LAPORAN TUGAS BESAR TAHAP 1: CLUSTERING

Disusun Untuk Memenuhi Mata Kuliah Pembelajaran Mesin

Disusun Oleh:
Kurniadi Ahmad Wijaya - 1301194024 - IF 43 09

PRODI S1 INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

BANDUNG

2021

DAFTAR ISI

DAFTAR ISI 2
1. Formulasi Masalah 3
2. Eksplorasi dan Persiapan Data 3
2.1 Proses EDA (Exploratory Data Analysis) 3
2.2 Pra-Pemrosesan Data 5
2.3 Keseluruhan Alur Pra-Pemrosesan 11
3. Pemodelan 12
3.1 Fungsi Euclidean Distance 13
3.2 Fungsi Pengecekan Kekonvergenan Centroid 13
3.3 Fungsi Model K-Means 13
4. Evaluasi 15
5.1 Metrik Perhitungan Silhoutte Score 15
5.2 Metrik Elbow Method Dengan Within Cluster Sum Of Squares (WCSS) 16
5.3 Evaluasi Model Terbaik 17
5. Eksperimen 19
5.1 Kolom Umur – Kanal Penjualan Z-Score Outlier 20
5.2 Kolom Umur – Kanal Penjualan Z-Score 21
5.3 Kolom Umur – Kanal Penjualan Z-Score PCA 23
5.4 Kolom Umur – Kanal Penjualan Z-Score Outlier PCA 24
5.5 Kolom Umur – Kanal Penjualan Min Max Outlier PCA 26
5.6 Kolom Umur – Kanal Penjualan Min Max PCA 27
5.7 Kolom Umur – Kanal Penjualan Min Max Outlier 29
5.8 Kolom Premi - Kanal Penjualan 30
5.9 Kolom Umur – Premi Min Max 32
5.10 Kolom Umur – Lama Berlangganan 33
6. Kesimpulan 35
7. Saran 35
LAMPIRAN 37

1. Formulasi Masalah

Diberikan dataset kendaraan_train.csv yang berisikan data-data pelanggan pada suatu dealer. Diberikan tugas untuk melakukan clustering (*unsupervised Learning*) dari dataset tersebut dengan mengelompokkan pelanggan berdasarkan data pelanggan di dealer tanpa memperhatikan label kelas apakah pelanggan tertarik untuk membeli kendaraan baru atau tidak.

2. Eksplorasi dan Persiapan Data

Didalam dataset terdapat beberapa macam jenis kolom yang akan digunakan untuk proses clustering yang diantaranya adalah:

Nama Kolom	Deskripsi
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 Dari Kendaraan
Kendaraan_Rusak	1: Kendaraan Pernah Rusak Sebelumnya. 0: Kendaraan Belum Pernah Rusak.
Premi	Jumlah Premi Yang Harus Dibayarkan Per Tahun.
Kanal_Penjualan	Kode Kanal Untuk Menghubungi Pelanggan
Lama_Berlangganan	Sudah Berapa Lama Pelanggan Menjadi Klien Perusahaan

Untuk memastikan lebih jauh lagi mengenai jenis data yang akan kita tangani kita dapat melakukan proses EDA (Exploratory Data Analysis) terlebih dahulu.

2.1 Proses EDA (Exploratory Data Analysis)

Pada bagian ini kita akan mencari insight dari dataset yang diberikan seluas-luasnya untuk memahami dataset yang diberikan. Adapun beberapa langkah yang dapat dilakukan yaitu:

- **Pengecekan Tipe Data Setiap Kolom**

  Untuk mengenal lebih jauh setiap kolom baik dari segi tipe data, jangkauan data, nilai maximum serta minimumnya kita dapat menggunakan fungsi pada library pandas yaitu dtypes**.** Fungsi ini akan menampilkan berbagai macam detail mengenai keseluruhan column yang akan kita analisis kemudian.

```
df_train = pd.read_csv("Dataset/kendaraan_train.csv")
df_train.drop(['id'], axis=1, inplace=True)

print("Total Dataset :", len(df_train))
df_train.sample(5)
```

0.6s

```
Total Dataset : 285831
```

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan
81532	Pria	72.0	1.0	28.0	0.0	1-2 Tahun	Pernah	46584.0	122.0	229.0
215527	Pria	25.0	1.0	28.0	0.0	< 1 Tahun	Pernah	47206.0	163.0	48.0
141356	Wanita	24.0	1.0	2.0	0.0	< 1 Tahun	Pernah	18940.0	152.0	115.0
41521	Pria	24.0	1.0	8.0	1.0	< 1 Tahun	Tidak	43505.0	152.0	28.0
78676	Pria	60.0	NaN	46.0	0.0	1-2 Tahun	Pernah	25347.0	124.0	50.0

```
df_train.dtypes
```

0.4s

```
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
```

Pada program diatas kita dapat melihat 5 sampel dari isi setiap kolom dataset menggunakan fungsi sample() serta hasil dari pemanggilan fungsi dtypes yaitu jenis tipe data dari setiap kolom yang ada. Diketahui bahwasannya terdapat 7 kolom yang bertipe float yaitu kolom Umur, SIM, Kode_Daerah, Sudah_Asuransi, Premi, Kanal_Penjualan, dan Lama_Berlangganan serta 3 kolom bertipe object atau string yaitu kolom Jenis_Kelamin, Umur_Kendaraan dan Kendaraan_Rusak. Disini terdapat juga kolom Tertarik akan tetapi tidak kita gunakan pada proses clustering.

- **Detail Value Unik Kategorikal Dan Value Numerik Dari Setiap Kolom**

Setelah melakukan pengecekan jenis-jenis tipe data untuk setiap kolom, tentunya untuk menambah pemahaman pada dataset terutama pada data bertipe object kita harus melakukan pengecekan jenis value apa saja yang ada pada setiap kolom. Hal ini dapat kita lakukan pengecekan dengan fungsi value_counts().

```
df_train['Jenis_Kelamin'].value_counts()
```

0.5s

```
Pria      146678
Wanita    124713
Name: Jenis_Kelamin, dtype: int64
```

```
df_train['Umur_Kendaraan'].value_counts()
```

0.6s

```
1-2 Tahun    142761
< 1 Tahun    117378
> 2 Tahun     11417
Name: Umur_Kendaraan, dtype: int64
```

```
df_train['Kendaraan_Rusak'].value_counts()

```

0.6s

```
Pernah    137123
Tidak     134520
Name: Kendaraan_Rusak, dtype: int64
```

Kemudian, meskipun memiliki tipe data float beberapa kolom seperti SIM dan Sudah_Asuransi hanya memiliki value 1 dan 0 yang menandakan sudah atau belum sehingga kolom tersebut termasuk jenis data kategorikal ordinal.

```
df_train['SIM'].value_counts()

```

0.3s

```
1.0    270843
0.0       584
Name: SIM, dtype: int64
```

```
df_train['Sudah_Asuransi'].value_counts()

```

0.3s

```
0.0    146997
1.0    124605
Name: Sudah_Asuransi, dtype: int64
```

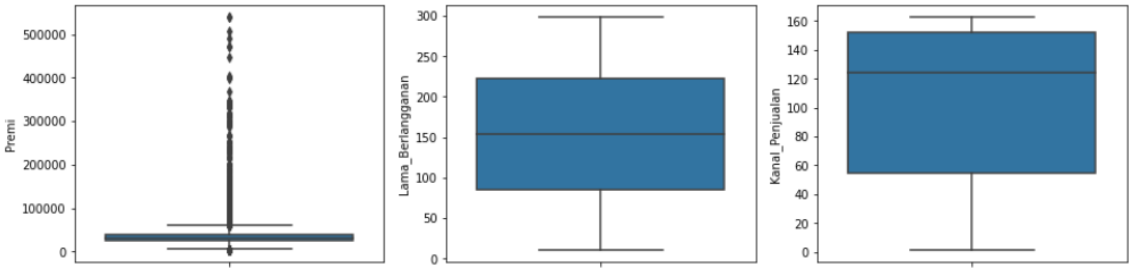
Selanjutnya setelah melakukan detail dari isi data kategorikal kita juga dapat melakukan pengecekan terhadap sebaran isi data numerikal dengan menggunakan fungsi describe(). Dapat kita lihat penyebaran data mulai dari jumlah data (count), rerataan (mean), standar deviasi (std), quartil bawah (25%), median (50%), quartil atas (75%) serta nilai maksimal (max) untuk setiap kolom return function tersebut.

```
1 df_train.describe()
✓ 0.2s
```

	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
count	271617.000000	271427.000000	271525.000000	271602.000000	271262.000000	271532.000000	271839.000000	285831.000000
mean	38.844336	0.997848	26.405410	0.458778	30536.683472	112.021567	154.286302	0.122471
std	15.522487	0.046335	13.252714	0.498299	17155.000770	54.202457	83.694910	0.327830
min	20.000000	0.000000	0.000000	0.000000	2630.000000	1.000000	10.000000	0.000000
25%	25.000000	1.000000	15.000000	0.000000	24398.000000	29.000000	82.000000	0.000000
50%	36.000000	1.000000	28.000000	0.000000	31646.000000	132.000000	154.000000	0.000000
75%	49.000000	1.000000	35.000000	1.000000	39377.750000	152.000000	227.000000	0.000000
max	85.000000	1.000000	52.000000	1.000000	540165.000000	163.000000	299.000000	1.000000

- **Pengecekan Distribusi Data Numerik**

Hal lain yang dapat kita lakukan dalam eksplorasi data yaitu adalah mengecek pendistribusian data pada data numerik. Salah satu hal yang bisa kita dapatkan pada proses ini yaitu mengetahui apakah terdapat outlier pada data atau tidak. Berikut merupakan visualisasi pendistribusian data menggunakan boxplot.



Dari hasil visualisasi boxplot diatas dapat kita lihat bahwasannya value dari attribut Premi memiliki jangkauan data yang sangat jauh sehingga outlier pada data sangat terlihat. Selanjutnya outlier ini akan kita hilangkan pada fase pembersihan data karena dapat membuat noise pada jangkauan data.

2.2 Pra-Pemrosesan Data

Setelah mengetahui bagaimana kondisi dari data yang diberikan maka langkah selanjutnya yang kita lakukan adalah pra-pemrosesan data. Pada proses ini dilakukan beberapa jenis pra-pemrosesan daat yaitu pembersihan data (data cleaning), transformasi data (data transformation), dan reduksi data (data reduction). Proses ini dilakukan untuk menghilangkan data yang tidak lengkap, terdapat noise, dan tidak konsisten sehingga memiliki data yang berkualitas.

- **Pembersihan Data**

Langkah awal yang harus kita lakukan dahulu dalam proses pembersihan data adalalah menghapus data duplikat. Disamping dapat meringankan proses komputasi, penghapusan data duplikasi juga dapat mereduksi noise sehingga dapat meningkatkan kualitas data yang akan di analisis [1]. Untuk menghapus data duplikat kita dapat menggunakan fungsi drop_duplicates pada pandas.

```
1  duplicate = list(df_train.duplicated())
2  print("Data Duplikasi :", duplicate.count(True))
```

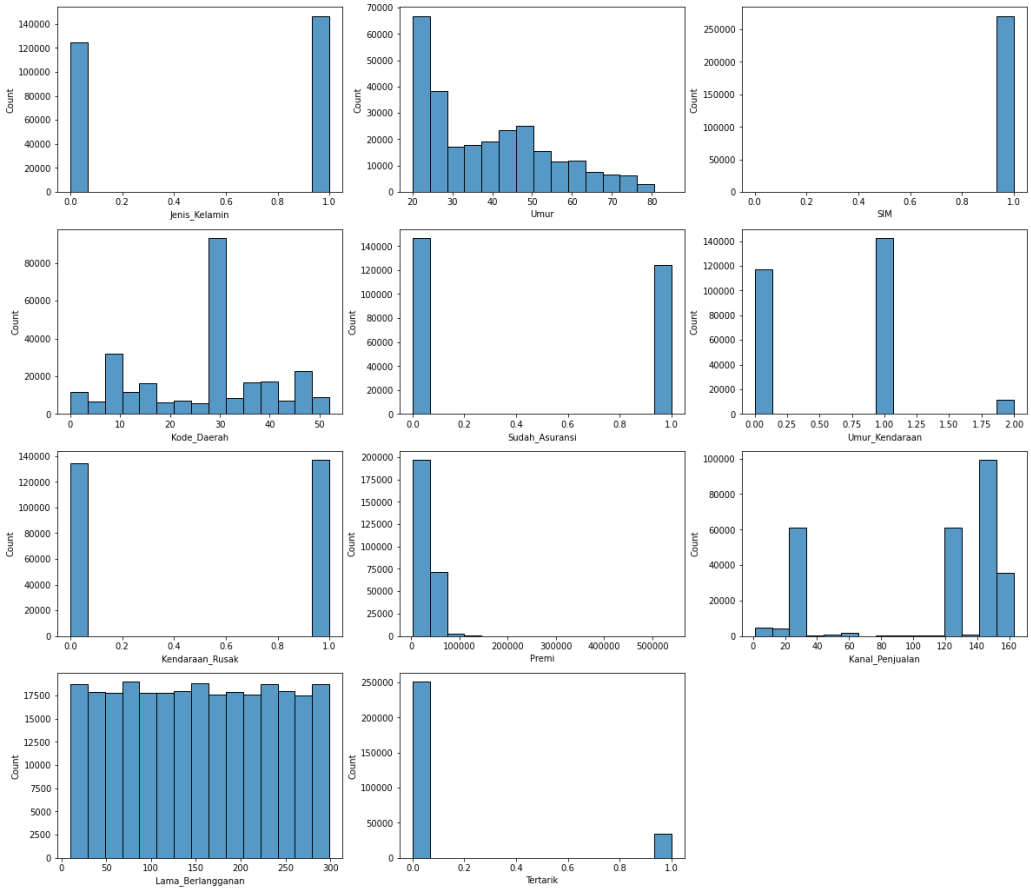
```
Data Duplikasi : 169
```

```
1  df_train.drop_duplicates(inplace=True)
```

```
1  duplicate = list(df_train.duplicated())
2  print("Data Duplikasi :", duplicate.count(True))
```

```
Data Duplikasi : 0
```

Setelah menghapus data yang duplikat selanjutnya kita akan melakukan pengisian data kosong dengan beberapa ukuran pemusatan data seperti mean, median dan mode adapun pengisian data tersebut dilakukan dengan melihat bentuk pendistribusian serta jenis data terlebih dahulu. Berikut merupakan gambar pendistribusian untuk setiap kolom pada data menggunakan distplot.



