9	1.0	0.000000	1.0	0.0	0.0

3. PEMODELAN

3.1. Data diubah dahulu ke bentuk list untuk mempermudah pemodelan

3.2. Nilai K yang dipakai K=11 (akan dijelaskan di **5.EKSPERIMEN**)

3.3. Melakukan pendefinisian fungsi yang dipakai

```python
#Mencari jarak antar titik menggunakan Euclidean Distance
def EuclideanDistance(titik1, titik2):
    jumlah = 0
    for i in range (0,5):
        hasil = (titik1[i+1] - titik2[i+1]) ** 2
        jumlah += hasil
    return np.sqrt(jumlah)

#Melakukan Perubahan Centroid setelah dihitung rata2 semua titik yang masuk cluster tersebut
def geserCentroid(cluster):
  new_centroid = [0.0]
  for i in range(0,5):
    jumlah = 0
    for j in range(0, len(cluster)-1):
      jumlah += cluster[j+1][i+1]
    new_titik = jumlah / (len(cluster))
    new_centroid.append(new_titik)
  return new_centroid

#Mencari apakah suatu nilai ada pada suatu array / list
def is_found(anggota, arr):
  for i in range(0,len(arr)):
    if (arr[i] == anggota):
      return True
  return False
```

Penghitungan jarak yang dipakai adalah Euclidean Distance

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

GeserCentroid untuk menghitung total nilai pada masing-masing koordinat (fitur/kolom) lalu dijumlahkan dan dicari rata-ratanya (mean) sehingga menghasilkan koordinat centroid baru.

Is_Found adalah fungsi bantuan untuk mencari terdapatnya anggota di suatu array / list.

3.4. Insiasi Centroid dan juga List Cluster

```python
#Inisiasi Centroid dengan angka random berdasarkan banyak data
calon_sentroid = []
acak = np.random.randint(0,len(model)-1)
calon_sentroid.append(acak)

#Melakukan inisiasi angka random sebanyak K dan menjamin tidak ada angka yang sama
for i in range(0,K-1):
  while True:
    acak = np.random.randint(0,len(model)-1)
    if not is_found(acak, calon_sentroid):
      calon_sentroid.append(acak)
      break

list_centroid = []     #Berisi koordinat centroid-centroid
list_cluster = []      #Berisi cluster (sebanyak K) dan menampung titik-titik pada cluster tersebut

for i in range(0,3*K):
  if (0 <= i <= K-1):
    list_centroid.append(model[calon_sentroid[i]])                  #Mengisi titik Centroid berdasarkan angka random (yang menunjukan indeks) di atas
  elif (K <= i <= 2*K-1):
    list_cluster.append(["Cluster" + str(i-K)])                     #Menambahkan nama "Cluster ke-n" di awal list_cluster
  else:
    list_cluster[i-2*K].append(list_centroid[i-2*K])                #Memasukkan centroid sebagai anggota pertama masing-masing cluster

list_cluster
```

Centroid didapatkan secara random berdasarkan banyak data. Lalu dipilih data berdasarkan indeks yang didapat secara random yang akan menjadi Centroid.

Lalu melakukan inisiasi cluster yang di awal berisi Centroid tersebut.

3.5. Menghitung jarak pada suatu titik i kepada cluster sebanyak K. Jika terdapat jarak terpendeknya, dimasukan sesuai cluster nya

```
for i in range(0,len(model)):          #Melakukan loop untuk menghitu
  for j in range(0,K):                 #Melakukan loop untuk menghitu
    list_jarak.append(EuclideanDistance(list_centroid[j], model[i]))
  #endfor

  jarak_terdekat = min(list_jarak)              #Memilih jarak terpen

  for j in range(0,K):                          #Mencari jarak terdek
    if (list_jarak[j] == jarak_terdekat):
      list_cluster[j].append(model[i])
  #endfor


  list_jarak.clear()                          #Melakukan clear pada
```

Lalu, lakukan perulangan sebanyak datanya sampai semua data mendapatkan clusternya masing-masing.

