LAPORAN TUGAS BESAR 1 MACHINE LEARNING (CLUSTERING)

Dibuat Oleh

- Artisa Bunga Syahputri (1301194007)

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG

2021

Daftar Isi

A. Formulasi Masalah

Masalah yang ingin dibahas pada Tugas besar tahap 1 pada mata kuliah pembelajaran mesin ini adalah untuk melakukan prediksi apakah pelanggan untuk membeli kendaraan atau tidak berdasarkan dataset yang disediakan dan membentuk beberapa cluster dari beberapa feature yang mungkin akan mempengaruhi pembeli untuk membeli kendaraan itu atau tidak.

B. Eksplorasi dan Pre-Processing Data

1. Data Eksplorasi

Pada saat eksplorasi data, saya melakukan pengecekan terhadap data train terlebih dahulu seperti melakukan cek ukuran dimensi ini untuk melihat ukuran dimensi dari data yang akan kita training, informasi mengenai data train untuk melihat tipe dari setiap feature pada data set , deskripsi statistik data train untuk melihat nilai minimum maksimum dan nilai-nilai quartile dari data , dan pendistribusian data train menggunakan boxplot hal ini untuk melihat apakah dalam setiap feature terdapat outlier atau tidak .

Read Data Pada Exel

```
df_train = pd.read_excel('kendaraan_train.xlsx')
#df_test = pd.read_excel('kendaraan_test.xlsx')
```

```
[ ] #tampilkan 5 data train teratas
    print(df_train.head())
```

```
   id Jenis_Kelamin  Umur  ...  Kanal_Penjualan  Lama_Berlangganan  Tertarik
0   1        Wanita  30.0  ...            152.0               97.0         0
1   2          Pria  48.0  ...             29.0              158.0         0
2   3           NaN  21.0  ...            160.0              119.0         0
3   4        Wanita  58.0  ...            124.0               63.0         0
4   5          Pria  50.0  ...             88.0              194.0         0

[5 rows x 12 columns]
```

Data Train Exploration

```
#tampilkan 15 data train teratas

df_train.head(10)
```

	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	< 1 Tahun	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
5	6	Pria	21.0	1.0	35.0	1.0	< 1 Tahun	Tidak	22735.0	152.0	171.0	0
6	7	Wanita	33.0	1.0	8.0	0.0	NaN	Pernah	32435.0	124.0	215.0	1
7	8	Pria	23.0	NaN	28.0	1.0	< 1 Tahun	Tidak	26869.0	152.0	222.0	0
8	9	Wanita	20.0	1.0	8.0	1.0	< 1 Tahun	Tidak	30786.0	160.0	31.0	0
9	10	NaN	54.0	1.0	29.0	0.0	> 2 Tahun	Pernah	88883.0	124.0	28.0	1

```
#melihat info data train
df_train.info()
```

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

```
#ukuran data train
df_train.shape
```

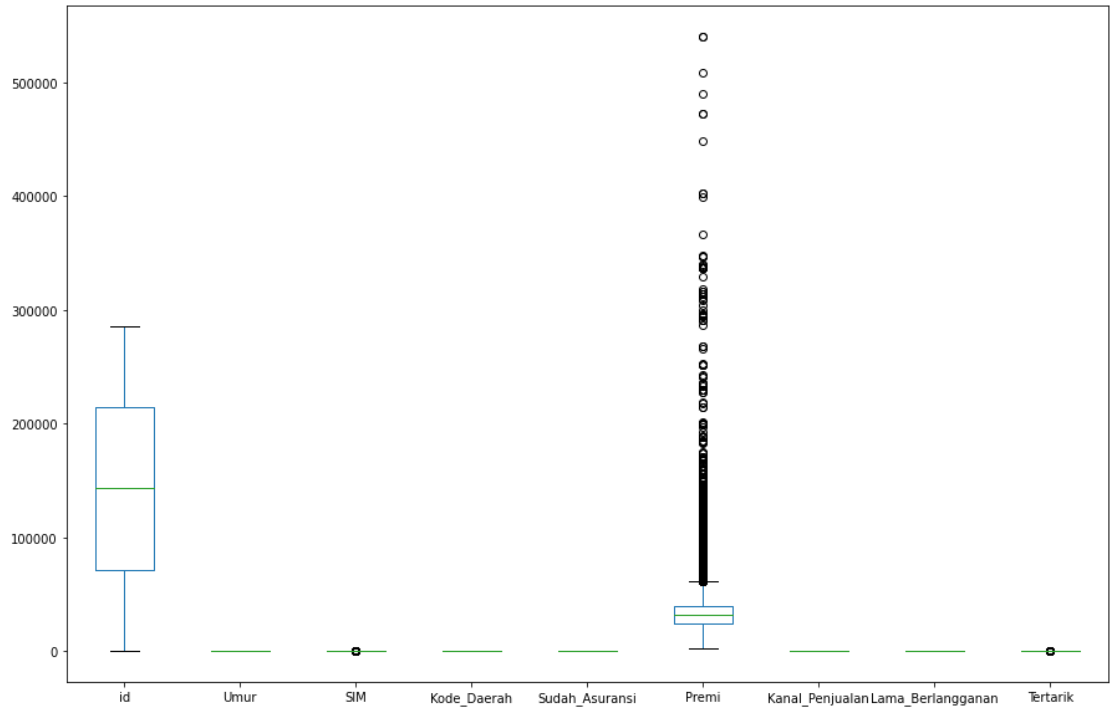
```
(285831, 12)
```

```
[ ] #lihat deskripsi data train
df_train.describe()
```

