Laporan Tubes 1 Machine Learning

Muhammad Rifqi Arrahim Natadikarta

1301190425

a. Formulasi Masalah

- Problem Statement
  Diberikan sebuah dataset kendaraan.csv yang memiliki 12 atribut.
  - Atribut apa yang memiliki korelasi kuat
  - Membuat klasterisasi berdasarkan atribut tersebut
  - Mencari nilai k yang optimal
- Goals
  - Mengetahui atribut mana yang memiliki korelasi paling kuat
  - Membuat model machine learning yang dapat mengklasterisasi atribut tersebut
  - Mengetahui nilai k yang optimal
- Solution Statements
  Untuk menyelesaikan permasalahan ini saya akan menggunakan K – Means clustering dan menggunakan elbow method untuk mencari nilai k yang optimal
- Data Understanding
  Id : id pelanggan
  Jenis_kelamin : jenis kelamin dari pelanggan
  Umur : umur dari pelanggan
  SIM : 1 (mempunyai SIM), 0 (tidak mempunyai SIM)
  Kode_Daerah : Kode Daerah pelanggan
  Sudah_Asuransi : 1 (mempunyai asuransi), 0 (tidak mempunyai asuransi)
  Umur_Kendaraan : Umur kendaraan dari pelanggan
  Premi : Jumlah premi dari pelanggan
  Kanal_Penjualan : Kanal Penjualan dari Pelanggan
  Lama_Berlangganan : Lamanya berlangganan dari pelanggan
  Tertarik : 1(tertarik membeli mobil), 0 (tidak tertarik membeli mobil)

b. Eksplorasi dan Persiapan Data

- Menghapus kolom Tertarik sesuai deskripsi Tugas Besar

```
[ ]  df = df.drop(columns='Tertarik')
```

```
[ ]  df.head()
```

	id	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan
0	1	Wanita	30.0	1.0	33.0	1.0	< 1 Tahun	Tidak	28029.0	152.0	97.0
1	2	Pria	48.0	1.0	39.0	0.0	> 2 Tahun	Pernah	25800.0	29.0	158.0
2	3	NaN	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	32733.0	160.0	119.0
3	4	Wanita	58.0	1.0	48.0	0.0	1-2 Tahun	Tidak	2630.0	124.0	63.0
4	5	Pria	50.0	1.0	35.0	0.0	> 2 Tahun	NaN	34857.0	88.0	194.0

- Menghapus kolom id karena tidak relevan untuk clustering

```
# drop kolom yang tidak diperlukan
data = df.drop(columns=['id'])
```

- Mengganti kolom dengan nilai numerical

```
# jalankan proses one-hot encoding dengan pd.get_dummies()
data = pd.get_dummies(data)
data
```

	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Premi	Kanal_Penjualan	Lama_Berlangganan	Jenis_Kelamin_Pria	Jenis_Kelamin_Wanita	Umur_Kendaraan_1-2 Tahun	Umur_Kendaraan_< 1 Tahun	Umur_Kendaraan_> 2 Tahun	Kendaraan_Rusak_Pernah
0	30.0	1.0	33.0	1.0	28029.0	152.0	97.0	0	1	0	1	0	0
1	48.0	1.0	39.0	0.0	25800.0	29.0	158.0	1	0	0	0	1	1
2	21.0	1.0	46.0	1.0	32733.0	160.0	119.0	0	0	0	1	0	0
3	58.0	1.0	48.0	0.0	2630.0	124.0	63.0	0	1	1	0	0	0
4	50.0	1.0	35.0	0.0	34857.0	88.0	194.0	1	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
285826	23.0	1.0	4.0	1.0	25988.0	152.0	217.0	0	1	0	1	0	0
285827	21.0	1.0	46.0	1.0	44686.0	152.0	50.0	0	1	0	1	0	0
285828	23.0	1.0	50.0	1.0	49751.0	152.0	226.0	0	1	0	1	0	0
285829	68.0	1.0	7.0	1.0	30503.0	124.0	270.0	1	0	1	0	0	0
285830	45.0	1.0	28.0	0.0	36480.0	26.0	44.0	1	0	1	0	0	1

