CLUSTERING : DATASET KENDARAAN

Observasi

oleh: Muhammad Shulhannur mshulhannur@gmail.com

1. PENGGUNAAN BAHASA PEMROGRAMAN, TOOLS, DAN LIBRARY

Penulis menggunakan bahasa pemrograman Python, karena diwajibkan, dengan menggunakan compiler online Google Colab, karena Google Colab memberikan kemudahan dalam menulis dokumentasi, markdown dan notes. Adapun hasil visualisasi dari seluruh data yang ditampilkan disini, terlampir langsung pada Google Colab. Sebagai disclaimer, penulis menambahkan bahwa kodingan penulis (1301180396_MUHAMMAD SHULHANNR_KENDARAAN.ipynb), beserta pemanggilan URL di dalamnya dapat berjalan dengan lancar menggunakan Google Colab.

Penulis memanggil library yang terdiri atas :

```
#untuk data analysis and manipulation
import pandas as pd

#untuk matematika
import numpy as np

#untuk matematika
import math

#untuk generate random number
import random

#untuk grafik 2D
import matplotlib.pyplot as plt

#untuk visualisasi data ke grafik 2D berbasis matplotlib
import seaborn as sns

#untuk url
import io

#untuk request file from url
import requests
```

2. FORMULASI MASALAH

Diberikan dua buah dataset berjudul ‘kendaraan_train.csv’ dengan 216815 baris 12 kolom, dan ‘kendaraan_test.csv’ dengan 47639 baris 11 kolom. Terhadap kedua dataset tersebut, dilakukan merger/penggabungan, lalu dilakukan klasterisasi atau memodelkan struktur data, supaya data tersebut dapat dipelajari lebih lanjut, serta mengklasifikasikan objek-objek yang sejenis atau berpola sama, dalam area tertentu. Kemudian dilakukan pencarian jumlah kelompok yang sebanyak k yang optimal, supaya mendapatkan hasil yang optimal pula.

3. EKSPLORASI DAN PERSIAPAN DATA

A. Read File Into Dataframe

Penulis melakukan download data secara online dengan pemanggilan URL.

```
# Read kendaraan_train.csv file into DataFrame
url_datatraining="https://cdn.discordapp.com/attachments/756550576640360469/833533583331164190/kendaraan_train.csv"
training=requests.get(url_datatraining).content
datatraining = pd.read_csv(io.StringIO(training.decode('utf-8')))
datatraining

# Read kendaraan_test.csv file into DataFrame
url_datatesting="https://cdn.discordapp.com/attachments/756550576640360469/833533558941024266/kendaraan_test.csv"
testing=requests.get(url_datatesting).content
datatesting = pd.read_csv(io.StringIO(testing.decode('utf-8')))
datatesting
```

B. Lakukan Pengecekkan Missing Data

Melakukan Pengecekkan terhadap adanya values yang bernilai Null/NaN. Data yang bernilai NaN akan berpengaruh pada pemrosesan data karena jika banyak data kosong maka pemrosesan data tidak akan berjalan optimal. Dengan hasil akhir sebagai berikut :

```
datatraining.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
```

```
datatesting.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47639 entries, 0 to 47638
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Jenis_Kelamin      47639 non-null  object
 1   Umur               47639 non-null  int64
 2   SIM                47639 non-null  int64
 3   Kode_Daerah        47639 non-null  int64
 4   Sudah_Asuransi     47639 non-null  int64
 5   Umur_Kendaraan     47639 non-null  object
 6   Kendaraan_Rusak    47639 non-null  object
 7   Premi              47639 non-null  int64
 8   Kanal_Penjualan    47639 non-null  int64
 9   Lama_Berlangganan  47639 non-null  int64
 10  Tertarik           47639 non-null  int64
dtypes: int64(8), object(3)
memory usage: 4.0+ MB
```

C. Drop Unused/Unecesary Column

Melakukan penghapusan kolom yang tidak relevan untuk menghilangkan redundancies.

```
[10] datatraining.drop('id', axis=1, inplace=True)
     datatraining.drop('Tertarik', axis=1, inplace=True)

     datatraining
```