	id	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
count	285831.000000	271617.000000	271427.000000	271525.000000	271602.000000	271262.000000	271532.000000	271839.000000	285831.000000
mean	142916.000000	38.844336	0.997848	26.405410	0.458778	30536.683472	112.021567	154.286302	0.122471
std	82512.446734	15.522487	0.046335	13.252714	0.498299	17155.000770	54.202457	83.694910	0.327830
min	1.000000	20.000000	0.000000	0.000000	0.000000	2630.000000	1.000000	10.000000	0.000000
25%	71458.500000	25.000000	1.000000	15.000000	0.000000	24398.000000	29.000000	82.000000	0.000000
50%	142916.000000	36.000000	1.000000	28.000000	0.000000	31646.000000	132.000000	154.000000	0.000000
75%	214373.500000	49.000000	1.000000	35.000000	1.000000	39377.750000	152.000000	227.000000	0.000000
max	285831.000000	85.000000	1.000000	52.000000	1.000000	540165.000000	163.000000	299.000000	1.000000

```
#melihat boxplot dari data train
df_train.plot(kind='box', figsize=(15, 10) )
```

```
/usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: Vi
  return array(a, dtype, copy=False, order=order)
<matplotlib.axes._subplots.AxesSubplot at 0x7f6f48a01290>
```



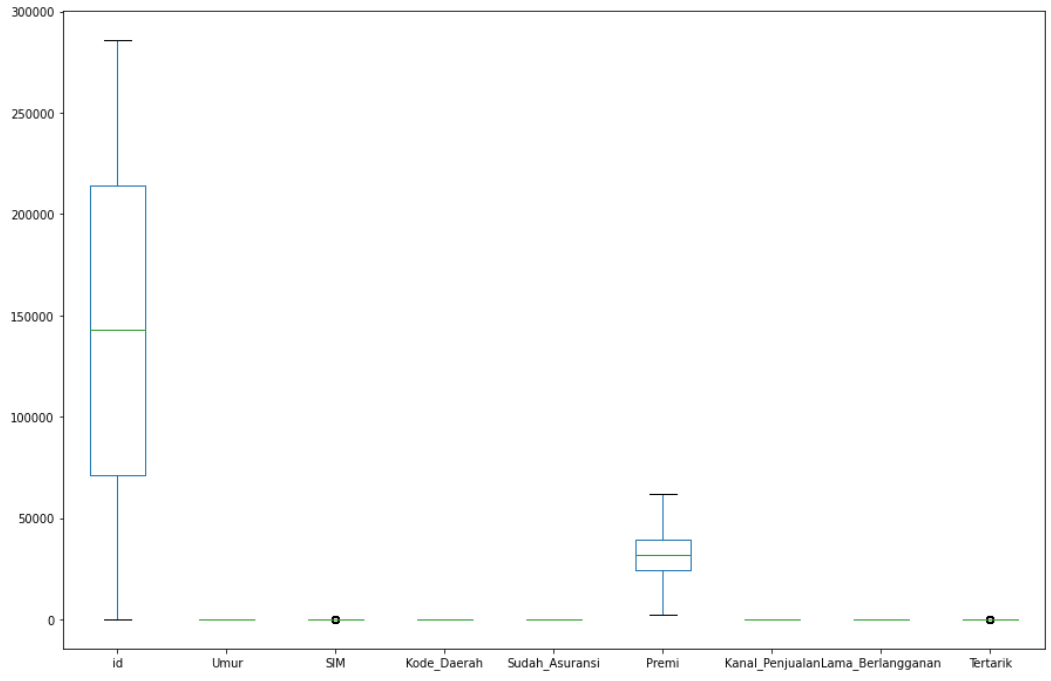
2. Data Pre-Processing

Hal yang saya lakukan dalam data preprocessing pertama kali adalah melakukan handling terhadap outlier, feature yang memiliki banyak outlier adalah feature premi, sehingga saya menghilangkan outlier agar data yang diproses overfitting atau underfitting . Teknik yang saya gunakan untuk menghandle outlier ini adalah dengan menggunakan metode interquartile, yaitu dengan menghilangkan data yang berada diluar Q1-Q3

DATA PRE-PROCESSING

```
#menghilangkan outlier pada data premi
#tentukan outlier
Q1 = df_train['Premi'].quantile(0.25)
Q3 = df_train['Premi'].quantile(0.75)
IQR = Q3 - Q1

Max = Q3 + (1.5 * IQR)
Min = Q1 - (1.5 * IQR)

more_than = df_train['Premi'] > Max
lower_than = df_train['Premi'] < Min
```

```
#hilangkan outlier
df_train['Premi'] = df_train['Premi'].mask(more_than, Max)
df_train['Premi'] = df_train['Premi'].mask(lower_than, Min)
df_train.plot(kind='box', figsize=(15, 10) )
```



Selanjutnya yang saya lakukan adalah mereplace data yang berbentuk kategorikal seperti Jenis_Kelamin, Umur_Kendaraan, Kendaraan_Rusak dengan angka antara 0,1,2 agar fiture dapat dieksekusi oleh model

```
#replace Umur_Kendaraan, Jenis_Kelamin, dan Kendaraan_Rusak to integer

df_train.replace(['< 1 Tahun', '1-2 Tahun', '> 2 Tahun'], [0,1,2], inplace=True)

df_train.replace(['Wanita', 'Pria'], [0,1], inplace= True)
df_train.replace(['Tidak', 'Pernah'], [0,1], inplace=True)

df_train.head(10)
```

