Implementasi K-Means Clustering Pada Data Kendaraan

Disusun Untuk Memenuhi Tugas Besar Mata Kuliah Pembelajaran Mesin

Disusun Oleh:

Fadhlurrahman Akbar Nasution (1301194258)

PROGRAM STUDI INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

2020/2021

Daftar Isi

1. Formulasi Masalah

Pada tugas besar ini penulis diminta untuk membuat pemodelan yang dapat mengelompokkan pelanggan berdasarkan data pelanggan di dealer tanpa memperhatikan label kelas apakah pelanggan tertarik untuk membeli kendaraan baru atau tidak. Dataset yang digunakan dalam pemodelan ini adalah kendaraan_train.csv. Pada dataset ini ada beberapa atribut seperti id, Jenis_Kelamin, Umur, SIM, Kode_Daerah, Sudah_Asuransi, Umur_Kendaraan, Kendaraan_Rusak, Premi, Kanal_Penjualan, Lama_Berlangganan, dan Tertarik. Data yang diberikan berisi 285.831 record yang dimana ada beberapa data yang tidak lengkap sehingga dibutuhkan pra-pemrosesan data sebelum melakukan clustering.

Ada beberapa algoritma clustering yang dapat digunakan dalam melakukan pemodelan clustering seperti : K-Means, Agglomerative, EM Clustering, Fuzzy C-Means, K-D Trees, Quality Threshold, dan lainya. Dilihat dari dataset yang memiliki data yang banyak, maka penulis memilih untuk menggunakan K-Means sebagai algoritma clustering untuk melakukan pemodelan dari data tersebut.

2. Eksplorasi dan Persiapan Data

A. Reading dan Understanding Data

Disini penulis melakukan pembacaan data dengan data yang digunakan yaitu dataset kendaraan_train.csv.

Memasukan dataset yang akan digunakan

```
df_train = pd.read_csv('Data/kendaraan_train.csv')
df_train
```

0.3s

	id	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	1	Wanita	30.0	1.0	33.0	1.0	< 1 Tahun	Tidak	28029.0	152.0	97.0	0
1	2	Pria	48.0	1.0	39.0	0.0	> 2 Tahun	Pernah	25800.0	29.0	158.0	0
2	3	NaN	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	32733.0	160.0	119.0	0
3	4	Wanita	58.0	1.0	48.0	0.0	1-2 Tahun	Tidak	2630.0	124.0	63.0	0
4	5	Pria	50.0	1.0	35.0	0.0	> 2 Tahun	NaN	34857.0	88.0	194.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
285826	285827	Wanita	23.0	1.0	4.0	1.0	< 1 Tahun	Tidak	25988.0	152.0	217.0	0
285827	285828	Wanita	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	44686.0	152.0	50.0	0
285828	285829	Wanita	23.0	1.0	50.0	1.0	< 1 Tahun	Tidak	49751.0	152.0	226.0	0
285829	285830	Pria	68.0	1.0	7.0	1.0	1-2 Tahun	Tidak	30503.0	124.0	270.0	0
285830	285831	Pria	45.0	1.0	28.0	0.0	1-2 Tahun	Pernah	36480.0	26.0	44.0	0

285831 rows × 12 columns

Setelah melakukan read data saya melakukan pengecekan terhadap data yang telah saya masukan tersebut. Tujuan dari pengecekan data ini untuk mengetahui tipe data dari kolom tersebut dan untuk mengetahui apakah ada data yang kosong, tidak lengkap, ataupun duplikat. Implementasi pada pengkodingannya sebagai berikut:

```
df_train.dtypes
✓ 0.3s
id                    int64
Jenis_Kelamin        object
Umur                float64
SIM                 float64
Kode_Daerah         float64
Sudah_Asuransi      float64
Umur_Kendaraan       object
Kendaraan_Rusak      object
Premi               float64
Kanal_Penjualan     float64
Lama_Berlangganan   float64
Tertarik              int64
dtype: object
```

```
df_train.isna().sum()
✓ 0.4s
Jenis_Kelamin        14440
Umur                 14214
SIM                  14404
Kode_Daerah          14306
Sudah_Asuransi       14229
Umur_Kendaraan       14275
Kendaraan_Rusak      14188
Premi                14569
Kanal_Penjualan      14299
Lama_Berlangganan    13992
dtype: int64
```

```
#pengecekan data yang duplikat
df_train[df_train.duplicated()]
✓ 0.2s
```

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan
24277	1.0	21.0	1.0	14.0	1.0	0.0	0.0	2630.0	160.0	202.000000
26597	1.0	50.0	1.0	28.0	0.0	1.0	1.0	31646.0	26.0	154.286302
26774	1.0	41.0	1.0	28.0	0.0	1.0	1.0	2630.0	124.0	76.000000
30907	1.0	25.0	1.0	28.0	0.0	0.0	1.0	2630.0	152.0	127.000000
34700	1.0	43.0	1.0	28.0	0.0	1.0	1.0	31646.0	124.0	285.000000
...	...	...	...	...	...	...	...	...	...	...
283840	1.0	40.0	1.0	28.0	0.0	1.0	1.0	2630.0	26.0	233.000000
284038	0.0	46.0	1.0	28.0	0.0	1.0	1.0	31646.0	24.0	154.286302
285037	1.0	23.0	1.0	8.0	1.0	0.0	0.0	2630.0	152.0	71.000000
285254	0.0	43.0	1.0	28.0	0.0	1.0	1.0	2630.0	124.0	252.000000
285499	1.0	52.0	1.0	28.0	0.0	1.0	1.0	31646.0	26.0	154.286302