3.6. Melakukan pergeseran centroid berdasarkan penghitungan function sebelumnya

```
list_new_centroid = []                          #Tempat calon
for i in range(0,K):
  list_new_centroid.append(geserCentroid(list_cluster[i]))
```

3.7. Melakukan loop secara terus-menerus dan kembali ke 3.5. sampai posisi centroid tidak bergeser lagi

```
if (list_new_centroid == list_centroid):
  break
else:
  list_centroid = list_new_centroid
```

3.8. Hasil output final dari pemodelan

```
Centroid 0  ada di titik  [0.0, 0.9999642013317105, 0.07323636483801124, 0.9999642013317105, 0.0, 0.03425932555308942]
Cluster 0  ada sebanyak  27933

Centroid 1  ada di titik  [0.0, 0.0, 0.07566342607112728, 0.9999514657348088, 0.0, 0.0]
Cluster 1  ada sebanyak  20603

Centroid 2  ada di titik  [0.0, 0.0, 0.07490174059517883, 0.0, 0.0, 0.999894213477203]
Cluster 2  ada sebanyak  9452

Centroid 3  ada di titik  [0.0, 0.0, 0.35866314652584136, 0.0, 0.5014235569422777, 0.9999609984399376]
Cluster 3  ada sebanyak  25639

Centroid 4  ada di titik  [0.0, 0.9999081642024061, 0.4663322901728629, 0.9999081642024061, 0.49995408210120307, 0.0]
Cluster 4  ada sebanyak  10888

Centroid 5  ada di titik  [0.0, 0.4725897920604915, 0.27085356987057757, 0.0, 0.2629962192816635, 0.0]
Cluster 5  ada sebanyak  10579

Centroid 6  ada di titik  [0.0, 0.0, 0.4786914909745347, 0.9999364231673978, 0.4999682115836989, 0.0]
Cluster 6  ada sebanyak  15728

Centroid 7  ada di titik  [0.0, 0.999903984637542, 0.06831271464973626, 0.0, 0.0, 0.999903984637542]
Cluster 7  ada sebanyak  10414

Centroid 8  ada di titik  [0.0, 0.0003732736095558044, 0.35169265225256463, 0.9996267263904441, 0.3516237402015677, 0.9996267263904441]
Cluster 8  ada sebanyak  2678

Centroid 9  ada di titik  [0.0, 0.9999589204288707, 0.44237578959674795, 0.033931725752783144, 0.5535472209670131, 0.9999589204288707]
Cluster 9  ada sebanyak  24342
```

4. EVALUASI

Menggunakan Silhouette Coefficient, alasah dipilih karena lebih cocok untuk tipe pembelajaran Clustering karena secara khusus menghitung kualitas klaster berdasarkan apakah data pada klaster tertentu bisa dengan mudah dibedakan dengan klaster lain dengan range nilai -1 sampai 1 di mana 1 yang terbaik, 0 berarti klaster tidak ada bedanya dengan klaster lain, dan -1 berarti klaster salah dimasukan ke klaster tertentu.

Rumusnya sebagai berikut

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Di mana nilai a(i) dan b(i)

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j) \qquad b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$$

Dengan source code sebagai berikut

```
def a_i(cluster1, titik_i):
  jumlah = 0
  for i in range(1,len(cluster1)):
    if (titik_i[0] != cluster1[i][0]):
      jumlah += EuclideanDistance(cluster1[i], titik_i)
  return (jumlah/(len(cluster1)-1))

def b_i(cluster_sisa, titik_i):
  jarak_antar_cluster = []
  for i in range(1,K):
    jumlah=0
    for j in range(1,len(cluster_sisa[i])-1):
      jumlah += EuclideanDistance(cluster_sisa[i][j], titik_i)
    jarak_antar_cluster.append(jumlah/len(cluster_sisa[i]))

  return min(jarak_antar_cluster)

def s_i(list_cluster):
  i = np.random.randint(1,len(list_cluster[0])-1)
  nilai_b_i = b_i(list_cluster, list_cluster[0][i])
  nilai_a_i = a_i(list_cluster[0], list_cluster[0][i])
  return (nilai_b_i - nilai_a_i) / (max(nilai_b_i, nilai_a_i))
```