	id	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	1	0.0	30.0	1.0	33.0	1.0	0.0	0.0	28029.000	152.0	97.0	0
1	2	1.0	48.0	1.0	39.0	0.0	2.0	1.0	25800.000	29.0	158.0	0
2	3	NaN	21.0	1.0	46.0	1.0	0.0	0.0	32733.000	160.0	119.0	0
3	4	0.0	58.0	1.0	48.0	0.0	1.0	0.0	2630.000	124.0	63.0	0
4	5	1.0	50.0	1.0	35.0	0.0	2.0	NaN	34857.000	88.0	194.0	0
5	6	1.0	21.0	1.0	35.0	1.0	0.0	0.0	22735.000	152.0	171.0	0
6	7	0.0	33.0	1.0	8.0	0.0	NaN	1.0	32435.000	124.0	215.0	1
7	8	1.0	23.0	NaN	28.0	1.0	0.0	0.0	26869.000	152.0	222.0	0
8	9	0.0	20.0	1.0	8.0	1.0	0.0	0.0	30786.000	160.0	31.0	0
9	10	NaN	54.0	1.0	29.0	0.0	2.0	1.0	61847.375	124.0	28.0	1

Selanjutnya yang saya lakukan adalah melakukan pengecekan terhadap missing value agar kita dapat mendapatkan hasil yang akurat. Teknik yang saya gunakan adalah dengan men drop seluruh missing value yang ada pada setiap feature

```
#cek missing value
print(df_train.isnull().sum())
```

```
id                   0
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
Tertarik             0
dtype: int64
```

```
#drop missing value pada datatrain
X_train = df_train.dropna()
```

```
print(X_train.isna().sum())
```

```
id                   0
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
```

Proses selanjutnya yang saya lakukan adalah men drop feature yang saya rasa tidak akan berpengaruh besar dalam proses learning pada model clustering yang dibangun. future yang saya drop adalah future id dan Tertarik

```
#drop kolom ID
X_train.drop(['id', 'Tertarik'], inplace=True, axis= 1)

/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4174: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
  errors=errors,

X_train.head()
```

	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
3	0.0	58.0	1.0	48.0	0.0	1.0	0.0	2630.0	124.0	63.0
5	1.0	21.0	1.0	35.0	1.0	0.0	0.0	22735.0	152.0	171.0
8	0.0	20.0	1.0	8.0	1.0	0.0	0.0	30786.0	160.0	31.0

dari preprocessing yang sudah dilakukan maka didapat hasil dataset yang sudah bersih yaitu data yang sudah tidak memiliki missing value, dan data kategorikal sudah di replace sehingga model lebih mudah untuk mengeksekusi dataset

3. Menghitung Korelasi Antar Data

Proses yang saya lakukan selanjutnya setelah preprocessing adalah melakukan drop terlebih dahulu terhadap data yang menurut saya bersifat kategorikal yang hanya terdiri dari 2 atau 3 kategori saja, yaitu Jenis_Kelamin, SIM, Sudah_Asuransi, dan Umur_Kendaraan
hal ini dilakukan karena saya merasa tidak terlalu berpengaruh terhadap model clustering ini. sehingga tersisa data yang akan di eksekusi sebagai berikut

```
X_train
```

	Umur	Kode_Daerah	Umur_Kendaraan	Premi	Kanal_Penjualan	Lama_Berlangganan
0	30.0	33.0	0.0	28029.0	152.0	97.0
1	48.0	39.0	2.0	25800.0	29.0	158.0
3	58.0	48.0	1.0	2630.0	124.0	63.0
5	21.0	35.0	0.0	22735.0	152.0	171.0
8	20.0	8.0	0.0	30786.0	160.0	31.0
...	...	...	...	...	...	...
285826	23.0	4.0	0.0	25988.0	152.0	217.0
285827	21.0	46.0	0.0	44686.0	152.0	50.0
285828	23.0	50.0	0.0	49751.0	152.0	226.0
285829	68.0	7.0	1.0	30503.0	124.0	270.0
285830	45.0	28.0	1.0	36480.0	26.0	44.0

171068 rows × 6 columns

Selanjutnya yang saya lakukan adalah dengan menghitung korelasi antar data yang tersisa dan didapatkan hasil korelasi seperti heatmap berikut:


```
#lihat korelasi antar data

correlation_X = X_train.corr()

plt.figure(figsize=(15,10))

plt.title('Correlation', y=1, size=25)
#mask = np.triu(np.ones_like(correlation_X, dtype=bool))

sns.heatmap(correlation_X, annot= True, linewidths= .5, square=True)

plt.show()
```

dari hitmap tersebut didapatkan data dengan korelasi tertinggi dimiliki oleh umur dan Umur_Kendaraan dengan nilai korelasi antar kedua future tersebut adalah 0.77 dan data dengan korelasi yang lebih rendah adalah antara Kanal_Penjualan dan Lama Berlangganan dengan nilai korelasi 0.0008

4. Feature Selection

Pada tahap ini saya akan memilih feature yang akan digunakan untuk memodelkan model clustering ini, dan saya memilih untuk tidak menggunakan feature yang memiliki korelasi tinggi mendekati 1 dan saya menggunakan data yang memiliki korelasi lebih rendah yaitu feature Premi dan Lama_Berlangganan , selain itu saya merasa bahwa premi dan lama berlangganan termasuk salah satu faktor yang mempengaruhi seseorang mau membeli mobil atau tidak.

```
#mengambil data dengan korelasi terendah
data_train = X_train[['Premi' , 'Lama_Berlangganan']]
```

```
[ ] data_train
```

	Premi	Lama_Berlangganan
0	28029.0	97.0
1	25800.0	158.0
3	2630.0	63.0
5	22735.0	171.0
8	30786.0	31.0
...	...	...
285826	25988.0	217.0
285827	44686.0	50.0
285828	49751.0	226.0
285829	30503.0	270.0
285830	36480.0	44.0

