PEMBELAJARAN MESIN

LAPORAN TUGAS BESAR TAHAP PERTAMA

(UNSUPERVISED CLUSTERING)

Laporan

Disusun untuk Memenuhi Tugas Besar Mata Kuliah
Pembelajaran Mesin

Oleh:

Muhammad Zalfa Thoriq (NIM
1301180018) Kelas : IF-43-01

S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS
TELKOM
BANDUNG

2021

1. **Formulasi Masalah**

Clustering adalah sebuah proses untuk mengelompokan data ke dalam beberapa cluster atau kelompok sehingga data dalam satu cluster memiliki tingkat kemiripan yang maksimum dan data antar cluster memiliki kemiripan yang minimum. Objek yang di dalam cluster memiliki kemiripan karakteristik antar satu sama lainnya dan berbeda dengan cluster yang lain.Clustering bertujuan untuk meminimumkan jarak data poin dan cetroid dengan menggunakan metode Within Cluster Sum of Square (WSCC).

Pada tugas kali ini saya melakukan clustering data pada pelanggan yang tertarik untuk membeli kendaraan baru / tidak berdasarkan data pelanggan di dealer tanpa memperhatikan lable kelas pelanggan tertarik membeli kendaraan baru atau tidak dengan menggunakan metode K-Means.

2. **Eksplorasi dan Persiapan Data**

- Import Data & Library

IMPORT DATA & LIBRARY

```
[ ]  import pandas as pd
     import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt
     from sklearn.preprocessing import MinMaxScaler
```

- Import Dataset

Data Train :

```
[ ] data_1 = pd.read_csv('kendaraan_train.csv')
    data_1.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

Data Test :

```
[ ] data_2 = pd.read_excel('kendaraan_test.xlsx')
    data_2.head()
```

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	Wanita	22	1	52	0	1-2 Tahun	Pernah	32895	124	71	1
1	Pria	54	1	52	0	1-2 Tahun	Pernah	43388	124	198	0
2	Wanita	24	1	52	0	1-2 Tahun	Pernah	45032	124	171	0
3	Wanita	78	1	52	0	> 2 Tahun	Pernah	42825	26	208	1
4	Wanita	45	1	52	0	1-2 Tahun	Pernah	2630	26	228	0

- Data Merging

  Menggabungkan antara dataset Train dan data Test yang mengasilkan 333470 data.

```
# Menggabungkan data train and data test
kolom = ['Jenis_Kelamin', 'Umur', 'SIM', 'Kode_Daerah', 'Sudah_Asuransi', 'Umur_Kendaraan', 'Kendaraan_Rusak', 'Premi', 'Kanal_Penjualan', 'Lama_Berlangganan', 'Tertarik']
data_gab = data_1[kolom].append(data_2, ignore_index=True)
data_gab
```

	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
...	...	...	...	...	...	...	...	...	...	...	...
333465	Pria	41.0	1.0	0.0	0.0	1-2 Tahun	Pernah	2630.0	78.0	20.0	0
333466	Pria	39.0	1.0	0.0	0.0	1-2 Tahun	Pernah	2630.0	124.0	200.0	0
333467	Pria	33.0	1.0	0.0	1.0	< 1 Tahun	Tidak	2630.0	152.0	152.0	0
333468	Wanita	71.0	1.0	0.0	0.0	1-2 Tahun	Pernah	2630.0	25.0	226.0	0
333469	Wanita	32.0	1.0	0.0	1.0	1-2 Tahun	Tidak	2630.0	26.0	202.0	0

333470 rows × 11 columns

- Detect Missing Values

  Selanjutnya kami melakukan pengecekan data yang bernilai kosong

```
#Check data null dari setiap kolom
print('Data Type Tiap Column: \n')
print(data_gab.dtypes, '\n')
print('Banyak Missing Values Tiap Column: \n')
print(data_gab.isna().sum())
```

```
Data Type Tiap Column:

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

Banyak Missing Values Tiap Column:

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 data

  Melakukan drop data utuk melakukan pengecekan data kosong yang berjumlah 218707 data. Selanjutnya menghilangkan data yang tidak dipakai yaitu hanya pada kolom tertarik untuk feature yang tidak digunakan.

```
[158] #Drop data
      data_gab = data_gab.dropna()
      len(data_gab)

      218707
```

```
[159] #drop kolom yang tidak digunakan
      data_gab = data_gab.drop(columns=['Tertarik'])
      data_gab.head()
```