Dapat kita lihat pada gambar diatas, beberapa kolom memiliki pendistribusian data yang berbeda-beda sehingga penangan selanjutnya pada fase pengisian data untuk setiap kolom akan berbeda-beda.

Adapun berdasarkan aturan ketidakseimbangan data (skewness) data kosong berbentuk interval / rasio akan diisi dengan mean, data yang seimbang berbentuk interval / rasio akan diisi dengan Median, sedangkan data nominal akan diisi dengan modus [2]. Adapun kolom SIM, kode_daerah, dan Kanal_Penjualan diisi dengan modus, kolom Lama_Berlangganan diisi dengan mean dan selain itu diisi dengan median. Berikut merupakan pengimplementasian kodenya.

```python
# Pengisian Data Sesuai Pendistribusiannya
df_train['SIM'].fillna(float(df_train['SIM'].mode()), inplace=True)
df_train['Kode_Daerah'].fillna(float(df_train['Kode_Daerah'].mode()), inplace=True)
df_train['Kanal_Penjualan'].fillna(float(df_train['Kanal_Penjualan'].mode()), inplace=True)
df_train['Lama_Berlangganan'].fillna(float(df_train['Lama_Berlangganan'].mean()), inplace=True)

# Data Nominal Kategorikal Diisi Menggunakan Modus
for i in df_train.columns:
    df_train[i].fillna(float(df_train[i].median()), inplace=True)

df_train.isna().sum()
```

```
Jenis_Kelamin       0
Umur                0
SIM                 0
Kode_Daerah         0
Sudah_Asuransi      0
Umur_Kendaraan      0
Kendaraan_Rusak     0
Premi               0
Kanal_Penjualan     0
Lama_Berlangganan   0
Tertarik            0
dtype: int64
```

Selanjutnya kita dapat melakukan penghapusan outlier pada data kita dengan menggunakan rumus interquartile untuk menentukan batas atas dan batas bawah dari value outlier yang akan dihapus.

```python
# Membuat Rumus Penghapusan Outlier Dengan Menentukan Batas Atas Dan Batas Bawahnya
Q1 = df_train["Premi"].quantile(0.25)
Q3 = df_train["Premi"].quantile(0.75)
IQR = Q3 - Q1

BB = Q1 - (1.5 * IQR)
BA = Q3 + (1.5 * IQR)

print("Batas Atas : ", BA)
print("Batas Bawah : ", BB)

df_train = df_train[~((df_train["Premi"] < BB) | (df_train["Premi"] > BA))]
```

✓ 0.8s

```
Batas Atas :  59721.625
Batas Bawah :  4066.625
```

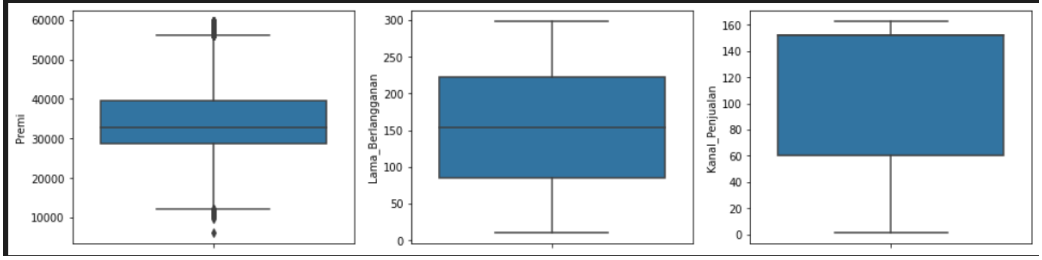
Disamping itu, pada percobaan **akan digunakan juga data yang memiliki outlier.** Hal ini disaranakan sehingga dapat dilihat hasil laporan perbandingan diantara yang menggunakan dan tidak menggunakan outlier sehingga didapatkan jaminan hasil yang maksimal .

Adapun hasil dari penghapusan outlier tersebut terlihat pada boxplot berikut.

```
print("Total Dataset :", len(df_train))

fig, ax = plt.subplots(ncols=3, figsize=(16, 4))
sns.boxplot(y='Premi', data=df_train, ax=ax[0])
sns.boxplot(y='Lama_Berlangganan', data=df_train, ax=ax[1])
sns.boxplot(y='Kanal_Penjualan', data=df_train, ax=ax[2])
plt.show()
```

✓ 0.6s

```
Total Dataset : 230631
```



- **Transformasi Data**

Transformasi data merupaakn salah satu cara menormalkan data dengan merubah skala pengukuran data asli menjadi bentuk lain yang masih memiliki nilai sama sehingga data dapat memenuhi kriteria uji asumsi klasik [3].

Pada tranformasi data projek ini dilakukan proses label encoding pada beberapa kolom yang memiliki data kategorikal seperti kolom Jenis_Kelamin, Umur_Kendaraan, dan Kendaraan_Rusak. Proses transformasi ini dilakukan sebelum pengisian data kosong sehingga kedepannya data kategorikal kosong dapat diisi dengan data yang telah ditransformasi. Penggantian label pada setiap kolom tersebut dilakukan dengan menggunakan fungsi pada pandas yaitu replace. Adapun hasil dan kode label encoding dijabarkan sebagai berikut.

```
def label_encoding(df):
    df['Jenis_Kelamin'] = df['Jenis_Kelamin'].replace(['Wanita', 'Pria'], [0, 1])
    df['Umur_Kendaraan'] = df['Umur_Kendaraan'].replace(['< 1 Tahun', '1-2 Tahun', '> 2 Tahun'], [0, 1, 2])
    df['Kendaraan_Rusak'] = df['Kendaraan_Rusak'].replace(['Tidak', 'Pernah'], [0, 1])

label_encoding(df_train)
label_encoding(df_test)

df_train.head()
```

✓ 0.5s

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak
0	0.0	30.0	1.0	33.0	1.0	0.0	0.0
1	1.0	48.0	1.0	39.0	0.0	2.0	1.0
2	NaN	21.0	1.0	46.0	1.0	0.0	0.0
3	0.0	58.0	1.0	48.0	0.0	1.0	0.0
4	1.0	50.0	1.0	35.0	0.0	2.0	NaN

Disamping melakukan laberl encoding hal lain yang dilakukan pada transformasi data adalah normalisasi. Adapun pada projek ini akan dilakukan penggunaan 2 normalisasi yaitu standart scaler dan min-max scaler sebagai acuan untuk eksperimen mendapatkan hasil maksimal yang terbaik.

```
numerical = ['Lama_Berlangganan', 'Umur', 'Kode_Daerah', 'Kanal_Penjualan', 'Premi']

scaler = MinMaxScaler()

df_train[numerical] = scaler.fit_transform(df_train[numerical].values)
df_train.sample(5)

```

✓ 0.3s

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan
88591	1.0	0.430769	1.0	0.961538	0.0	1.0	1.0	0.494620	0.154321
252024	1.0	0.292308	1.0	0.538462	0.0	1.0	1.0	0.431547	0.759259
88228	0.0	0.692308	1.0	0.538462	0.0	1.0	1.0	0.835139	0.364198
6422	1.0	0.076923	1.0	0.500000	0.0	0.0	1.0	0.376737	0.981481
66135	1.0	0.169231	1.0	0.538462	0.0	1.0	1.0	0.786893	0.154321

```
numerical = ['Lama_Berlangganan', 'Umur', 'Kode_Daerah', 'Kanal_Penjualan', 'Premi']

scaler = StandardScaler()

df_train[numerical] = scaler.fit_transform(df_train[numerical].values)
df_train.sample(5)

```

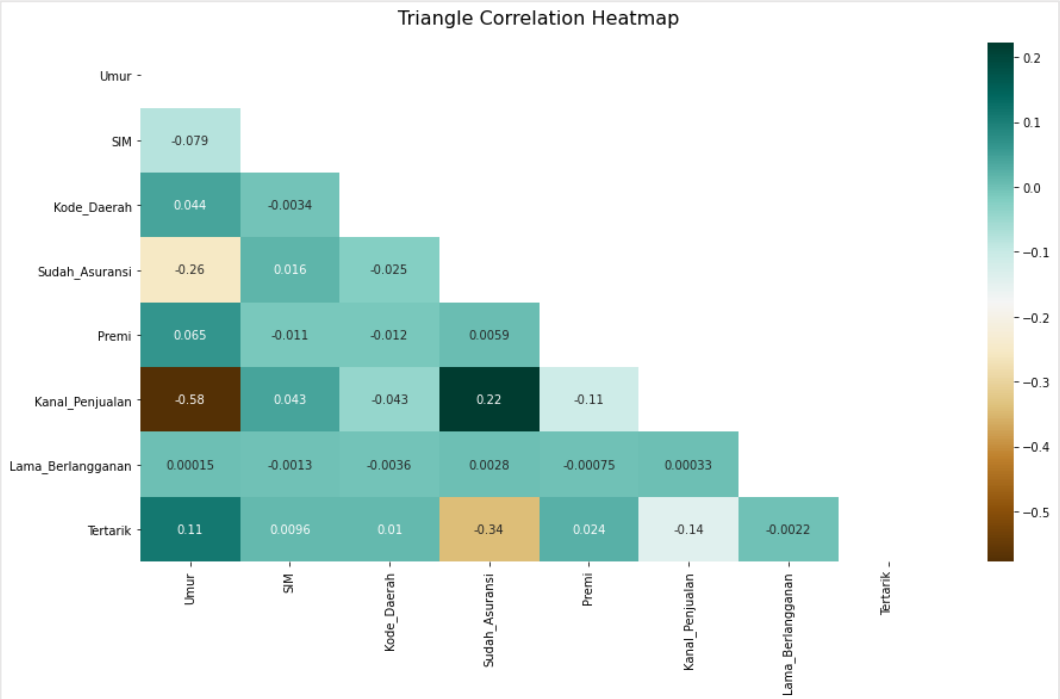
✓ 0.2s

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan
271251	0.0	-0.667122	1.0	0.113303	1.0	0.0	0.0	2.259207	0.710168
44542	0.0	-0.865617	1.0	-1.302278	1.0	1.0	0.0	-1.066390	0.710168
170779	1.0	-0.931782	1.0	-0.909061	1.0	0.0	0.0	0.207328	0.710168
45572	1.0	0.457685	1.0	-0.201270	0.0	1.0	0.0	-0.232436	0.182793
229792	1.0	2.574968	1.0	1.135667	1.0	1.0	1.0	-0.337914	-1.926705

- **Reduksi Data**

Dalam prosesnya seringkali algoritma klasifikasi dan clustering menjadi bermasalah pada data dengan dimensi tinggi berupa menurunnya akurasi klasifikasi dan kualitas cluster. Disamping itu juga yang paling penting adalah sangat berpengaruh pada waktu komputasi yang lama. Permasalahan ini dapat diatasi dengan melakukan proses reduksi dimensi. Proses reduksi dimensi terdiri dari dua yaitu ekstrasi fitur dan seleksi fitur. Seleksi fitur dapat digunakan untuk mengurangi biaya pengumpulan data, reduksi waktu komputasi, melakukan visualisasi data maupun menggali informasi tentang penyebab masalah [4].

Pada projek ini dilakukan dua macam reduksi dimensi tersebut. Pada proses seleksi fitur, fitur atau kolom akan dipilih untuk dijadikan cluster dilakukan dengan cara melihat korelasi tertinggi antar kolomnya. Dengan menggunakan heatmap correlation dapat ditentukan kolom yang paling berkorelasi yang selanjutnya akan kita gunakan untuk melakukan percobaan clustering.



Disamping seleksi fitur, untuk ekstrasi fitur digunakan teknik PCA (Principal Component Analysis). Principal Component Analysis (PCA) digunakan untuk mereduksi dimensi tanpa mengurangi karakteristik data secara signifikan. Metode ini mengubah sebagian besar variabel asli yang berkorelasi menjadi satu himpunan variabel baru yang lebih kecil dan saling bebas [4].

Adapun proses penggunan PCA ini dilakukan menggunakan library sklearn dengan melakukan reduksi hingga menjadi 2 komponen untuk keseluruhan feature. Berikut merupakan kode dan hasil dari penggunaan PCA.

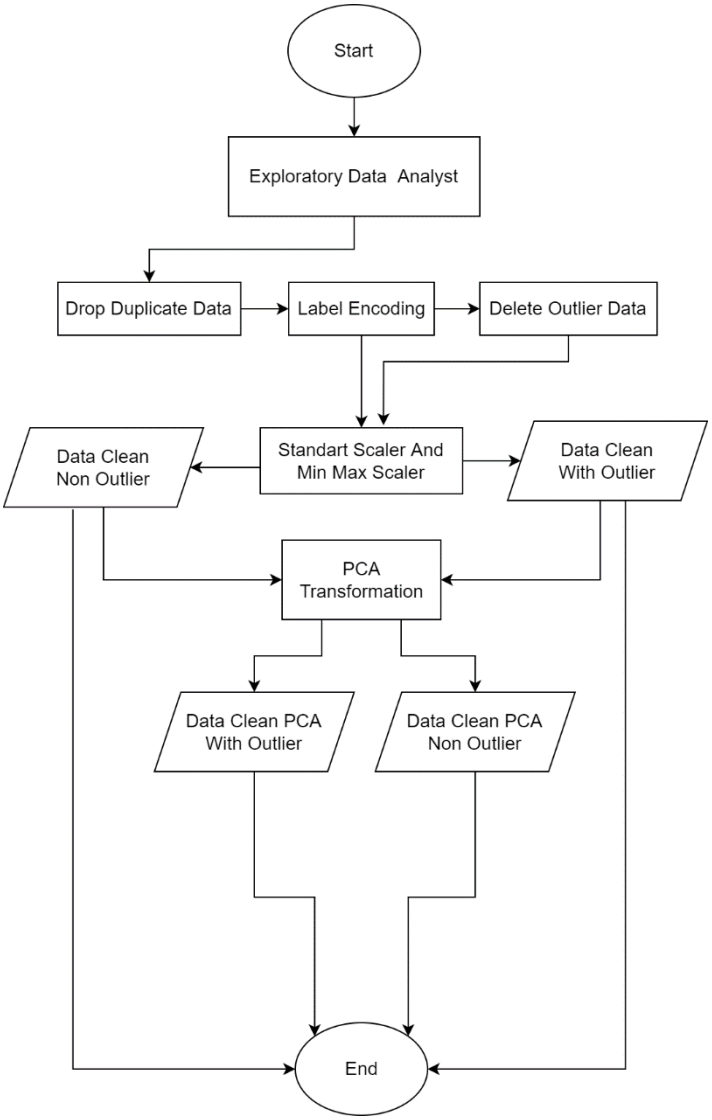
```
pca = PCA(n_components=2)
component = pca.fit_transform(df_train_pca)
component_test = pca.transform(df_test_pca)

df_train_pca = pd.DataFrame(data=component, columns=['Component_1', 'Component_2'])
df_train_pca['Tertarik'] = df_label_train.values

df_test_pca = pd.DataFrame(data=component_test, columns=['Component_1', 'Component_2'])
df_test_pca['Tertarik'] = df_label_test.values

df_train_pca.sample(5)
```

✓ 0.8s

	Component_1	Component_2	Tertarik
119680	-0.256363	0.898518	0
213225	0.868486	0.098863	0
169606	-0.311025	0.842598	0
222511	0.638952	-0.525892	0
206831	0.624045	-0.544173	1

2.3 Keseluruhan Alur Pra-Pemrosesan

Adapun keseluruhan dari alur pemrosesan data pada projek ini dijabarkan melalui flowchart sebagai berikut.