171068 rows × 2 columns

5. Scalling

Feature yang sudah saya pilih tadi kemudian saya lakukan scalling untuk menghindari nilai feature yang berbeda jauh antara 1 feature dengan feature yang lain. saya melakukan scaling dengan menggunakan metode Min Max Scaller untuk mengkonversi nilai setiap feature menjadi berada pada rentang 0-1

+ Code + Text

```
#melakukan sceling terhadap data train yang akan di gunakan
scalling = MinMaxScaler()

data_train = pd.DataFrame(scalling.fit_transform(data_train))

data_train.columns = ['Premi', 'Lama_Berlangganan']
data_train
```

	Premi	Lama_Berlangganan
0	0.428911	0.301038
1	0.391270	0.512111
2	0.000000	0.183391
3	0.339512	0.557093
4	0.475469	0.072664
...	...	...
171063	0.394445	0.716263
171064	0.710197	0.138408
171065	0.795729	0.747405
171066	0.470690	0.899654
171067	0.571623	0.117647

```
data_train.describe()
```

	Premi	Lama_Berlangganan
count	171068.000000	171068.000000
mean	0.463974	0.499146
std	0.261228	0.290182
min	0.000000	0.000000
25%	0.367118	0.245675
50%	0.489265	0.498270
75%	0.619637	0.750865
max	1.000000	1.000000

C. Pemodelan

Model yang saya buat menggunakan metode K Means Clustering yang bersifat centroid based. Konsep dasar dari metode ini adalah melakukan inisialisasi jumlah K sebagai centroid awal. kemudian dilakukan perhitungan dan membandingkan jarak dari data ke centroid terdekat, kemudian centroid akan terupdate sesuai dengan mean dari jarak antar data , iterasi ini akan terus dilakukan sampai centroid tidak melakukan update/perubahan lagi. Dalam membangun model ini saya membuat beberapa fungsi yaitu:

- fungsi Euclidean Distance : fungsi ini berguna untuk menghitung jarak data ke centroid

```
#fungsi menghitung jarak data ke centroid menggunakan eucledian distance
def euclidianDistance(instance1,instance2):
  distance = 0
  x=0
  for x in range(len(instance2)):
    distance += pow((instance1[x]-instance2[x]), 2 )
  return math.sqrt(distance)
```

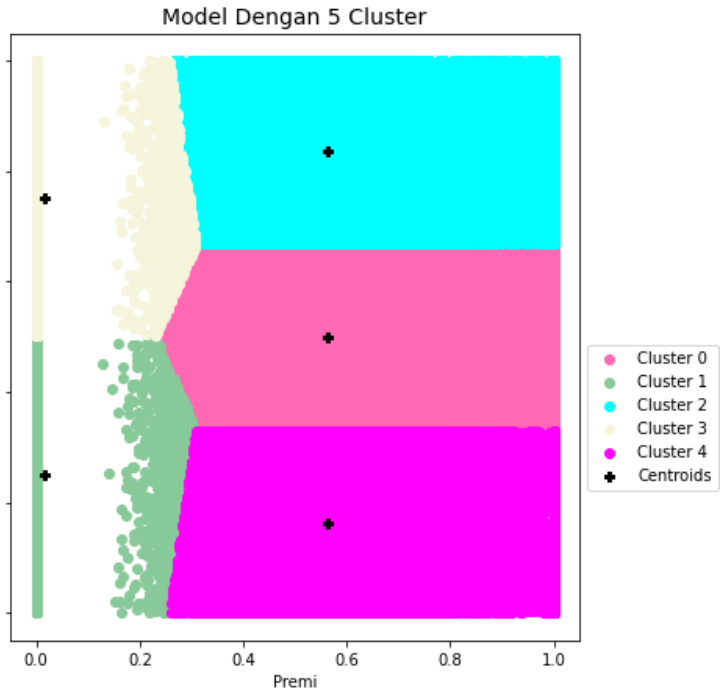
- fungsi convergent : fungsi yang berguna untuk mengecek apakan centrois saat ini sudah sama dengan centroid sebelumnya, ini berguna untuk mengecek apakah centroid sudah stabil dan tidak berubah lagi atau belum

```
# fungsi untuk menghitung apakan centroid sudah tetap atau belum
def convergent(temp_centroids, centroids):
  if( np.array(centroids).all != np.array(temp_centroids).all):
    return False

  return True
```

- lalu saya membuat fungsi Kmeans : fungsi yang akan mentraining data train dan mengeluarkan list centroid, cluster dan jumlah dari sum square error dari data train yang kita training