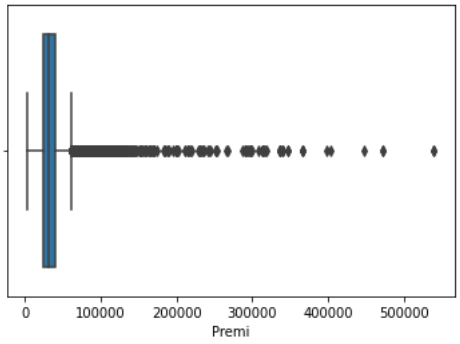
	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan
0	Wanita	30.0	1.0	33.0	1.0	< 1 Tahun	Tidak	28029.0	152.0	97.0
1	Pria	48.0	1.0	39.0	0.0	> 2 Tahun	Pernah	25800.0	29.0	158.0
3	Wanita	58.0	1.0	48.0	0.0	1-2 Tahun	Tidak	2630.0	124.0	63.0
5	Pria	21.0	1.0	35.0	1.0	< 1 Tahun	Tidak	22735.0	152.0	171.0
8	Wanita	20.0	1.0	8.0	1.0	< 1 Tahun	Tidak	30786.0	160.0	31.0

- Outliers

  Check pada Outliers / pencilan untuk melakukan handle dataset, selanjutnya menemukan outlier pada data premi.

```
#plot boxplot untuk melihat data outlier
sns.boxplot(x = 'Premi',data = data_gab)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbd9615fa10>
```



- UpperBound & LowerBound

  menggunakan metode upperbound dan lowerbound yang berfungsi untuk menghilangkan nilai Outliers.

  LowerBound : 1894.75

  UpperBound : 61836.75

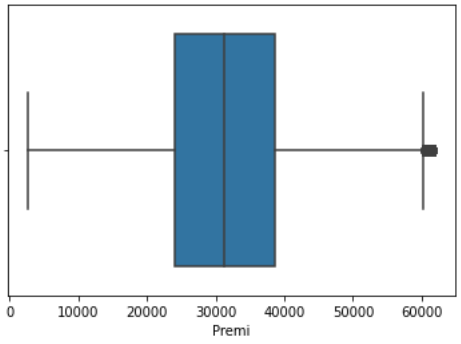
```
#menentukan nilai upperbound dan lower bound
#diambil dari nilai quantile data

q1 = data_gab['Premi'].quantile(0.25)
q3 = data_gab['Premi'].quantile(0.75)
med = q3-q1
lower = q1-1.5*med
upper = q3+1.5*med
print(lower, upper)

#untuk data outlier diatas dari upperbound, maka akan direplace dengan nilai upperbound
data_gab = data_gab[(data_gab['Premi']>lower) & (data_gab['Premi']<upper)]

#plot data setelah tidak ada outlier
sns.boxplot(x = 'Premi', data = data_gab)
```

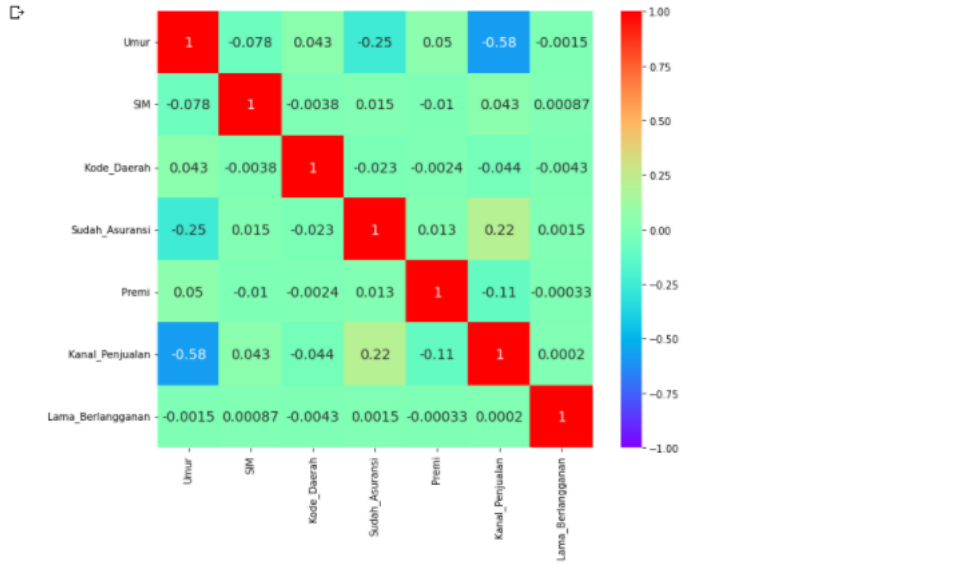
```
1894.75 61836.75
<matplotlib.axes._subplots.AxesSubplot at 0x7fbda2489750>
```



