OLEH :

Meyzo Naufal R (1301184299)

IF-42-04

Tugas Besar MK Pembelajaran Mesin

Clustering 2021

1. Formulasi Masalah

a. Problem

Memprediksi apakah pelanggan tertarik untuk membeli kendaraan baru / berdasarkan data pelanggan dealer

b. Data Set

Dataset yang digunakan adalah Kendaraan.

c. Identify Potential Learning Problems

Solusi yang ditampilkan adalah pengelompokan atribut yang memungkinkan membeli kendaraan baru

2. Ekplorasi dan Persiapan Data

Pertama dilakukan persiapan dan eksplorasi data sebagai berikut:

a. import library

Mengimport library yang dibutuh kan untuk pengerjaan

```
]: #import library untuk mendukung pengerjaan

import numpy as np
from matplotlib import pyplot as plt
from matplotlib.pyplot import figure
import pandas as pd
import math
import seaborn as sns
from pandas import DataFrame
import random as rd
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
```

b. Membaca & MenampilkanData

Disediakan 2 dataset , yaitu data train dan data test yang memiliki 285831 baris x 12 kolom

```
#Lakukan pemanggilan dataset yang sudah disiapkan
df_train = pd.read_csv('kendaraan_train.csv')
df_test = pd.read_csv('kendaraan_test.csv')
display(df_train)
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
285826	285827	Wanita	23.0	1.0	4.0	1.0	< 1 Tahun	Tidak	25988.0	152.0	217.0
285827	285828	Wanita	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	44686.0	152.0	50.0
285828	285829	Wanita	23.0	1.0	50.0	1.0	< 1 Tahun	Tidak	49751.0	152.0	226.0
285829	285830	Pria	68.0	1.0	7.0	1.0	1-2 Tahun	Tidak	30503.0	124.0	270.0
285830	285831	Pria	45.0	1.0	28.0	0.0	1-2 Tahun	Pernah	36480.0	26.0	44.0

285831 rows × 12 columns

c. Mengganti type data dari kategorikal menjadi numerical

type data diganti agar lebih mudah untuk dihitung pada kmeans.

```
#untuk mengganti type data dari kategorikal menjadi numerical
df_train['Kendaraan_Rusak'] = df_train['Kendaraan_Rusak'].replace(['Pernah','Tidak'],[1,0])
df_train['Umur_Kendaraan'] = df_train['Umur_Kendaraan'].replace(['< 1 Tahun','1-2 Tahun','> 2 Tahun'],[0,0.5,1])
df_train['Jenis_Kelamin'] = df_train['Jenis_Kelamin'].replace(['Wanita','Pria'],[0,1])
```

d. Mencari missing value pada dataset kita

Menampilkan informasi tentang missing value pada dataset kita

```
#untuk mengecek apakah ada handling missing value atau tidak
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
```

e. Mengassign nilai yang missing

Karena terdapat banyak sekali data dengan nilai NaN sehingga tidak memungkinkan untuk didrop semua, dilakukan *impute*

missing values. Untuk atribut dengan numeric value diisi dengan nilai mean (nilai rerata), sementara atribut dengan string value diisi dengan modus (value yang sering muncul).

In [5]:
```
#untuk mengassign value yang hilang dengan nilai mean atau modus disesuaikan data nya lebih cocok menggunakan mean atau modus
df_train['Jenis_Kelamin'].fillna(value=df_train['Jenis_Kelamin'].mode()[0],inplace=True)
df_train['Umur'].fillna(value=math.ceil(df_train['Umur'].mean()),inplace=True)
df_train['SIM'].fillna(value=df_train['SIM'].mode()[0],inplace=True)
df_train['Kode_Daerah'].fillna(value=df_train['Kode_Daerah'].mode()[0],inplace=True)
df_train['Sudah_Asuransi'].fillna(value=df_train['Sudah_Asuransi'].mode()[0],inplace=True)
df_train['Umur_Kendaraan'].fillna(value=df_train['Umur_Kendaraan'].mode()[0],inplace=True)
df_train['Kendaraan_Rusak'].fillna(value=df_train['Kendaraan_Rusak'].mode()[0],inplace=True)
df_train['Premi'].fillna(value=math.ceil(df_train['Premi'].mean()),inplace=True)
df_train['Kanal_Penjualan'].fillna(value=df_train['Kanal_Penjualan'].mode()[0],inplace=True)
df_train['Lama_Berlangganan'].fillna(value=math.ceil(df_train['Lama_Berlangganan'].mean()),inplace=True)