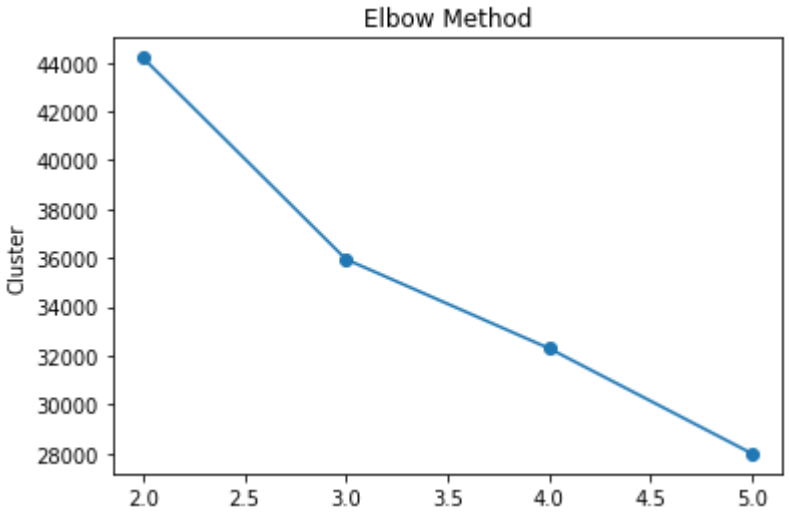
```python
def kMeans(K , max_iterasi,data):
  sum_square_error = 0

  # inisialisasi centroid
  centroids = {i: data[random.randint(0, len(data))] for i in range(K)}
  # copy centroid sekarang untuk dibandingkan dengan centroid setelahnya
  temp_centroids = centroids.copy()

  for _ in range(max_iterasi):
    #mendefinisikan cluster
    cluster = {i : [] for i in range(K)}
    sse = []

    #mencari nilai terdekat data dengan euclidiant distance dan masukan ke cluster
    for x in data:
      jarak = [euclidianDistance(x, centroids[c]) for c in centroids]
      sse.append(np.min(jarak))

      cluster[jarak.index(min(jarak))].append(x)
    #merecompute centroid ulang dari setiap cluster dengan mean nya
    for i in cluster:
      centroids[i] = np.mean(cluster[i], axis= 0)
    #mengecek apakah centroid suda h stabil atau belum
    if convergent(temp_centroids, centroids) : break
    #copy centroid sekarang untuk dibandingkan dengan centroid berikutnya
    temp_centroids = centroids.copy()

  return centroids, cluster, sum(sse)
```

Hasil dari model K Means akan divisualisasikan menggunakan scatter plot agar persebaran data dan letak centroid dapat terlihat. Dalam percobaan kali ini saya menggunakan k = 5 dengan iterasi maksimal sebesar 100x.

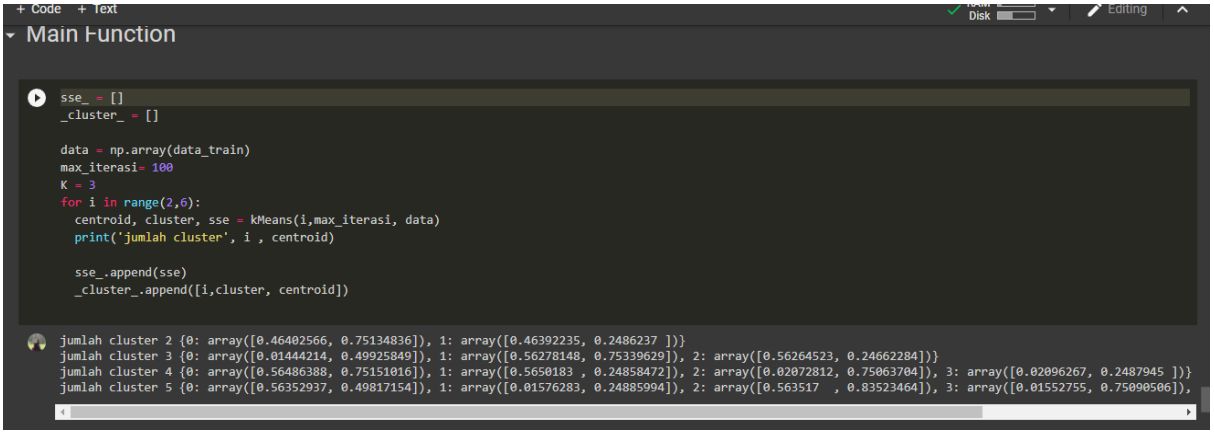


D. Evaluasi

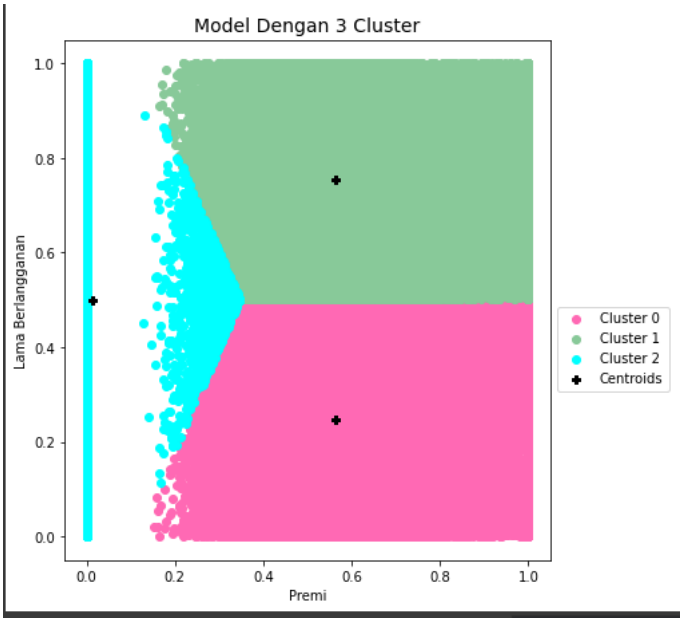
Evaluasi yang saya lakukan menggunakan metode Elbow Method. Elbow Method digunakan untuk menentukan jumlah K yang optimal untuk model clustering, caranya adalah dengan melihat penurunan dari nilai cluster sum of square SEE antar tiap cluster dan centroidnya . Pada model K Means yang saya buat dapat ditunjukan grafik elbow Method yang dihasilkan adalah seperti pada gambar dibawah:



dengan hasil training setiap cluster adalah seperti gambar berikut:



pada grafik tersebut nilai yang stabil adalah saat nilai K = 3.
berikut adalah visualisasi ketika menggunakan K=3 dengan jumlah iterasi maksimal sebanyak 100 kali



saya juga melakukan pengecekan hasil clustering untuk feature yang saya pilih tersebut menggunakan Silhouette Score sehingga dapat hasil silhouette score dari setiap cluster adalah seperti gambar berikut