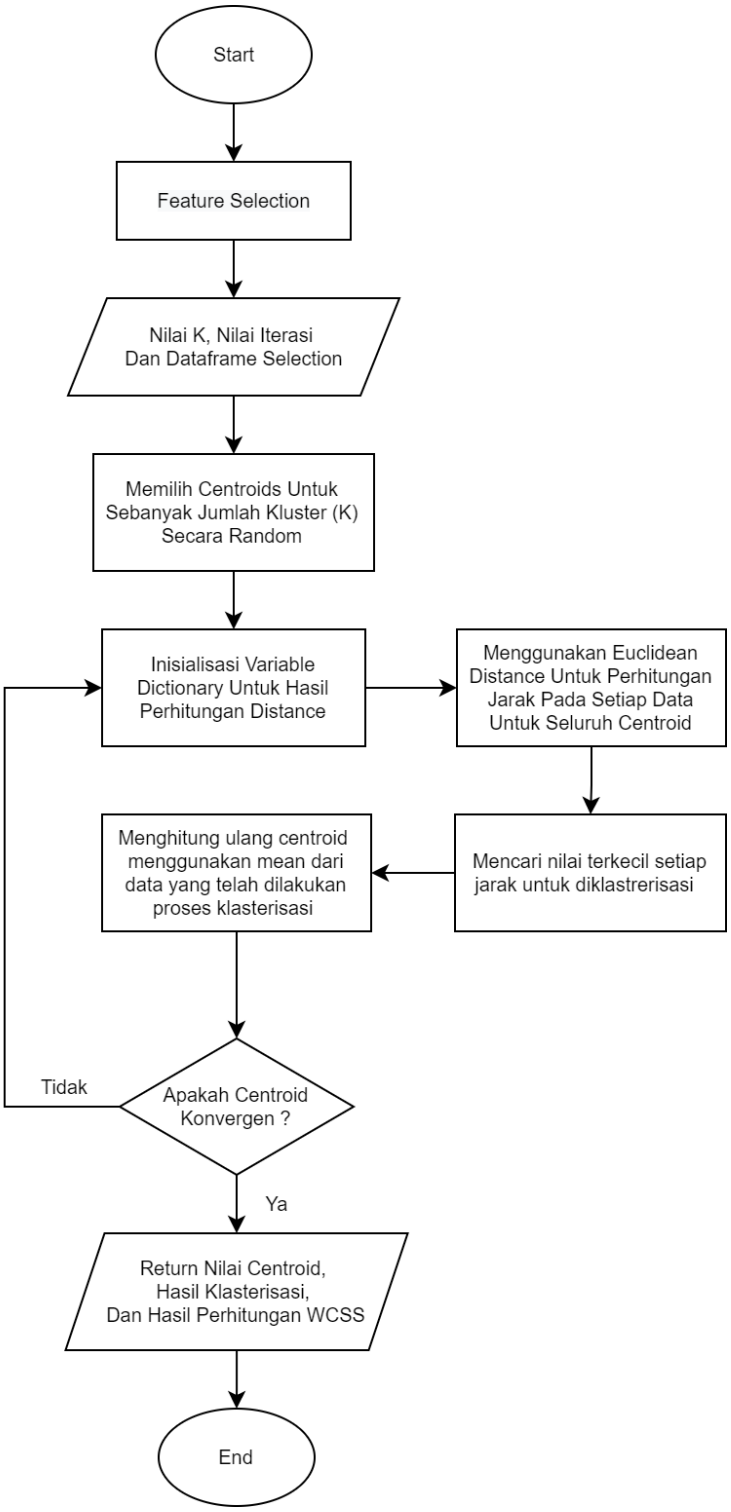


Dari alur pemrosesan diatas akan dihasilkan **8 macam dataset** dari hasil langkah pra-pemrosesan data yang selanjutnya akan digunakan untuk proses eksperimen. Adapun keseluruhan dataset tersebut diantaranya yaitu:

- Dataset Min Max Scaler Tanpa Outlier
- Dataset Min Max Scaler Dengan Outlier
- Dataset Min Max Scaler PCA Tanpa Outlier
- Dataset Min Max Scaler PCA Dengan Outlier
- Dataset Standard Scaler Tanpa Outlier
- Dataset Standard Scaler Dengan Outlier
- Dataset Standard Scaler PCA Tanpa Outlier
- Dataset Standard Scaler PCA Dengan Outlier

3. Pemodelan

Sistem clustering yang akan dibuat oleh saya adalah clustering menggunakan algoritma K-Means. Algoritma ini saya pilih karena dapat mencapai kekonvergenan, mudah untuk diimplementasikan dan bagus digunakan pada dataset skala besar. [5] Adapun algoritma K-Means yang dibuat dijabarkan pada flowchart sebagai berikut.



3.1 Fungsi Euclidean Distance

Dalam mencari perhitungan jarak terdekat untuk menentukan cluster digunakan metode euclidean distance sebagai bawaan. Basis dasar dari euclidean distance sendiri didasarkan dari perhitungan teorema Pythagoras. Metode ini digunakan karena memiliki efisiensi dalam komputasi, mudah untuk digunakan, serta memberikan hasil terbaik pada perhitungan jarak. [6]

$$Euclidean\ Distance\ = \sqrt{\sum(x_i - y_i)^2}$$

Adapun pengimplementasian metode euclidean distance dalam proses pengkodean dijabarkan sebagai berikut.

```
# Mendefinisikan Metode Euclidean Untuk Perhitungan Jarak Dari Centroid
def euclidean(x, y):
    return math.sqrt(sum((x - y)**2))
```

3.2 Fungsi Pengecekan Kekonvergenan Centroid

Salah satu bentuk keunggulan menggunakan algoritma K-Means adalah akan selalu menghasilkan hasil yang konvergen atau biasa disebut sebagai local optima pada algoritma k-means yaitu centroid dari semua cluster tidak berubah lagi [7]. Adapun cara yang dilakukan untuk mencapai hal tersebut yaitu dengan membandingkan seluruh centroid sebanyak jumlah cluster (k) sebelumnya dengan seluruh centroid sebanyak jumlah cluster (k) saat ini.

```
# Pemberhentian Proses Iterasi Ketika Centroid Saat Ini Sama Dengan Nilai Centroid Sebelumnya
# Dilakukan Dengan Pengecekan Masing-Masing Value Dari Dictionary
def is_convergent(temp_centroid, centroid, k):
    for i in range(k):
        if (temp_centroid.get(i) != centroid.get(i))[0]:
            return False

    return True
```

3.3 Fungsi Model K-Means

Pada pemodelan K-Means digunakan standar dari algoritma K-Means sesuai dengan flowchart yang telah dijabarkan diatas. Untuk parameter maksimal iterasi yang digunakan adalah 100 hal ini dikarenakan rata-rata proses hingga mencapai kekonvergenan tertinggi hanya dicapai pada proses iterasi ke 60 saja.

Untuk parameter-paramater yang digunakan yaitu jumlah cluster (k) dalam projek ini dibatasi dari kluster ke-2 hingga kluster ke-5 karena keterbatasan dalam segi komputasi yang tinggi dan yang terakhir untuk feature-feature yang akan dipilih selanjutnya akan dijelaskan pada bagian evaluasi dan eksperimen.

Adapun hasil dari pengkodean seluruh proses dari algoritma K-Means dijabarkan sebagai berikut.

- **Kontrol Program Utama**

```python
# Inisialisasi Array Untuk WCSS Dan Keseluruhan Hasil Cluster
wcss_all = []
cluster_all = []

# Merubah DataFrame Menjadi Numpy Array
data = df.to_numpy()

# Memasukkan Nilai Maksimum Iterasi
max_iter = 100

# Inisialisasi Range K
K = range(2, 6)

# Melakukan Perulangan Untuk Setiap Range
for i in K:

    # Menggunakan Model Yang Telah Dibangun Untuk Mendapatkan Centroid, Cluster Dan Nilai WCSS
    centroid, cluster, wcss = kmeans(i, max_iter, data)
    print(f'Centroid Cluster-{i}: \n{centroid}\n')

    # Memasukkan Setiap Nilai Yang Direturn Kedalam Array
    wcss_all.append(wcss)
    cluster_all.append([i, cluster, centroid])
```

- **Fungsi K-Means**

```python
def kmeans(k, max_iter, df):
    # Inisialisasi Dictionary Centroid Kemudian Memilih Centroid Secara Random Pada Data
    # Sesuai Banyaknya Cluster Yang Didefinisikan
    centroid = {i: df[random.randint(0, len(df))] for i in range(k)}

    # Mengcopy Centroid Saat Ini Untuk Acuan Pemberhentian Proses Iterasi Centroid Maksimal
    temp_centroid = centroid.copy()

    for _ in range(max_iter):
        # Mendefinisikan Dictionary Cluster Dan Mengisi Setiap Key
        # Pada Cluster Berisi Array kosong, contoh : {0: [], 1: [], 2: []}
        cluster = {i: [] for i in range(k)}
        wcss = []

        # Mencari Nilai Terdekat Pada Data Untuk Setiap Centroid Yang Telah Di Definisikan Sebelumnya
        # Menggunakan Metode Euclidean Dan Mencari Nilai Minimum Pada Setiap Distance
        # Untuk Dimasukkan Kedalam Dictionary Cluster
        for x in df:
            dist = [euclidean(x, centroid[c]) for c in centroid]
            wcss.append(np.min(dist))
            cluster[dist.index(min(dist))].append(x)

        # Mencari Nilai Centroid Ulang Dari Setiap Cluster
        # Yang Telah Dicari Sebelumnya Menggunakan Mean
        for cl in cluster:
            centroid[cl] = np.mean(cluster[cl], axis=0)

        # Mengecek Jika Centroid Telah Mencapai Nilai Maksimum / Konvergen
        if is_convergent(temp_centroid, centroid, k): break

        # Mengcopy Centroid Yang Telah Berubah Dari Perhitungan Sebelumnya
        temp_centroid = centroid.copy()

    # Mengembalikan Nilai Dari Variabel Centroid, Cluster Dan WCSS
    return centroid, cluster, sum(wcss)
```

4. Evaluasi

Pada proses evaluasi dari model cluster yang telah dibuat, digunakan dua macam metriks evaluasi yaitu dengan menggunakan Silhoutte Score dan Elbow Method dengan Within Cluster Sum Of Squares (WCSS).

5.1 Metrik Perhitungan Silhoutte Score

Pengujian Silhoutte Score pada sebuah model bertujuan untuk mendapatkan informasi sedekat apa hubungan antara satu objek dengan objek lain pada sebuah cluster dan sejauh apa sebuah cluster dengan yang lainnya. Adapun Silhouette Coefficient terdapat pada angka antara nilai -1 sampai 1 dimana nilai Silhouette Coefficient semakin mendekati nilai 1, maka semakin bagus pengelompokan objek-objek kedalam sebuah cluster [8]. Adapun penghitungan Silhoutte Coefficient untuk suatu titik tunggal melalui proses tiga tahap yaitu:

- Pada objek ke i, dihitung rata-rata jarak antar satu objek dengan objek yang lain dalam satu cluster.

$$a(i) = \frac{\sum Distance(i,j)}{|C_i|-1}$$

- Pada objek ke i, dihitung rata-rata jarak terhadap semua objek-objek yang ada pada di cluster lain.

$$b(i) = \min\left(\frac{\sum Distance(i,j)}{|C_j|}\right)$$

- Sehingga Silhoutte Coeffisien untuk suatu titik didefinisikan dengan:

$$S(i) = \min\left(\frac{(b_{(i)} - a_{(i)})}{max(a_{(i)},\ b_{(i)})}\right)$$

$S = 1 - \frac{a}{b}$, jika a < b

$S = 1 - \frac{b}{a}$, jika a ≥ b

Hasil clustering dapat disebutkan bagus jika Silhoutte Coeffisien harus bernilai positif (ai < bi) dan ai harus mendekati 0, sehingga mendapatkan angka Silhoutte Coeffisien yang paling tinggi yaitu 1 saat a(i) = 0, Sehingga jika nilai S(i) = 1 artinya objek i berada didalam cluster yang tepat [8]. Dalam pengimplementasiannya akan digunakan library silhoutte score untuk mempermudah proses evaluasi metriks ini.

```
# Perulangan Setiap Cluster Untuk Menghitung Dan Menampilkan Hasil Silhoutte Score
for i, cluster in enumerate(clusterize):
    score = silhouette_score(cluster[['Umur', 'Kanal_Penjualan']], cluster['Cluster'])
    print(f'Cluster-{i+2}, Silhouette Score = {score}')
```

```
Cluster-2, Silhouette Score = 0.6897537450307077
Cluster-3, Silhouette Score = 0.6455086018983847
Cluster-4, Silhouette Score = 0.6147766012999412
Cluster-5, Silhouette Score = 0.6213020960247723
```

5.2 Metrik Elbow Method Dengan Within Cluster Sum Of Squares (WCSS)

Dalam pengukuran metrik menggunakan metode Elbow, kita akan memvariasikan jumlah cluster (K) yang kita inginkan. Untuk setiap nilai K, kita akan melakukan perhitungan WCSS (Within-Cluster Sum of Square). WCSS adalah jumlah kuadrat jarak antara setiap titik dan pusat massa dalam sebuah cluster. Dengan bertambahnya jumlah cluster, nilai WCSS akan mulai berkurang. Nilai WCSS terbesar adalah ketika nilai K = 1.

