Fakultas Informatika
School of Computing
Telkom University

ASSIGNMENT COVER SHEET

NIM	Nama	Kelas
1301190374	Azriel Naufal Aulia	IF-43-10

Catatan: Jika tugas ini merupakan tugas kelompok, maka tulis semua anggota kelompok.

Nama MK	Pembelajaran Mesin	
Judul Tugas	Tugas Unsupervised learning, membuat klusterisasi pada dataset pelanggan historis dealer kendaraan ☐ ☐	
Dosen/Tutor	Dr. Gamma Kosala, S.Si, M. Si ☐ ☐	
Tanggal Pengumpulan		**13/11/2021**

Harap perhatikan bahwa Saudara bertanggung jawab untuk menyimpan salinan tugas Saudara.

Plagiarisme: Plagiarisme adalah perbuatan secara sengaja atau tidak sengaja dalam memperoleh atau mencoba memperoleh kredit atau nilai untuk suatu tugas, dengan mengutip sebagian atau seluruh karya dan/atau karya ilmiah pihak lain yang diakui sebagai karya ilmiahnya, tanpa menyatakan sumber secara tepat dan memadai.

Kolusi: Kolusi adalah perbuatan bekerjasama dengan orang lain tanpa izin dari Dosen/Tutor mengenai tugas tertulis, lisan, atau praktik dan termasuk membayar orang lain untuk menyelesaikan semua atau sebagian dari tugas.

Plagiarisme dan/atau Kolusi merupakan bentuk kecurangan akademik. Jika ada indikasi kuat bahwa plagiarisme atau kolusi telah terjadi, maka akan dilaporkan kepada Komisi Etik Fakultas Informatika untuk ditindak lanjuti sesuai dengan peraturan yang berlaku.

Pernyataan Mahasiswa:

- Saya memahami peraturan akademik Universitas Telkom terkait integritas akademik.
- Saya memahami konsekuensi jika terlibat dalam kecurangan.
- Saya menyatakan bahwa saya tidak menjiplak karya orang lain ketika mengerjakan tugas ini.
- Saya telah berupaya untuk memastikan tugas ini tidak dapat dijiplak.
- Saya menyatakan bahwa saya tidak bekerjasama tanpa izin dengan orang lain dan termasuk membayar orang lain untuk menyelesaikan semua atau sebagian dari tugas ini.

Tanda tangan .....  *Tanggal 13/11/2021*

Formulasi Masalah: jelaskan apa permasalahan yang akan diselesaikan.

Tugas clustering (unsupervised Learning) adalah mengelompokkan pelanggan berdasarkan data pelanggan di dealer tanpa memperhatikan label kelas apakah pelanggan tertarik untuk membeli kendaraan baru atau tidak

Objektivitas :

Diberikan data historis pelanggan dealer mobil yang tertarik / membeli dan tidak tertarik / tidak jadi beli mobil. data terdiri dari berbagai macam fitur. Dari dataset tersebut, akan dilakukan proses clustering menggunakan algoritma K-means. K-means adalah algoritma Unsupervised meachine learning yang melakukan proses pengelompokkan data ke dalam k jumlah cluster. Jumlah k / cluster nantinya akan ditentukan oleh pengguna. Penentuan jumlah k yang paling optimal nantinya akan menggunakan Elbow method.

Metode Elbow adalah teknik yang sangat popular dan konsepnya adalah menjalankan pengelompokan k-means untuk rentang cluster k (katakanlah dari 1 hingga 10) dan untuk setiap nilai k, akan dihitung nilai distorsi-nya. Nilai distorsi dihitung sebagai rata-rata jarak (Sum Square Distance) kuadrat antara pusat cluster (titik centroid) ke masing-masing titik didalam cluster.

Penjelasan fitur-fitur dari dataset:

1. Id : identitas pelanggan yang terdaftar
2. Jenis_kelamin : jenis kelamin pelanggan
3. Umur : umur pelanggan
4. SIM : kepemilikan SIM pelanggan (1 berarti pelanggan mempunyai SIM, dan 0 untuk sebaliknya)
5. kode daerah : kode daerah tempat tinggal pelanggan
6. Sudah_asuransi : Asuransi pelanggan (1 berarti pelanggan sudah pernah asuransi, dan 0 untuk sebaliknya.
7. Umur_kendaraan : Umur kendaraan pelanggan (0 untuk umur kendaraan dibawah 1 tahun. 1 untuk umur kendaraan diatas 1 tahun, dan dibawah 2 tahun. 2 untuk umur kendaraan diatas 2 tahun).

8. Kendaraan_rusak : 1 untuk mobil pelanggan pernah rusak dan 0 untuk sebaliknya.
9. Premi : jumlah premi yang harus dibayarkan per tahun
10. Kanal_penjualan : kode kanal untuk menghubungi pelanggan (email, telpon, dll)
11. lama_berlangganan : durasi pelanggan menjadi klien perusahaan
12. tertarik : 1 untuk pelanggan tertarik dan membeli kendaraan dan 0 untuk sebaliknya.

Note : pembuatan model dibuat di dalam environment jupyter notebook

Eksplorasi dan Persiapan Data (termasuk data splitting): lakukan semua teknik eksplorasi dan persiapan data yang menurut Anda perlu dilakukan. Jelaskan mengapa perlu melakukan teknik tersebut, dan lakukan analisis terhadap hasilnya.

Sebelumnya mari kita siapkan library untuk keperluan pembuatan model

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

plt.style.use('bmh')
```

Setelah itu, memuat dataset dan disimpan didalam variable dataTrain

```
# Load data training
dataTrain = pd.read_csv('kendaraan_train.csv')
dataTrain.head()
```

	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

```
dataTrain.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
```

Data Cleaning / pembersihan data

Setelah memuat dataset, untuk keperluan modelling yang lebih optimal, saya akan membuang fitur-fitur yang saya kira tidak berguna / tidak memberikan pengaruh cukup signifikan kepada hasil modeling.

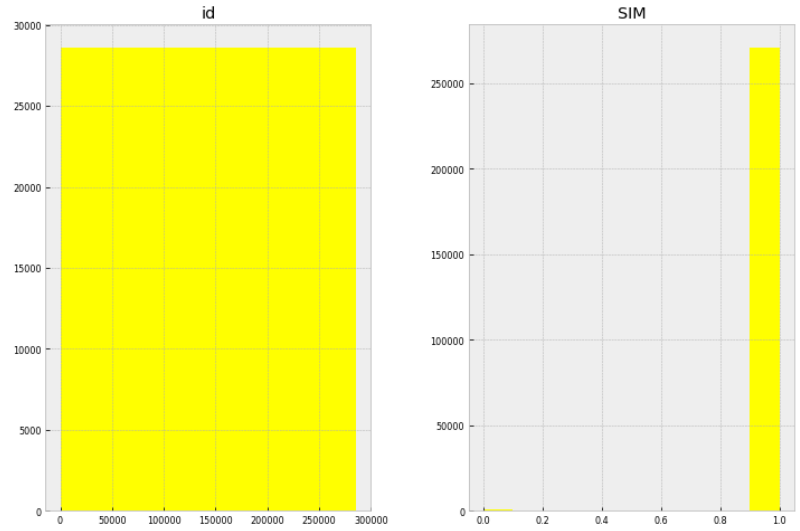
Membuang fitur “tertarik” / label karena pada penilitian model unsupervised learning tidak membutuhkan label untuk masing-masing data.