Pengecekan Menggunakan Silhouette Score

```
[ ] from sklearn.metrics import silhouette_score

    for i, cluster in enumerate(clusterize):
        score = silhouette_score(cluster[['Premi', 'Lama_Berlangganan']], cluster['Cluster'])
        print(f'Cluster-{i+3}, Silhouette Score = {score}')
```

```
Cluster-3, Silhouette Score = 0.39131843815495393
Cluster-4, Silhouette Score = 0.45339244990336675
Cluster-5, Silhouette Score = 0.4398033619676446
Cluster-6, Silhouette Score = 0.39535527959564326
```

E. Eksperimen

Eksperimen yang saya lakukan adalah dengan melakukan clustering pada data train dan sebelum melakukan proses clustering, data akan direduksi terlebih dahulu dengan menggunakan PCA. PCA (Principal Component Analysis) adalah teknik reduksi dimensi linear yang dapat digunakan untuk mengekstraksi informasi dari ruang berdimensi tinggi dengan memproyeksikannya ke subruang berdimensi lebih rendah. Ini dilakukan untuk menjaga bagian-bagian penting yang memiliki lebih banyak variasi data dan menghapus bagian bagian yang tidak penting dengan variasi yang lebih sedikit.

Proses clustering dengan menggunakan metode PCA dimulai dengan preprocessing yang sama. lalu data yang sudah dibersihkan kemudian akan scaling, disini saya menggunakan standard scaler untuk melakukan scaling terhadap data set yang sudah di cleaning.

```
df = pd.DataFrame(StandardScaler().fit_transform(X_train))
df
```

	0	1	2	3	4	5	6	7	8	9
0	-1.084918	-0.567753	0.045537	0.496795	1.083923	-1.071127	-1.008925	-0.134222	0.734473	-0.682703
1	0.921729	0.591171	0.045537	0.949204	-0.922575	2.452328	0.991154	-0.278315	-1.538819	0.044680
2	-1.084918	1.235018	0.045537	1.627818	-0.922575	0.690601	-1.008925	-1.776129	0.216975	-1.088129
3	0.921729	-1.147215	0.045537	0.647598	1.083923	-1.071127	-1.008925	-0.476450	0.734473	0.199696
4	-1.084918	-1.211599	0.045537	-1.388242	1.083923	-1.071127	-1.008925	0.044003	0.882329	-1.469707
...	...	...	...	...	...	...	...	...	...	...
171063	-1.084918	-1.018445	0.045537	-1.689848	1.083923	-1.071127	-1.008925	-0.266162	0.734473	0.748214
171064	-1.084918	-1.147215	0.045537	1.477015	1.083923	-1.071127	-1.008925	0.942562	0.734473	-1.243145
171065	-1.084918	-1.018445	0.045537	1.778621	1.083923	-1.071127	-1.008925	1.269986	0.734473	0.855532
171066	0.921729	1.878864	0.045537	-1.463643	1.083923	0.690601	-1.008925	0.025708	0.216975	1.380202
171067	0.921729	0.398017	0.045537	0.119788	-0.922575	0.690601	0.991154	0.412089	-1.594265	-1.314691

171068 rows × 10 columns

Selanjutnya saya menggunakan PCA untuk melakukan reduksi dimensi linear dengan menggunakan PCA yang sudah tersedia di google collab

Feature Selection Dengan PCA

```
pca_train = PCA(n_components=2)
principalComponents_train = pca_train.fit_transform(df)
```

```
[23] data = pd.DataFrame(data = principalComponents_train
                , columns = ['principal component 1', 'principal component 2'])
```