	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
2	NaN	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	32733.0	160.0	119.0
3	Wanita	58.0	1.0	48.0	0.0	1-2 Tahun	Tidak	2630.0	124.0	63.0
4	Pria	50.0	1.0	35.0	0.0	> 2 Tahun	NaN	34857.0	88.0	194.0
...	...	...	...	...	...	...	...	...	...	...
285826	Wanita	23.0	1.0	4.0	1.0	< 1 Tahun	Tidak	25988.0	152.0	217.0
285827	Wanita	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	44686.0	152.0	50.0
285828	Wanita	23.0	1.0	50.0	1.0	< 1 Tahun	Tidak	49751.0	152.0	226.0
285829	Pria	68.0	1.0	7.0	1.0	1-2 Tahun	Tidak	30503.0	124.0	270.0
285830	Pria	45.0	1.0	28.0	0.0	1-2 Tahun	Pernah	36480.0	26.0	44.0

285831 rows × 10 columns

```
[11] datatesting.drop('Tertarik', axis=1, inplace=True)

     datatesting
```

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan
0	Wanita	49	1	8	0	1-2 Tahun	Pernah	46963	26	145
1	Pria	22	1	47	1	< 1 Tahun	Tidak	39624	152	241
2	Pria	24	1	28	1	< 1 Tahun	Tidak	110479	152	62
3	Pria	46	1	8	1	1-2 Tahun	Tidak	36266	124	34
4	Pria	35	1	23	0	1-2 Tahun	Pernah	26963	152	229
...	...	...	...	...	...	...	...	...	...	...
47634	Pria	61	1	46	0	> 2 Tahun	Pernah	31039	124	67
47635	Pria	41	1	15	0	1-2 Tahun	Pernah	2630	157	232
47636	Pria	24	1	29	1	< 1 Tahun	Tidak	33101	152	211
47637	Pria	59	1	30	0	1-2 Tahun	Pernah	37788	26	239
47638	Pria	52	1	31	0	1-2 Tahun	Tidak	2630	124	170

D. Merge Data Testing Dan Data Training

Gabungkan kedua dataset untuk pemangglan prosedur lebih efisien.

```
DataKendaraan = datatraining.append(datatesting, ignore_index=True)

DataKendaraan
```

	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
2	NaN	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	32733.0	160.0	119.0
3	Wanita	58.0	1.0	48.0	0.0	1-2 Tahun	Tidak	2630.0	124.0	63.0
4	Pria	50.0	1.0	35.0	0.0	> 2 Tahun	NaN	34857.0	88.0	194.0
...	...	...	...	...	...	...	...	...	...	...
333465	Pria	61.0	1.0	46.0	0.0	> 2 Tahun	Pernah	31039.0	124.0	67.0
333466	Pria	41.0	1.0	15.0	0.0	1-2 Tahun	Pernah	2630.0	157.0	232.0
333467	Pria	24.0	1.0	29.0	1.0	< 1 Tahun	Tidak	33101.0	152.0	211.0
333468	Pria	59.0	1.0	30.0	0.0	1-2 Tahun	Pernah	37788.0	26.0	239.0
333469	Pria	52.0	1.0	31.0	0.0	1-2 Tahun	Tidak	2630.0	124.0	170.0

333470 rows × 10 columns

E. Handling Missing Value

Lakukan eksperimen terhadap dua data yang sekiranya paing berkorelasi. Lalu isi semua missing value dengan mean dan modus. Kemudian ubah data kategorikal menjadi numerikal