```
# Drop column tertarik ~ karena tugasnya tentang unsupervised
dataTrain_Unlabel = dataTrain.drop(['Tertarik'], axis = 1)
dataTrain_Unlabel.head()
```

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

Seperti pada fitur “Id” yang hanya merupakan primary key dari dataset (untuk membedakan setiap datanya) dan fitur “SIM” karena jika diteliti, hampir 99% data pada fitur “SIM” memiliki nilai yang sama, yaitu 1 / pelanggan mempunyai SIM.

```
cols = ['id','SIM']
dataTrain_Unlabel[cols].hist(figsize=(12,8), bins=10, xlabelsize=8, ylabelsize=8, color='yellow');
```




```
dataTrain_Unlabel = dataTrain_Unlabel.drop(['id'], axis = 1)
dataTrain_Unlabel = dataTrain_Unlabel.drop(['SIM'], axis = 1)
dataTrain_Unlabel.head()
```

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

Selain itu, saya akan membuang fitur yang bersifat kategorial dan hanya memiliki sedikit nilai kategori, seperti fitur “jenis kelamin” yang hanya meng-kategorikan 2 kategori, pria dan Wanita. Karena untuk kelancaran proses normalisasi dan modeling, yang dimana, dengan adanya fitur kategori yang memiliki nilai ketegori yang sedikit akan membuat proses normalisasi menghasilkan nilai yang tidak selaras. Dan membuat proses modeling menggunakan algoritma K-means clustering tidak optimal.

```
# melihat berapa banyak total kategori pada masing-masing kolom
dataTrain_Unlabel.nunique()
```

```
Jenis_Kelamin            2
Umur                    66
Kode_Daerah             53
Sudah_Asuransi           2
Umur_Kendaraan           3
Kendaraan_Rusak          2
Premi                45114
Kanal_Penjualan        151
Lama_Berlangganan      290
dtype: int64
```

Karena untuk kolom Jenis_kelamin, Sudah_Asuransi, umur_kendaraan, kendaraan_rusak hanya memiliki 2-3 nilai kategori. saya memutuskan untuk mengdrop kolom tersebut supaya tidak menganggu proses normalisasi

```
dataTrain_Unlabel = dataTrain_Unlabel.drop(['Jenis_Kelamin'], axis = 1)
dataTrain_Unlabel = dataTrain_Unlabel.drop(['Sudah_Asuransi'], axis = 1)
dataTrain_Unlabel = dataTrain_Unlabel.drop(['Umur_Kendaraan'], axis = 1)
dataTrain_Unlabel = dataTrain_Unlabel.drop(['Kendaraan_Rusak'], axis = 1)
dataTrain_Unlabel
```

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

285831 rows × 5 columns

Setelah itu, saya akan mengelola masing-masing data yang memiliki nilai kosong (Null / NaN)

```
#Finding Missing values percentage in all columns
miss = pd.DataFrame(dataTrain_Unlabel.isnull().sum())
miss = miss.rename(columns={0:"miss_count"})
miss["miss_%"] = (miss.miss_count/len(dataTrain.index))*100
miss
```

	miss_count	miss_%
Umur	14214	4.972869
Kode_Daerah	14306	5.005055
Premi	14569	5.097068
Kanal_Penjualan	14299	5.002606
Lama_Berlangganan	13992	4.895200

Jika diperhatikan, persentase nilai yang hilang tidak begitu banyak, bahkan hanya sekitar 5% dari total data. Oleh karena itu, saya memutuskan untuk melakukan pembuangan terhadap data yang memiliki nilai fitur yang hilang.

```
# menuliskan berapa banyak rows pada dataframe sebelum NaN value di drop
index = dataTrain_Unlabel.index
rows_before = len(index)
rows_before
```

```
285831
```

```
# drop data yang ada NaN value nya
dataTrain_Unlabel.dropna(inplace = True)
# reset index
dataTrain_Unlabel.reset_index(inplace=True,drop=True)
# show data
dataTrain_Unlabel.head()
```

	Umur	Kode_Daerah	Premi	Kanal_Penjualan	Lama_Berlangganan
0	30.0	33.0	28029.0	152.0	97.0
1	48.0	39.0	25800.0	29.0	158.0
2	21.0	46.0	32733.0	160.0	119.0
3	58.0	48.0	2630.0	124.0	63.0
4	50.0	35.0	34857.0	88.0	194.0

```
# menuliskan berapa banyak rows pada dataframe setelah NaN value di drop
index = dataTrain_Unlabel.index
rows_after = len(index)
rows_after
```

```
221199
```

```
persentase_drop = ((rows_before - rows_after)/rows_before)*100
persentase_drop
```

```
22.611963013109147
```

Persentase data yang dihapus masih cukup wajar, sekitar 22,6 %

```
# dataTrain.isnull().sum().sum()
dataTrain_Unlabel
```

	Umur	Kode_Daerah	Premi	Kanal_Penjualan	Lama_Berlangganan
0	30.0	33.0	28029.0	152.0	97.0
1	48.0	39.0	25800.0	29.0	158.0
2	21.0	46.0	32733.0	160.0	119.0
3	58.0	48.0	2630.0	124.0	63.0
4	50.0	35.0	34857.0	88.0	194.0
...	...	...	...	...	...
221194	23.0	4.0	25988.0	152.0	217.0
221195	21.0	46.0	44686.0	152.0	50.0
221196	23.0	50.0	49751.0	152.0	226.0
221197	68.0	7.0	30503.0	124.0	270.0
221198	45.0	28.0	36480.0	26.0	44.0

221199 rows × 5 columns

Setelah proses pembersihan dataset dari membuang fitur-fitur yang tidak berguna dan membuang baris data yang memiliki nilai kosong (null / NaN), dataset tersisa 221199 baris data (awalnya terdapat 285831 baris data).

Normalisasi

Untuk proses normalisasi, saya menggunakan metode normalisasi nim-max

Formula normalisasi min-max

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

NORMALIZATION