Ketika kita menganalisis grafik plottingan hasil elbow, kita akan dapat melihat bahwa grafik akan berubah dengan cepat pada suatu titik dan akan menciptakan bentuk siku. Dari titik ini, grafik mulai bergerak hampir sejajar dengan sumbu X. Nilai K yang sesuai dengan titik ini adalah nilai K optimal atau jumlah cluster yang optimal [9].

Pada perhitungan WCSS dilakukan dengan mencari jarak antara titik tertentu yang paling minimum dengan titik centroid yang telah ditetapkan. Kemudian proses tersebut diulangi untuk semua titik di kluster, lalu menjumlahkan nilai untuk keseluruhan hasil kluster. Adapun proses perhitungan tersebut dilakukan bersamaan pada fungsi kmeans yang dijabarkan sebagai berikut.

```
def kmeans(k, max_iter, df):
    # Inisialisasi Dictionary Centroid Kemudian Memilih Centroid Secara Random Pada Data
    # Sesuai Banyaknya Cluster Yang Didefinisikan
    centroid = {i: df[random.randint(0, len(df))] for i in range(k)}

    # Mengcopy Centroid Saat Ini Untuk Acuan Pemberhentian Proses Iterasi Centroid Maksimal
    temp_centroid = centroid.copy()

    for _ in range(max_iter):
        # Mendefinisikan Array Penyimpanan Jarak Perhitungan Untuk WCSS
        wcss = []

        # Mendefinisikan Dictionary Cluster Dan Mengisi Setiap Key
        # Pada Cluster Berisi Array kosong, contoh : {0: [], 1: [], 2: []}
        cluster = {i: [] for i in range(k)}

        # Mencari Nilai Terdekat Pada Data Untuk Setiap Centroid Yang Telah Di Definisikan Sebelumnya
        # Menggunakan Metode Euclidean Dan Mencari Nilai Minimum Pada Setiap Distance
        # Untuk Dimasukkan Kedalam Dictionary Cluster
        for x in df:
            dist = [euclidean(x, centroid[c]) for c in centroid]
            wcss.append(np.min(dist))
            cluster[dist.index(min(dist))].append(x)

        # Mencari Nilai Centroid Ulang Dari Setiap Cluster
        # Yang Telah Dicari Sebelumnya Menggunakan Mean
        for cl in cluster:
            centroid[cl] = np.mean(cluster[cl], axis=0)

        # Mengecek Jika Centroid Telah Mencapai Nilai Maksimum / Konvergen
        if is_convergent(temp_centroid, centroid, k): break

        # Mengcopy Centroid Yang Telah Berubah Dari Perhitungan Sebelumnya
        temp_centroid = centroid.copy()

    # Mengembalikan Nilai Dari Variabel Centroid Dan Cluster
    return centroid, cluster, sum(wcss)
```

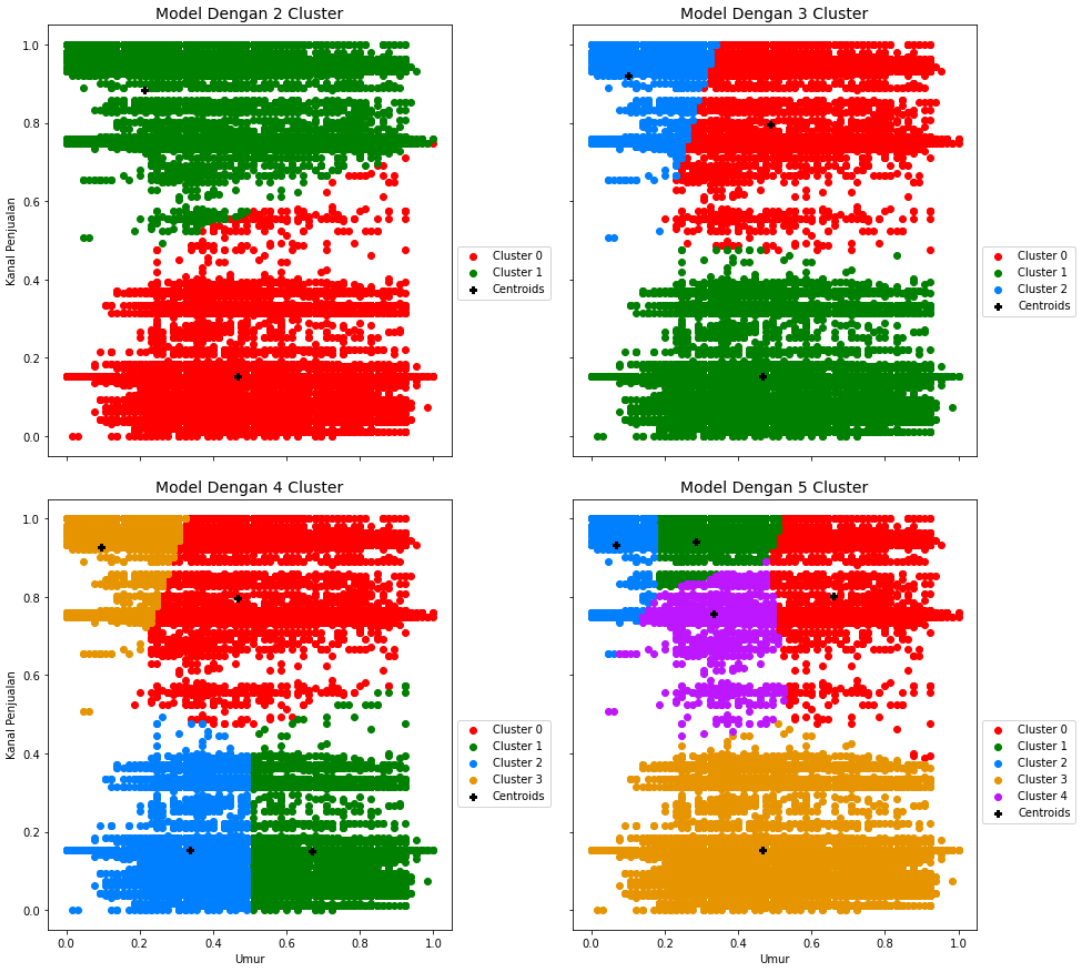
5.3 Evaluasi Model Terbaik

Setelah keseluruhan proses pembersihan data, pembangunan model dan melakukan keseluruhan pengukuran metriks langkah selanjutnya adalah melakukan evaluasi untuk mengukur tingkat keberhasilan program yang dibuat. Adapun parameter model terbaik yang dihasilkan terdapat pada kolom Umur – Kanal Penjualan dengan jumlah Kluster sebanyak 3. Model ini dipilih berdasarkan hasil dari metriks pengukuran WCSS dan hasil dari perhitungan Silhoutte Score. Adapun parameter beserta hasil dari perhitungan dan tampilan metriks tersebut dijabarkan sebagai berikut.

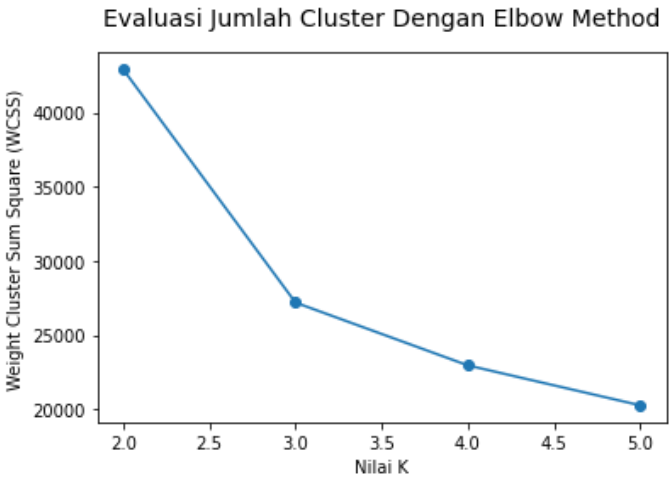
- **Parameter-Parameter Yang Digunakan**

Jenis	Value
Dataframe Kolom 1	Umur
Dataframe Kolom 2	Kanal_Penjualan
Scaler	Min Max Scaling (0 – 1)
Pembersihan Outlier	Ya

- **Hasil Plot Pengklasterisasian Model Berdasarkan Jumlah Cluster**



- **Hasil Plot Elbow Method Dengan WCSS**



- **Silhoutte Score Setiap Cluster**

Cluster (K)	Silhoutte Score	Silhoutte Score (Round)
2	0.6897537450307077	0.69
3	0.6455086018983847	0.65
4	0.6147766012999412	0.61
5	0.6213020960247723	0.62

- **Kesimpulan Evaluasi Model Terbaik**

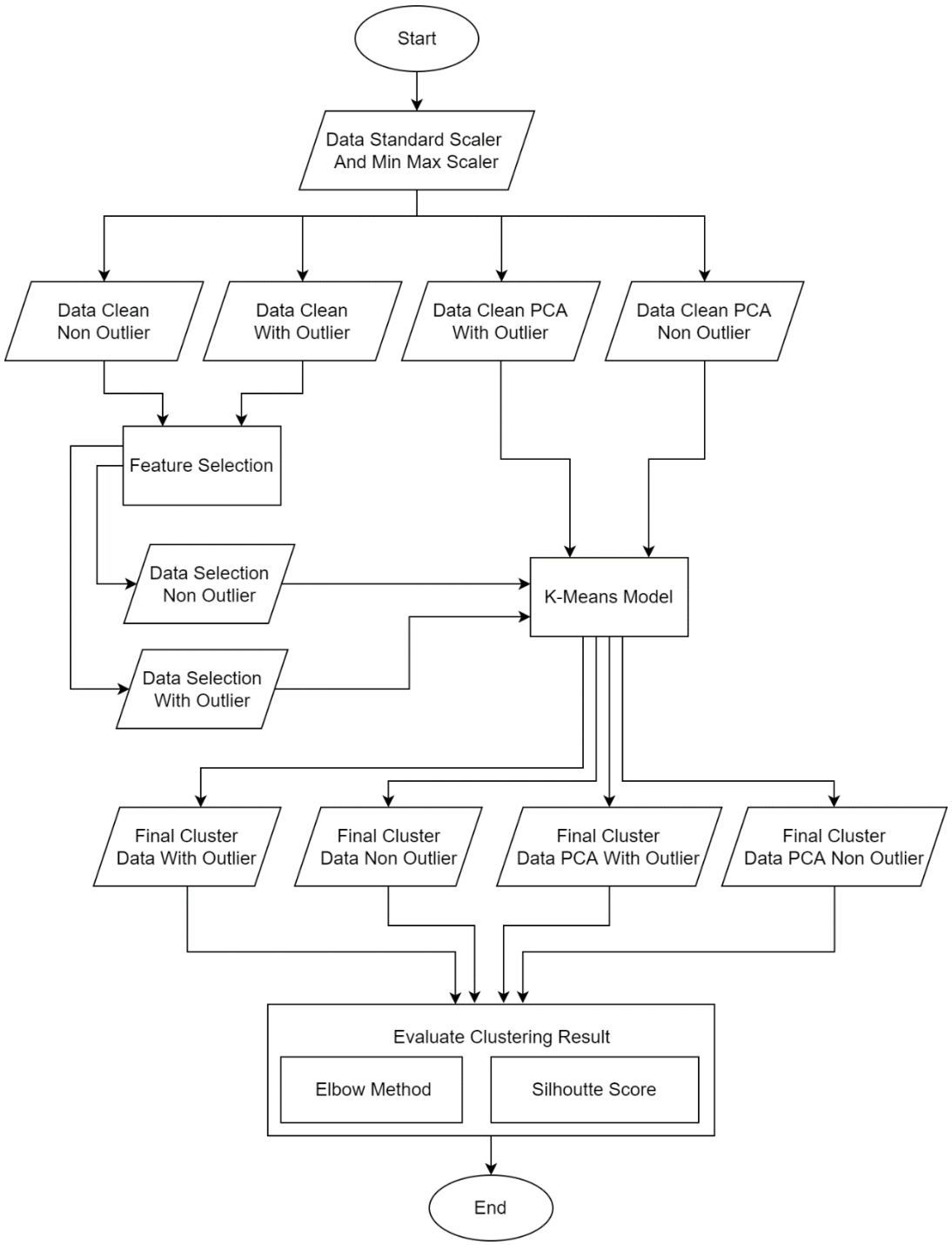
Dari hasil plot diatas dapat kita lihat pada elbow method dengan wcss diatas bahwa elbow (kemiringan) terbentuk ketika nilai K=3. Hal ini mengindikasikan bahwa maksimal kluster terbaik dapat kita dapatkan pada nilai K=3. Disamping itu, hal ini diperkuat dengan turunnya nilai Silhoutte Score selanjutnya yang ada pada pada kluster ke 4 dan ke 5 yang semakin rendah nilainya serta semakin berdekatan persebarannya pada hasil plot klasterisasi. Dari keseluruhan pernyataan tersebut dapat disimpulkan titik maksimal jumlah kluster terbaik ada pada nilai K=3 dengan Silhoutte Score Sebesar 0.65.

Hasil dari kluster model ini dapat diakses pada:

https://github.com/ShinyQ/Tugas-Besar_Pembelajaran-Mesin_KMeans-Clustering/blob/main/Min%20Max%20Scaler/Tubes_Clustering_Kmeans%20Non%20Outlier.ipynb

5. Eksperimen

Pada bagian eksperimen dilakukan pemrosesan secara keseluruhan dari hasil metode pra-pemrosesan data yang telah dibuat dan dengan memilih beberapa fitur (kolom) pada dataset dengan menggunakan acuan parameter-parameter pada model terbaik. Adapun proses tersebut dijabarkan pada flowchart sebagai berikut.