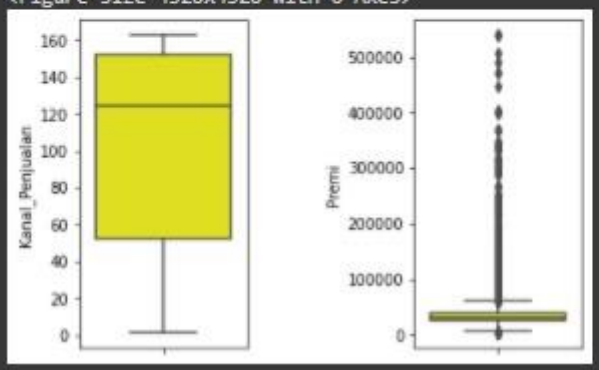
```
[ ]  # UMUR & JENIS KELAMIN => MODUS
     # 24 & PRIA
     DataKendaraan['Umur'].mode()
     DataKendaraan['Jenis_Kelamin'].mode()

[ ]  DataKendaraan["Umur"].replace(np.nan, 24, inplace=True)
     DataKendaraan['Jenis_Kelamin'].replace(np.nan, "Pria", inplace=True)

     DataKendaraan.head()

[ ]  # UMUR KENDARAAN, SIM, SUDAH ASURANSI, KENDARAAN RUSAK => MODUS
     DataKendaraan['Umur_Kendaraan'].mode()

     DataKendaraan['Umur_Kendaraan'].replace(np.nan, "1-2 Tahun", inplace=True)

[ ]  # UMUR KENDARAAN, SIM, SUDAH ASURANSI => MODUS
     DataKendaraan['SIM'].mode()

     DataKendaraan['SIM'].replace(np.nan, 1.0, inplace=True)

[ ]  # UMUR KENDARAAN, SIM, SUDAH ASURANSI => MODUS
     DataKendaraan['Sudah_Asuransi'].mode()

     DataKendaraan['Sudah_Asuransi'].replace(np.nan, 0.0, inplace=True)

[ ]  # UMUR KENDARAAN, SIM, SUDAH ASURANSI => MODUS
     DataKendaraan['Kendaraan_Rusak'].mode()

     DataKendaraan['Kendaraan_Rusak'].replace(np.nan, "Pernah", inplace=True)

[ ]  # UMUR KENDARAAN, SIM, SUDAH ASURANSI => MODUS
     DataKendaraan['Kode_Daerah'].mode()

     DataKendaraan['Kode_Daerah'].replace(np.nan, 28.0, inplace=True)

[ ]  # PREMI KANAL_PENJUALAN LAMA_BERLANGGANAN => MEAN

     avg_premi = DataKendaraan['Premi'].astype('float').mean(axis=0)
     avg_Kanal = DataKendaraan['Kanal_Penjualan'].astype('float').mean(axis=0)
     avg_Berlangganan = DataKendaraan['Lama_Berlangganan'].astype('float').mean(axis=0)

     DataKendaraan['Premi'].replace(np.nan, avg_premi, inplace=True)
     DataKendaraan['Kanal_Penjualan'].replace(np.nan, avg_Kanal, inplace=True)
     DataKendaraan['Lama_Berlangganan'].replace(np.nan, avg_Berlangganan, inplace=True)

[ ]  missing_data = DataKendaraan.isnull()

[ ]  for column in missing_data.columns.values.tolist():
       print(column)
       print(missing_data[column].value_counts())
       print("")
```

F. Lakukan Pengecekkan Pencilan

Visualisasikan dengan boxplot, dimana outliers terdapat.

```
[29] def check_outlier(data):
       plt.figure(figsize=(60, 60))
       f, axes = plt.subplots(1, 2)
       sns.boxplot(y= data['Kanal_Penjualan'], ax= axes[0], color='yellow')
       sns.boxplot(y= data['Premi'], ax=axes[1], color='yellow')
       plt.subplots_adjust(wspace=1)

     check_outlier(dataCluster)
```

```
<Figure size 4320x4320 with 0 Axes>
```