#setelah di assign dengan nilai baru cek lagi apakah masih ada handling missing value atau tidak
df_train.isnull().sum()
```

f. Lakukan pengecekan data apakah tipe data nya sudah benar dan data nya sudah tidak hillang

]:
```
#cek apakah masih ada data kategorikal atauu tidak
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      285831 non-null  float64
 2   Umur               285831 non-null  float64
 3   SIM                285831 non-null  float64
 4   Kode_Daerah        285831 non-null  float64
 5   Sudah_Asuransi     285831 non-null  float64
 6   Umur_Kendaraan     285831 non-null  float64
 7   Kendaraan_Rusak    285831 non-null  float64
 8   Premi              285831 non-null  float64
 9   Kanal_Penjualan    285831 non-null  float64
 10  Lama_Berlangganan  285831 non-null  float64
 11  Tertarik           285831 non-null  int64  
dtypes: float64(10), int64(2)
memory usage: 26.2 MB
```

g. Menampilkan korelasi dari masing-masing data

```
plt.figure(figsize=(12,8))
sns.heatmap(df_train.corr(),annot= True)
```

```
7]: <matplotlib.axes._subplots.AxesSubplot at 0x21f65850700>
```

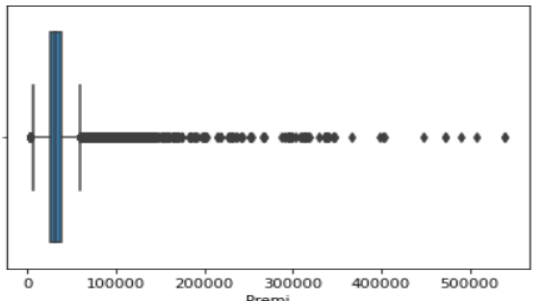
h. cek pencilan pada data

mnampilkan pencilan pada data

```
#cek ada pencilan atau outliers pada data atau tidak
sns.boxplot('Premi', data=df_train)
```

```
c:\python38\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
ersion 0.12, the only valid positional argument will be `data`, and pass
sult in an error or misinterpretation.
  warnings.warn(
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x21f079f0400>
```



i. Menghilangkan dan mencari nilai pencilan pada data

Dikarenakan ada data yang memiliki pencilan maka pencilan harus dihilangkan

```
#fungsi untuk mencari pencilan nya berada dimana
def finding_outlier(df):
    Q1= df.quantile(0.25)
    Q3= df.quantile(0.75)
    IQR= Q3-Q1
    df_final= df[(df<(Q1-(1.5*IQR))) | (df<(Q3+(1.5*IQR)))]
    return df_final

print(finding_outlier(df_train['Umur']))
print(finding_outlier(df_train['Kendaraan_Rusak']))
print(finding_outlier(df_train['Kanal_Penjualan']))
print(finding_outlier(df_train['Lama_Berlangganan']))
```