404 rows × 10 columns

B. Mengubah Data String Menjadi Integer Dan Drop Data

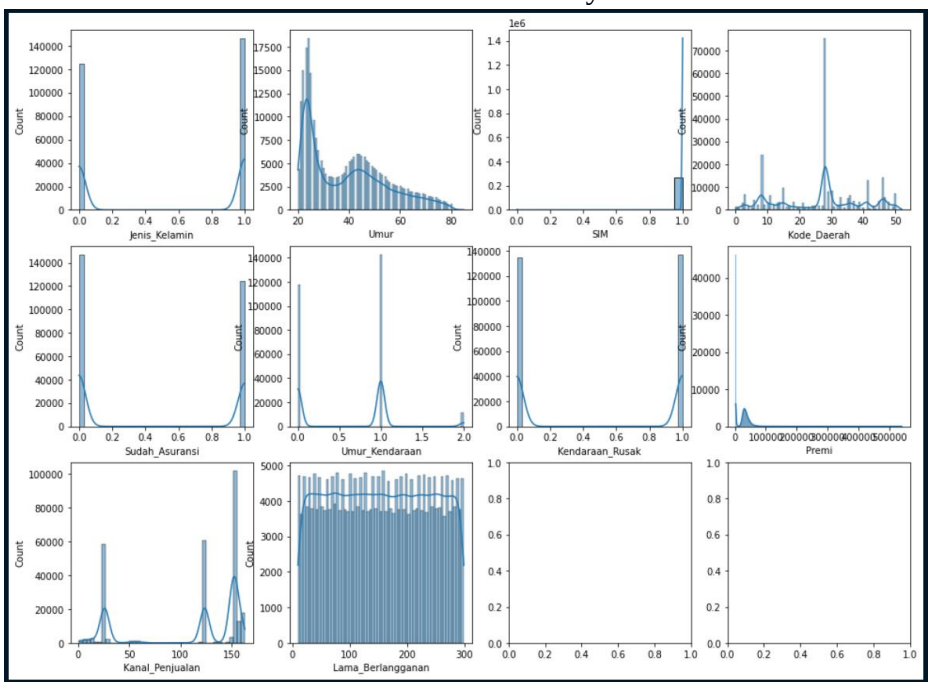
Setelah melakukan pengecekan data tersebut penulis mengubah data yang bertipe string ke integer. Pada Data tersebut penulis mengubah value dari kolom Jenis_Kelamin, Umur_Kendaraan, dan Kendaraan_Rusak. Dan penulis juga melakukan drop terhadap data yang tidak diperlukan seperti data ID dan Tertarik. Implementasi pada pengkodingannya sebagai berikut.

```
#mengubah data dari string enjadi integer agar mudah diproses
df_train['Jenis_Kelamin']= df_train['Jenis_Kelamin'].str.replace('Pria','1')
df_train['Jenis_Kelamin']= df_train['Jenis_Kelamin'].str.replace('Wanita','0')
df_train['Kendaraan_Rusak']= df_train['Kendaraan_Rusak'].str.replace('Pernah','1')
df_train['Kendaraan_Rusak']= df_train['Kendaraan_Rusak'].str.replace('Tidak','0')
df_train['Umur_Kendaraan']= df_train['Umur_Kendaraan'].str.replace('< 1 Tahun','0')
df_train['Umur_Kendaraan']= df_train['Umur_Kendaraan'].str.replace('1-2 Tahun','1')
df_train['Umur_Kendaraan']= df_train['Umur_Kendaraan'].str.replace('> 2 Tahun','2')

#drop kolom yang tidak digunakan
del df_train['id']
del df_train['Tertarik']
#ubah Nilai menjadi numeric
df_train['Jenis_Kelamin']= pd.to_numeric(df_train['Jenis_Kelamin'])
df_train['Kendaraan_Rusak']= pd.to_numeric(df_train['Kendaraan_Rusak'])
df_train['Umur_Kendaraan']= pd.to_numeric(df_train['Umur_Kendaraan'])
✓ 1.4s
```

C. Handling Missing Value

Penulis berencana menggunakan mean/modus/median dalam melakukan handling missing value yang ada pada data tersebut. Sebelum penulis melakukan handling missing value, penulis terlebih dahulu melihat distribusi datanya terlebih dahulu.



Setelah melihat distribusi data tersebut maka saya melakukan beberapa pemilihan terhadap handling missing value dengan memperhatikan center tendency. Penjelasan pemilihannya sebagai berikut:

1) Median

Penulis menggunakan median jika distribusi data mengalami kemiringan(skewness) atau value dari kolom yang dipilih bertipe ordinal.

2) Mean

Penulis menggunakan mean jika distribusi data normal yang tidak memiliki kemiringan(skewness) atau bisa dibilang distribusi data kontinu dan simetris dan value dari kolom bertipe interval/ratio.

3) Modus

Penulis menggunakan modus jika data bertipe nominal atau kategorik. Perbedaanya dengan median adalah karena jika valuenya memiliki jarak yang jauh maka penulis menggunakan modus dalam handling missing valuenya.

Berikut merupakan implementasi dari codingan data dari handling missing value:

```
#Menggunakna Mean Ketika data memiliki distribusi normal
df_train['Lama_Berlangganan'].fillna(df_train['Lama_Berlangganan'].mean(), inplace=True)

#Menggunakna Mode(modus) jika data kategorikal dan perbedaan hasilnya sangat jauh
col_mode = ['SIM','Kode_Daerah','Kanal_Penjualan']
for x in col_mode:
    df_train[x].fillna(df_train[x].mode()[0], inplace=True)
#Menggunakan Median jika distribusi data memiliki kemiringan(skewness)
col_median = ['Umur','Jenis_Kelamin','Umur_Kendaraan','Premi','Sudah_Asuransi','Kendaraan_Rusak']
for i in col_median:
    df_train[i].fillna(df_train[i].median(), inplace=True)
```

0.1s

D. Drop Duplikat Data

Kemudian penulis melakukan drop duplikat data agar data yang akan dimodel kan nanti dapat menghasilkan hasil yang lebih baik. Berikut implementasinya pada pengkodeannya:

```
#Melakukan drop dengan data yang duplikat
df_train=df_train.drop_duplicates()
df_train
```

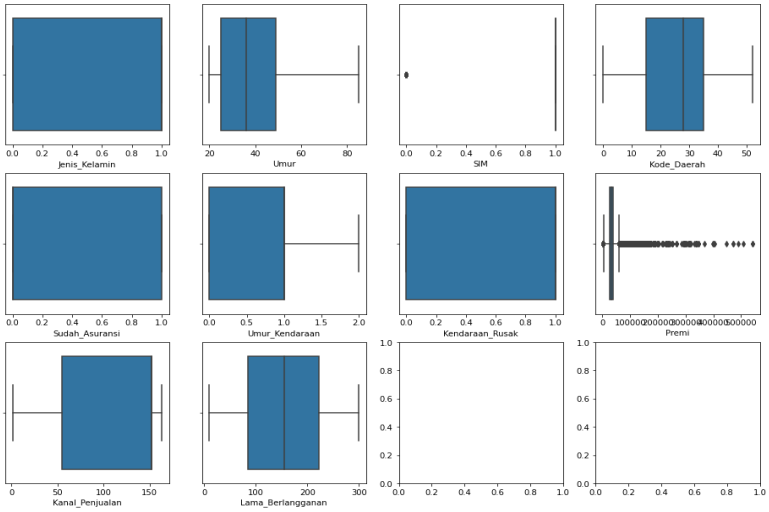
0.2s

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan
0	0.0	30.0	1.0	33.0	1.0	0.0	0.0	28029.0	152.0	97.0
1	1.0	48.0	1.0	39.0	0.0	2.0	1.0	25800.0	29.0	158.0
2	1.0	21.0	1.0	46.0	1.0	0.0	0.0	32733.0	160.0	119.0
3	0.0	58.0	1.0	48.0	0.0	1.0	0.0	2630.0	124.0	63.0
4	1.0	50.0	1.0	35.0	0.0	2.0	1.0	34857.0	88.0	194.0
...	...	...	...	...	...	...	...	...	...	...
285826	0.0	23.0	1.0	4.0	1.0	0.0	0.0	25988.0	152.0	217.0
285827	0.0	21.0	1.0	46.0	1.0	0.0	0.0	44686.0	152.0	50.0
285828	0.0	23.0	1.0	50.0	1.0	0.0	0.0	49751.0	152.0	226.0
285829	1.0	68.0	1.0	7.0	1.0	1.0	0.0	30503.0	124.0	270.0
285830	1.0	45.0	1.0	28.0	0.0	1.0	1.0	36480.0	26.0	44.0

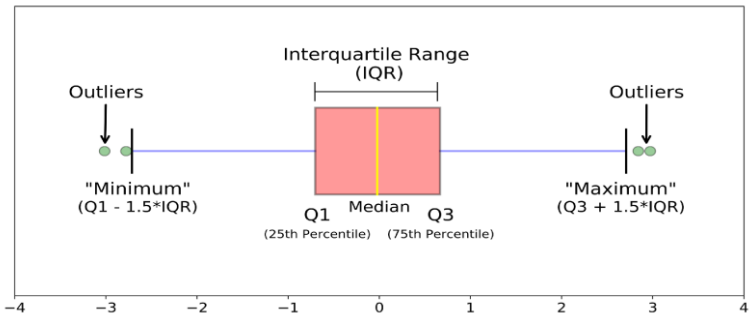
285427 rows × 10 columns

E. Handling Outlier

Penulis melakukan penampilan data menggunakan boxplot untuk mengetahui apakah ada pencilan(outlier) yang ada pada data tersebut.