G. Handle Outliers Dengan Data Cleansing

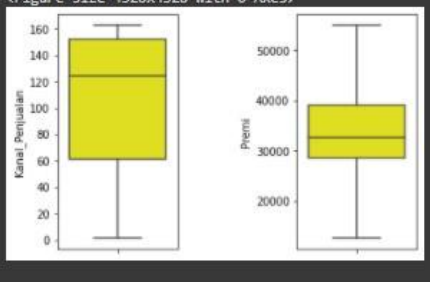
Lakukan data cleansing, lalu cek kembali apakah ada outliers pada premi

```
[30] #HANDLE OUTLIER PREMI
     while True:
       qlo1, qlo3 = np.percentile(dataCluster['Premi'],[25,75])
       iqrlo = qlo3 - qlo1
       lowerlo = qlo1 - (1.5 * iqrlo)
       upperlo = qlo3 + (1.5 * iqrlo)
       outlierlo = dataCluster[(dataCluster['Premi'] < (lowerlo)) | (dataCluster['Premi'] > (upperlo))]
       print('amount of outlier data',outlierlo.shape[0]) #JUMLAH OUTLIER DATA
       idxlo = outlierlo.index
       dataCluster.drop(idxlo, inplace=True) #DROP OUTLIER DATA
       if (outlierlo.shape[0] <= 0):
         break

     dataCluster['Premi'].describe()
```

```
amount of outlier data 64480
amount of outlier data 4132
amount of outlier data 1247
amount of outlier data 405
amount of outlier data 125
amount of outlier data 40
amount of outlier data 12
amount of outlier data 1
amount of outlier data 0
count    263028.000000
mean      34017.495005
std        7986.237800
min       12470.000000
25%       28458.000000
50%       32642.000000
75%       39123.000000
max       55120.000000
Name: Premi, dtype: float64
```

```
[31] def check_outlier_encode(data):
       plt.figure(figsize=(60, 60))
       f, axes = plt.subplots(1, 2)
       sns.boxplot(y=data['Kanal_Penjualan'], ax= axes[0], color='yellow')
       sns.boxplot(y= data['Premi'], ax=axes[1], color='yellow')
       plt.subplots_adjust(wspace=1)

     check_outlier_encode(dataCluster)

     dataCluster.to_csv(r'/content/data_ready.csv', index=False, header=True)
```

```
<Figure size 4320x4320 with 0 Axes>
```



H. Normalisasikan Tabel

Lakukan data cleansing, lalu cek kembali apakah ada outliers pada premi

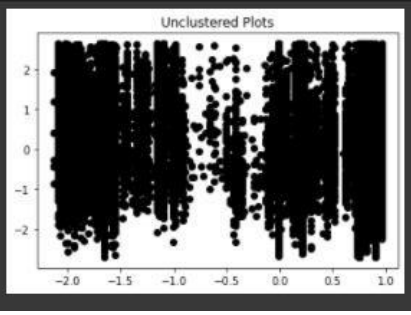
Normalization

```
[32] from sklearn.preprocessing import StandardScaler

     normalize = StandardScaler()
     df = normalize.fit_transform(dataCluster)

     df
```

```
array([[ 0.74905745, -0.74985325],
       [-1.59773366, -1.02895892],
       [ 0.90169427, -0.16083887],
       ...,
       [ 0.21482858, -0.37295417],
       [ 0.74905745, -0.11475951],
       [-1.65497247,  0.47212621]])
```

4. PERMODELAN A. Generate Unclustered Plot

```
[33] plt.scatter(df[:,0],df[:,1],c='black',label='unclustered data')

     plt.title('Unclustered Plots')

     plt.show()
```

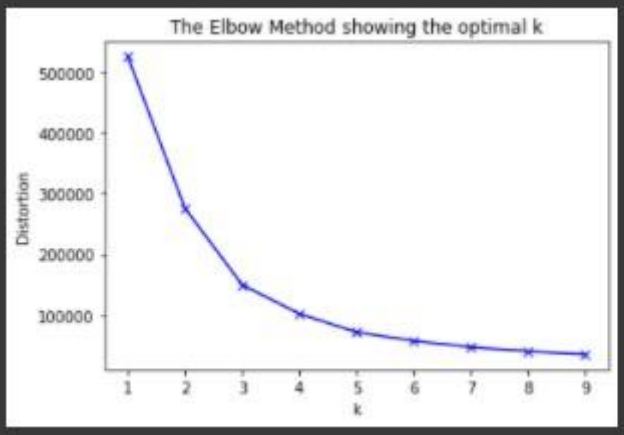


B. Buat K-Means Functions

```
def k_means(x,k, no_iterations):
    idx = np.random.choice(len(x), k)
    EuclidianDistance = np.array([]).reshape(x.shape[0],0)
    #PILIH CENTROID SECARA RANDOM
    centroids = x[idx, :]

    #MENCARI JARAK ANTARA TIAP CENTROID DAN SEMUA DATA POINT
    for i in range(k):
      tempDist = np.sum((x-centroids[i])**2,axis = 1)
      EuclidianDistance = np.c_[EuclidianDistance, tempDist]

    points = np.array([np.argmin(i) for i in EuclidianDistance])

    #MENGULANGI CARA DIATAS SEBANYAK ITERASI YANG DIINPUT
    for _ in range(no_iterations):
        centroids = []
        EuclidianDistance = np.array([]).reshape(x.shape[0],0)
        for idx in range(k):
            #MELAKUKAN UPDATE CENTROID DENGAN MENGAMBIL RATA RATA CLUSTER
            temp_cent = x[points==idx].mean(axis=0)
            centroids.append(temp_cent)

        centroids = np.vstack(centroids) #UPDATE CENTROID

        for i in range(k):
          tempDist = np.sum((x-centroids[i])**2,axis = 1)
          EuclidianDistance = np.c_[EuclidianDistance, tempDist]

        points = np.array([np.argmin(i) for i in EuclidianDistance])

    return points, centroids
```