- Heatmap Plot

Korelasi data antar feature menggunakan metode heatmap, akan tetapi dapat dilihat antar feature tidak berkorelasi, karena feature dataset ini sendiri banyak yang berbentuk categorical.

```
#heatmap plot untuk melihat korelasi dari dataset
fig, ax = plt.subplots(figsize=(10,8))
sns.heatmap(data_gab.corr(), annot=True, ax=ax, square=True, cmap='rainbow', vmin=-1,vmax=1, annot_kws={'size': 14})
plt.show()
```



- Labeling Data & Categorial

  Handle pada labelling dan categorial data yang tersedia yaitu dengan membagi kedalam kategori nominal dan ordinal.

a. Jenis_Kelamin
b. Sim
c. Kode_Daerah
d. Sudah_Asuransi
e. Kendaraan_Rusak
f. Umur_kendaraan

Nominal : jenis_kelamin, Sim, Kode_daerah, Sudah_Asuransi, Kendaraan_rusak

Ordinal : Umur_Kendaraan dengan labelEncoder

```
[ ] from sklearn.preprocessing import LabelEncoder

    #labeling untuk data categorical ordinal
    le_umur_kendaraan = LabelEncoder()
    data_gab['Umur_Kendaraan'] = le_umur_kendaraan.fit_transform(data_gab['Umur_Kendaraan'])
```

```
[ ] dummies = pd.get_dummies(data_gab['Jenis_Kelamin'], prefix='kelamin')
    data_gab = pd.concat([data_gab, dummies], axis=1)
    data_gab = data_gab.drop(['Jenis_Kelamin'], axis = 1)

    dummies = pd.get_dummies(data_gab['Kode_Daerah'], prefix='daerah')
    data_gab = pd.concat([data_gab, dummies], axis=1)
    data_gab = data_gab.drop(['Kode_Daerah'], axis = 1)

    dummies = pd.get_dummies(data_gab['SIM'], prefix='sim')
    data_gab = pd.concat([data_gab, dummies], axis=1)
    data_gab = data_gab.drop(['SIM'], axis = 1)

    dummies = pd.get_dummies(data_gab['Sudah_Asuransi'], prefix='asuransi')
    data_gab = pd.concat([data_gab, dummies], axis=1)
    data_gab = data_gab.drop(['Sudah_Asuransi'], axis = 1)

    dummies = pd.get_dummies(data_gab['Kendaraan_Rusak'], prefix='kendaraan_rusak')
    data_gab = pd.concat([data_gab, dummies], axis=1)
    data_gab = data_gab.drop(['Kendaraan_Rusak'], axis = 1)
```

```
[ ] data_gab.head()
```

	Umur	Umur_Kendaraan	Premi	Lama_Berlangganan	kelamin_Pria	kelamin_Wanita	daerah_0.0	daerah_1.0	daerah_2.0	daerah_3.0	daerah_4.0	daerah_5.0	daerah_6.0	daerah_7.0	daerah_8.0	daerah_9.0	daerah_10.0	daerah_11.0	daerah_12.0	daerah_13.0	daerah_14.0	daerah_15.0	daerah_16.0	daerah_17.0	daerah_18.0	d
0	30.0	1	28029.0	97.0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	48.0	2	25800.0	158.0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	58.0	0	2630.0	63.0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	21.0	1	22735.0	171.0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	20.0	1	30786.0	31.0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	

Perbandingan Before & After Labeling

Before :

```
# Menggabungkan data train and data test
kolom = ['Jenis_Kelamin', 'Umur', 'SIM', 'Kode_Daerah', 'Sudah_Asuransi', 'Umur_Kendaraan', 'Kendaraan_Rusak', 'Premi', 'Kanal_Penjualan', 'Lama_Berlangganan', 'Tertarik']
data_gab = data_1[kolom].append(data_2, ignore_index=True)
data_gab
```