```
# menggunakan min-max normalization ~ dalam nilai 0 - 1
dataTrain_Normal = dataTrain_Unlabel.copy()
for i in dataTrain_Unlabel.columns:
    dataTrain_Normal[i] = (dataTrain_Unlabel[i] - dataTrain_Unlabel[i].min()) / (dataTrain_Unlabel[i].max() - dataTrain_Unlabel[i
```

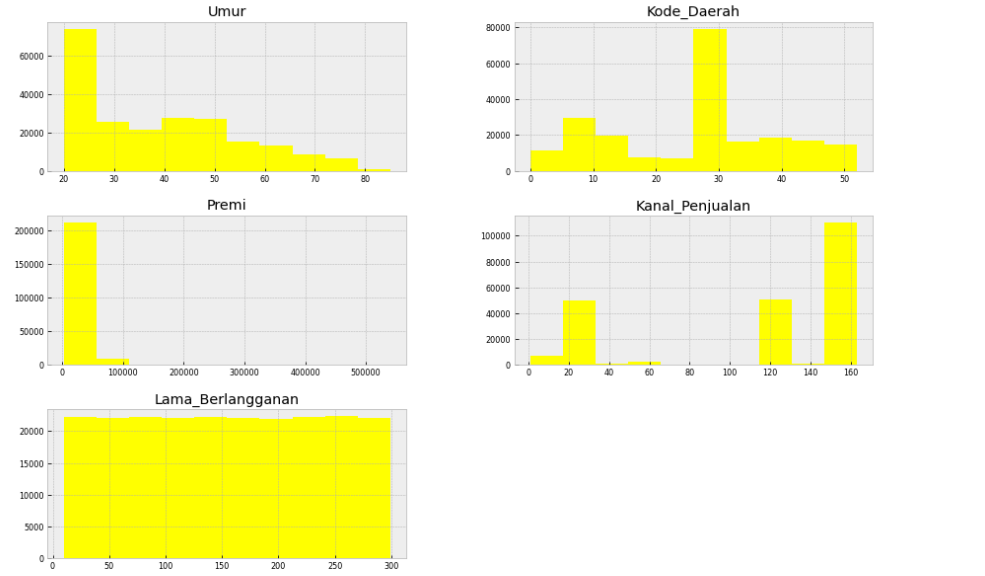
```
dataTrain_Normal.head()
```

	Umur	Kode_Daerah	Premi	Kanal_Penjualan	Lama_Berlangganan
0	0.153846	0.634615	0.047251	0.932099	0.301038
1	0.430769	0.750000	0.043104	0.172840	0.512111
2	0.015385	0.884615	0.056002	0.981481	0.377163
3	0.584615	0.923077	0.000000	0.759259	0.183391
4	0.461538	0.673077	0.059953	0.537037	0.636678

Mencari nilai pencilan (Outliers value)

Dalam mencari nilai outlier, saya menggunakan visualisasi histogram dan boxpot

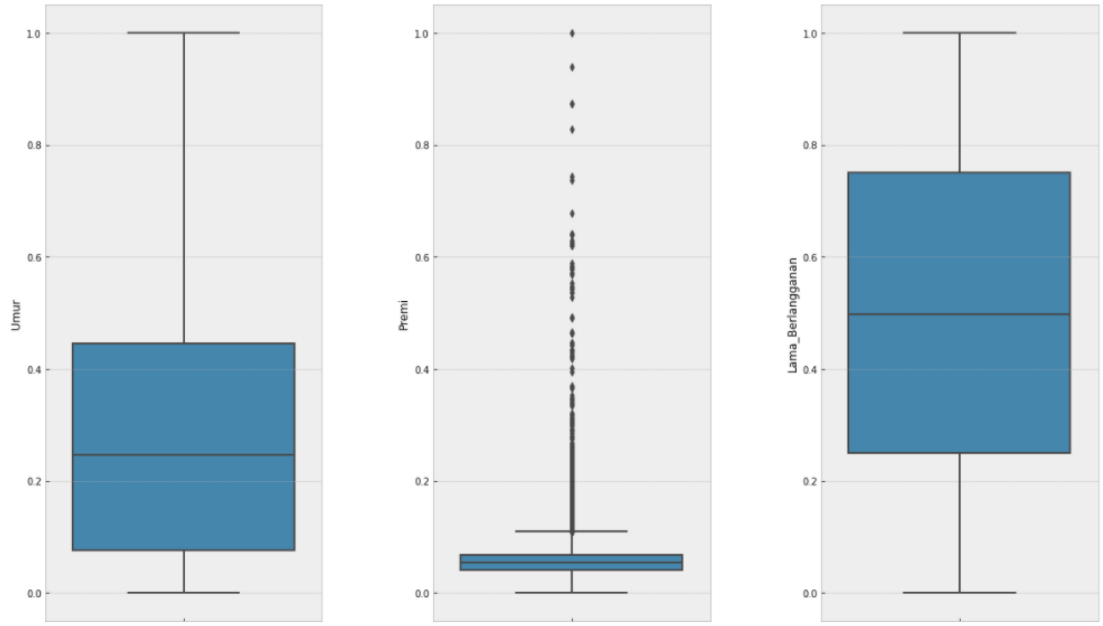
```
dataTrain_Unlabel.hist(figsize=(15,10), bins=10, xlabelsize=8, ylabelsize=8, color='yellow');
```



note : visualisasi distribusi hanya berlaku untuk melihat kolom umur, premi, dan lama berlangganan. sisanya merupakan data kategorial.