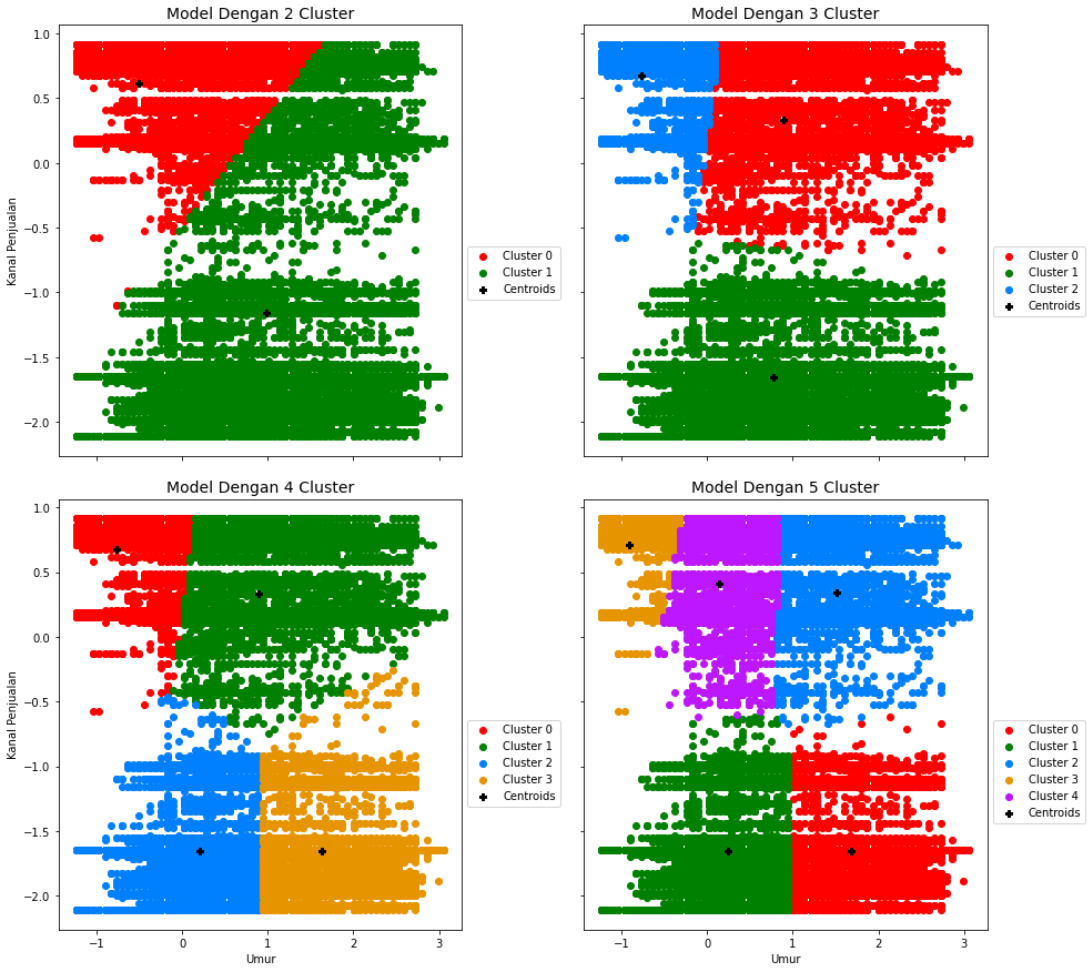


5.1 Kolom Umur – Kanal Penjualan Z-Score Outlier

- **Parameter-Parameter Yang Digunakan**

Jenis	Value
Dataframe Kolom 1	Umur
Dataframe Kolom 2	Kanal_Penjualan
Scaler	Standard Scaling (-3, 3)
Pembersihan Outlier	Tidak

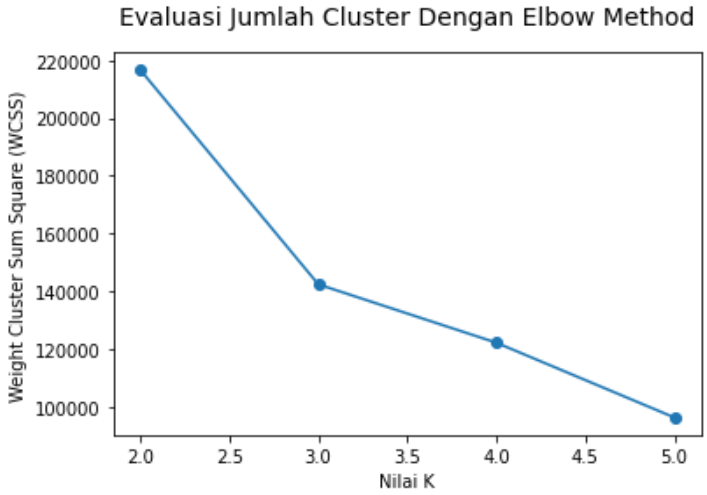
- **Hasil Plot Pengklasterisasian Model Berdasarkan Jumlah Cluster**



- **Silhoutte Score Setiap Cluster**

Cluster (K)	Silhoutte Score	Silhoutte Score (Round)
2	0.5933172464263627	0.59
3	0.6118127327806678	0.61
4	0.6055453841962751	0.61
5	0.592139429397299	0.59

- **Hasil Plot Elbow Method Dengan WCSS**

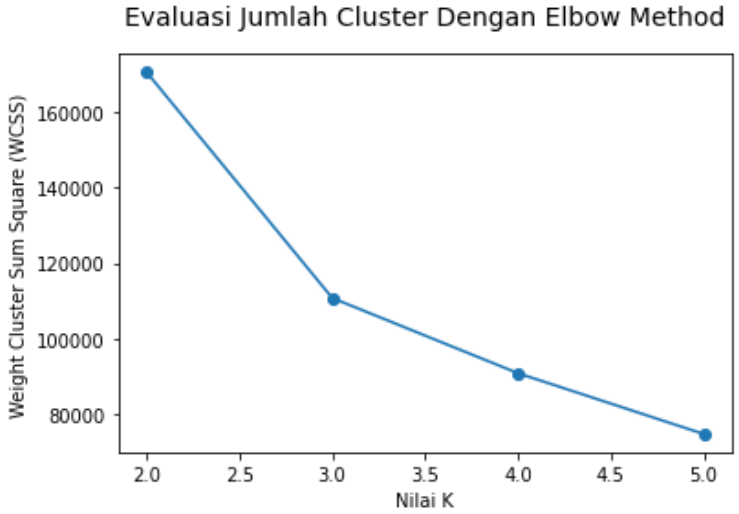


5.2 Kolom Umur – Kanal Penjualan Z-Score

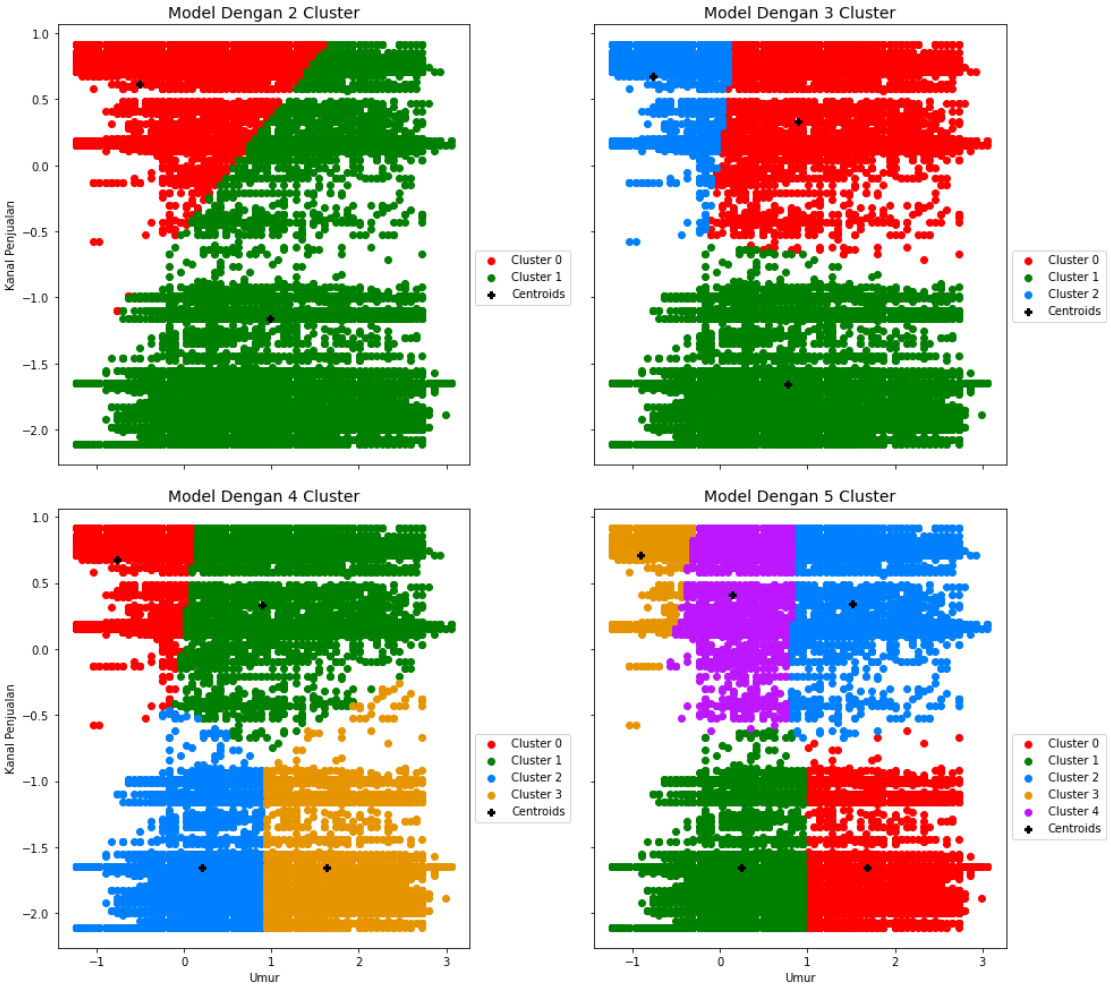
- **Parameter-Parameter Yang Digunakan**

Jenis	Value
Dataframe Kolom 1	Umur
Dataframe Kolom 2	Kanal_Penjualan
Scaler	Standard Scaler (-3 , 3)
Pembersihan Outlier	Ya

- **Hasil Plot Elbow Method Dengan WCSS**



- **Hasil Plot Pengklasterisasian Model Berdasarkan Jumlah Cluster**



- **Silhoutte Score Setiap Cluster**

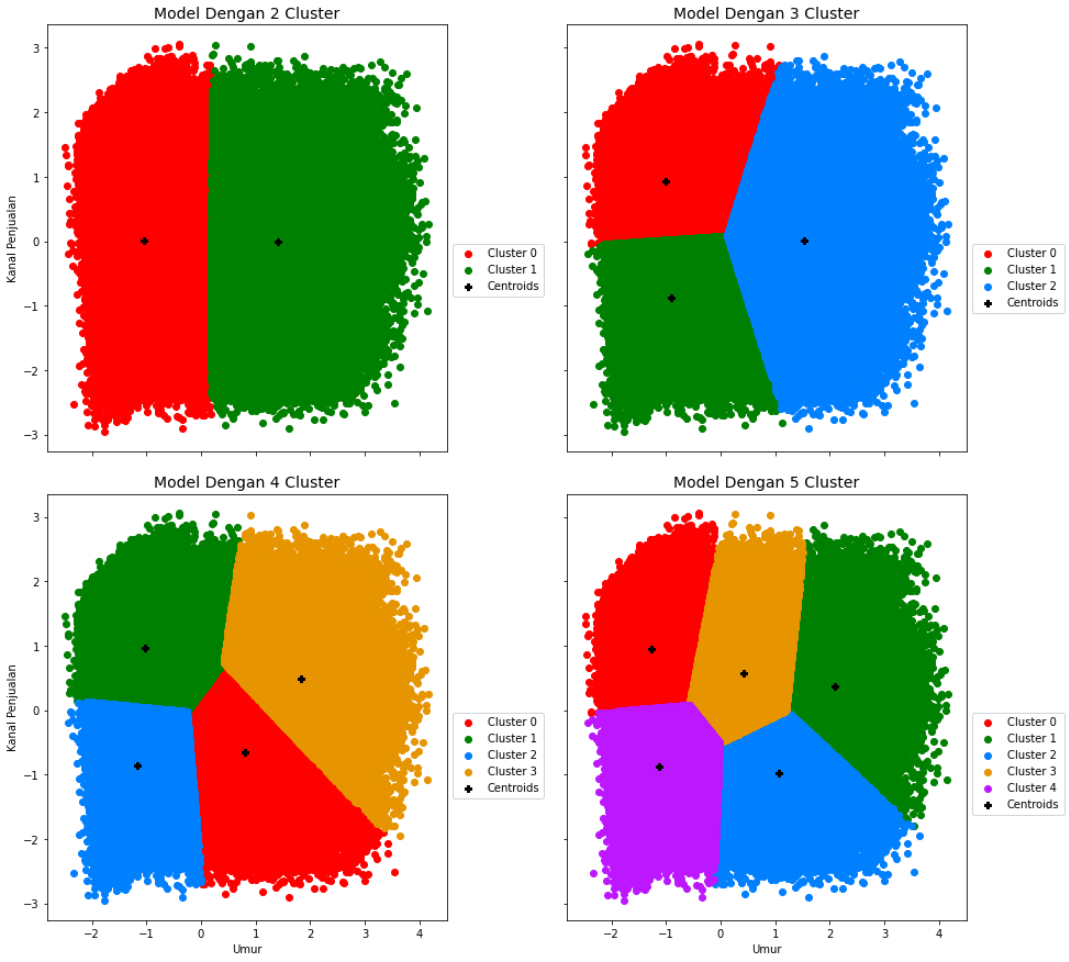
Cluster (K)	Silhoutte Score	Silhoutte Score (Round)
2	0.6053422096756218	0.61
3	0.623572794546726	0.62
4	0.602834282904698	0.60
5	0.6083513884901214	0.61

5.3 Kolom Umur – Kanal Penjualan Z-Score PCA

- **Parameter-Parameter Yang Digunakan**

Jenis	Value
Dataframe Kolom 1	Umur
Dataframe Kolom 2	Kanal_Penjualan
Scaler	Standard Scaling (-3, 3)
Pembersihan Outlier	Ya

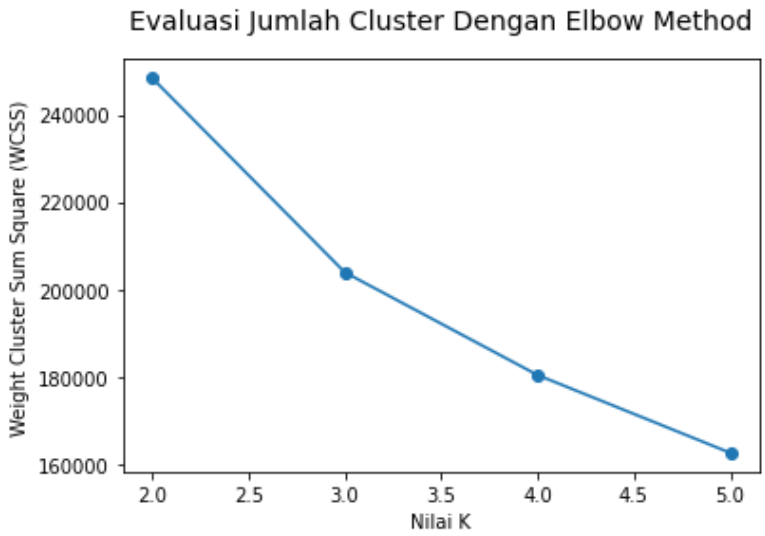
- **Hasil Plot Pengklasterisasian Model Berdasarkan Jumlah Cluster**