```
0          30.0
1          48.0
2          21.0
3          58.0
4          50.0
           ...
285826     23.0
285827     21.0
285828     23.0
285829     68.0
285830     45.0
Name: Umur, Length: 285823, dtype: float64
```

In [10]:
```
#fungsi untuk menghilangkan outliers
def remove_outlier(df):
    Q1= df.quantile(0.25)
    Q3= df.quantile(0.75)
    IQR= Q3-Q1
    df_final= df[((df<(Q1-(1.5*IQR))) | (df<(Q3+(1.5*IQR))))]
    return df_final
```

In [11]:
```
df = remove_outlier(df_train[['Umur', 'Kendaraan_Rusak', 'Kanal_Penjualan', 'Lama_Berlangganan']])
df.dropna(axis= 0)
df
```

Out[11]:

	Umur	Kendaraan_Rusak	Kanal_Penjualan	Lama_Berlangganan
0	30.0	0.0	152.0	97.0
1	48.0	1.0	29.0	158.0
2	21.0	0.0	160.0	119.0
3	58.0	0.0	124.0	63.0
4	50.0	1.0	88.0	194.0
...	...	...	...	...
285826	23.0	0.0	152.0	217.0
285827	21.0	0.0	152.0	50.0
285828	23.0	0.0	152.0	226.0
285829	68.0	0.0	124.0	270.0
285830	45.0	1.0	26.0	44.0

285831 rows × 4 columns

k. Scaling data

Dikarenakan untuk mempermudah hitungan maka data di scaling dan dijadikan array

```
data = df_train[['Umur', 'Kendaraan_Rusak', 'Kanal_Penjualan', 'Lama_Berlangganan' ]] #Memakai Data yang akan dipakai untuk clust
data_idx = data.index
scale = MinMaxScaler() #Scale dengan MinMax Scaler
data_scale = scale.fit_transform(data)
data_scale
```

```
array([[0.15384615, 0.        , 0.93209877, 0.30103806],
       [0.43076923, 1.        , 0.17283951, 0.51211073],
       [0.01538462, 0.        , 0.98148148, 0.37716263],
       ...,
       [0.04615385, 0.        , 0.93209877, 0.74740484],
       [0.73846154, 0.        , 0.75925926, 0.89965398],
       [0.38461538, 1.        , 0.15432099, 0.11764706]])
```

l. Menggunakan metode pca

Menggunakan metode pca yaitu metode untuk mereduksikan kolom yang tadi nya bejumlah 4 menjadi 2

7]:
```
pca = PCA(n_components=2) #Menggunakan method pca untuk merubah data menjadi 2 dimensi
principal = pca.fit_transform(df_cluster)
df_princ = pd.DataFrame(data=principal, columns=['pca1','pca2'])
df_princ
```

7]:

	pca1	pca2
0	-0.587458	-0.091621
1	0.607830	0.358986
2	-0.625537	-0.191640
3	-0.464125	0.236148
4	0.519134	0.061812
...	...	...
285826	-0.607453	-0.135987
285827	-0.612555	-0.150030
285828	-0.607480	-0.135926
285829	-0.436708	0.302098
285830	0.604552	0.354628

285831 rows × 2 columns

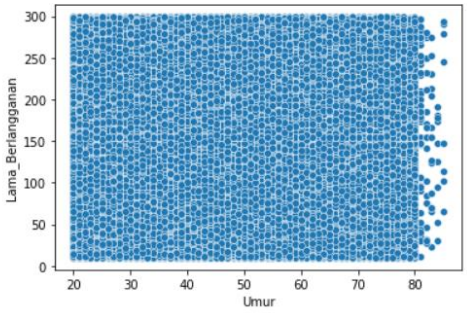
m. Menampilkan Penyebaran data

Persebaran data juga dicari untuk atribut yang bernilai numerik.