Dari boxplot diatas didapatkan bahwa yang memiliki pencilan adalah premi dan SIM. Alasan penulis tidak menggunakan atribut SIM karena atribut SIM berjenis kategori yang dimana saat menggunakan K-Means tidak baik untuk melakukan klasterisasi kelompok dengan data yang berjenis kategorikal. Sehingga saya melakukan pembersihan outlier atribut dengan menggunakan Interquartile Range. Formula dan pengkodeannya Sebagai berikut:



Formula:

Lowerbound = Q1 - 1.5(IQR)

Upperbound = Q3 + 1.5(IQR)

Interquartile Range = Q3 - Q1

Code :

```
#Membuat pembersihaan outlier
def outlier_detection(data_column):
    sorted(data_column)
    Q1 ,Q3 = np.percentile(data_column,[25,75])
    IQR = Q3 - Q1
    lower_range = Q1-(1.5 * IQR)
    upper_range = Q3+(1.5 * IQR)
    return lower_range, upper_range
```

✓ 0.3s

Pada Interquartile ini hanya berfokus dalam mendeteksi outliernya. Maka kita akan menggunakan formula untuk menghapus outlier yang ada pada data tersebut. Berikut formula yang digunakan.

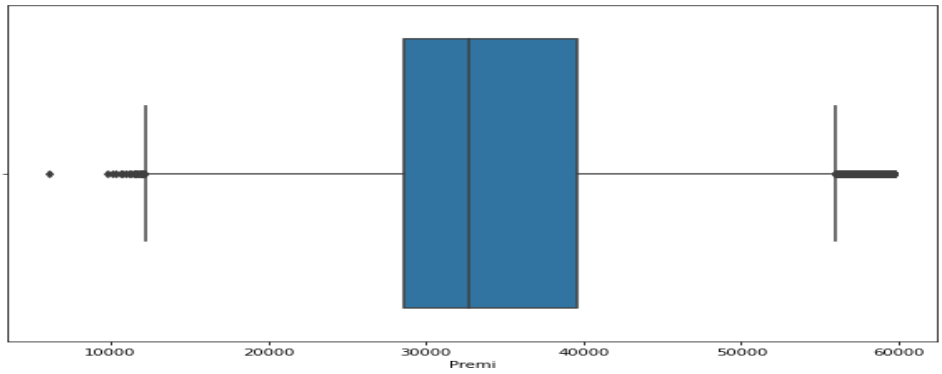
$$Data \in (x \geq Lowerbound) and (x \leq Upperbound)$$

Maka Implementasinya pada pengkodean akan menjadi seperti gambar dibawah ini.

```
# Pembersihan data outlier premi
lowerbound, upperbound = outlier_detection(df_train['Premi'])
df_train = df_train[~((df_train['Premi']<lowerbound) | (df_train['Premi']>upperbound))]
```

✓ 0.1s

Setelah melakukan proses diatas didapatkan hasil pembersihan sebagai berikut.



F. Normalisasi

Disini penulis menggunakan metode z-score untuk melakukan normalisasi data. Adapun formula yang dimiliki oleh z-score adalah sebagai berikut.

$$x_{std} = \frac{x-\mu}{\sigma}$$

Dari formula yang telah didapatkan penulis membuat implementasinya pada pengkodean sebagai berikut.

```
# Melakukan Normalisasi Data Menggunakan z-score method
df_trainNormalisasi = (df_train - df_train.mean())/(df_train.std())
```

Setelah melakukan normalisasi didapatkan hasil sebagai berikut.

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan
0	-1.122799	-0.534762	0.044711	0.506524	1.092527	-1.069754	-1.024379	-0.763296	0.710121	-0.701261
1	0.890628	0.656147	0.044711	0.978349	-0.915305	2.491438	0.976197	-1.025075	-1.606579	0.045787
2	0.890628	-1.130216	0.044711	1.528812	1.092527	-1.069754	-1.024379	-0.210846	0.860800	-0.431833
4	0.890628	0.788470	0.044711	0.663799	-0.915305	2.491438	0.976197	0.038602	-0.495316	0.486668
5	0.890628	-1.130216	0.044711	0.663799	1.092527	-1.069754	-1.024379	-1.385036	0.710121	0.204994
...	...	...	...	...	...	...	...	...	...	...
285826	-1.122799	-0.997893	0.044711	-1.773964	1.092527	-1.069754	-1.024379	-1.002996	0.710121	0.768342
285827	-1.122799	-1.130216	0.044711	1.528812	1.092527	-1.069754	-1.024379	1.192945	0.710121	-1.276855
285828	-1.122799	-0.997893	0.044711	1.843362	1.092527	-1.069754	-1.024379	1.787791	0.710121	0.878562
285829	0.890628	1.979378	0.044711	-1.538051	1.092527	0.710842	-1.024379	-0.472743	0.182742	1.417417
285830	0.890628	0.457662	0.044711	0.113337	-0.915305	0.710842	0.976197	0.229211	-1.663084	-1.350335

230567 rows × 10 columns

Dapat dilihat setelah kita melakukan normalisasi data maka data tersebut akan berubah seperti yang dapat dilihat pada gambar diatas.

G. Pembuatan PCA

Principal Component Analysis (PCA) adalah salah satu algoritma unsupervised learning yang paling umum digunakan untuk diaplikasikan sebagai: analisis data eksplorasi, pengurangan dimensi, kompresi informasi, de-noising data, dan lainnya. Karena Penulis ingin melakukan percobaan menggunakan maka hal pertama yang akan dilakukan adalah pembuatan PCAnya terlebih dahulu dengan cara sebagai berikut

```
# Implementasi PCA
pca = PCA(n_components=2)
principalComponent = pca.fit_transform(df_trainNormalisasi)
#memasukan data implementasi PCA ke dataset baru
df_pca = pd.DataFrame(data = principalComponent)
df_pca.columns = ['Component_1','Component_2']
df_pca
```

Disini penulis mengunakan data yang telah dinormalisasikan sebelumnya untuk melakukan pengujian PCA. Dalam PCA ini penulis melakukan pengurangan dimensi yang dimana awalnya atributnya ada 10 menjadi 2 atribut yaitu Component_1 dan Component_2 yang dari implementasi itu dihasilkan data sebagai berikut.

	Component_1	Component_2
0	-2.184836	0.329283
1	2.917861	0.161468
2	-2.089879	0.416098
3	2.702796	0.017162
4	-2.287977	0.263063
...	...	...
230562	-2.547640	0.081705
230563	-2.045069	0.446805
230564	-1.860918	0.611804
230565	0.318384	2.022060
230566	2.173323	-0.123210