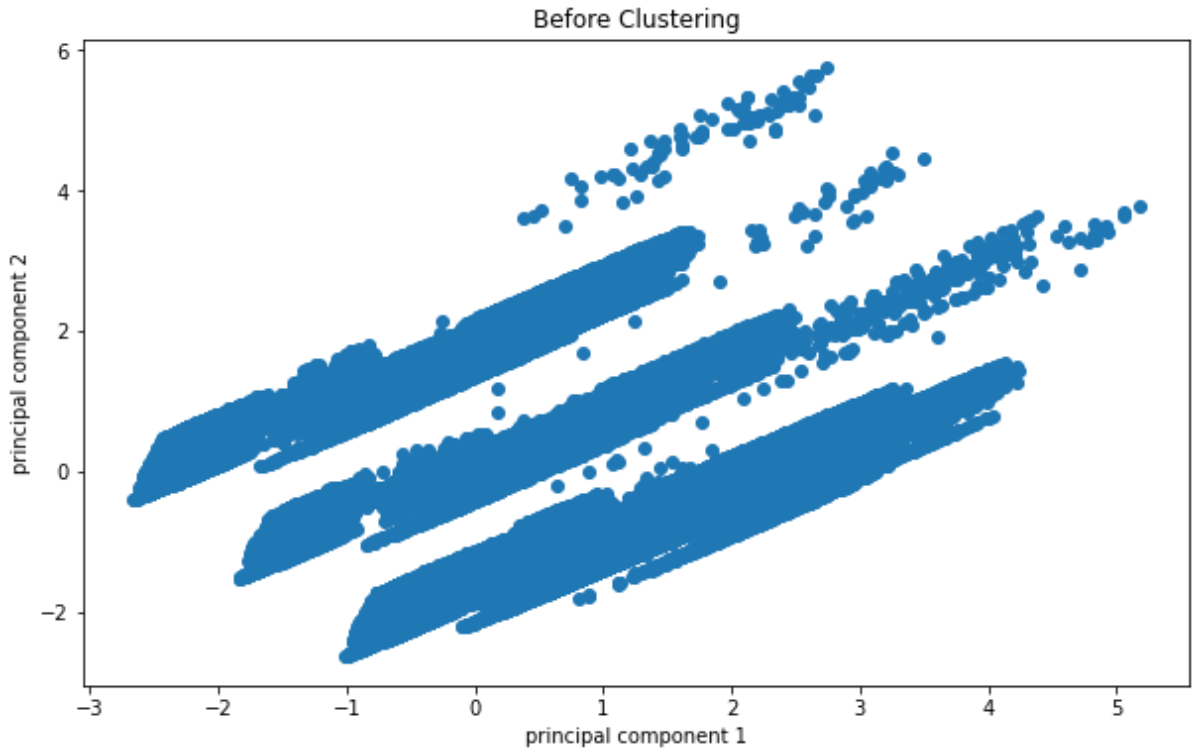
+ Code + Text

```
[24] data.head()
```

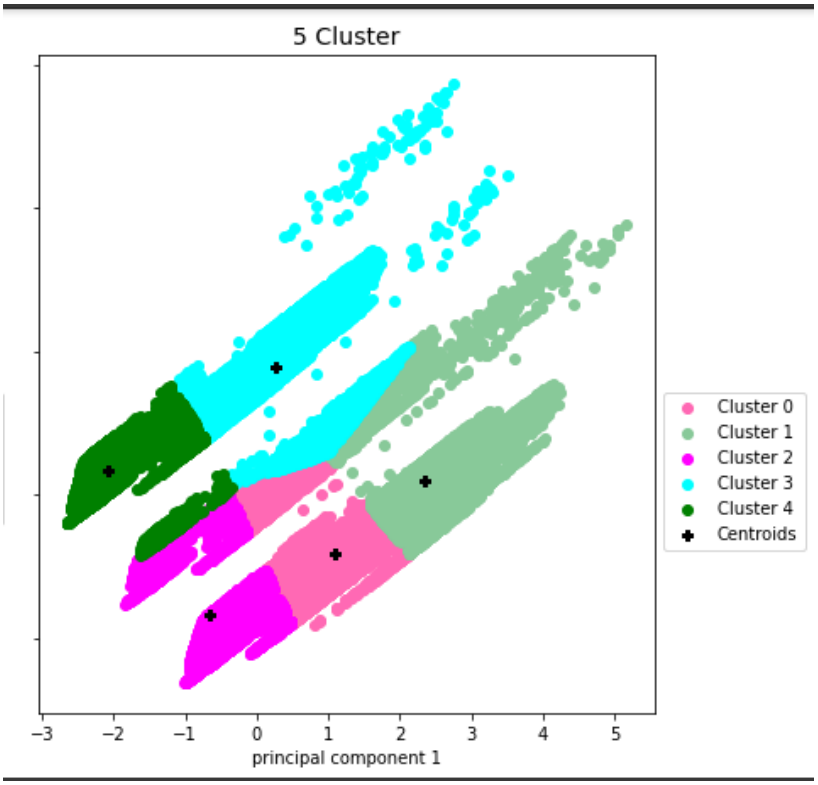
	principal component 1	principal component 2
0	-2.103030	0.328918
1	3.075764	0.300825
2	0.615086	0.099066
3	-2.096901	0.257983
4	-2.534807	0.014708

```
[25] # Mengecek Total Data Pada Dataframe
print("Total Data :", len(data))

Total Data : 171068
```

sehingga ketika divisualisasikan menggunakan scatter plot sebaran datanya akan seperti gambar dibawah ini:



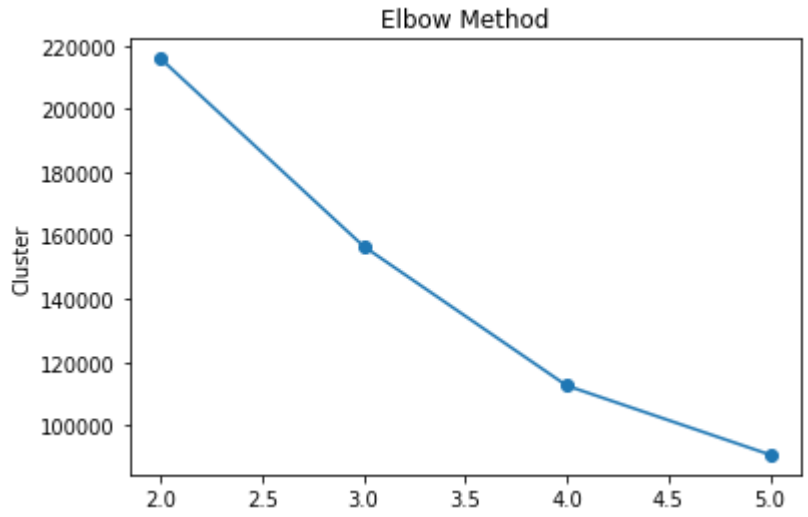
Selanjutnya saya melakukan training data dengan menggunakan model K Means dengan menghitung jarak data dengan centroid terdekat dengan euclidean distance seperti yang dilakukan sebelumnya , Hasil dari model K Means akan divisualisasikan menggunakan scatter plot agar persebaran data dan letak centroid dapat terlihat. Dalam percobaan kali ini saya menggunakan k = 5 dengan iterasi maksimal sebesar 100x.