```
# membuat boxplot untuk melihat outliers, hanya pada kolom Premi, Umur, dan Lama_Berlangganan
# (karena merupakan data numerik ~ diskrit / kontinu), sisanya merupakan kolom data kategorial.
f, axes = plt.subplots(1,3, figsize=(20,12), gridspec_kw={'wspace': 0.4, 'hspace':0.4})

sns.boxplot(y=dataTrain_Normal['Umur'], orient='v', ax=axes[0])
sns.boxplot(y=dataTrain_Normal['Premi'], orient='v', ax=axes[1])
sns.boxplot(y=dataTrain_Normal['Lama_Berlangganan'], orient='v', ax=axes[2])
```

```
<AxesSubplot:ylabel='Lama_Berlangganan'>
```



HASIL : outliers pada kolom premi, merupakan outlier yang normal karena premi hanya merupakan biaya yang harus dibayar pertahun

Dari hasil boxplot, dapat dilihat terdapat nilai pencilan (outlier) pada fitur ‘premi’. Namun jika kita analisa fitur ‘premi’, premi merupakan jumlah premi yang harus dibayarkan pelanggan per tahunnya, dan karena pada dataset tidak dijelaskan lebih lanjut untuk produk asuransi dari masing-masing pelanggan, maka kita tidak bisa menentukan apakah nilai pencilan pada fitur ‘premi’ merupakan nilai pencilan yang valid. Masing-masing pelanggan bisa saja membeli produk asuransi yang berbeda sehingga nilai pada fitur ‘premi’ menjadi terdistribusi merata seperti pada di boxplot. Sehingga saya memutuskan bahwa nilai pencilan (outliers) pada fitur ‘premi’ merupakan nilai yang normal dan tidak perlu untuk melakukan pembuangan.

Pada penelitian saya kali ini, saya memutuskan untuk membuat 2 experimen berdasarkan data input yang digunakan, experiment pertama saya membuat model dengan menggunakan data input, fitur dataset setelah dilakukan reduksi dimensi memakai PCA (principal component analysis). Sedangkan untuk experiment kedua, saya membuat model dengan menggunakan data input, dari 2 fitur dataset yang memiliki korelasi yang tinggi (berkorelasi paling posiitif ataupun negative)

Menggunakan Reduksi Dimensi dengan PCA (principal component analysis) untuk data input model (experimen pertama)

Saya melakukan reduksi dimensi dibantu dengan bantuan library sklearn.decomposition. Dengan reduksi dimensi, saya meng-reduksi 5 fitur pada dataset (umur, kode_daerah, premi, kanal_penjualan, lama_berlangganan) menjadi hanya 2 fitur (PCA_1 dan PCA_2)

Reduksi Dimensi menggunakan PCA (diexperimen pertama)

```
from sklearn.decomposition import PCA
```

```
dataTrain_Normal.head()
```

	Umur	Kode_Daerah	Premi	Kanal_Penjualan	Lama_Berlangganan
0	0.153846	0.634615	0.047251	0.932099	0.301038
1	0.430769	0.750000	0.043104	0.172840	0.512111
2	0.015385	0.884615	0.056002	0.981481	0.377163
3	0.584615	0.923077	0.000000	0.759259	0.183391
4	0.461538	0.673077	0.059953	0.537037	0.636678

```
# kita membuat class PCA() dengan memberikan parameter n_components yang mendefinisikan jumlah komponen
# atau kolom baru yang diinginkan.
pca = PCA(n_components=2)
```

```
fit_pca = pca.fit_transform(dataTrain_Normal)
```

```
df_pca = pd.DataFrame(data = fit_pca, columns = ['PCA_1', 'PCA_2'])
df_pca.head()
```

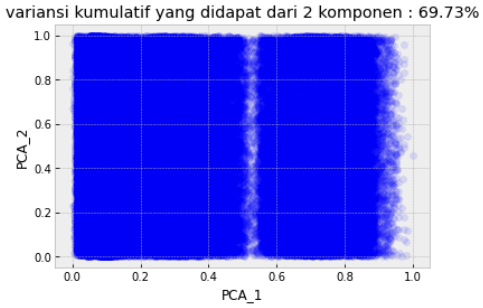
	PCA_1	PCA_2
0	-0.272957	-0.200354
1	0.529372	0.010251
2	-0.368520	-0.127250
3	0.107200	-0.320851
4	0.223982	0.135387

Hasil reduksi diatas, masih terdapat nilai negative, sehingga harus dilakukan normalisasi terlebih dahulu supaya nilai hanya dikisaran 0-1.

```
# Karena terdapat nilai yang negatif, maka harus dinormalisasi terlebih dahulu
# menggunakan min-max normalization ~ dalam nilai 0 - 1
dfPca_Normal = df_pca.copy()
for i in df_pca.columns:
    dfPca_Normal[i] = (df_pca[i] - df_pca[i].min()) / (df_pca[i].max() - df_pca[i].min())
```

Kemudian saya plot hasil reduksi dimensi-nya

```
# Plot PCA
plt.scatter(dfPca_Normal['PCA_1'], dfPca_Normal['PCA_2'], alpha= .1, color='blue')
plt.title("variansi kumulatif yang didapat dari 2 komponen : {:.2%}".format(np.sum(pca.explained_variance_ratio_)))
plt.xlabel("PCA_1")
plt.ylabel("PCA_2")
plt.show()
```



Dengan melihat secara kasat mata, plot hasil reduksi dimensi diatas sudah cukup terkelompokkan secara jelas, walaupun masih ada poin-poin yang belum bisa ditentukan tanpa menggunakan algoritma K-means clustering. Hasil reduksi dimensi dimasukkan ke dalam variable dfPca_Normal untuk digunakan dalam pembuatan model