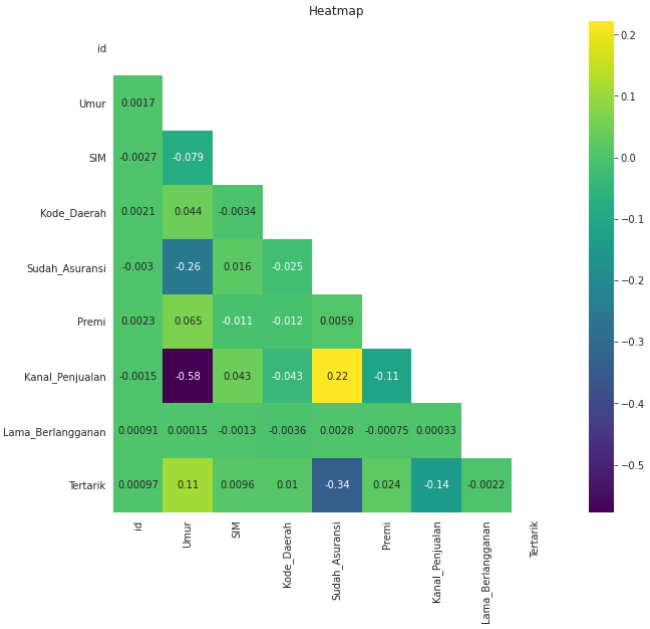
230567 rows × 2 columns

3. Pemodelan

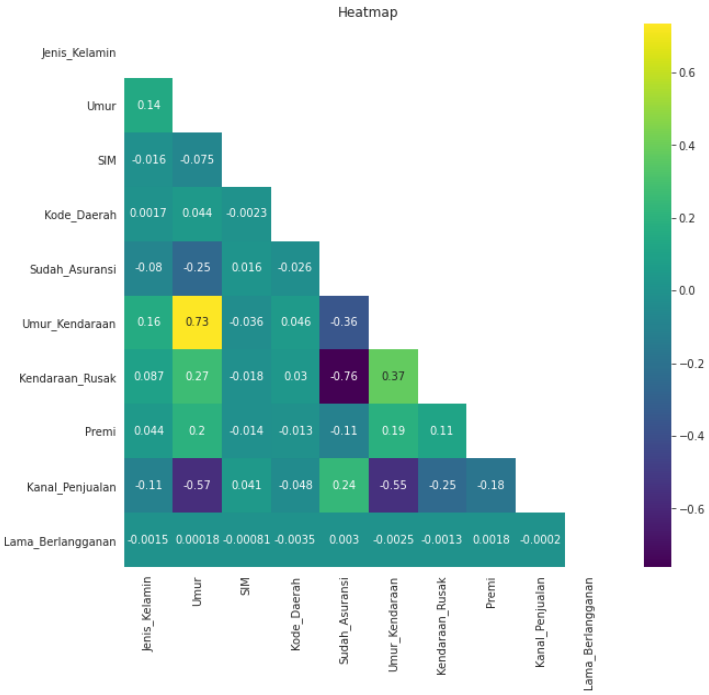
A. Pemilihan atribut Untuk Pengujian

Setelah Melakukan Cleaning data penulis memutuskan untuk memilih atribut apa saja yang akan digunakan dalam penelitian ini. Dalam memilih atribut apa saja yang akan digunakan kita akan menggunakan heatmap korelasi dalam penentuan atribut apa saja yang akan dipilih. Berikut heatmap korelasi yang didapatkan.

- Heatmap Korelasi Sebelum Preprocessing



- Heatmap Korelasi Sesudah Preprocessing



Dari heatmap yang didapatkan, penulis akan melakukan pemilihan atribut sebagai berikut.

1. Pengujian Utama

Pada pengujian utama ini penulis memilih atribut Umur dan Kanal_Penjualan untuk menjadi pengujian utamanya karena setelah melihat heatmap korelasinya itu kuat baik sebelum atau sesusdah preprocessing sehingga menjadi alasan penulis untuk menggunakan atribut tersebut sebagai Pengujian Utamanya.

2. Pengujian Eksperimen

Untuk eksperimen penulis memilih beberapa atribut yang akan dilakukan pengujian sebagai berikut.

- Attribut Premi dan Umur memilih atribut secara acak.
- atribut Lama_Berlangganan dan Umur alasannya karena korelasinya sangat kecil
- Pengujian menggunakan Data PCA.

B. K-Means

Disini Penulis menggunakan algoritma clustering K-Means yang digunakan untuk menjadi pemodelan yang akan digunakan. Berikut implementasi pengkodean

```python
#Metode Euclidean Untuk Menghitung jarak
def euclidean(x, centroid):
  return np.linalg.norm(x - centroid)
```

```python
def kmeans(k, data,max_iterasion):
    #inisiasi centroid dengan memilihnya secara random
    centroid = {i: data[random.randint(0, len(data))] for i in range(k)}
    # untuk mendapatkan wcss untuk membuat elbow method nantinya
    inertia = 0
    # inisiasi variable sameCentroid dengan false
    sameCentroid = False
    i = 0
    while(sameCentroid!=True) and (i < max_iterasion):
        #Mencopy centroid sebelumnya
        temp = centroid.copy()
        #Mengisi dictonary cluster yang setiap keynya memiliki array kosong sebanyak k
        cluster = {}
        for j in range(k):
            cluster[j] = []
        # Array untuk menyimpan nilai perhitungan jarak
        T = []

        #mencari nilai centroid terdekat dengan menggunakan metode euclidean
        #dan mencari nilai minimum pada setiap distancenya yang akan dimasukan kedalam cluster
        for row in data:
            dist = [euclidean(row,centroid[x]) for x in centroid]
            T.append(np.min(dist))#menyimpan nilai perhitungan jarak
            cluster[dist.index(min(dist))].append(row)

        #melakukan hasil tambah dari nilai perhitungan jarak yang telah diambil sebelumnya
        inertia = sum(T)

        #Update centroid menggunakan mean dengan mengambil centroid dari cluster sebelumnya
        for cl in cluster:
            centroid[cl] = np.mean(cluster[cl],axis = 0)

        #Melakukan pengecekan apakah centroid telah mencapai nilai yang maksimal / konvergen
        for key in cluster.keys():
            if(temp.get(key)==centroid.get(key))[0]:
                sameCentroid = True
                break
        i+=1

    return centroid,cluster,inertia
```

Pada saat mengimplementasikan K-Means dengan scratch ini ada beberapa tahapan dalam melakukan K-Means ini. Berikut merupakan tahapan K-Means.

1) Inisiasi Centroid Secara Random

Pada awal kita akan menggunakan K-Means kita harus melakukan inisialisasi centroid awal dengan mengambil value dari data yang telah kita pilih sebelumnya.

```python
#inisiasi centroid dengan memilihnya secara random
centroid = {i: data[random.randint(0, len(data))] for i in range(k)}
```

2) Mencari Centroid Terdekat dan Klasterisasi Data

Untuk mencari centroid terdekat memerlukan euclidean distance sebagai formula menghitung jaraknya. Sehingga penulis memerlukan formula dari euclidean distance agar bisa melakukan perhitungan jarak. Berikut bagaimana formula dari euclidean distance dan bagaimana cara pengimplementasiannya pada codenya.

$$d(i,j)=\sqrt{(x_{i1}-x_{j1})^2+(x_{i2}-x_{j2})^2+...+(x_{ip}-x_{jp})^2}$$

```
#Metode Euclidean Untuk Menghitung jarak
def euclidean(x, centroid):
    return np.linalg.norm(x - centroid)
```

Setelah menyediakan fungsi euclidean distance yang akan dipakai untuk mencari centroid terdekat terlebih dahulu untuk melakukan copy centroid sebelumnya karena akan digunakan untuk pengujian kekonvergenan terhadap centroid baru nantinya. Kemudian menginisiasi dictionary untuk cluster nantinya, kemudian melakukan pencarian centroid terdekat dengan euclidean yang telah dibuat sebelumnya agar menyimpan perhitungan jarak yang minimum untuk mencari elbow method dan juga untuk memasukan nilai minimum pada setiap jaraknya ke dalam cluster. Dan untuk perhitungan jarak yang disimpan sebelumnya akan menjumlahkan perhitungan jarak buat menentukan elbow method dari perhitungan jarak yang minimum yang kita simpan untuk mencari elbow tersebut akan kita jumlahkan(Sum). Berikut implementasi dari penjelasan sebelumnya ke dalam pencobaan.

```
#Mencopy centroid sebelumnya
temp = centroid.copy()
#Mengisi dictonary cluster yang setiap keynya memiliki array kosong sebanyak k
cluster = {}
for j in range(k):
    cluster[j] = []
# Array untuk menyimpan nilai perhitungan jarak
T = []

#mencari nilai centroid terdekat dengan menggunakan metode euclidean
#dan mencari nilai minimum pada setiap distancenya yang akan dimasukan kedalam cluster
for row in data:
    dist = [euclidean(row,centroid[x]) for x in centroid]
    T.append(np.min(dist))#menyimpan nilai perhitungan jarak
    cluster[dist.index(min(dist))].append(row)

#melakukan hasil tambah dari nilai perhitungan jarak yang telah diambil sebelumnya
inertia = sum(T)
```