```
#untuk melihat data nya seperti apa

sns.scatterplot(df_train['Umur'], df_train['Lama_Berlangganan'])
plt.show()
```

```
c:\python38\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the followir
m version 0.12, the only valid positional argument will be `data`, and passing other argu
result in an error or misinterpretation.
  warnings.warn(
```



n. KMeans Clustering

Menggunakan metode KMeans untuk untuk melakukan clustering dikarenakan metode KMeans lebih mudah dipahami dan hasil nya akurat\

Untuk step by step KMeans sendiri sebagai berikut:

1. inisiai jumlah kluster secara random missal kita memilih k=3 k adalah sebuah parameter yang mewakilkan angka dari cluster yang data points nya akan dilakukan pengelompokan

2. tentukan posisi centroid secara randomg

3. Hitung jarak antara data point ke centroid. Assign setiap data ke centroid terdekat (hal ini akan membentuk k clusters).Perhitungan jarak antara titik data ke centroid menggunakan formula euclidean distance.

4. Kemudian hitung ulang centroid berdasarkan label-label data sebelumnya. Disini kita buat kembali centroid baru dan menghitung kembali jarak antar data poin ke centroid.

5. Kemudian di assign kembali data poin ke centroid terdekat yang terbaru, dan kita cek apakah ada perubahan dalam cluster atau tidak, jika ada maka kita perlu kembali ke poin 4 mencari posisi centroid yang tidak mengubah cluster. Hal ini kita lakukan perulangan sampai posisi centroid tidak berubah lagi.

In [54]:
```
import numpy as np
from numpy.linalg import norm


class Kmeans:
    '''Implementasi Kmeans algorithm.'''

    def __init__(self, n_clusters, max_iter=100, random_state=123):
        self.n_clusters = n_clusters
        self.max_iter = max_iter
        self.random_state = random_state

    def inisiasi_centroids(self, x):
        np.random.RandomState(self.random_state)
        random_idx = np.random.permutation(x.shape[0])
        centroids = x[random_idx[:self.n_clusters]]
        return centroids

    def menghitung_centroids(self, x, labels):
        centroids = np.zeros((self.n_clusters, x.shape[1]))
        for k in range(self.n_clusters):
            centroids[k, :] = np.mean(x[labels == k, :], axis=0)
        return centroids

    def menghitung_distance(self, x, centroids):
        distance = np.zeros((x.shape[0], self.n_clusters))
        for k in range(self.n_clusters):
            row_norm = norm(x - centroids[k, :], axis=1)
            distance[:, k] = np.square(row_norm)
        return distance
```

```
def find_closest_cluster(self, distance):
    return np.argmin(distance, axis=1)

def menghitung_sse(self, x, labels, centroids):
    distance = np.zeros(x.shape[0])
    for k in range(self.n_clusters):
        distance[labels == k] = norm(x[labels == k] - centroids[k], axis=1)
    return np.sum(np.square(distance))

def fit(self, x):
    self.centroids = self.inisiasi_centroids(x)
    for i in range(self.max_iter):
        old_centroids = self.centroids
        distance = self.menghitung_distance(x, old_centroids)
        self.labels = self.find_closest_cluster(distance)
        self.centroids = self.menghitung_centroids(x, self.labels)
        if np.all(old_centroids == self.centroids):
            break
    return self.menghitung_sse(x, self.labels, self.centroids)

def prediksi(self, x):
    distance = self.menghitung_distance(x, old_centroids)
    return self.find_closest_cluster(distance)
```

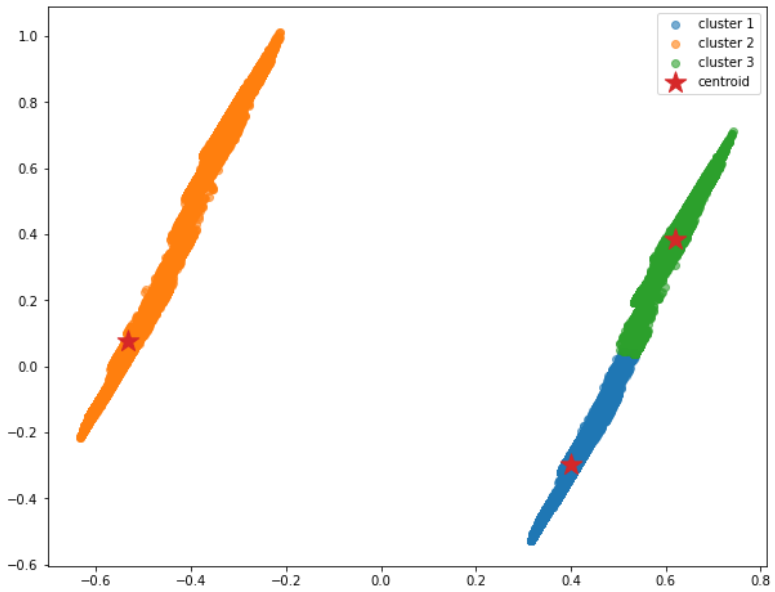
]:
```
z = df_princ.values
km = Kmeans(n_clusters=3, max_iter=100) #Modelling data dengan KMeans
km.fit(z)
centroids = km.centroids
```

o. menampilkan data yang sudah di plot