285831 rows × 14 columns

- Menghapus kolom yang kurang relevan

```
[ ] data = data.drop(columns=['SIM','Sudah_Asuransi','Jenis_Kelamin_Pria','Jenis_Kelamin_Wanita','Umur_Kendaraan_1-2 Tahun','Umur_Kendaraan_< 1 Tahun','Umur_Kendaraan_> 2 Tahun','Kendaraan_Rusak_Pernah','Kendaraan
```

- Menghapus missing value

```
[ ] count_nan_in_data = data.isnull().sum().sum()
    print ('Count of NaN: ' + str(count_nan_in_data))
```

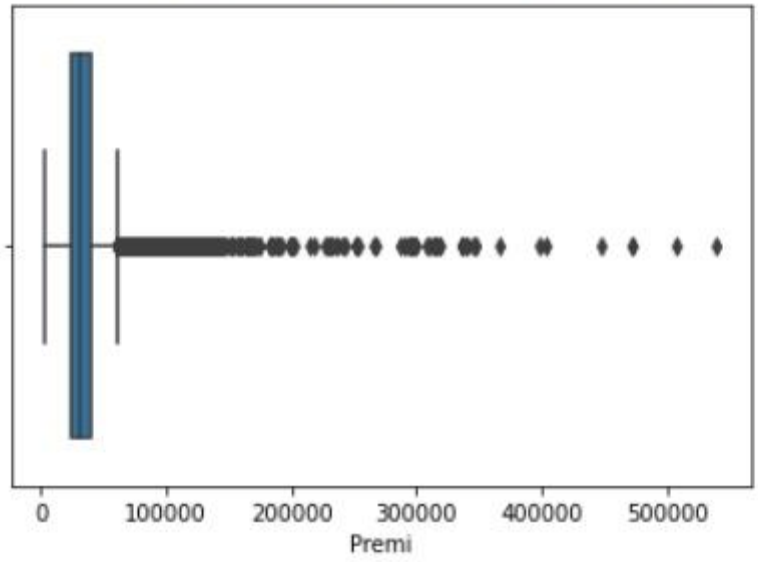
```
Count of NaN: 71380
```

```
[ ] data=data.dropna()
```

- Mengecek outliers

```
[ ]  sns.boxplot(x=data['Premi'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f876cef6cd0>
```



- Menghapus outliers

menghapus data outliers/pencilan

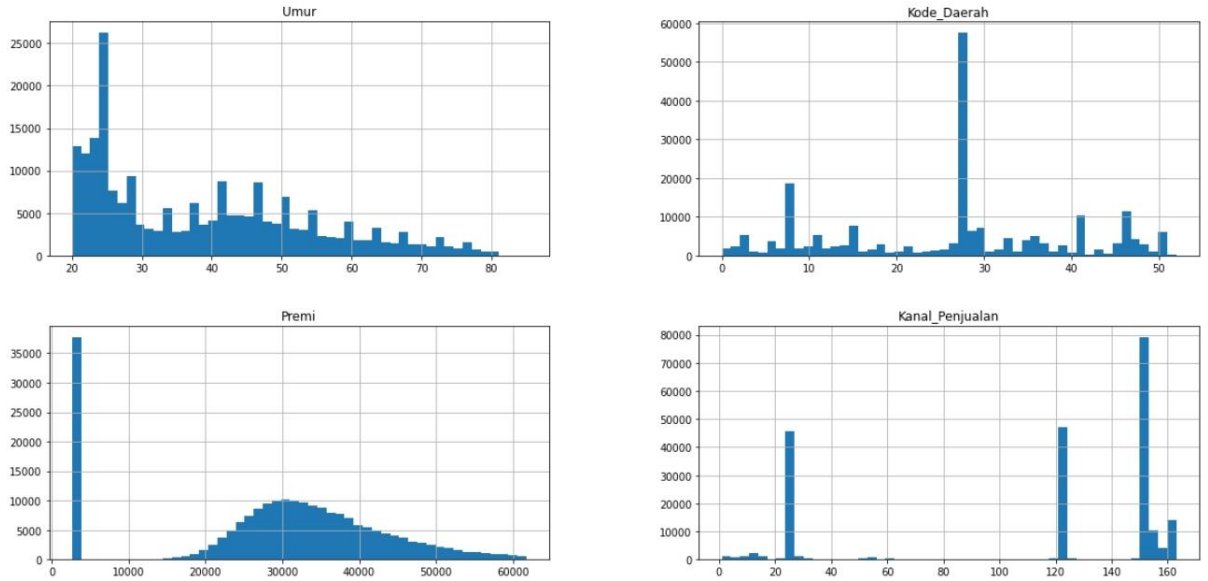
```
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
IQR=Q3-Q1
data=data[~((data<(Q1-1.5*IQR))|(data>(Q3+1.5*IQR))).any(axis=1)]

# Cek ukuran dataset setelah kita drop outliers
data.shape
```

```
(215198, 5)
```