	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
...	...	...	...	...	...	...	...	...	...	...	...
333465	Pria	41.0	1.0	0.0	0.0	1-2 Tahun	Pernah	2630.0	78.0	20.0	0
333466	Pria	39.0	1.0	0.0	0.0	1-2 Tahun	Pernah	2630.0	124.0	200.0	0
333467	Pria	33.0	1.0	0.0	1.0	< 1 Tahun	Tidak	2630.0	152.0	152.0	0
333468	Wanita	71.0	1.0	0.0	0.0	1-2 Tahun	Pernah	2630.0	25.0	226.0	0
333469	Wanita	32.0	1.0	0.0	1.0	1-2 Tahun	Tidak	2630.0	26.0	202.0	0

333470 rows × 11 columns

After :

	Umur	Umur_Kendaraan	Premi	Kanal_Penjualan	Lama_Berlangganan	kelamin_Pria	kelamin_Wanita	daerah_0.0	daerah_1.0	daerah_2.0	daerah_3.0	daerah_4.0	daerah_5.0	daerah_6.0	daerah_7.0	daerah_8.0	daerah_9.0	daerah
0	0.153846	1	0.429030	0.932099	0.301038	0	1	0	0	0	0	0	0	0	0	0	0	
1	0.430769	2	0.391379	0.172840	0.512111	1	0	0	0	0	0	0	0	0	0	0	0	
3	0.584615	0	0.000000	0.759259	0.183391	0	1	0	0	0	0	0	0	0	0	0	0	
5	0.015385	1	0.339606	0.932099	0.557093	1	0	0	0	0	0	0	0	0	0	0	0	
8	0.000000	1	0.475600	0.981481	0.072664	0	1	0	0	0	0	0	0	0	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
333465	0.323077	0	0.000000	0.475309	0.034602	1	0	1	0	0	0	0	0	0	0	0	0	
333466	0.292308	0	0.000000	0.759259	0.657439	1	0	1	0	0	0	0	0	0	0	0	0	
333467	0.200000	1	0.000000	0.932099	0.491349	1	0	1	0	0	0	0	0	0	0	0	0	
333468	0.784615	0	0.000000	0.148148	0.747405	0	1	1	0	0	0	0	0	0	0	0	0	
333469	0.184615	0	0.000000	0.154321	0.664360	0	1	1	0	0	0	0	0	0	0	0	0	

212772 rows × 66 columns

- Scaling Data

  Menggunakan metode MinMax yang memakai range nilai 0 sampai 1 untuk mempercepat proses perhitungan pada K-means. Alasan memakai metode ini adalah karena K-means melakukan perhitungan pada jarak antar data.

```
from sklearn.preprocessing import MinMaxScaler

#scaling data dengan minmax scaler agar menghasilkan pada rentang yang sama[0,1]
scaler = MinMaxScaler()
data_gab['Premi'] = scaler.fit_transform(data_gab[['Premi']])
data_gab['Umur'] = scaler.fit_transform(data_gab[['Umur']])
data_gab['Lama_Berlangganan'] = scaler.fit_transform(data_gab[['Lama_Berlangganan']])
data_gab['Kanal_Penjualan'] = scaler.fit_transform(data_gab[['Kanal_Penjualan']])
data_gab
```

	Umur	Umur_Kendaraan	Premi	Kanal_Penjualan	Lama_Berlangganan	kelamin_Pria	kelamin_Wanita	daerah_0.0	daerah_1.0	daerah_2.0	daerah_3.0	daerah_4.0	daerah_5.0	daerah_6.0	daerah_7.0	da
0	0.153846	1	0.429030	0.932099	0.301038	0	1	0	0	0	0	0	0	0	0	
1	0.430769	2	0.391379	0.172840	0.512111	1	0	0	0	0	0	0	0	0	0	
3	0.584615	0	0.000000	0.759259	0.183391	0	1	0	0	0	0	0	0	0	0	
5	0.015385	1	0.339606	0.932099	0.557093	1	0	0	0	0	0	0	0	0	0	
8	0.000000	1	0.475600	0.981481	0.072664	0	1	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
333465	0.323077	0	0.000000	0.475309	0.034602	1	0	1	0	0	0	0	0	0	0	
333466	0.292308	0	0.000000	0.759259	0.657439	1	0	1	0	0	0	0	0	0	0	
333467	0.200000	1	0.000000	0.932099	0.491349	1	0	1	0	0	0	0	0	0	0	
333468	0.784615	0	0.000000	0.148148	0.747405	0	1	1	0	0	0	0	0	0	0	
333469	0.184615	0	0.000000	0.154321	0.664360	0	1	1	0	0	0	0	0	0	0	