- **Silhoutte Score Setiap Cluster**

Cluster (K)	Silhoutte Score	Silhoutte Score (Round)
2	0.44910819198362345	0.45
3	0.4177727741630425	0.42
4	0.37707613786314864	0.38
5	0.38397641868594057	0.38

- **Hasil Plot Elbow Method Dengan WCSS**

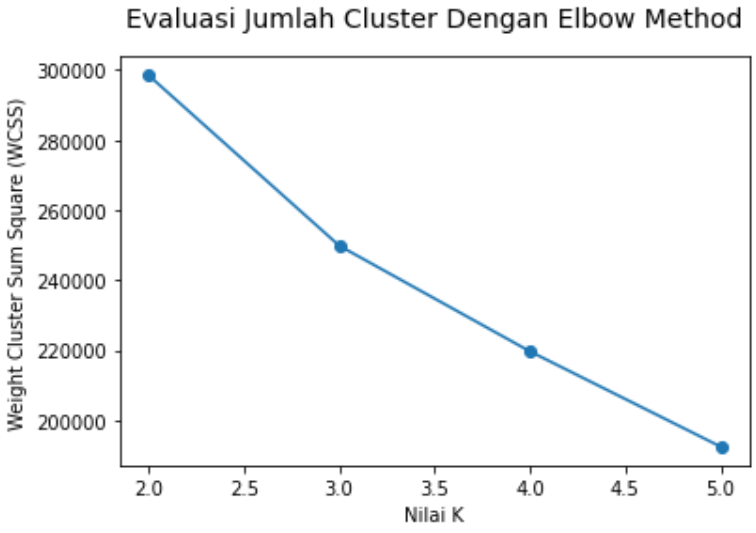


5.4 Kolom Umur – Kanal Penjualan Z-Score Outlier PCA

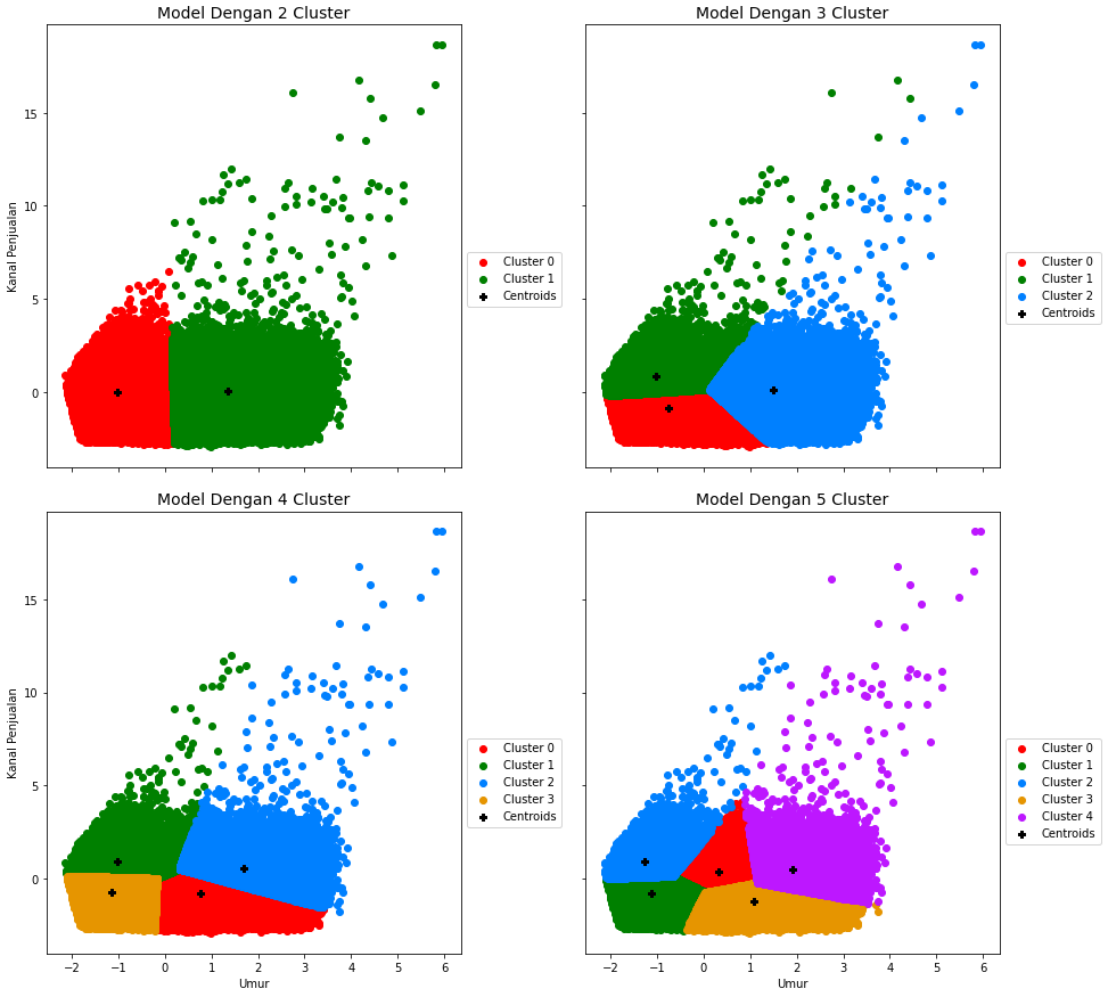
- **Parameter-Parameter Yang Digunakan**

Jenis	Value
Dataframe Kolom 1	Umur
Dataframe Kolom 2	Kanal_Penjualan
Scaler	Standard Scaler (-3 , 3)
Pembersihan Outlier	Tidak

- **Hasil Plot Elbow Method Dengan WCSS**



- **Hasil Plot Pengklasterisasian Model Berdasarkan Jumlah Cluster**



- **Silhoutte Score Setiap Cluster**

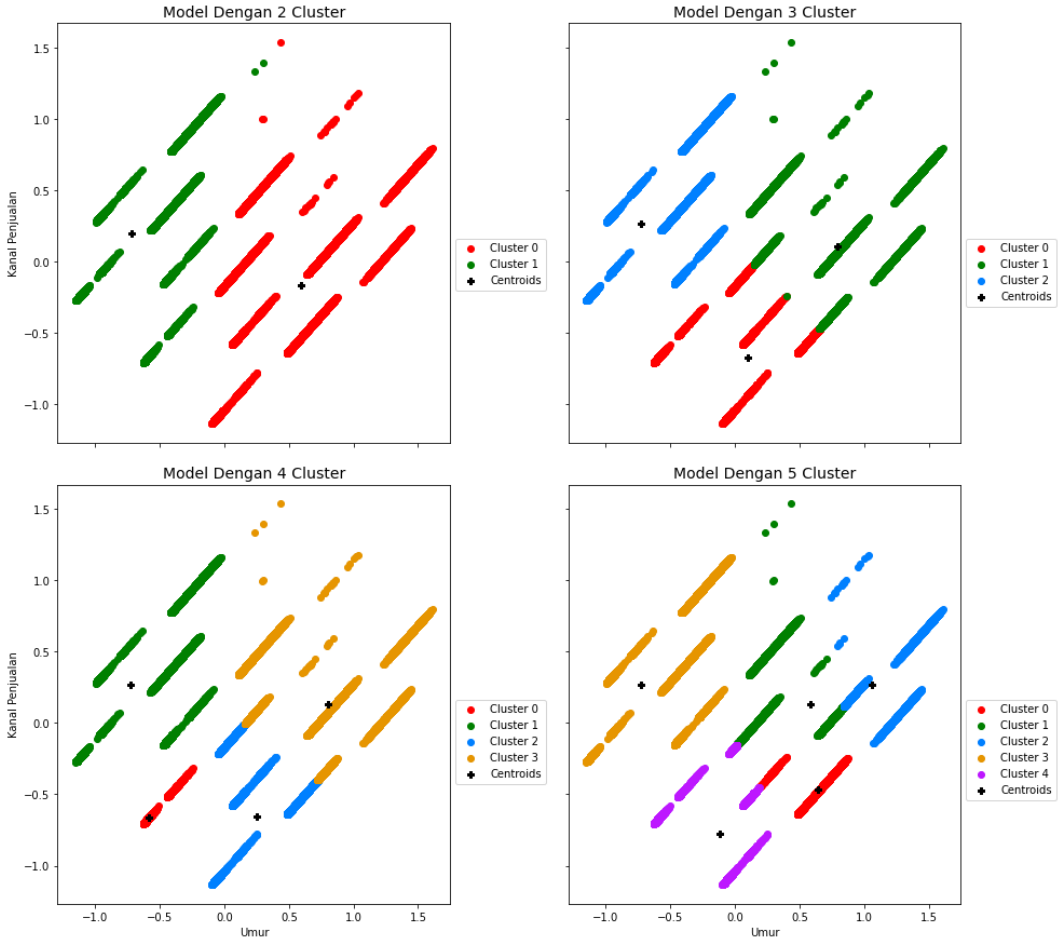
Cluster (K)	Silhoutte Score	Silhoutte Score (Round)
2	0.44646039259522163	0.45
3	0.40092668028779815	0.40
4	0.3725226527699101	0.37
5	0.40083592852009703	0.40

5.5 Kolom Umur – Kanal Penjualan Min Max Outlier PCA

- **Parameter-Parameter Yang Digunakan**

Jenis	Value
Dataframe Kolom 1	Umur
Dataframe Kolom 2	Kanal_Penjualan
Scaler	Min Max (0 – 1)
Pembersihan Outlier	Tidak

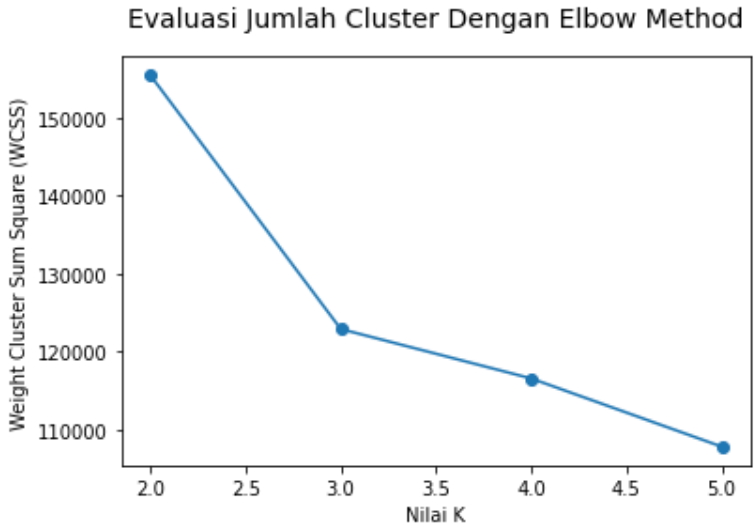
- **Hasil Plot Pengklasterisasian Model Berdasarkan Jumlah Cluster**



- **Silhoutte Score Setiap Cluster**

Cluster (K)	Silhoutte Score	Silhoutte Score (Round)
2	0.5206031182046656	0.52
3	0.4913909319585213	0.49
4	0.3839338250023323	0.38
5	0.44146386756909606	0.44

- **Hasil Plot Elbow Method Dengan WCSS**

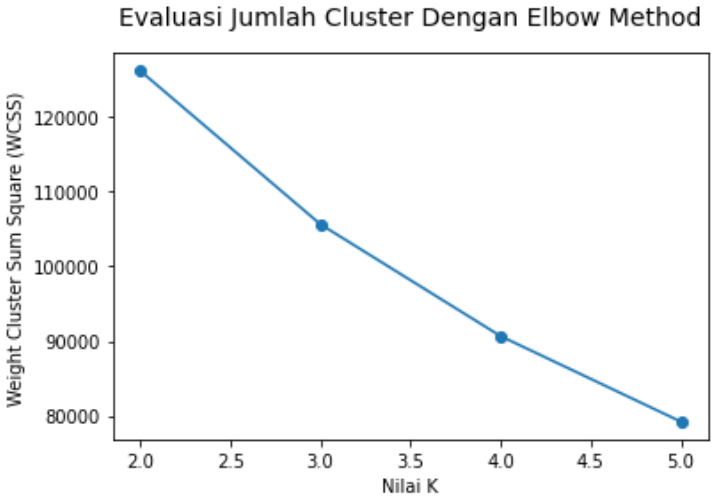


5.6 Kolom Umur – Kanal Penjualan Min Max PCA

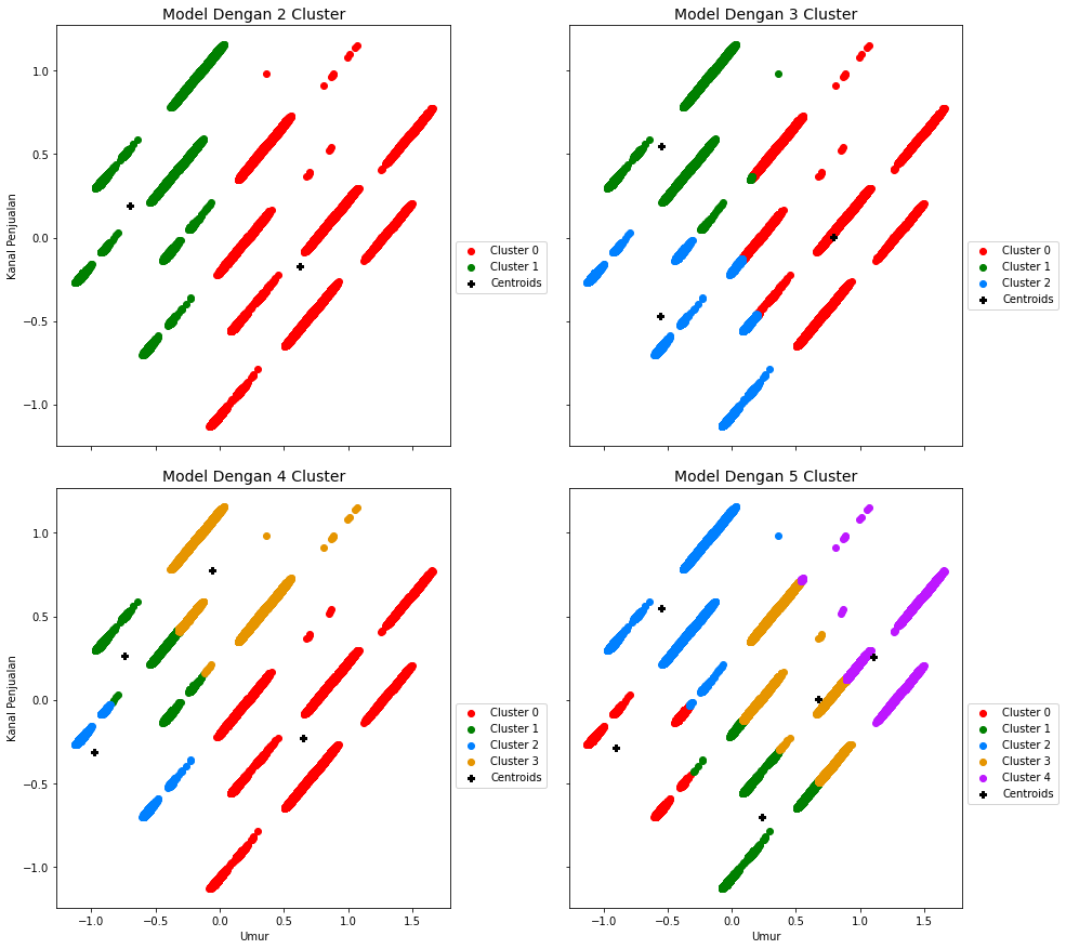
- **Parameter-Parameter Yang Digunakan**