- Melihat sebaran data

```
[ ]  data.hist(bins=50, figsize=(20,15))
     plt.show()
```

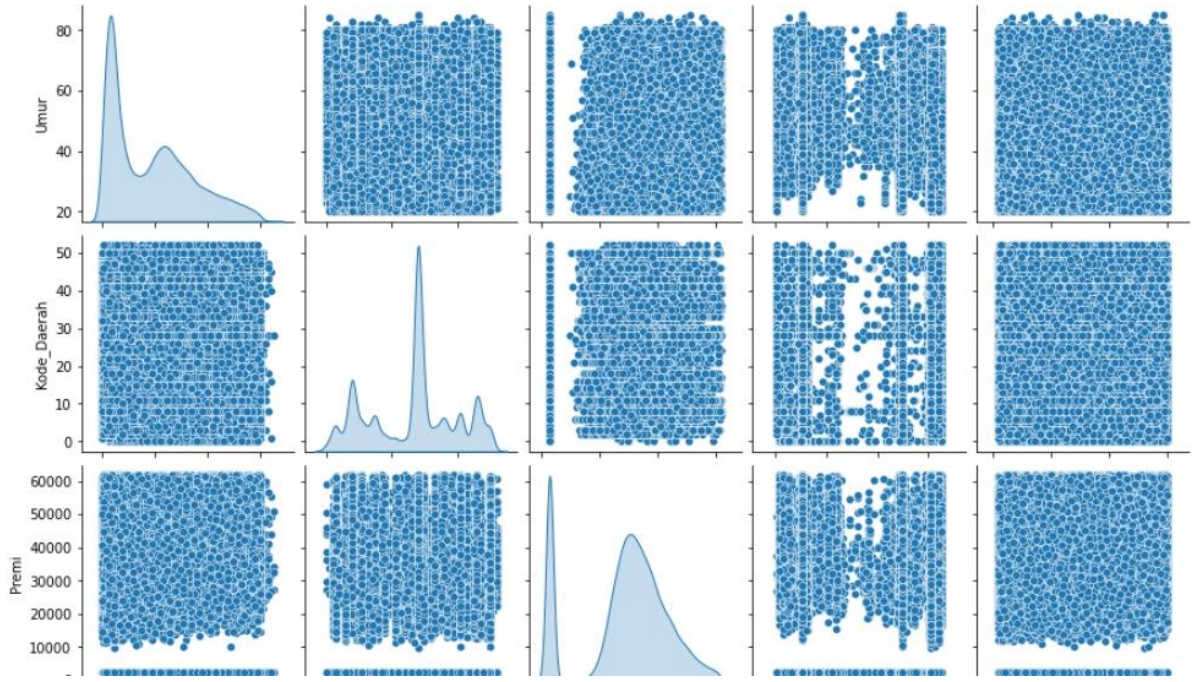


- Melihat korelasi

menggunakan pairplot untuk melihat korelasi antar fitur

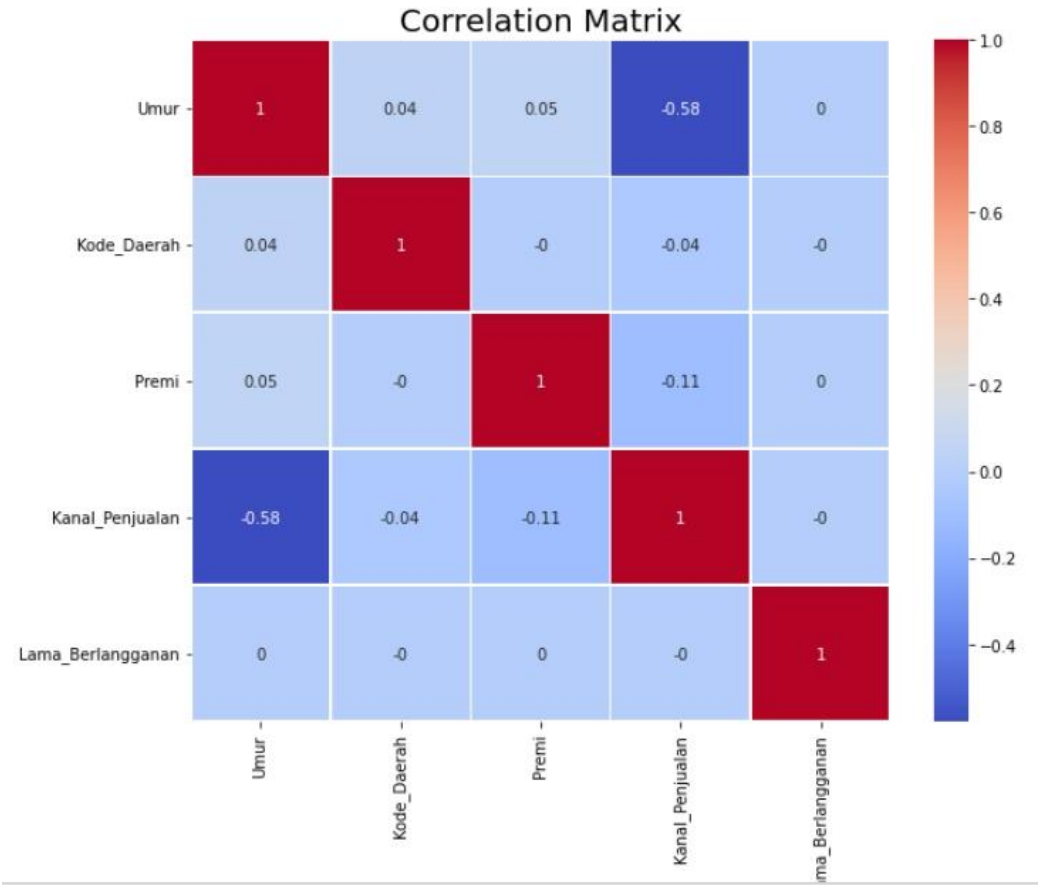
```
# mengamati hubungan antar fitur numerik dengan fungsi pairplot()
sns.pairplot(data, diag_kind = 'kde')
```

```
<seaborn.axisgrid.PairGrid at 0x7f876c7b1150>
```



```
sns.heatmap(data=correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5, )
plt.title("Correlation Matrix", size=20)
```

```
Text(0.5, 1.0, 'Correlation Matrix')
```



- Memilih atribut dengan nilai korelasi paling besar

  berdasarkan visualisasi diatas, atribut yang memiliki nilai korelasi paling besar adalah antara umur dan Kanal_Penjualan

```
[ ] data.drop(['Kode_Daerah','Premi','Lama_Berlangganan'], axis=1, inplace=True)
    data.head()
```

	Umur	Kanal_Penjualan
0	30.0	152.0
1	48.0	29.0
2	21.0	160.0
3	58.0	124.0
4	50.0	88.0

- Mengubah dataframe menjadi array

Mengubah data menjadi numpy array

```
[ ]  data.to_numpy()
```

```
array([[ 30., 152.],
       [ 48.,  29.],
       [ 21., 160.],
       ...,
       [ 23., 152.],
       [ 68., 124.],
       [ 45.,  26.]])
```

- Melakukan standarisasi

memanggil MinMaxScaler dari library sklearn, standarisasi adalah teknik yang digunakan untuk memusatkan kolom fitur pada mean 0 dengan standar deviasi 1 sehingga kolom fitur memiliki parameter yang sama dengan distribusi normal standar

```
[ ]  from sklearn.preprocessing import MinMaxScaler
     scaler = MinMaxScaler()
     scaler.fit(data)
     data = scaler.transform(data)
```

c. Pemodelan

- Inisiasi nilai K, nilai toleransi perpindahan centroids, banyaknya iterasi

```
def __init__(self,k):
  self.k = k
  # self.tol merupakan nilai toleransi yang sesuai dengan library sk.learn
  self.tol = 0.0001
  # self.iter merupakan banyaknya iterasi yang akan dilakukan sesuai dengan library sk.learn
  self.max_iter = 300
  # self.sse merupakan nilai sum square error dari hasil clustering
  self.sse = 0
```