3) Melakukan Update Centroid

Disini akan dilakuikan pengupdatean centroid dengan cara mengambil rata-rata(mean) dari nilai centroid dari setiap cluster yang telah dibuat sebelumnya.

```
#Update centroid menggunakan mean
for cl in cluster:
    centroid[cl] = np.mean(cluster[cl],axis = 0)
```

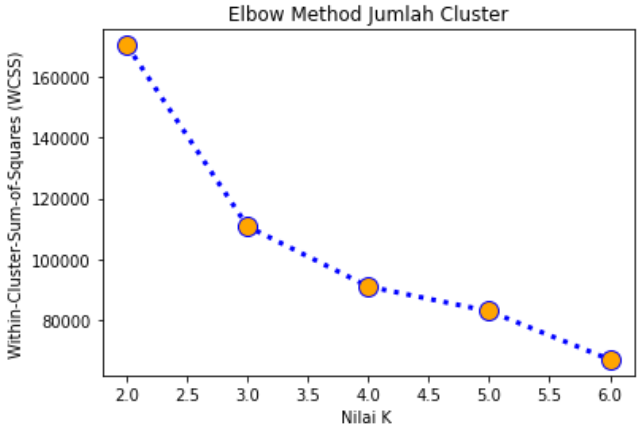
4) Memeriksa Konvergen

Untuk melakukan optimasi terhadap algoritma K-Means dari scratch ini saya membuat pemeriksaan konvergen. Tujuan dari pemeriksaan konvergen ini jika centroid sebelumnya sama dengan centroid baru maka proses akan berhenti dan centroid baru disebut telah dianggap sebagai centroid maksimalnya. Berikut implementasinya.

```
#Melakukan pengecekan apakah centroid telah mencapai nilai yang maksimal / konvergen
for key in cluster.keys():
    if(temp.get(key)==centroid.get(key))[0]:
        sameCentroid = True
        break
```

4. Evaluasi Pengujian Utama

Pada evaluasi ini penulis menggunakan elbow method untuk mencari nilai K optimal dari data yang ada. Pada pengujian ini penulis melakukan pengujian K dari 2 – 6.




```
# Pengecekan nilai WCSS dari elbow method
for i in range(2,7):
    print(f'K = {i}, WCSS = {jumlah_elbow[i-2]}')
```

```
K = 2, WCSS = 170492.35163033454
K = 3, WCSS = 110752.73888744244
K = 4, WCSS = 90930.24456038652
K = 5, WCSS = 82808.39546928425
K = 6, WCSS = 66860.45919112509
```

Dari hasil yang didapatkan diatas didapatkan nilai K optimal berada di K = 3 dengan nilai WCSS = 110752.73888744244. Kemudian dengan nilai K optimal yang telah didapatkan maka didapatkan result cluster sebagai berikut.

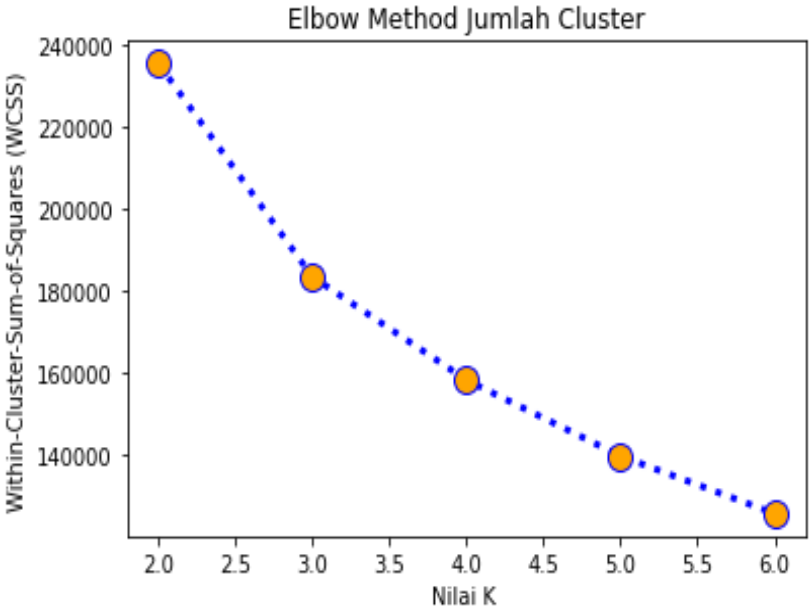
Dari result diatas dapat dilihat bahwa data terbagi dan tersebar dengan sangat jelas karena data tidak semuanya menumpuk pada setiap clusternya. Titik berwarna hitam diresult tersebut adalah centroid dari kluster.

5. Eksperimen

A. Nilai K dan Hasil Clustering Data Eksperimen

Sama dengan pengujian utama, pada eksperimen ini penulis melakukan pengujian nilai K dengan nilai 2 - 6 untuk eksperimen terhadap data pengujian atribut lainnya. Dan untuk eksperimen PCA penulis menggunakan K 2 - 7 untuk melakukan pengujian nilai K nya.

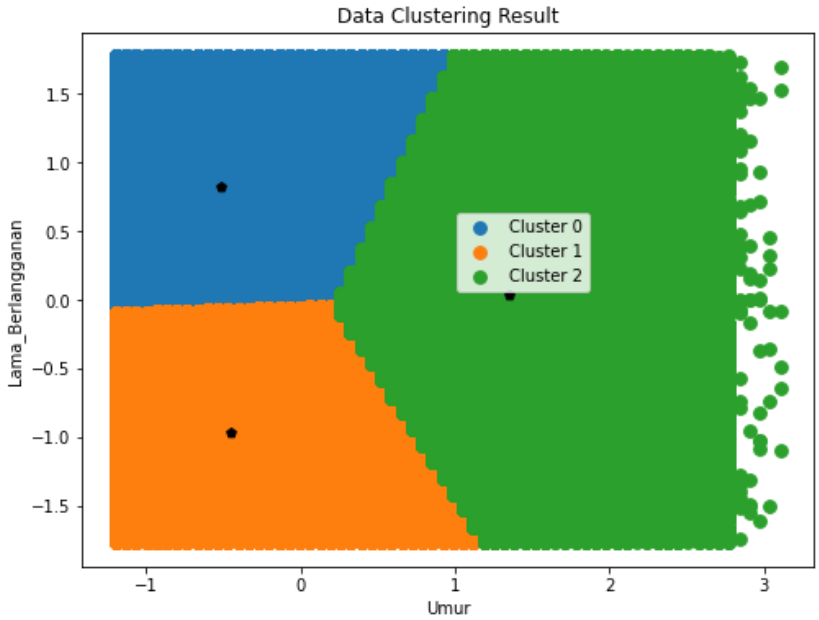
1) Data Umur dan Lama_Berlangganan



```
# Pengecekan nilai WCSS dari elbow method
for i in range(2,7):
    print(f'K = {i}, WCSS = {jumlah_elbow[i-2]}')
```

