LAPORAN TUGAS BESAR MK PEMBELAJARAN MESIN
TAHAP 1 : CLUSTERING

Oleh :

Hilman Bayu Aji – 1301180397

IFX-44-GAB

S1 INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

2021/2022

1. Formulasi Masalah

Pada Tugas Besar Tahap 1 ini, saya diberikan dataset kendaraan yang akan diselesaikan dengan *clustering* (*unsupervised Learning*), dalam kolom tersebut. Tugas *clustering* (*unsupervised Learning*) kali ini adalah mengelompokkan pelanggan berdasarkan data pelanggan di dealer tanpa memperhatikan label kelas apakah pelanggan tertarik untuk membeli kendaraan baru atau tidak. Kemudian, klasterisasi menggunakan fitur yang memiliki korelasi yang baik dan kurang baik, setelah itu akan dibandingkan. Sebelumnya ada 2 dataset yaitu kendaraan_test dan kendaraan_train yang akan kita gabungkan. Berikut data ketika digabungkan, terlihat ada sekitar 333470 baris dan 11 kolom.

```
df = pd.concat([df,df2])
df
```

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	Wanita	49.0	1.0	8.0	0.0	1-2 Tahun	Pernah	46963.0	26.0	145.0	0
1	Pria	22.0	1.0	47.0	1.0	< 1 Tahun	Tidak	39624.0	152.0	241.0	0
2	Pria	24.0	1.0	28.0	1.0	< 1 Tahun	Tidak	110479.0	152.0	62.0	0
3	Pria	46.0	1.0	8.0	1.0	1-2 Tahun	Tidak	36266.0	124.0	34.0	0
4	Pria	35.0	1.0	23.0	0.0	1-2 Tahun	Pernah	26963.0	152.0	229.0	0
...	...	...	...	...	...	...	...	...	...	...	...
285826	Wanita	23.0	1.0	4.0	1.0	< 1 Tahun	Tidak	25988.0	152.0	217.0	0
285827	Wanita	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	44686.0	152.0	50.0	0
285828	Wanita	23.0	1.0	50.0	1.0	< 1 Tahun	Tidak	49751.0	152.0	226.0	0
285829	Pria	68.0	1.0	7.0	1.0	1-2 Tahun	Tidak	30503.0	124.0	270.0	0
285830	Pria	45.0	1.0	28.0	0.0	1-2 Tahun	Pernah	36480.0	26.0	44.0	0

333470 rows × 11 columns

2. Eksplorasi dan Persiapan Data

```
df2 = pd.read_csv("Dataset/kendaraan_train.csv")
df2 = df2.drop(columns=["id"])
df2
```

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

285831 rows × 11 columns

Membaca data train yang telah diberikan, terdapat sekitar 285831 baris dan 11 kolom.

```
df.dtypes
```

```
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
```

Atribut-atribut yang digunakan pada dataset kendaraan.

```
# Cleaning data
# Mengecek nilai null
df.isnull().sum()
```

```
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
```