```
dfPca_Normal
```

	PCA_1	PCA_2
0	0.121965	0.301300
1	0.705098	0.509401
2	0.052510	0.373534
3	0.398263	0.182236
4	0.483140	0.633048
...	...	...
221194	0.057927	0.717824
221195	0.083859	0.137676
221196	0.098267	0.738536
221197	0.417899	0.898991
221198	0.690882	0.122040

221199 rows × 2 columns

Menggunakan Pasangan fitur dengan korelasi paling besar, positif dan negative sebagai data input model (Experimen kedua)

Membuat plot korelasi untuk melihat korelasi antara masing-masing fitur.

FIND CORRELATION (experimen kedua) ~ menggunakan pasangan kolom dengan korelasi paling positif ataupun negatif

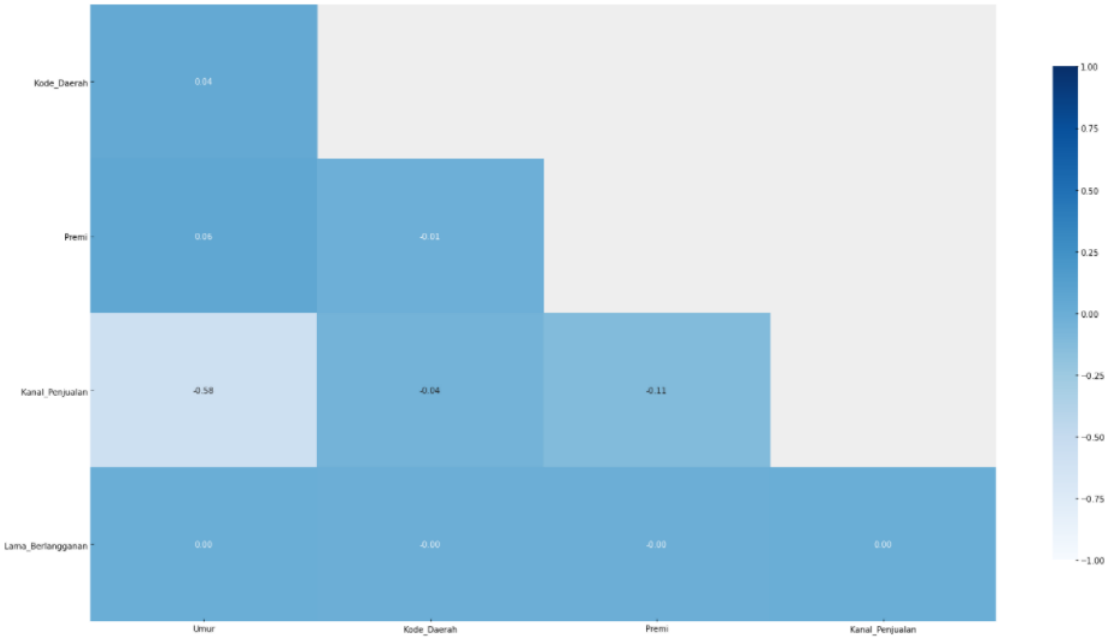
```
# dataTrain_Unlabel # belum di normalisasi
# dataTrain_Normal # sudah dinormalisasi
# reset index
dataTrain_Normal.reset_index(inplace=True,drop=True)
dataTrain_Normal.head()
```

	Umur	Kode_Daerah	Premi	Kanal_Penjualan	Lama_Berlangganan
0	0.153846	0.634615	0.047251	0.932099	0.301038
1	0.430769	0.750000	0.043104	0.172840	0.512111
2	0.015385	0.884615	0.056002	0.981481	0.377163
3	0.584615	0.923077	0.000000	0.759259	0.183391
4	0.461538	0.673077	0.059953	0.537037	0.636678

```
# untuk mencari korelasi, kita menggunakan dfNum_Unlabel
df1_corr = dataTrain_Normal.corr()
df1_corr
```

	Umur	Kode_Daerah	Premi	Kanal_Penjualan	Lama_Berlangganan
Umur	1.000000	0.044031	0.063716	-0.577513	0.000117
Kode_Daerah	0.044031	1.000000	-0.012261	-0.043869	-0.003012
Premi	0.063716	-0.012261	1.000000	-0.111770	-0.001510
Kanal_Penjualan	-0.577513	-0.043869	-0.111770	1.000000	0.000141
Lama_Berlangganan	0.000117	-0.003012	-0.001510	0.000141	1.000000

Supaya lebih mudah, saya membuat heatmap berdasarkan nilai korelasi yang didapat



Dari hasil korelasinya, dapat diperhatikan bahwa antara fitur-fitur, tidak ada yang memiliki korelasi positif yang tinggi, korelasi tertinggi ada pada fitur umur dan premi yang hanya sebesar 0.06 (disimpan di dalam variable dataFinal2a). sedangkan untuk nilai yang paling negative ada pada fitur kanal_penjualan dan fitur, sebesar -0,57 (disimpan di dalam variable dataFinal2). Sehingga untuk experiment kedua ini, saya akan menggunakan pasangan fitur umur dengna premi dan fitur kanal_penjualan dan umur

Pemodelan: bangunlah model menggunakan data hasil praproses, dan lakukan proses training untuk mendapatkan hasil terbaik. Jelaskan secara detail semua proses yang Anda lakukan dilengkapi dengan justifikasi dan analisis hasilnya.

```python
# perhitungan jarak menggunakan formula euclidian
def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2) ** 2))


class Kmeans:
    # konstruktor
    def __init__(self, K=5, max_iters=100, plot_steps=False):
        self.K = K
        self.max_iters = max_iters
        self.plot_steps = plot_steps

        # list untuk poin-poin data pada masing-masing kluster
        self.clusters = [[] for _ in range(self.K)]
        # centroid untuk masing-masing kluster
        self.centroids = []

    def predict(self, X):
        self.X = X
        self.n_samples, self.n_features = X.shape

        # inisialisasi centroid awal. centroid awal diletakkan secara random
        random_sample_idxs = np.random.choice(self.n_samples, self.K, replace=False)
        self.centroids = [self.X[idx] for idx in random_sample_idxs]

        # looping sampai proses klusterisasi sudah optimal / tidak terjadi perpindahan centroid
        for _ in range(self.max_iters):
            # membuat kluster / untuk masing-masing data poin memilih centroid terdekat
            self.clusters = self._create_clusters(self.centroids)

            if self.plot_steps:
                self.plot()

            # mencari centroid baru di setiap kluster
            centroids_old = self.centroids
            self.centroids = self._get_centroids(self.clusters)

            # cek apakah posisi centroid masih berubah
            if self._is_converged(centroids_old, self.centroids):
                break

            if self.plot_steps:
                self.plot()

        # meng-klasifikasikan kluster masing-masing data poin
        return self._get_cluster_labels(self.clusters)
```

```
    def _get_cluster_labels(self, clusters):
        # setiap sampel akan mendapatkan label dari cluster yang didapat
        labels = np.empty(self.n_samples)

        for cluster_idx, cluster in enumerate(clusters):
            for sample_index in cluster:
                labels[sample_index] = cluster_idx
        return labels

    def _create_clusters(self, centroids):
        # Assign sampel ke centroid terdekat untuk membuat cluster
        clusters = [[] for _ in range(self.K)]
        for idx, sample in enumerate(self.X):
            centroid_idx = self._closest_centroid(sample, centroids)
            clusters[centroid_idx].append(idx)
        return clusters

    def _closest_centroid(self, sample, centroids):
        # return jarak sampel saat ini ke setiap centroid
        distances = [euclidean_distance(sample, point) for point in centroids]
        closest_index = np.argmin(distances)
        return closest_index

    def _get_centroids(self, clusters):
        # assign nilai rata-rata cluster ke centroid
        centroids = np.zeros((self.K, self.n_features))
        for cluster_idx, cluster in enumerate(clusters):
            cluster_mean = np.mean(self.X[cluster], axis=0)
            centroids[cluster_idx] = cluster_mean
        return centroids

    def _is_converged(self, centroids_old, centroids):
        # jarak antara masing-masing centroid lama dan baru, untuk semua centroids
        distances = [
            euclidean_distance(centroids_old[i], centroids[i]) for i in range(self.K)
        ]
        return sum(distances) == 0

    def plot(self):
        fig, ax = plt.subplots(figsize=(12, 8))

        for i, index in enumerate(self.clusters):
            point = self.X[index].T
            ax.scatter(*point)

        for point in self.centroids:
            ax.scatter(*point, marker="x", color="black", linewidth=2)

        plt.show()
```

Untuk memudahkan proses clustering kedepannya, saya membuat kelas Kmeans dengan prinsip Object Orientation program. Dimana Kmeans adalah nama kelas dan metode predict untuk menghasilkan label cluster (0,1…,k)

Untuk penggunaan kelas Kmeans, input parameter pada kelas Kmeans harus dengan tipe array.