- Inisiasi centroid

```
def fit(self, data):
  #self.centroids digunakan untuk menyimpan nilai centroids
  self.centroids = {}
  #mengacak sample data agar mendapat centroid baru
  np.random.shuffle(data)
  for i in range(self.k):
    #mengisi centroid sebanyak nilai self.k
    self.centroids[i] = data[i]
```

- Menghitung jarak dengan centroid, lalu memilih jarak terpendek

```
for i in range(self.k):
  #membuat array sebanyak self.k
  self.classifications[i] = []
for featureset in data:
  #distances untuk menghitung jarak data dengan centroid
  distances = [np.linalg.norm(featureset-self.centroids[centroid])for centroid in self.centroids]
  #mengambil index jarak terpendek dari array distances
  classification = distances.index(min(distances))
  #masukan kedalam array self.current_distances
  self.current_distances.append(distances[classification])
  #masukan data ke dalan self.classification
  self.classifications[classification].append(featureset)
```

- Tentukan centroid baru

```
for classification in self.classifications:
  #menghitung rata rata data untuk mendapatkan centroid baru
  self.centroids[classification] = np.average(self.classifications[classification], axis = 0)
```

- Berhenti melakukan training jika centroid tidak berubah secara signifikan (dibawa nilai toleransi), lalu hitung nilai SSE

```
for c in self.centroids:
  original_centroid = prev_centroids[c]
  current_centroid = self.centroids[c]
  print(c)
  print(original_centroid)
  print(current_centroid)
  #jika perubahan centroid baru dengan centroid lama melebihi self.tol maka hasil clustering belum optimal
  if np.sum(((current_centroid-original_centroid))/original_centroid*100.0)>self.tol:
    optimized = False
#jika data hasil clustering sudah optimal hitung sse dan keluar dari looping
if optimized:
  for se in self.current_distances:
    self.sse = self.sse + (se**2)
  break
```

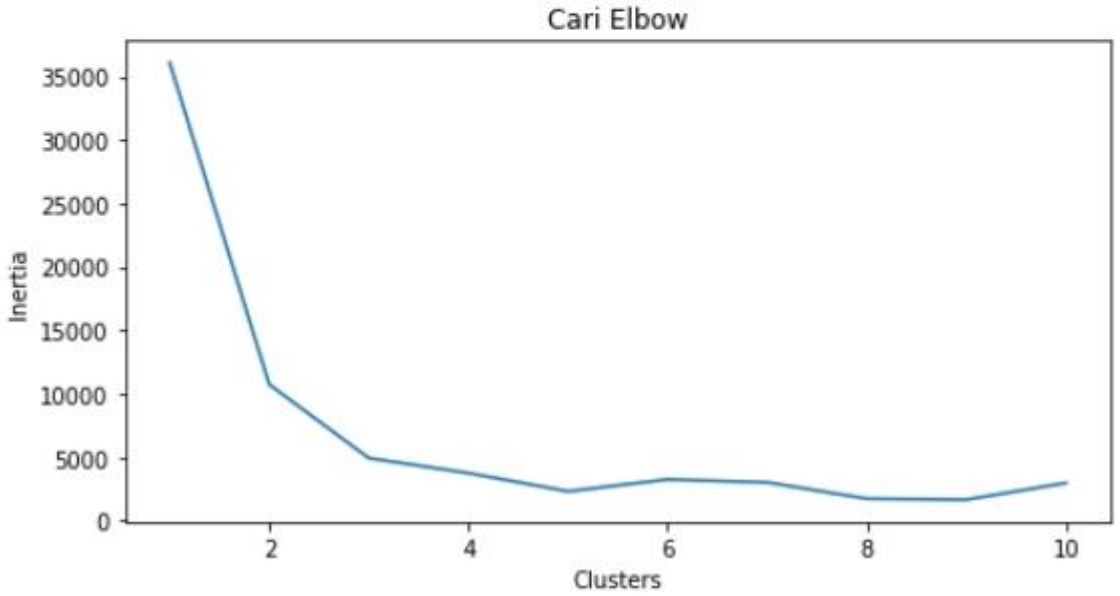
d. Evaluasi

Kita akan menentukan nilai k yang optimal dengan menggunakan elbow method. Hitung nilai SSE k = 1 sampai k = 10, lalu visualisasikan

```
[ ]  #sse untuk menampung nilai sse dari k = 1 sampai k = 10
     sse = []
     for i in range(1,11):
         clf=K_Means(i)
         clf.fit(data)
         sse.append(clf.sse)
     print(sse)
```

```
[ ]  fig, ax = plt.subplots(figsize=(8, 4))
     sns.lineplot(x=list(range(1, 11)), y=sse, ax=ax)
     ax.set_title('Cari Elbow')
     ax.set_xlabel('Clusters')
     ax.set_ylabel('Inertia')
```

```
Text(0, 0.5, 'Inertia')
```