Dan dilakukan penghitungan untuk beberapa titik pada suatu klaster

```
total = 0

for i in range(0,200):
  silhoutte_score = s_i(list_cluster)
  total += silhoutte_score
  print("Perulangan ke-", i, " dengan Silhoutte Score ", total/(i+1))

print("\nAverage Silhoutte Score = ", total/200)
```

Menghasilkan output sebagai berikut:

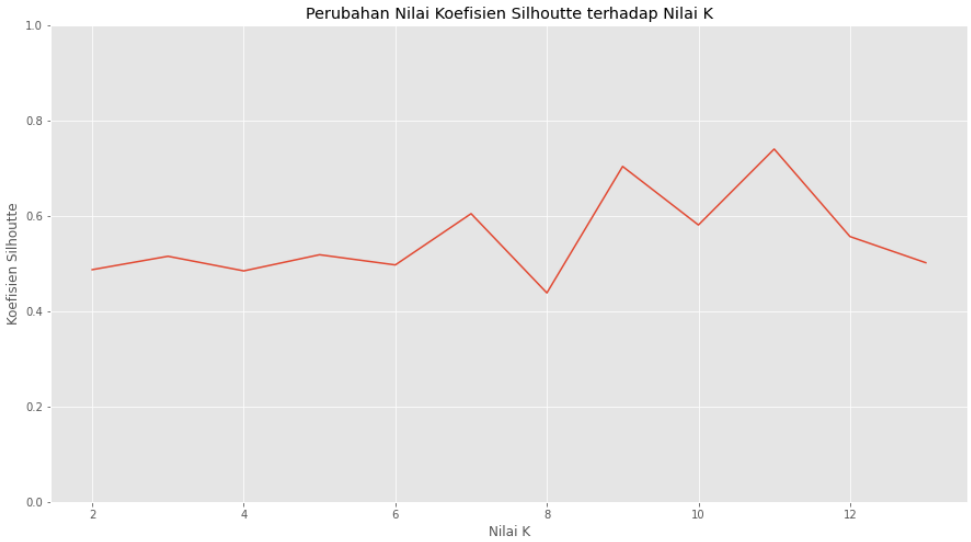
```
Perulangan ke- 197  dengan Silhoutte Score  0.8561522253315921
Perulangan ke- 198  dengan Silhoutte Score  0.8563092093702559
Perulangan ke- 199  dengan Silhoutte Score  0.8560950050866635

Average Silhoutte Score =  0.8560950050866635
```

Hasil Silhouette Coefficient yang didapatkan adalah 0.7402133287539158. Perlu diperhatikan contoh di atas adalah salah satu percobaan dan angka 0.74 didapatkan dengan melakukan beberapa kali percobaan dengan mengganti angka centroid awal dan menghitung rata-ratanya.

5. EKSPERIMEN

5.1. Mengganti nilai K

Nilai K diganti-ganti untuk mencari yang terbaik dengan Silhouette Coefficient



5.2. Mengubah handling missing value

Dengan mendrop baris dari 5 kolom yang dijadikan parameter. Data yang tersisa adalah 77.38%, hasil nilai yang dihasilkan tidak terlalu jauh dengan drop missing value sebelumnya dengan range silhouette yang dihasilkan 0.6 – 0.8.

5.3. Melakukan normalisasi kolom Premi, bukan standarisasi

Tidak memberikan perubahan yang signifikan.

6. VIDEO PRESENTASI

https://youtu.be/sj3-uHRt-Ho

7. KESIMPULAN

Metode Clustering dengan menggunakan K-Means-Clustering merupakan teknik machine learning yang baik untuk data yang tidak memiliki label untuk mengelompokan (mengklasterkan) pelanggan pada dari dealer mobil berdasarkan data yang ada. Hasil klaster tersebut dapat dianalisis lebih lanjut agar dapat mengambil keputusan dan tindakan yang tepat.

Semua proses diperlukan dari mendapatkan data, mengeksplorasi data di mana bisa melihat keanehan ataupun persebaran data, melakukan persiapan (pra-pemrosesan) data penting dilakukan agar hasil pembelajaran lebih maksimal, membangun model, dan menghasilkan nilai terbaik Silhouette Coefficient 0.74 dan nilai K=11.