```
dataFinal = dfPca_Normal.to_numpy()
dataFinal.shape
```

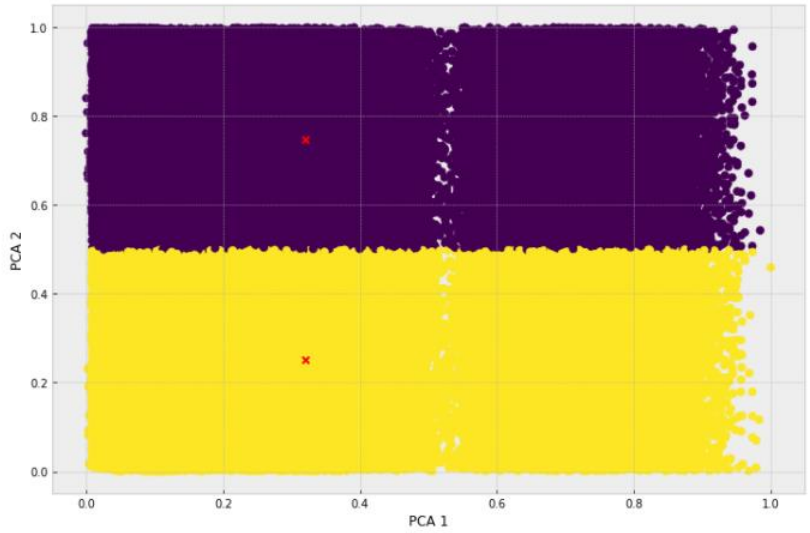
```
(221199, 2)
```

Untuk masing-masing percobaan, algoritma K-means clustering akan menetapkan iterasi maksimum sebanyak 10 kali, yang berarti proses klusterisasi / perpindahan centroid akan dilakukan maksimal sebanyak 10 kali perulangan.

- Eksperimen clustering dengan menggunakan data input dari hasil reduksi dimensi

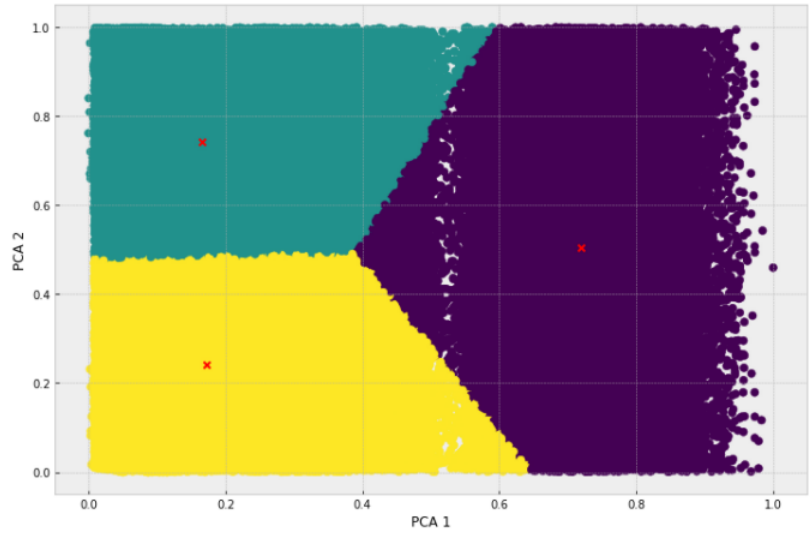
1. Percobaan K = 2

```
# K = 2
model = Kmeans(K=2, max_iters=10)
predik = model.predict(dataFinal)
predik
plt.figure(figsize=(12,8))
plt.scatter(dataFinal[:,0], dataFinal[:,1], c=predik, s=50)

centroid = model.centroids
plt.scatter(centroid[:,0], centroid[:,1], c='red', marker='x', linewidth=2)
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.show()
```



2. Percobaan K = 3

```
# K = 3
model = Kmeans(K=3, max_iters=10)
predik = model.predict(dataFinal)

plt.figure(figsize=(12,8))
plt.scatter(dataFinal[:,0], dataFinal[:,1], c=predik, s=50)