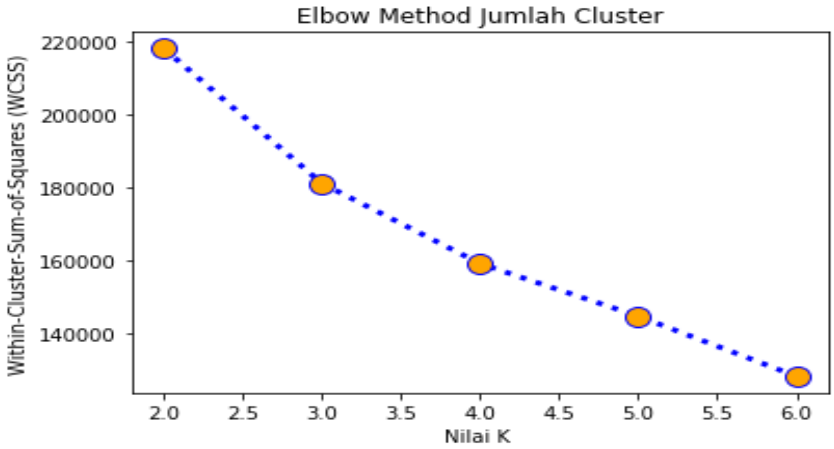
```
K = 2, WCSS = 235534.10930994517
K = 3, WCSS = 183392.4803004012
K = 4, WCSS = 158167.4999266057
K = 5, WCSS = 139604.19390359707
K = 6, WCSS = 125553.91684180155
```

Dari hasil yang didapatkan diatas didapatkan nilai K optimal berada di K = 3 dengan nilai WCSS = 183392.4803004012. Kemudian dengan nilai K optimal yang telah didapatkan maka didapatkan result cluster sebagai berikut.



Dari cluster result diatas didapatkan kesimpulan bahwa hasilnya sangat buruk karena data tidak tersebar dengan baik. Dan penyebaran datanya hanya terjadi di cluster 2.

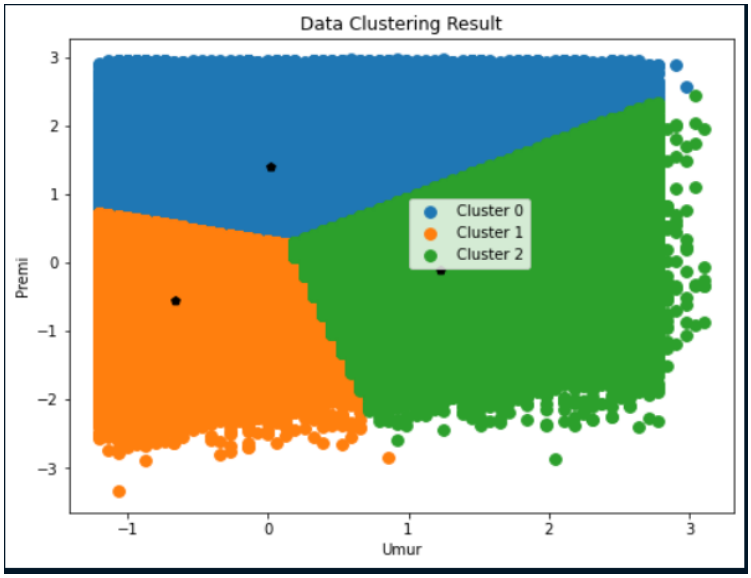
2) Data Umur dan Premi



```python
# Pengecekan nilai WCSS dari elbow method
for i in range(2,7):
    print(f'K = {i}, WCSS = {jumlah_elbow[i-2]}')
```

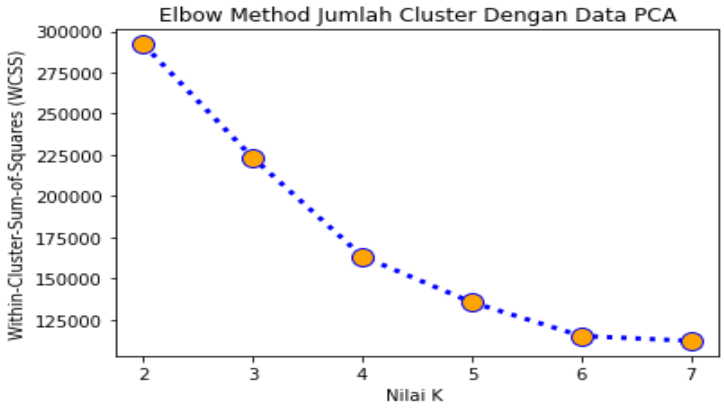
```
K = 2, WCSS = 218073.04673421528
K = 3, WCSS = 181143.04504155606
K = 4, WCSS = 159165.7501030075
K = 5, WCSS = 144768.2257599271
K = 6, WCSS = 128317.64027066813
```

Dari hasil yang didapatkan diatas didapatkan nilai K optimal berada di K = 3 dengan nilai WCSS = 181143.04504155606. Kemudian dengan nilai K optimal yang telah didapatkan maka didapatkan result cluster sebagai berikut.



Dari cluster result diatas didapatkan kesimpulan bahwa hasilnya sedikit lebih baik dibandingkan dengan eksperimen atribut Umur dan Lama_Berlangganan namun tidak lebih baik dari percoban dengan atribut Umur dan Kanal_Penjualan. Penyebaran data hanya terjadi di cluster 1 dan cluster 2 namun masih banyak data yang masih bertumpuk pada setiap clusternya terutama pada cluster 0.

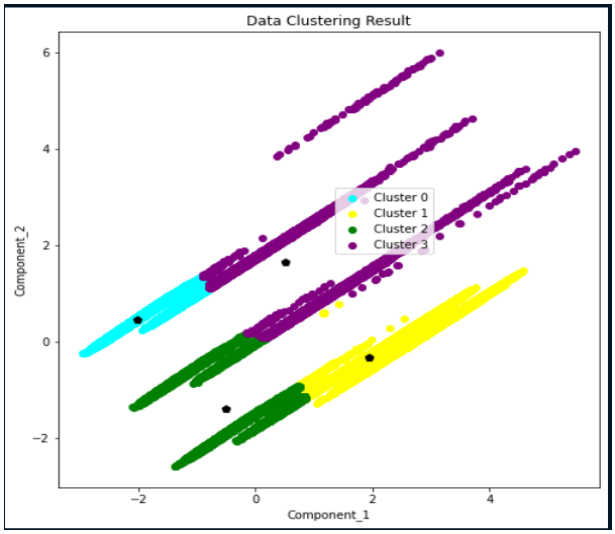
3) Data PCA



```
    # Pengecekan nilai WCSS dari elbow method
    for i in range(2,8):
        print(f'K = {i}, WCSS = {jumlah_elbow[i-2]}')

K = 2, WCSS = 292074.14493204677
K = 3, WCSS = 223241.80845604386
K = 4, WCSS = 163403.80440347194
K = 5, WCSS = 135429.9109077085
K = 6, WCSS = 115006.92172748066
K = 7, WCSS = 112018.15415745322
```

Dari hasil yang didapatkan diatas didapatkan nilai K optimal berada di K = 4 dengan nilai WCSS =163403.80440347194. Kemudian dengan nilai K yang telah didapatkan maka didapatkan result cluster sebagai berikut.