Jenis	Value
Dataframe Kolom 1	Umur
Dataframe Kolom 2	Kanal_Penjualan
Scaler	Min Max Scaler (0 – 1)
Pembersihan Outlier	Ya

- **Hasil Plot Elbow Method Dengan WCSS**



- **Hasil Plot Pengklasterisasian Model Berdasarkan Jumlah Cluster**



- **Silhoutte Score Setiap Cluster**

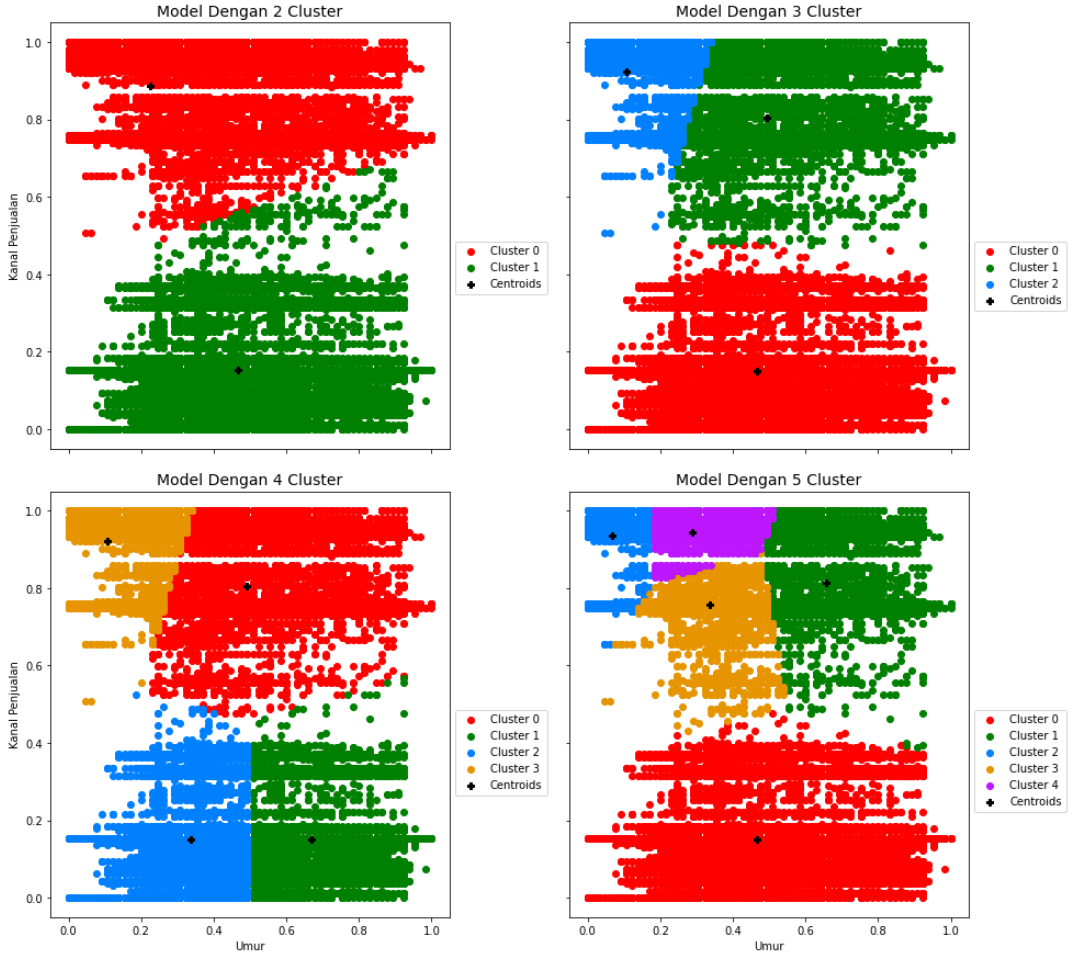
Cluster (K)	Silhoutte Score	Silhoutte Score (Round)
2	0.5237115109963842	0.52
3	0.4906018092662969	0.49
4	0.5147390896569016	0.51
5	0.4290976218111357	0.43

5.7 Kolom Umur – Kanal Penjualan Min Max Outlier

- **Parameter-Parameter Yang Digunakan**

Jenis	Value
Dataframe Kolom 1	Umur
Dataframe Kolom 2	Kanal_Penjualan
Scaler	Min Max (0 – 1)
Pembersihan Outlier	Tidak

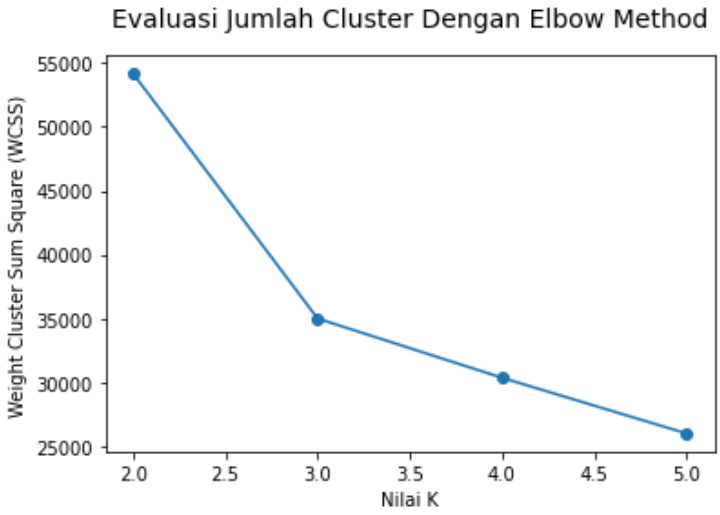
- **Hasil Plot Pengklasterisasian Model Berdasarkan Jumlah Cluster**



- **Silhoutte Score Setiap Cluster**

Cluster (K)	Silhoutte Score	Silhoutte Score (Round)
2	0.6834913654254513	0.68
3	0.6323481388391808	0.63
4	0.5992407176254866	0.60
5	0.609695435299693	0.60

- **Hasil Plot Elbow Method Dengan WCSS**

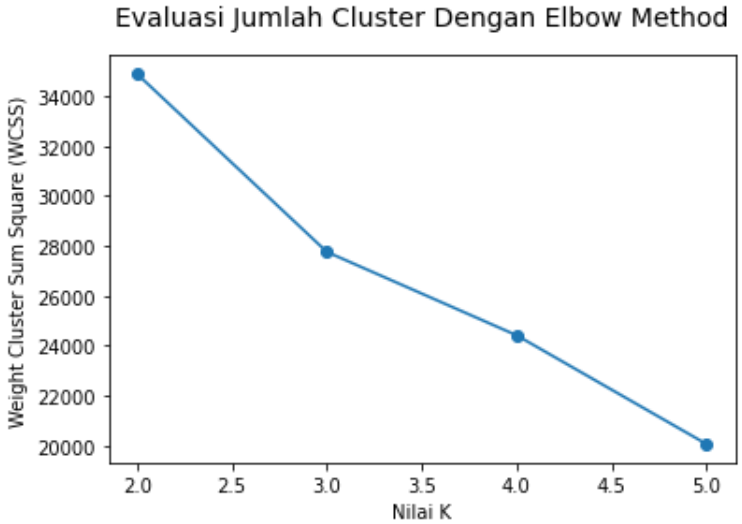


5.8 Kolom Premi - Kanal Penjualan

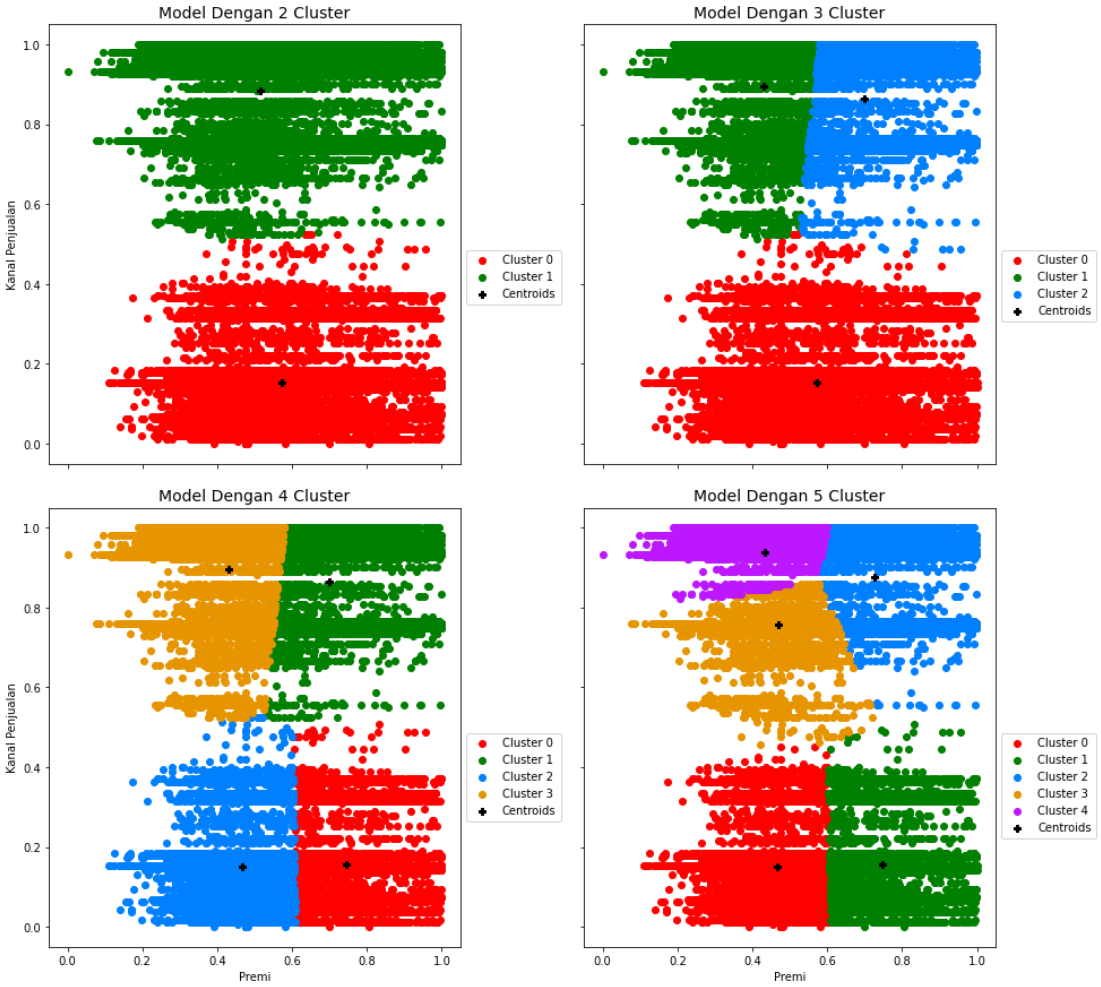
- **Parameter-Parameter Yang Digunakan**

Jenis	Value
Dataframe Kolom 1	Premi
Dataframe Kolom 2	Kanal_Penjualan
Scaler	Min Max Scaler (0 – 1)
Pembersihan Outlier	Ya

- **Hasil Plot Elbow Method Dengan WCSS**



- **Hasil Plot Pengklasterisasian Model Berdasarkan Jumlah Cluster**



- **Silhoutte Score Setiap Cluster**

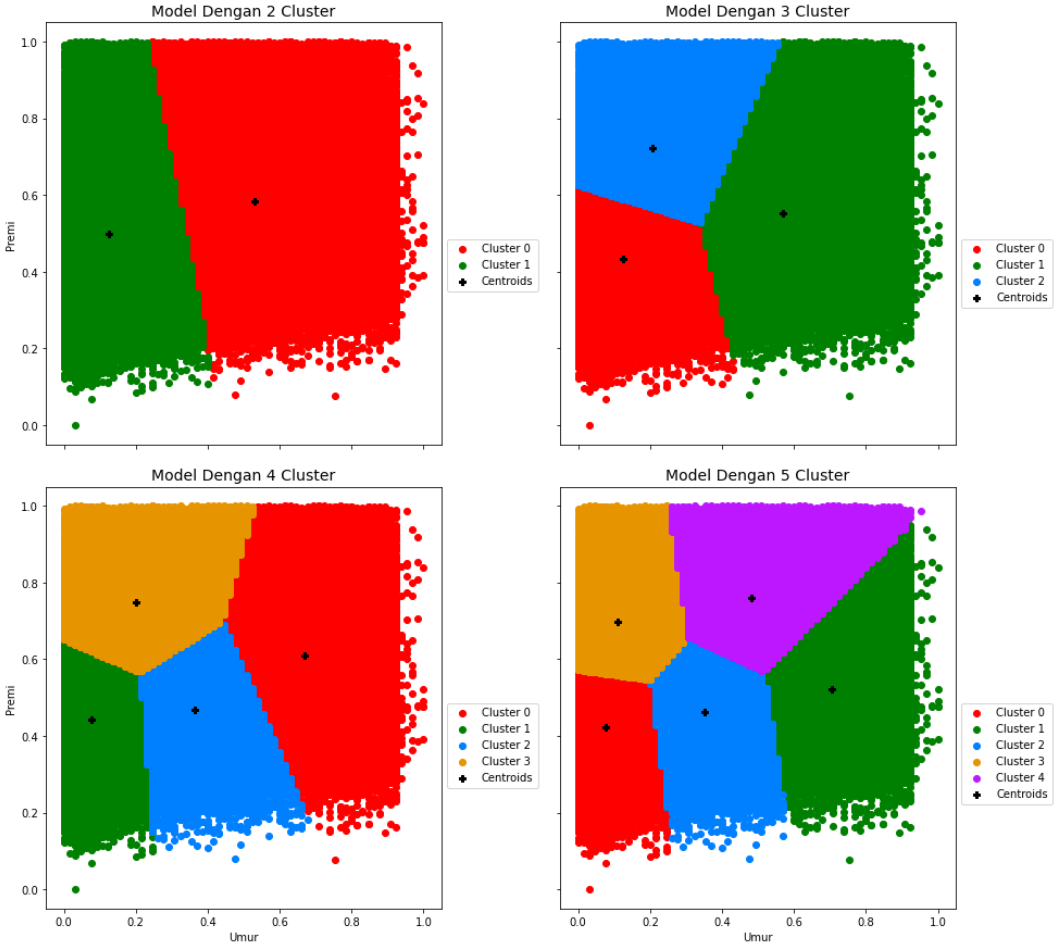
Cluster (K)	Silhoutte Score	Silhoutte Score (Round)
2	0.7231849777347742	0.72
3	0.5336388860136475	0.53
4	0.4880807816325216	0.49
5	0.49812921474844923	0.50