212772 rows × 66 columns

3. Permodelan

Permodelan yang dipakai adalah K-means, selanjutnya menggunakan rumus Euclidean untuk menghitung jarak antar centroid dan objek.

- K-Means

  adalah suatu metode penganalisaan data atau metode Data Mining yang melakukan proses pemodelan tanpa supervisi (unsupervised) dan merupakan salah satu metode yang melakukan pengelompokan data dengan sistem partisi.

- Euclidean

  Rumus ini dipakai pada perhitungan jarak antar centroid dan objek.

1.**Euclidean Distance**

$$d(x,y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}$$

ide rumus ini dari rumus pythagoras

$$c = \sqrt{a^2 + b^2}$$

*$d(x,y)$ dibaca *distance* antara x dan y.

```python
#model kmeans dengan bentuk class
class Kmeans:
  def __init__(self, k, max_iter):
    self.k = k
    self.max_iter = max_iter
    self.centroid = []

  #euclidean
  def euclidean(self, x1, x2):
    distance = np.sqrt(((x2-x1)**2).sum(axis=0))
    return distance

  def fit(self,data):

    #label cluster awal diset default menjadi -1 terlebih dahulu
    labels = [-1]*len(data)

    #pemilihan cluster awal secara random sebanyak k
    self.centroid = data.sample(self.k)
    self.centroid = self.centroid.to_numpy()

    data = data.to_numpy()

    for i in range(self.max_iter):
      #nilai inertia untuk menghitung wcss yang berguna untuk elbow method
      self.inertia = 0
      #clusters untuk pengumpulan data berdasarkan cluster, digunakan untuk menghitung mean pada centroid nanti
      clusters = [[] for i in range(self.k)]

      for x in range(len(data)):

        #menghitung jarak tiap data ke tiap centroid
        distance = []
        for indeks in range(len(self.centroid)):
          distance.append(self.euclidean(data[x], self.centroid[indeks]))
        #memasukkan data ke cluster terdekat
        label = distance.index(min(distance))
        clusters[label].append(data[x])
        labels[x] = label

        self.inertia += (min(distance))**2

      #pergantian tiap centroid dengan mean tiap cluster
      temp = np.copy(self.centroid)
      for indeks in range(self.k):
        self.centroid[indeks] = np.mean(clusters[indeks], axis=0)

      #pengecekan kondisi berhenti kmeans, apakah centroid sama dengan sebelumnya atau tidak
      condition = temp == self.centroid
      if condition.all():
        break

    #function fit mengembalikan label cluster tiap data
    return labels
```

4. Evaluasi

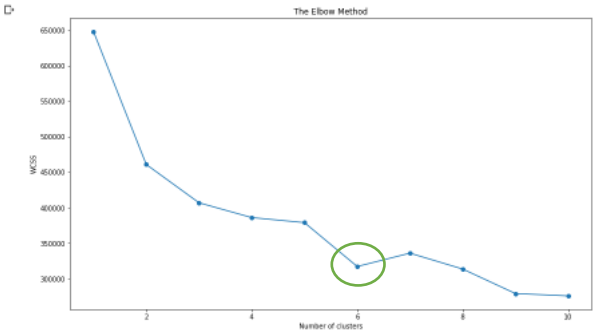
- **Elbow Method**

  Elbow method adalah metoda yang sering dipakai untuk menentukan jumlah cluster yang akan digunakan pada k-means clustering.Seperti yang sudah dibahas sebelumnya, clustering adalah meminimumkan jarak antara data point dan centroid, serta memaksimumkan jarak antara centroid yang dihitung menggunakan within-cluster sum of squares atau WCSS.Tujuannya adalah menghitung WCSS se-minimum dengan jumlah cluster yang kecil agar bisa dilakukan interpretasi data.

```
wcss = []
for i in range(1, 11):

  kmeans = Kmeans(i, 500)
  kmeans.fit(data_gab)
  wcss.append(kmeans.inertia)

plt.figure(figsize=(14,8))
plt.plot(range(1, 11), wcss)
plt.scatter(range(1, 11), wcss)
#memberikan label plot dengan judul The Elbow Method
plt.title('The Elbow Method')
#memberikan label x dengan nama Number of clusters
plt.xlabel('Number of clusters')
#memberikan label y dengan nama WCSS
plt.ylabel('WCSS')
#menampilkan plot
plt.show()
```