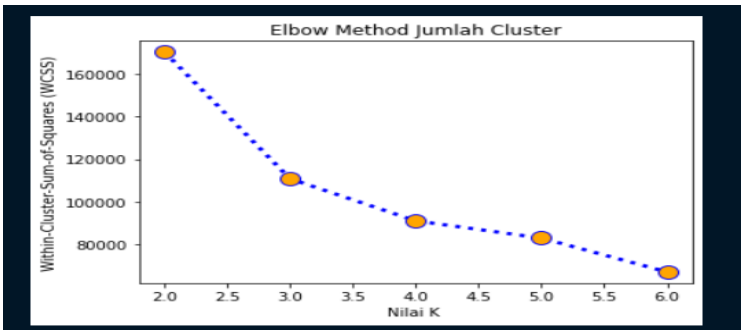


Dari cluster result diatas didapatkan hasil yang cukup baik karena tidak banyak data yang menumpuk diantara clusternya dan penyebaran datanya terlihat dengan jelas.

B. Pengujian Nilai K Menggunakan K-Means Scratch Dengan Nilai K Menggunakan Library K-Means

1) Elbow Method Untuk Data Menggunakan K-Means Scratch

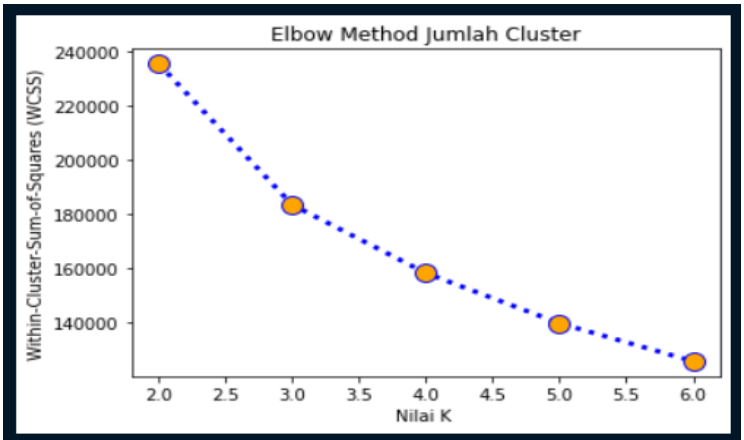
a. Data Umur Dan Kanal_Penjualan




```
    # Pengecekan nilai WCSS dari elbow method
    for i in range(2,7):
        print(f'K = {i}, WCSS = {jumlah_elbow[i-2]}')

K = 2, WCSS = 170492.35163033454
K = 3, WCSS = 110752.73888744244
K = 4, WCSS = 90930.24456038652
K = 5, WCSS = 82808.39546928425
K = 6, WCSS = 66860.45919112509
```

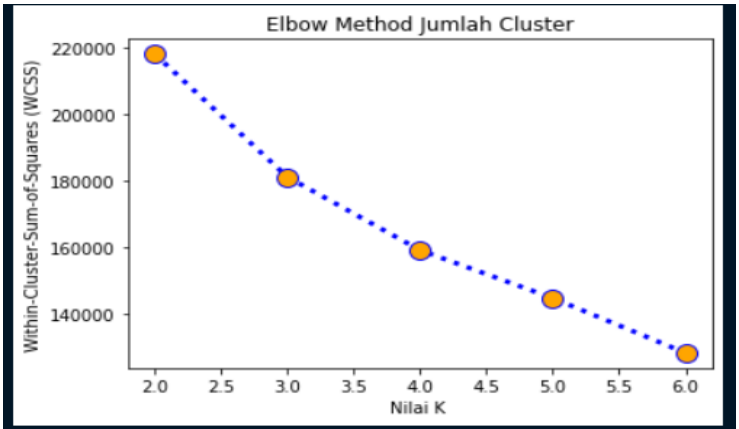
b. Data Umur Dan Lama_Berlangganan




```
# Pengecekan nilai WCSS dari elbow method
for i in range(2,7):
    print(f'K = {i}, WCSS = {jumlah_elbow[i-2]}')
```

```
K = 2, WCSS = 235534.10930994517
K = 3, WCSS = 183392.4803004012
K = 4, WCSS = 158167.4999266057
K = 5, WCSS = 139604.19390359707
K = 6, WCSS = 125553.91684180155
```

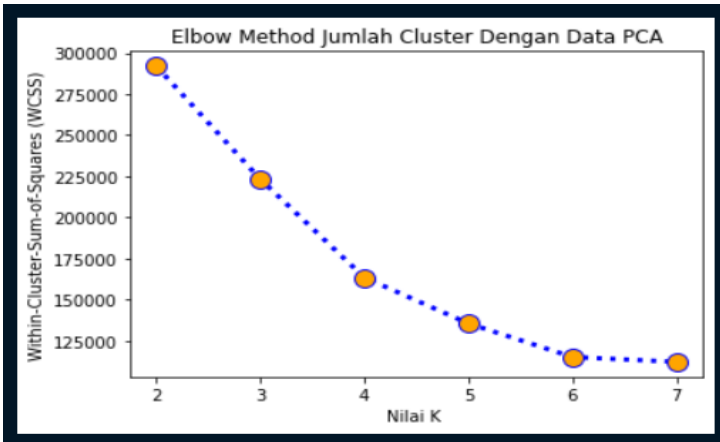
c. Data Umur Dan Premi




```
# Pengecekan nilai WCSS dari elbow method
for i in range(2,7):
    print(f'K = {i}, WCSS = {jumlah_elbow[i-2]}')
```

```
K = 2, WCSS = 218073.04673421528
K = 3, WCSS = 181143.04504155606
K = 4, WCSS = 159165.7501030075
K = 5, WCSS = 144768.2257599271
K = 6, WCSS = 128317.64027066813
```

d. Data PCA

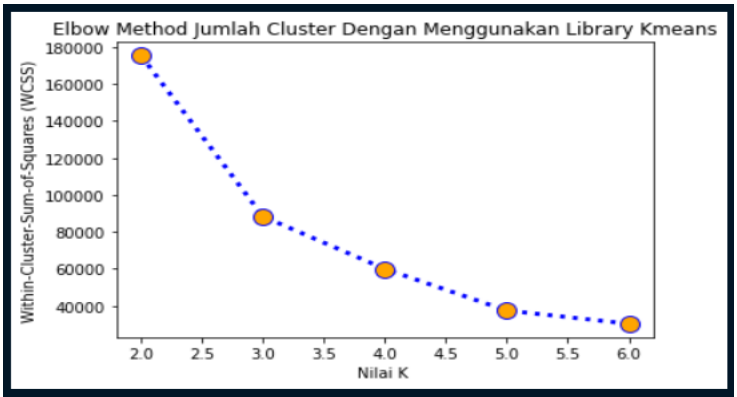



```
# Pengecekan nilai WCSS dari elbow method
for i in range(2,8):
    print(f'K = {i}, WCSS = {jumlah_elbow[i-2]}')
```

```
K = 2, WCSS = 292074.14493204677
K = 3, WCSS = 223241.80845604386
K = 4, WCSS = 163403.80440347194
K = 5, WCSS = 135429.9109077085
K = 6, WCSS = 115006.92172748066
K = 7, WCSS = 112018.15415745322
```