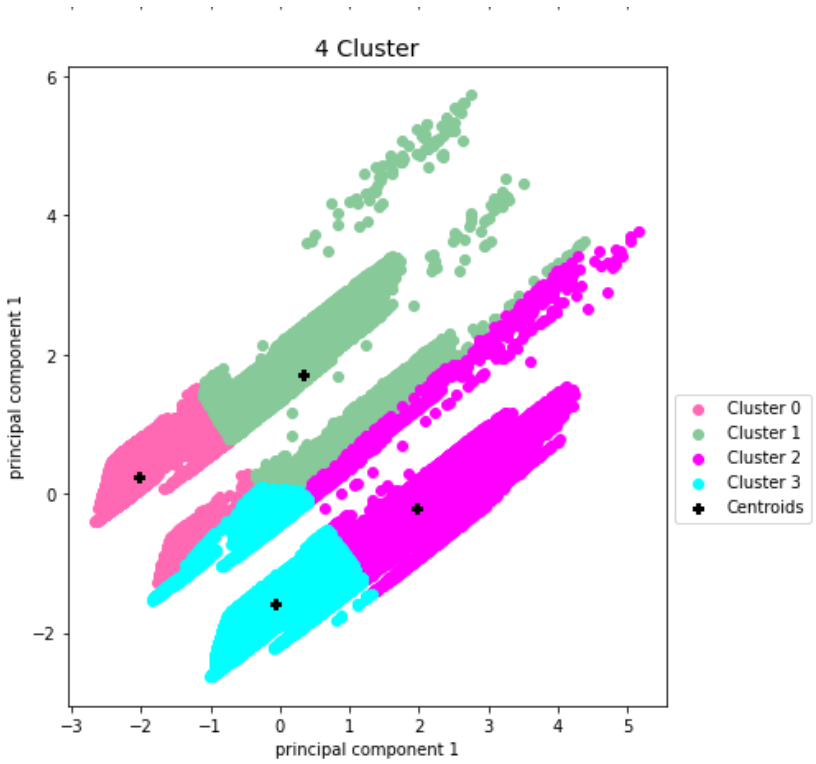
dari hasil clustering tersebut lalu saya melakukan evaluasi dengan menggunakan metode elbow method yang digunakan untuk menentukan jumlah K yang optimal untuk model clustering. Hasil running dari data trining dapat dilihat pada gambar berikut

```
[ ] sse_ = []
    _cluster_ = []

    data_train = np.array(data)
    max_iterasi= 100
    K = 3
    for i in range(2,6):
      centroid, cluster, sse = kMeans(i,max_iterasi, data_train)
      print('jumlah cluster', i , centroid)

      sse_.append(sse)
      _cluster_.append([i,cluster, centroid])

    jumlah cluster 2 {0: array([ 1.20399988, -0.62981834]), 1: array([-1.35029852,  0.70634789])}
    jumlah cluster 3 {0: array([0.38945483, 1.77505425]), 1: array([-1.99871592,  0.20993268]), 2: array([ 1.18565547, -0.7466975 ])}
    jumlah cluster 4 {0: array([-2.02448746,  0.24450674]), 1: array([0.33598201, 1.70489957]), 2: array([ 1.97717363, -0.19587871]), 3: array([-0.06411671, -1.587565
    jumlah cluster 5 {0: array([ 1.09899816, -0.83162072]), 1: array([2.35011721, 0.18510421]), 2: array([-0.655481  , -1.66925831]), 3: array([0.26930477, 1.7789577
```

Hasil Elbow method yang didapatkan dapat dilihat pada grafik dibawah



pada grafik tersebut nilai yang stabil adalah saat nilai K = 4.
berikut adalah visualisasi ketika menggunakan K=4 dengan jumlah iterasi maksimal sebanyak 100 kali



dan saya melakukan pengecekan dengan menggunakan Silhouette Score sehingga dapat hasil silhouette score dari setiap cluster adalah seperti gambar berikut

dari eksperimen tersebut lalu saya bandingkan dengan hasil pada training awal didapat hasil silhouette score untuk data yang pemilihan featurenya menggunakan PCA lebih besar silhouette score nya dari pada data yang dipilih manual, walaupun kedua hasil perhitungan silhouette score nya tidak mendekati 1.

F. Kesimpulan

Berdasarkan hasil percobaan yang susah saya lakukan. proses Clustering dan metode Clustering yang digunakan setiap orang dapat berbeda-beda. Pembuatan preprocessing yang dilakukan juga dapat berbeda-beda untuk setiap orang. seperti metode scaling dapat menggunakan metode MinMaxScaller, atau ada yang menggunakan standar Scaller, selain itu setiap orang dapat memilih feature yang berbeda yang akan digunakan untuk melakukan training terhadap model clustering nya seperti menggunakan metode PCA atau dilakuan dengan manual berdasarkan asumsi masing-masing. dan berdasarkan percobaan yang saya lakukan, pemilihan feature sangatlah penting dalam men generate model clustering yang kita bangun. Hal ini terlihat dari hasil silhouette score dari setiap cluster bahwa fitur yang berbeda dan jumlah dimensi yang berbeda akan memberikan keakuratan yang berbeda juga terhadap hasil analisis apakah pembeli akan membeli mobil atau tidak

Link Google Colab:
Clustering tanpa PCA:
https://colab.research.google.com/drive/1mmwMQEOKl7_-gpPXLKB_q-oZS_tdztVa?usp=sharing

Clustering dengan PCA:
https://colab.research.google.com/drive/1K3s-snv8_WJ9-tbB3Tptrgo82mriNZWZ?usp=sharing