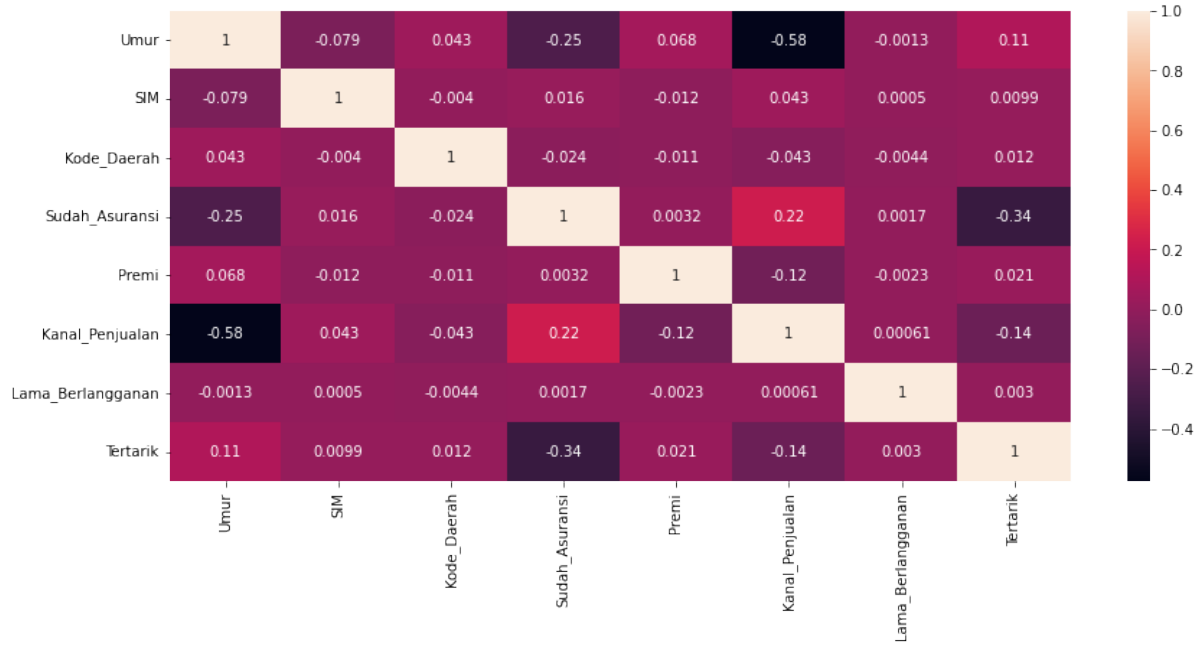
Setelah mendapatkan data, kita disini mengecek mana nilai data yang kosong/null. Ada beberapa data yang kosong dan terlihat hasilnya.

```
# Menghapus baris yang memiliki nilai null
df = df.dropna()
df.isnull().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
```

Setelah kita cek dan ada data yang masih null, maka kita hapus semua baris yang datanya masih null.

3. Pemodelan



Berdasarkan dari korelasi diatas, terlihat salah satu fitur yang memiliki korelasi yang tinggi adalah **umur dan premi**. Dan akan dilakukan percobaan antara 2 fitur yang memiliki korelasi yang kurang baik, salah satunya adalah **umur dan lama berlangganan**.

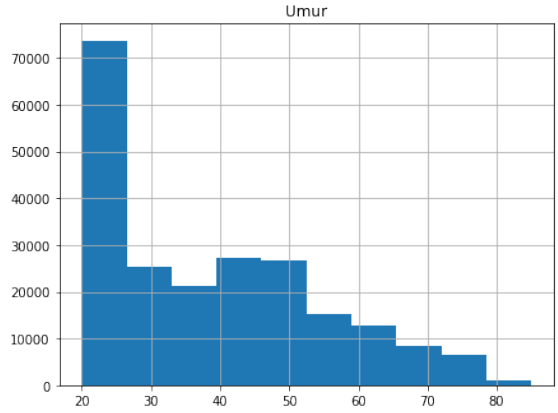
```
# Menentukan fitur
df = df.loc[:, ("Umur", "Premi", "Lama_Berlangganan")]
df
```

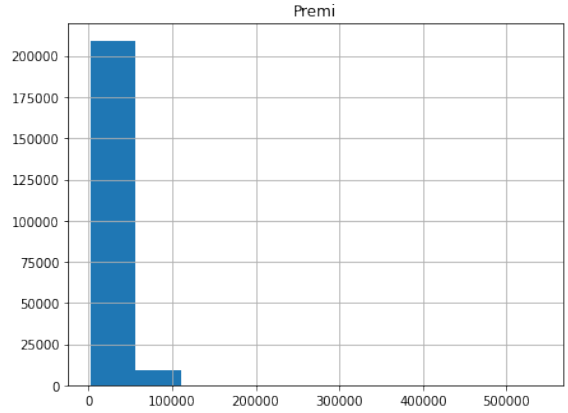
	Umur	Premi	Lama_Berlangganan
0	49.0	46963.0	145.0
1	22.0	39624.0	241.0
2	24.0	110479.0	62.0
3	46.0	36266.0	34.0
4	35.0	26963.0	229.0
...	...	...	...
285826	23.0	25988.0	217.0
285827	21.0	44686.0	50.0
285828	23.0	49751.0	226.0
285829	68.0	30503.0	270.0
285830	45.0	36480.0	44.0