centroid = model.centroids
plt.scatter(centroid[:,0], centroid[:,1], c='red', marker='x', linewidth=2)
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.show()
```



Dengan melihat secara kasat mata, hasil clustering menggunakan data input hasil reduksi dimensi sudah cukup terkelompok dengan jelas, dari K = 2 dan 3. Selain itu, pada hasil clustering K = 2, hasil plot terlihat cukup sama dengan plot hasil reduksi dimensi diatas. Yang memperlihatkan dari hasil reduksi dimensi, data sudah sedikit terkelompokkan menjadi 2 bagian.

Untuk eksperimen menggunakan data input pasangan fitur akan ditampilkan di bagian Eksperimen.

Evaluasi: pilih metode evaluasi yang sesuai beserta justifikasinya. Lakukan evaluasi terhadap model yang telah dihasilkan. Berikan analisis terhadap hasil evaluasi.

Untuk mengevaluasi K / cluster yang paling optimal, saya akan menggunakan metode Elbow. Dikarenakan perhitungannya yang sederhana dan merupakan metode evaluasi yang baru saja saya pelajari di kelas mata kuliah. Dengan menggunakan metode Elbow, untuk setiap nilai k, akan dihitung nilai distorsi-nya. Nilai distortion dihitung sebagai rata-rata jarak (Sum Square Distance) kuadrat antara pusat cluster (titik centroid) ke masing-masing titik di dalam cluster. Nantinya nilai distortion akan di plot dengan nilai K, dan akan diperhatikan dimana nilai K yang akan mulai mengalami perlambatan penurunan.

Pada proses evaluasi, saya dibantu dengan menggunakan library sklearn.cluster

evaluation

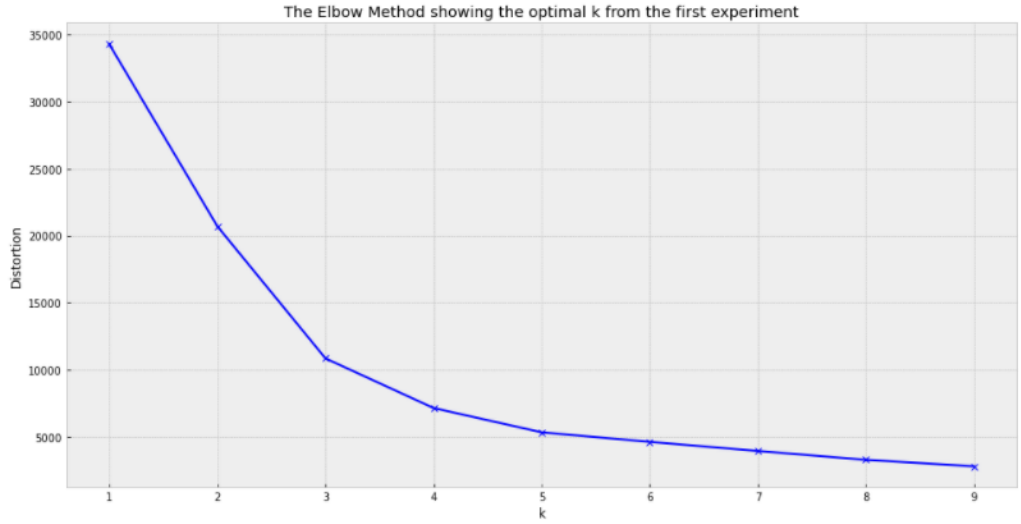
to find optimum number of clusters use ELBOW METHOD

```
from sklearn.cluster import KMeans
```

experimen pertama, modeling menggunakan reduksi dimensi

```
distortions = []
K = range(1,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(dataFinal)
    distortions.append(kmeanModel.inertia_)
```

```
plt.figure(figsize=(16,8))
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k from the first experiment')
plt.show()
```



Berdasarkan hasil plot diatas, bisa disimpulkan bahwa K / cluster yang paling optimal untuk digunakan dalam mengelompokkan dataset adalah sebanyak 3 kluster. Karena Ketika kluster sudah lebih dari 3, grafik diatas, sudah mulai menunjukkan perlambatan penurunan.

Eksperimen: lakukan berbagai eksperimen yang melibatkan tahapan Eksplorasi dan Persiapan Data, Pemodelan, dan Evaluasi untuk mendapatkan hasil terbaik. Laporkan semua Eksperimen yang Anda lakukan beserta analisis hasil dan perbandingannya.

Saya melakukan 2 eksperimen yang berdasarkan 2 data input yang berbeda ke dalam model. Eksperimen pertama, data input merupakan hasil dari reduksi dimensi menggunakan PCA (principal component analysis), hasil dari eksperimen pertama sudah di tampilkan pada bagian permodelan dan evaluasi . Kemudian untuk, eksperimen kedua, data input merupakan pasangan fitur yang dipilih berdasarkan nilai korelasinya. Pada eksperimen kedua, saya memilih 2 pasangan fitur data, yaitu pasangan fitur umur dengan fitur premi, pasangan fitur yang memiliki nilai korelasi yang paling positive dan pasangan fitur umur dengan fitur kanal_penjualan, pasangan fitur yang memiliki nilai korelasi yang paling negative. Pada eksperimen kedua, saya memutuskan memilih 2 pasangan fitur data ketimbang hanya memilih satu, dikarenakan kedua pasangan fitur data tersebut bisa dikatakan mempengaruhi nilai antara fitur satu sama lainnya, maupun itu korelasi yang dihasilkan positif maupun negative.

- Eksperimen clustering dengan menggunakan data input pasangan fitur data yang memiliki keterkaitan korelasi yang paling positif ataupun negatif pada dataset.

Percobaaan 1 menggunakan fitur umur dan premi (K = 2)

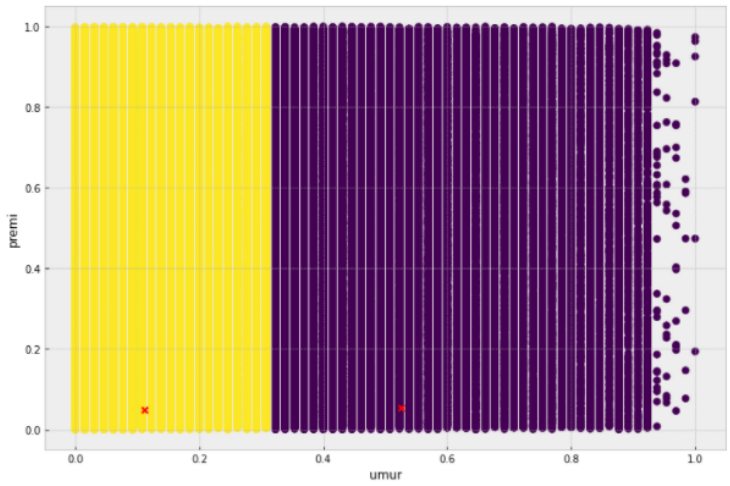
mengambil kolom premi dan umur sebagai data input ke dalam model

```
cols = ['Umur','Premi']
df_exp2 = dataTrain_Normal[cols]
df_exp2.head()
```

```
dataFinal2 = df_exp2.to_numpy()
dataFinal2.shape
```

```
(221199, 2)
```

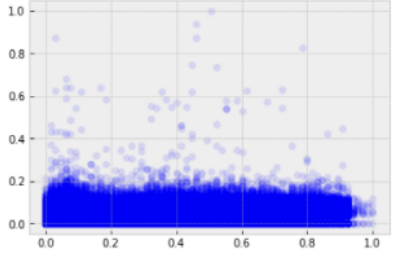
```
model = Kmeans(K=2, max_iters=10)
predik = model.predict(dataFinal2)

plt.figure(figsize=(12,8))
plt.scatter(dataFinal2[:,0], dataFinal[:,1], c=predik, s=50)

centroid = model.centroids
plt.scatter(centroid[:,0], centroid[:,1], c='red', marker='x', linewidth=2)
plt.xlabel('umur')
plt.ylabel('premi')
plt.show()
```



Dengan kasat mata, hasil klusterisasi, sudah cukup baik dalam mengelompokkan data, namun, hasil klusterisasi diatas tidak begitu padat dan simetris jika kita bandingkan dengan hasil kluster pada eksperimen pertama (Ketika menggunakan fitur hasil reduksi dimensi sebagai data input), selain itu, juga plot sebelum dan sesudah klusterisasi tidak sama atau tidak representasi dengan satu sama lain.