2) Elbow Method Untuk Data Menggunakan K-Means Library

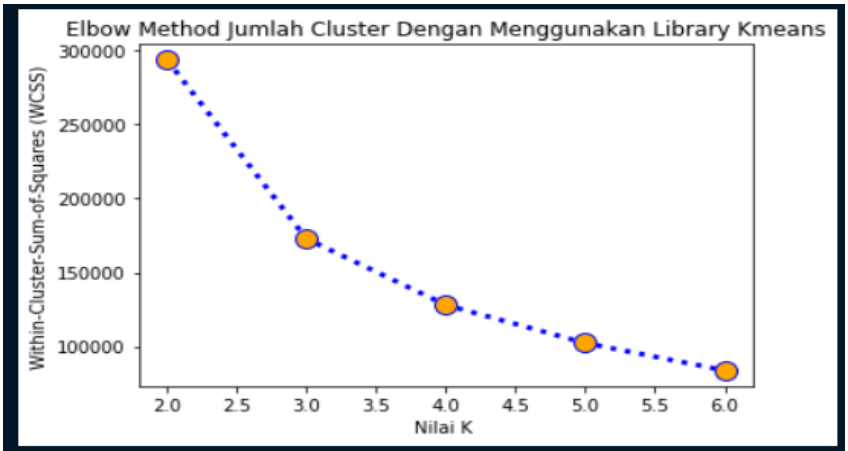
a. Data Umur Dan Kanal_Penjualan




```
# Pengecekan nilai WCSS dari elbow method
for i in range(2,7):
    print(f'K = {i}, WCSS = {distance[i-2]}')
```

```
K = 2, WCSS = 175358.18441026044
K = 3, WCSS = 88346.38782945066
K = 4, WCSS = 59968.410943579314
K = 5, WCSS = 37450.7395198147
K = 6, WCSS = 30340.52846504609
```

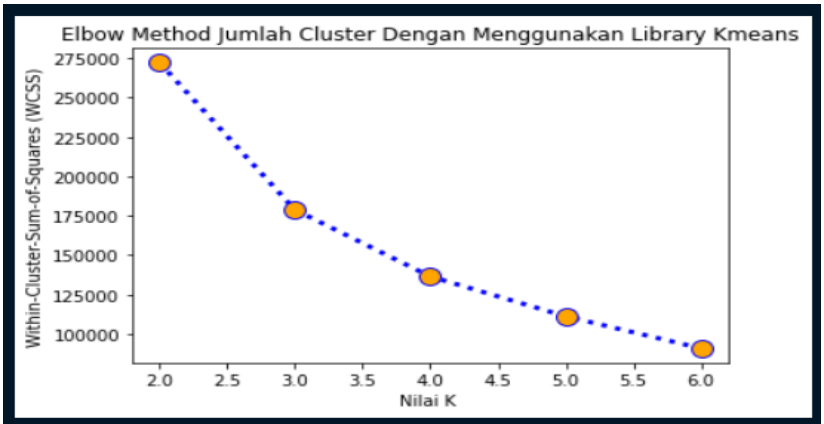
b. Data Umur Dan Lama_Berlangganan




```
# Pengecekan nilai WCSS dari elbow method
for i in range(2,7):
    print(f'K = {i}, WCSS = {distance[i-2]}')
```

```
K = 2, WCSS = 293504.49103906885
K = 3, WCSS = 172785.7048039948
K = 4, WCSS = 127988.104599012
K = 5, WCSS = 102292.90287814058
K = 6, WCSS = 83788.68617225935
```

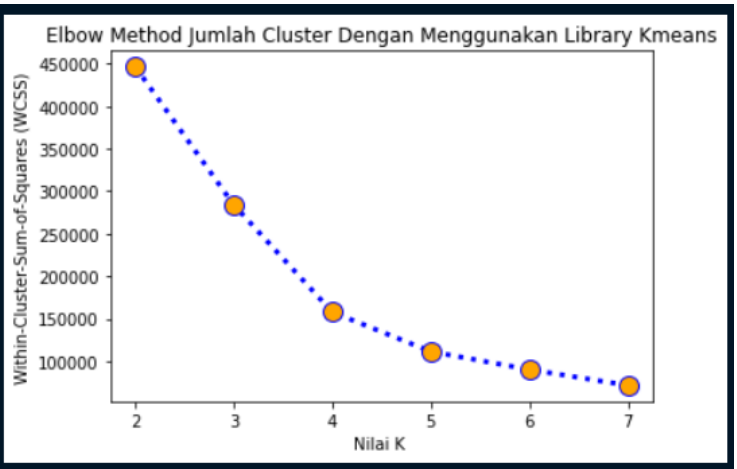
c. Data Umur Dan Premi




```
# Pengecekan nilai WCSS dari elbow method
for i in range(2,7):
    print(f'K = {i}, WCSS = {distance[i-2]}')
```

```
K = 2, WCSS = 272143.56565356324
K = 3, WCSS = 178812.79639432466
K = 4, WCSS = 136475.85063518691
K = 5, WCSS = 111189.93485313153
K = 6, WCSS = 90908.06317419883
```

d. Data PCA




```
# Pengecekan nilai WCSS dari elbow method
for i in range(2,8):
    print(f'K = {i}, WCSS = {distance[i-2]}')
```

```
K = 2, WCSS = 446289.7875253065
K = 3, WCSS = 283840.1175971618
K = 4, WCSS = 157816.20939494699
K = 5, WCSS = 111352.44359211033
K = 6, WCSS = 90198.9011687659
K = 7, WCSS = 71601.0667912781
```

Dilihat pada setiap elbow method dapat dikatakan bahwa grafik elbow method yang menggunakan K-Means dari scratch hampir sama dengan grafik elbow method menggunakan library K-Means. Akan tetapi nilai Within Cluster Sum of Square (WCSS) yang dimiliki terjadi perbedaaan antara K-Means Library dengan K-Means Scratch. Hal ini disebabkan karena algoritma K-Means Scratch melakukan inisiasi centroid awal menggunakan nilai yang random sehingga membuat WCSS dan grafik elbow method tidak konsisten dan sering berganti jika melakukan running ulang. Berbeda dengan K-Means dengan menggunakan library yang dimana diasumsikan bahwa library tersebut sudah menambahkan antisipasi terhadap insiasi centroid awal dengan data yang random menggunakan algoritma K-Means++ yang membuat WCSS dan grafik elbow methodnya lebih konsisten dibandingkan dengan K-Means yang dibuat secara scratch.

6. Kesimpulan

- Data memiliki hasil clustering yang baik jika menggunakan atribut yang memiliki koorelasi yang kuat dan akan memberikan hasil yang buruk jika korelasi antara atribut itu lemah.
- Saat ingin menggunakan K-Means dalam clusteringnya harus dilakukan pembersihan pencilan(outlier) telebih dahulu karena inisiasi random centroidya sangat sensitif terhadap pencilan(outlier).
- Hasil Within Cluster Sum of Square (WCSS) untuk elbow method dan hasil centroidnya tergantung dengan K-Meansnya. Jika menggunakan K-Means from scratch dengan algoritma K-Means grafik elbow method dan WCSS nya akan tidak konsisten karena insiasi centroid awalnyanya yang dilakukan secara random. Sedangkan jika menggunakan algoritma K-Means++ didapatkan elbow method dan WCSS yang konsisten karena inisiasi centroid awal yang dilakukan secara random telah diminimalisir.
- Elbow Method digunakan untuk melihat nilai K mana yang optimal.

7. Lampiran

Link Youtube:
https://youtu.be/n6WxZUkhM6U
Link Github:
https://github.com/s17Happiness/Clustering_Tubes
Link Dataset:
https://drive.google.com/drive/folders/1OO6UizXCYQxUoyGEDwrZPTG6kcAVEAcT?usp=sharing
Referensi :

https://www.keboola.com/blog/pca-machine-learning
https://towardsdatascience.com/data-normalization-with-pandas-and-scikit-learn-7c1cc6ed6475
https://statistics.laerd.com/statistical-guides/measures-central-tendency-mean-mode-median.php