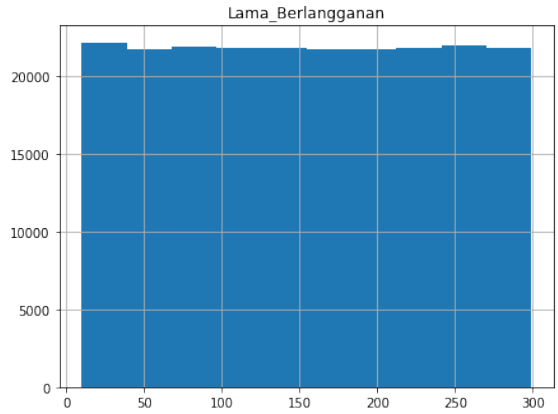
218707 rows × 3 columns

Fitur yang saya gunakan yaitu umur, premi dan lama berlangganan.

```
# Mengecek distribusi dari setiap fitur yang dipakai
df.hist(layout=(2,2), figsize=(16,12))
plt.show()
```








```
# Melakukan Normalisasi
# Data diperkecil agar lebih mudah dilihat
scaler = MinMaxScaler()
hasil = scaler.fit_transform(df)
df.loc[:, [column for column in df.columns]] = hasil
df
```

	Umur	Premi	Lama_Berlangganan
0	0.446154	0.082475	0.467128
1	0.030769	0.068822	0.799308
2	0.061538	0.200636	0.179931
3	0.400000	0.062575	0.083045
4	0.230769	0.045268	0.757785
...	...	...	...
285826	0.046154	0.043454	0.716263
285827	0.015385	0.078239	0.138408
285828	0.046154	0.087661	0.747405
285829	0.738462	0.051853	0.899654
285830	0.384615	0.062973	0.117647

218707 rows × 3 columns

Normalisasi yang saya gunakan menggunakan rentang angka 0 sampai 1, tujuannya agar lebih mudah dalam visualisasi dan klasterisasi.

4. Evaluasi

```python
def euclidean_distance (x1,x2) :
    return np.sqrt(np.sum((x1-x2)**2))
```

Menggunakan algoritma Euclidean distance untuk penghitungan jaraknya.

```python
class KMeans_scratch :
    def __init__(self, K=2, maks_iteration=150):
        self.K = K
        self.maks_iteration = maks_iteration
        self.centroids = []
        self.clusters = [[] for i in range(self.K)]
```

Menggunakan class k-means dengan parameter K=2. Maks iteration itu gunanya untuk menentukan maksimal dalam melakukan iterasi, disini kita set di 150 defaultnya.

```python
    def predict(self, X) :
        self.X = X
        self.baris , self.column = X.shape
```

Untuk menghasilkan label dari hasil klasterisasi.

```python
        # Generate random centroid
        centroid_indexes = np.random.choice(self.baris,self.K,replace=False)
        for index in centroid_indexes :
            self.centroids.append(self.X[index])

        # Iteration sebanyak maks iteration atau centroid tidak berubah posisi
        for i in range(self.maks_iteration) :

            # Menentukan clusters / centroid terdekat dari setiap titik
            clusters = [[] for i in range(self.K)]
            for idx, row in enumerate(self.X) :
                # Mencari centroid terdekat dengan membandingkan jarak setiap centroid dan mencari jarak terpendek
                closest_centroids_index = self.select_nearest_centroid(row)
                clusters[closest_centroids_index].append(idx)

            self.clusters = clusters

            # Update centroid
            old_centroids = self.centroids
            self.centroids = self.generate_new_centroids()

            # Cek centroid berubah atau tidak
            isChange = False
            for i,old_centroid in enumerate(old_centroids) :
                # Menghitung jarak old centroid ke new centroid
                distance = euclidean_distance(old_centroid, self.centroids[i])
                if (distance != 0) :
                    isChange = True
            if (isChange == False) :
                break
```

Setelah itu, tahap pertama yaitu generate random centroid, kemudian lakukan looping sebanyak max iteration atau selama centroid tidak berubah posisinya.