5.9 Kolom Umur – Premi Min Max

- **Parameter-Parameter Yang Digunakan**

Jenis	Value
Dataframe Kolom 1	Umur
Dataframe Kolom 2	Premi
Scaler	Min Max (0 – 1)
Pembersihan Outlier	Ya

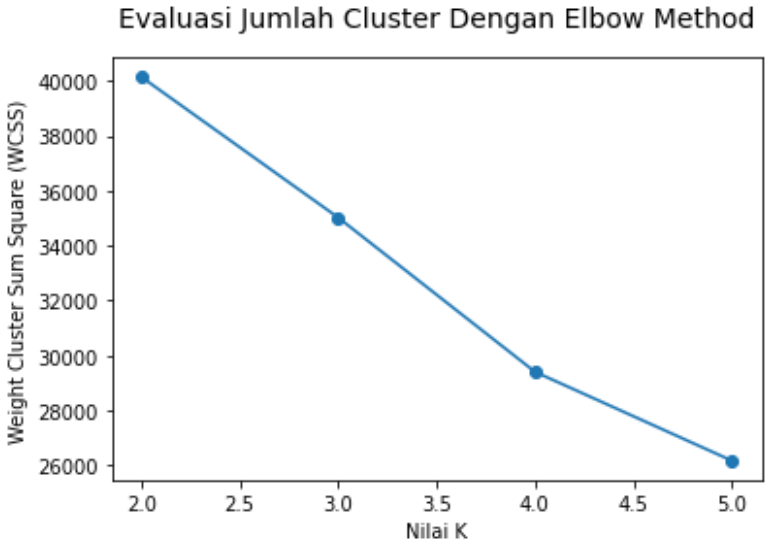
- **Hasil Plot Pengklasterisasian Model Berdasarkan Jumlah Cluster**



- **Silhoutte Score Setiap Cluster**

Cluster (K)	Silhoutte Score	Silhoutte Score (Round)
2	0.46104959338643886	0.46
3	0.401704558910869	0.40
4	0.4086359830208996	0.41
5	0.40947044376325725	0.41

- **Hasil Plot Elbow Method Dengan WCSS**

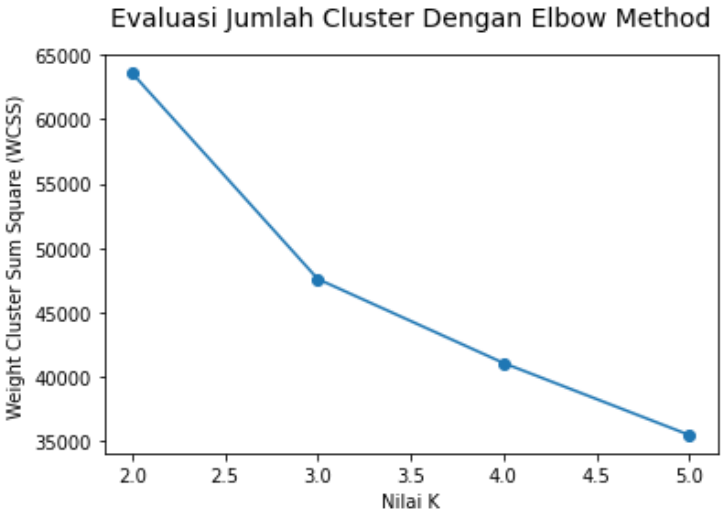


5.10 Kolom Umur – Lama Berlangganan

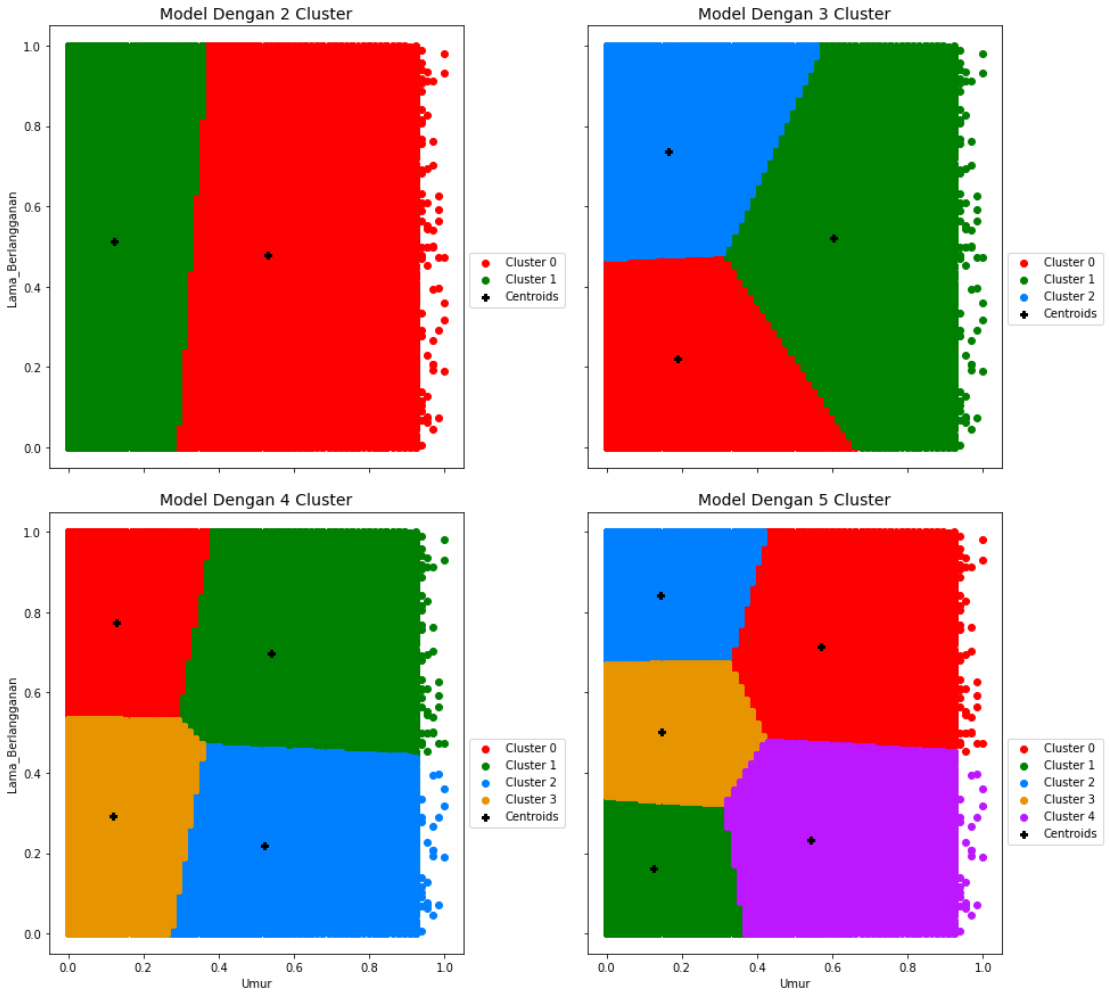
- **Parameter-Parameter Yang Digunakan**

Jenis	Value
Dataframe Kolom 1	Umur
Dataframe Kolom 2	Lama Berlangganan
Scaler	Min Max Scaler (0 – 1)
Pembersihan Outlier	Ya

- **Hasil Plot Elbow Method Dengan WCSS**



- **Hasil Plot Pengklasterisasian Model Berdasarkan Jumlah Cluster**



- **Silhoutte Score Setiap Cluster**

Cluster (K)	Silhoutte Score	Silhoutte Score (Round)
2	0.3156309729027889	0.32
3	0.40110643705035165	0.40
4	0.3845919374388585	0.38
5	0.38954191014110917	0.39

6. Kesimpulan

Dalam projek tugas besar tahap pertama ini digunakan metode K-Means yang berfungsi untuk melakukan proses klasterisasi terhadap dataset yang telah diberikan. Adapun berdasarkan hasil analisis dan eksperimen pada projek ini didapatkan hasil:

1. Metode K-Means Cukup Efektif Dalam Melakukan Proses Klasterisasi. Hal ini ditandai dengan hasil penggunaan metriks Elbow Method Dan Silhoutte Score yang baik pada keseluruhan proses klasterisasi.

2. Model Terbaik Dalam Proses Eksperimen Dan Analisis Terdapat Pada Kolom Umur – Kanal Penjualan Dengan Nilai K=3 dan Silhoutte Score Sebesar 0.65

3. Parameter terbaik berdasarkan silhoutte score untuk proses keseluruhan kluster didapatkan dengan cara penghapusan outlier, penggunaan min-max scaler, dan tanpa menggunakan PCA.

7. Saran

Ada baiknya dataset yang diberikan tidak dalam jumlah yang cukup banyak terutama dalam proses clustering. Hal ini menyebabkan waktu yang sangat lama untuk melakukan eksplorasi serta eksperimen setiap proses yang dilakukan karena membutuhkan proses komputasi yang sangat besar.

8. Daftar Pustaka

[1] "How to Drop Duplicates in Pandas," Medium, 25 November 2020. [Online]. Available: https://towardsdatascience.com/how-to-drop-duplicates-in-pandas-b1d4a5f4d2c6. [Accessed 10 11 2021].

[2] "Measures of Central Tendency," Laerd Statistics, 2018. [Online]. Available: https://statistics.laerd.com/statistical-guides/measures-central-tendency-mean-mode-median.php.

[3] I. Ghozali, "Aplikasi Analisis Multivariate Dengan Program IBM SPSS 25," *Aplikasi Analisis Multivariate Dengan Program IBM SPSS 25,* vol. 9, 2018.

[4] D. T. C. Sirait, Adiwijaya and W. Astuti, *Analisis Perbandingan Reduksi Dimensi Principal Component Analysis (PCA) dan Partial Least Square (PLS) untuk Deteksi Kanker menggunakan Data Microarray,* p. 2, 2019.

[5] G. Developers, "K-Means Advantages and Disadvantages," Google Developers, 13 January 2021. [Online]. Available: https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages. [Accessed 10 November 2021].

[6] A. Singh, A. Yadav and A. Rana, "K-means with Three different Distance Metrics," vol. 67, no. 10, p. 14, 2013.

[7] I. Dabbura, "K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks," Towards Data Science, 18 September 2018. [Online]. Available: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a.

[8] K. SARI, "ANALISIS EVALUASI PERHITUNGAN JARAK TERHADAP NILAI SILHOUETTE COEFFICIENT PADA ALGORITMA K-MEANS," p. 15, 2020.

[9] B. Saji, "In-depth Intuition of K-Means Clustering Algorithm in Machine Learning," Analytics Vidhya, January 20 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/01/in-depth-intuition-of-k-means-clustering-algorithm-in-machine-learning/.

LAMPIRAN

- **Video Presentasi:**
  -

- **Github Proses Pengerjaan:**
  https://github.com/ShinyQ/Tugas-Besar_Pembelajaran-Mesin_KMeans-Clustering

- **List Dataset Hasil Pra-Pemrosesan Setiap Langkah:**

  1. Penghilangan Data Duplikasi
     https://drive.google.com/file/d/1CkTiQNPrUH5InQR6eukv7hLgu_71t0uV/view?usp=sharing

  2. Label Encoding
     https://drive.google.com/file/d/1ofZ8UrijyDzoGeSgmwghqL_Hzs7yMkiH/view?usp=sharing

  3. Drop Outlier Data
     https://drive.google.com/file/d/1TShzhfAQmG_UFGLMoLycAHbVN1szRLYg/view?usp=sharing

  4. Data Scaler Min-Max Tanpa Outlier
     https://drive.google.com/file/d/1jtUqM8xq98gIcZt6Rgo_vIzQzZxCvxHk/view?usp=sharing

  5. Data Scaler Min-Max Dengan Outlier
     https://drive.google.com/file/d/1M-FBGiUypbnK7fFjAbW8s7BEsUkSbHTn/view?usp=sharing

  6. Data Scaler Min-Max PCA Tanpa Outlier
     https://drive.google.com/file/d/1lZ8bXfQH0j-mxrQ6ZaXfOYEMaiR-y2Os/view?usp=sharing

  7. Data Scaler Min-Max PCA Dengan Outlier
     https://drive.google.com/file/d/1P2IUmee8poSkXxZlE9n3XBNZydTYgVdK/view?usp=sharing

  8. Data Scaler Standard PCA Dengan Outlier
     https://drive.google.com/file/d/1_jbcqjuWgZ9KZjpJJ3Cn4iMC_6drrvJE/view?usp=sharing

9. Data Scaler Standard PCA Tanpa Outlier
   https://drive.google.com/file/d/1P2IUmee8poSkXxZlE9n3XBNZydTYgVdK/view?usp=sharing

10. Data Scaler Standard (Z-Score) Tanpa Outlier
    https://drive.google.com/file/d/1pSt7baSqNoq6D7MjLYpGVfbcCIFIJbKJ/view?usp=sharing

11. Data Scaler Standard (Z-Score) Dengan Outlier
    https://drive.google.com/file/d/1F7bKovkmGmXxqdQy5mOpdubnQPvmYQ7f/view?usp=sharing