Pada diagram diatas dapat disimpulkan bahwa k = 6 adalah jumlah cluster yang paling efektif, akan tetapi yang diambil adalah k = 2 karena hanya mengkelompokkan 2 cluster saja yaitu tertarik dan tidak tertarik.

5. Eksperimen

- Percobaan K-means

1. Memanggil K-means menggunakan parameter K = 2 dan iterasi = 500.
2. Melakukan plot dataset menggunakan Principal Component Analysis(PCA) untuk mereduksi data dan menginterpretasikan feature menjadi 2 dimensi yang berfungsi menampilkan visual plot yang mudah dilihat.

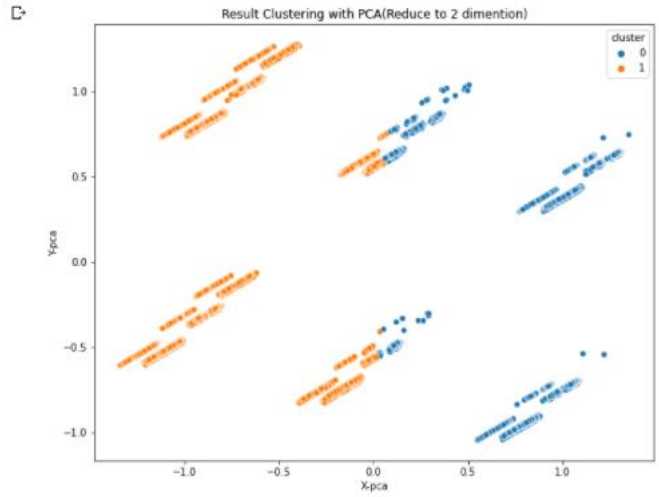
```
#percobaan kmeans dengan k=2 dan max_iterasi = 500
max_iter = 500
k = 2
model_1 = Kmeans(k, max_iter)
label_1 = model_1.fit(data_gab)
```

```
[ ] from sklearn.decomposition import PCA
    #penggunaan PCA untuk mereduksi semua kolom feature menjadi 2 dimensi agar dapat dilihat persebaran cluster dengan baik
    pca = PCA(n_components=2)
    pca = pca.fit_transform(data_gab)
    data_frame = pd.DataFrame()

    data_frame['X-pca'] = pca[:,0]
    data_frame['Y-pca'] = pca[:,1]
    data_frame['cluster'] = label_1
```

- Hasil Percobaan

```
plt.figure(figsize=(10,8))
sns.scatterplot(data=data_frame, x="X-pca", y="Y-pca", hue="cluster")
plt.title('Result Clustering with PCA(Reduce to 2 dimention)')
plt.show()
```



Menghasilkan warna Orange dan biru yang menandakan

Orange(1) : Tertarik

Biru(0) : Tidak Tertarik

6. Kesimpulan

1. Pada proses Labeling & Categorial dilakukan agar data dievaluasi dengan metode K-means.
2. **Scaling data** berfungsi untuk mempercepat proses perhitungan pada K-means dengan menggunakan metode MinMax.
3. Percobaan K-means menggunakan Principal Component Analysis(PCA) untuk mereduksi data dan menginterpretasikan menjadi 2 dimensi yang berfungsi menampilkan visual plot yang mudah dilihat dalam hal sebaran cluster.
4. Elbow method dilakukan menggunakan within-cluster sum of squares atau WCSS.Tujuannya adalah menghitung WCSS se-minimum dengan jumlah cluster yang kecil agar bisa dilakukan interpretasi data untuk mengetahui K mana yang paling efektif dan hasil yang dibuat menunjukan K = 2 adalah yang paling efektif.

Link Video : https://youtu.be/nfQAloOx4G8