Lalu tentukan cluster terdekat, centroid itu pusat terdekat titiknya. Terakhir kita update untuk melihat rata-rata dari cluster set.

```
        # Menentukan label dari hasil klusterisasi
        labels = self.generate_labels()
        return labels
```

Kita gunakan label ini untuk menentukan titik yang termasuk cluster dimana.

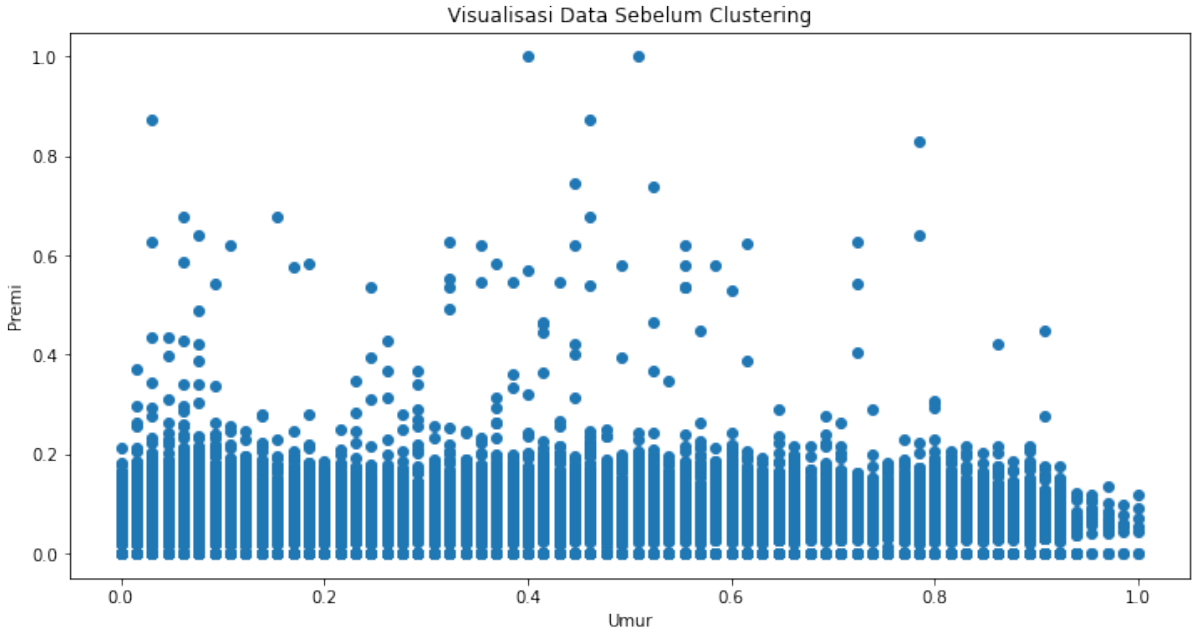
5. Eksperimen

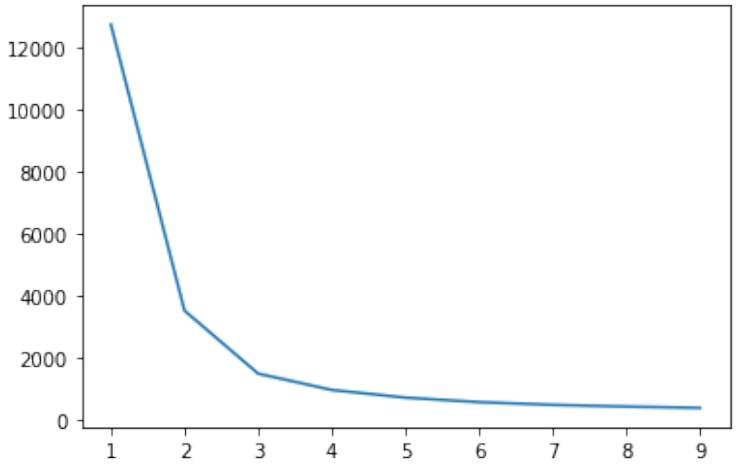
Eksperimen ke-1 :

```
df1 = df.loc[:, ("Umur", "Premi")]
df1.head()
```