```
plt.scatter(df_exp2['Umur'], df_exp2['Premi'], alpha= .1, color='blue')
```

```
<matplotlib.collections.PathCollection at 0x1d4fbd7b610>
```



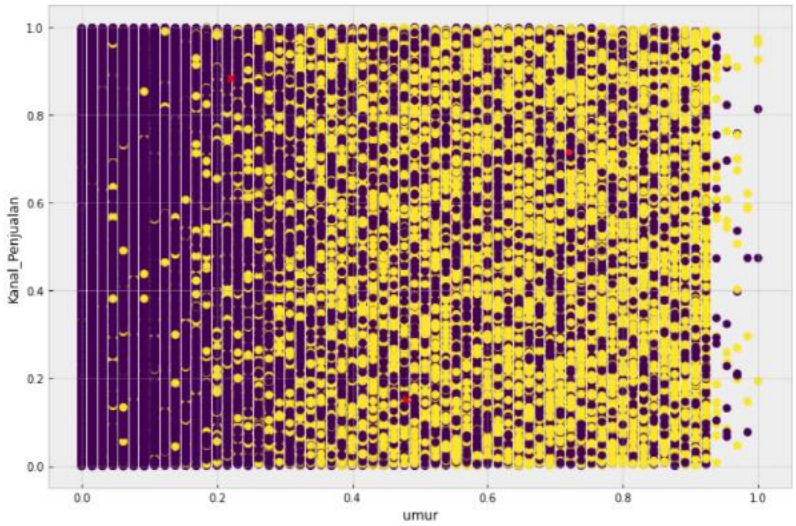
Percobaan 2 menggunakan fitur umur dan kanal_penjualan (K=2)

mengambil kolom kanal_penjualan dan umur sebagai data input ke dalam model

```
cols = ['Umur','Kanal_Penjualan']
df_exp2a = dataTrain_Normal[cols]
df_exp2a.head()
```

```
dataFinal2a = df_exp2a.to_numpy()
dataFinal2a.shape
```

```
model = Kmeans(K=2, max_iters=10)
predik = model.predict(dataFinal2a)

plt.figure(figsize=(12,8))
plt.scatter(dataFinal2a[:,0], dataFinal[:,1], c=predik, s=50)

centroid = model.centroids
plt.scatter(centroid[:,0], centroid[:,1], c='red', marker='x', linewidth=2)
plt.xlabel('umur')
plt.ylabel('Kanal_Penjualan')
plt.show()
```



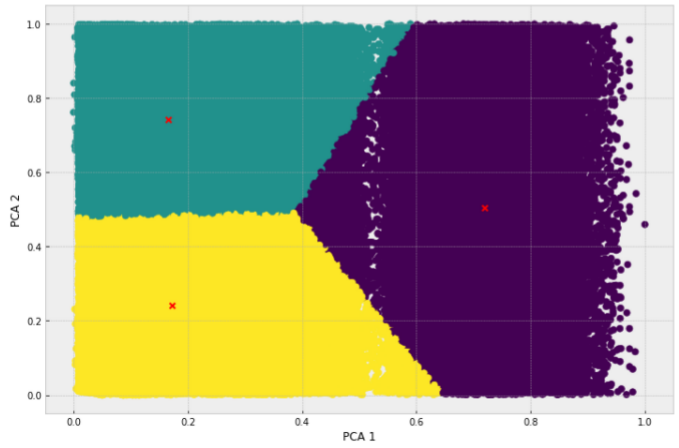
Dari hasil klusterisasi diatas, dapat kita simpulkan bahwa proses klusterisasi diatas membutuhkan iterasi klusterisasi yang lebih banyak. Atau dengan kata lain membutuhkan waktu dan proses yang lebih Panjang dari proses klusterisasi di eksperimen lainnya. Sehingga saya memutuskan untuk tidak menggunakan eksperimen tersebut karena dapat memakan waktu yang lebih lama dari hasil eksperimen lainnya.

Kesimpulan

Dari hasil penelitian Unsupervised learning, saya mendapatkan bahwa untuk mengelompokkan dataset yang diberikan menggunakan algoritma K-means Clustering, cluster yang paling optimal adalah sebanyak 3 cluster dengan input data menggunakan fitur yang sudah di reduksi dimensi memakai PCA (Principal Component Analysis).

Hasil kluster yang paling optimal

```
# K = 3
model = Kmeans(K=3, max_iters=10)
predik = model.predict(dataFinal)

plt.figure(figsize=(12,8))
plt.scatter(dataFinal[:,0], dataFinal[:,1], c=predik, s=50)

centroid = model.centroids
plt.scatter(centroid[:,0], centroid[:,1], c='red', marker='x', linewidth=2)
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.show()
```



Hasil dataset yang sudah ditambahkan kolom kluster

hasil : dataset yang sudah ada hasil klusterisasinya

```
dataTrain_Unlabel['kluster'] = predik
dataTrain_Unlabel
```

	Umur	Kode_Daerah	Premi	Kanal_Penjualan	Lama_Berlangganan	kluster
0	30.0	33.0	28029.0	152.0	97.0	2.0
1	48.0	39.0	25800.0	29.0	158.0	1.0
2	21.0	46.0	32733.0	160.0	119.0	2.0
3	58.0	48.0	2630.0	124.0	63.0	2.0
4	50.0	35.0	34857.0	88.0	194.0	1.0
...	...	...	...	...	...	...
221194	23.0	4.0	25988.0	152.0	217.0	0.0
221195	21.0	46.0	44686.0	152.0	50.0	2.0
221196	23.0	50.0	49751.0	152.0	226.0	0.0
221197	68.0	7.0	30503.0	124.0	270.0	0.0
221198	45.0	28.0	36480.0	26.0	44.0	1.0

221199 rows × 6 columns

Link Video presentasi dan Drive (jika dibutuhkan) terkati dokumen dan source code

Link video presentasi : https://youtu.be/K1nc07uKklc

Link Drive : https://drive.google.com/drive/folders/13uiDwGIrKC40Ie7CZOD_i3xhnczDtuST?usp=sharing