```
#Plotiing data yang sudah di clustering untuk di visualisasi

fig, ax = plt.subplots(figsize=(10, 8))
plt.scatter(z[km.labels == 0, 0], z[km.labels == 0, 1],
            cmap='viridis', label='cluster 1', alpha=0.6)
plt.scatter(z[km.labels == 1, 0], z[km.labels == 1, 1],
            cmap='copper', label='cluster 2', alpha=0.6)
plt.scatter(z[km.labels == 2, 0], z[km.labels == 2, 1],
            cmap='wistia', label='cluster 3', alpha=0.6)
plt.scatter(centroids[:, 0], centroids[:, 1], marker='*', s=300,
            cmap='black', label='centroid')
plt.legend()
```

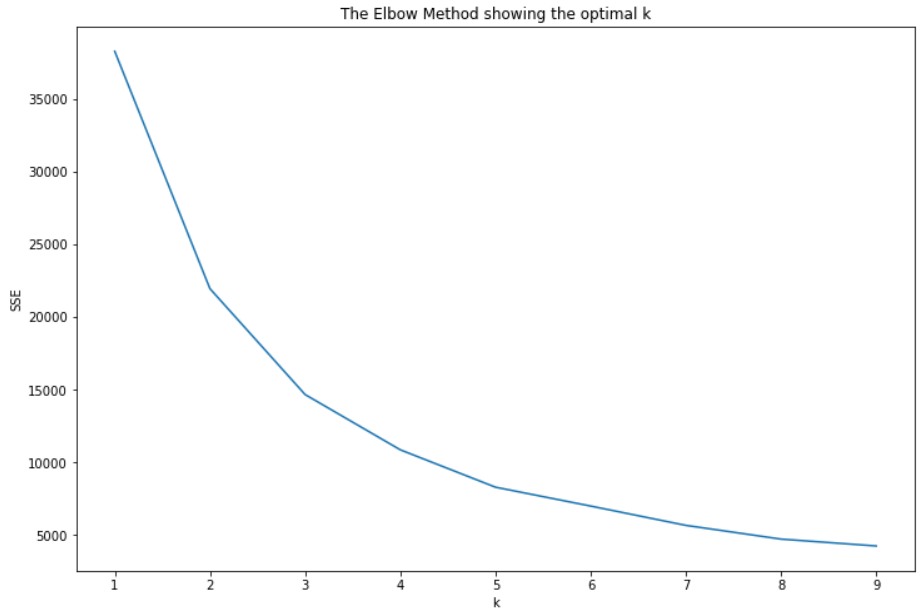
```
<matplotlib.legend.Legend at 0x21f0843d190>
```



p. Elbow Method

Menampilkan elbow method untuk mencari nilai k terbaik

```
plt.figure(figsize=(12,8))
plt.plot(K, sse)
plt.xlabel('k')
plt.ylabel('SSE')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```



q. menentukan nilai silhoute score

```
[7]: from sklearn.metrics import silhouette_score

     print(silhouette_score(z, labels=km.labels))

     0.7207614094475696
```

3. Kesimpulan

Berdasarkan pengolahan data yang dilakukan menggunakan metode clustering KMeans dan optimasi metode elbow method dengan menggunakan SSE(Sum of Square Error) dihasilkan 3 kelompok cluster berdasaykan (umur,Kendaraan rusak, kanal penjualan dan lama berlangganan)