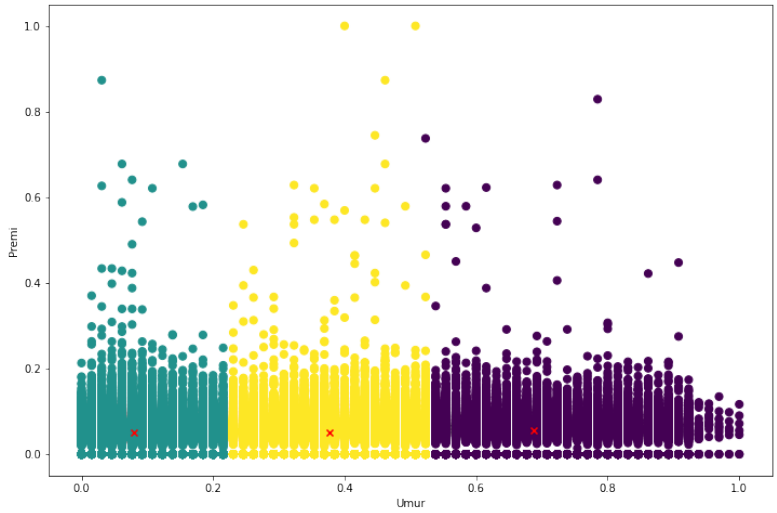
	Umur	Premi
0	0.446154	0.082475
1	0.030769	0.068822
2	0.061538	0.200636
3	0.400000	0.062575
4	0.230769	0.045268

Pada eksperimen pertama, kita gunakan adalah umur dan premi (yang korelasinya baik).





Dapat dilihat diatas bahwa nilai k terbaik sesuai dengan elbow method adalah 3.



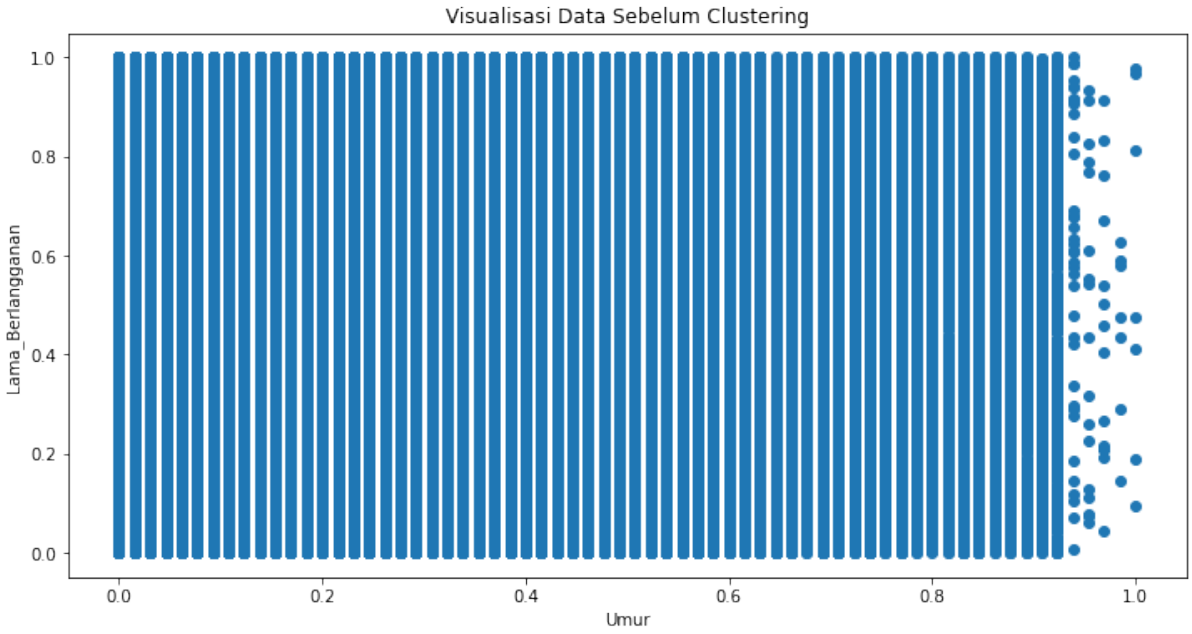
Hasil klasterisasi eksperimen ke-1.