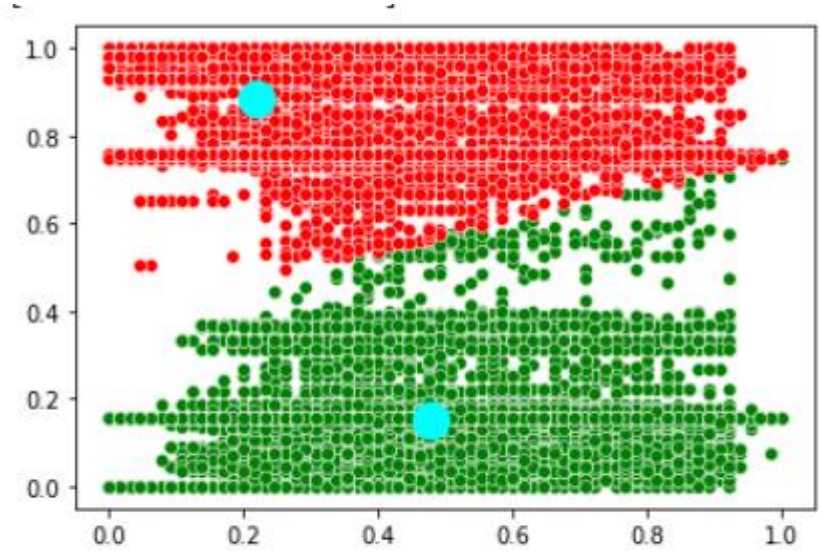


Nilai k yang optimal adalah 2, karena memiliki sudut terkecil diantara yang lainnya

e. Eksperimen

Memanggil K_Means dengan nilai k = 2

```
clf = K_Means(2)
clf.fit(data)
#k1 untuk menampung data cluster pertama
k1 = np.reshape(clf.classifications[0], (-1))
#k2 untuk menampung data cluster kedua
k2 = np.reshape(clf.classifications[1], (-1))
#x1 untuk menampung umur dari k1
x1 = []
#y1 untuk menampung kanal_penjualan dari k1
y1 = []
for i in range(len(k1)):
  if (i%2)==0:
    x1.append(k1[i])
  else:
    y1.append(k1[i])
#x2 untuk menampung umur dari k2
x2 = []
#y2 untuk menampung kanal_penjualan daru k2
y2 = []
for i in range(len(k2)):
  if (i%2)==0:
    x2.append(k2[i])
  else:
    y2.append(k2[i])
#tentukan warna
sns.scatterplot(x=x1,y=y1,color=colors[0])
sns.scatterplot(x=x2,y=y2,color=colors[1])
#beri tanda untuk centroid
for centroid in clf.centroids:
  plt.scatter(x=clf.centroids[centroid][0], y=clf.centroids[centroid][1],
              marker="o", color="cyan", s=150, linewidths=5)
```



f. Kesimpulan

Berdasarkan hasil diatas, atribut umur dank anal penjualan memiliki nilai korelasi paling tinggi diantara pasangan atribut yang lainnya, namun kanal_penjualan memiliki sebaran data yang kurang bagus sehingga bisa menjadi bias dan mempengaruhi hasil akhir. Setelah menghitung SSE dari nilai k = 1 sampai k = 10, nilai k = 2 merupakan nilai k yang optimal, karena nilai k = 2 membentuk sudut paling lancip diantara nilai k yang lain. Setelah itu kita lakukan clustering dengan nilai k = 2, warna hijau untuk kluster 1, warna merah untuk kluster 2, warna cyan untuk centroid.

Lampiran

Source Code : https://colab.research.google.com/drive/1AOnIUvTn2t4tv8n_P-YW979s-GmNX_nP?usp=sharing

Dataset: https://drive.google.com/drive/folders/14QPe3o6LeSjfYj-kGhCZJM4pn-I55YsJ

Video : **https://youtu.be/3anWT3c1kz0**