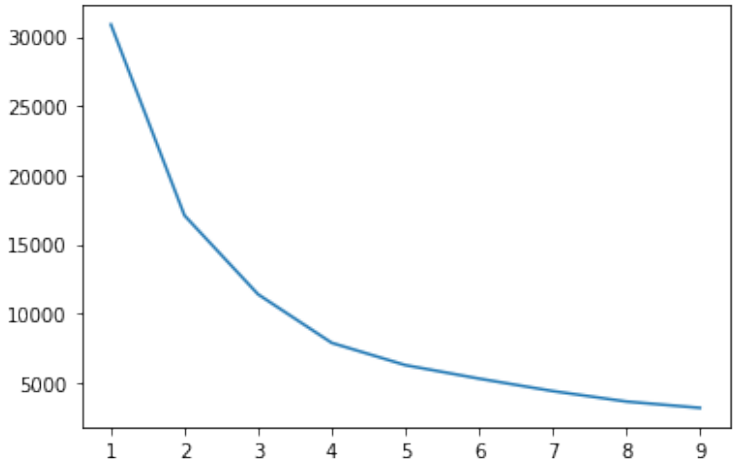
Eksperimen ke-2 :

```
df2 = df.loc[:, ("Umur", "Lama_Berlangganan")]
df2.head()
```

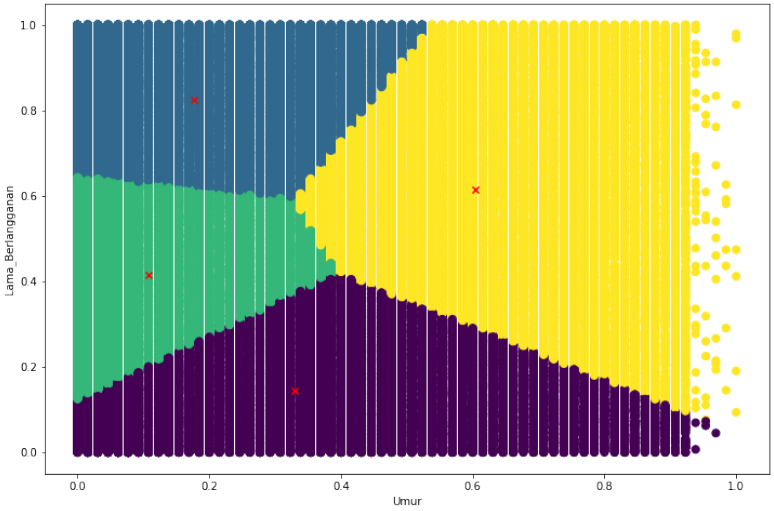
	Umur	Lama_Berlangganan
0	0.446154	0.467128
1	0.030769	0.799308
2	0.061538	0.179931
3	0.400000	0.083045
4	0.230769	0.757785

Pada eksperimen pertama, kita gunakan adalah umur dan lama berlangganan (yang korelasinya kurang baik).





Dapat dilihat diatas bahwa nilai k terbaik sesuai dengan elbow method adalah 4.



Hasil klasterisasi eksperimen ke-2.

6. Evaluasi

```
score_df1 = silhouette_score(X1,y1_pred)
score_df2 = silhouette_score(X2,y2_pred)
```

```
print("score eksperimen 1 : " + str(score_df1))
print("score eksperimen 2 : " + str(score_df2))
```

```
score eksperimen 1 : 0.6221765623482106
score eksperimen 2 : 0.35039496939771664
```

Menggunakan silhouette score.

7. Kesimpulan

Dalam tugas besar tahap 1 ini, dataset yang banyak dan pengolahan *clustering*, kita dapat menggunakan metode *k-means* untuk *clustering* supaya mempermudah dan cukup umum pada metode pengelompokan data. Dan juga, kita menggunakan metode *silhouette score* karena metode tersebut biasa dipakai untuk *clustering*.

8. Link Video Presentasi

Youtube : https://youtu.be/vyLhMmLj9Hc