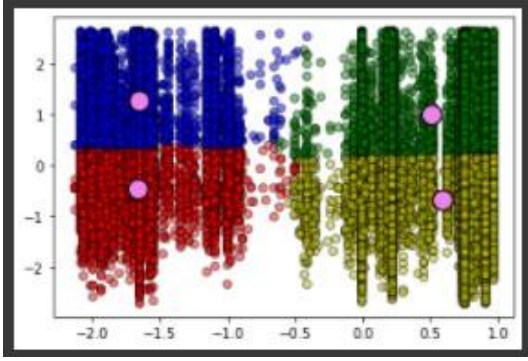
C. Buat Elbow Method

```
[35] from sklearn.cluster import KMeans #HANYA UNTUK ELBOW METHOD
     distortions = []
     K = range(1,10)
     for k in K:
         kmeanModel = KMeans(n_clusters=k, max_iter=300)
         kmeanModel.fit(df)
         distortions.append(kmeanModel.inertia_)

     plt.plot(K, distortions, 'bx-')
     plt.xlabel('k')
     plt.ylabel('Distortion')
     plt.title('The Elbow Method showing the optimal k')
     plt.show()
```



D. Lakukan Plotting Hasil K-Means

```
#DARI ELBOW METHOD SEBELUMNYA K TERBAIK ADALAH 4
label, centroid = k_means(df,4,300)
label_centre = ['centroid 1', 'centroid 2', 'centroid 3', 'centroid 4']
colors=['r', 'g', 'b', 'y', 'teal']
j = 0

u_labels = np.unique(label)

for i in u_labels:
    plt.scatter(df[label == i , 0] , df[label == i , 1] , c = colors[i], linewidths=1 ,alpha=0.5, edgecolors= 'k', label = i )

for centre in centroid:
    plt.scatter(centroid[: , 0] , centroid[: , 1] , s=200, c = 'violet',linewidths=1, edgecolors= 'k', label = label_centre[j])
    j += 1

plt.show()
```



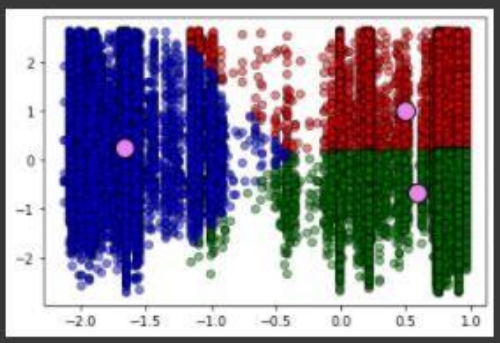
E. Lakukan Plotting Hasil K-Means

```
[37] from sklearn.metrics import silhouette_score
     sil_score = []

     for n_cluster in range(3, 7):
         kmeans = KMeans(n_clusters=n_cluster).fit(df)
         label = kmeans.labels_
         sil_coeff = silhouette_score(df, label, metric='euclidean')
         sil_score.append(sil_coeff)
         print('Nilai Silhoutte Method untuk n_clusters = {} adalah {}'.format(n_cluster, sil_coeff))

     Nilai Silhoutte Method untuk n_clusters = 3 adalah 0.5160827954857263
     Nilai Silhoutte Method untuk n_clusters = 4 adalah 0.5281681256655532
     Nilai Silhoutte Method untuk n_clusters = 5 adalah 0.46079546795069876
     Nilai Silhoutte Method untuk n_clusters = 6 adalah 0.4479244314730548
```

5. OBSERVASI A. Lakukan Eksperimen Menggunakan Nilai K Yang Didapat

K = 3, TETAPI DILAKUKAN DATA CLEANSING

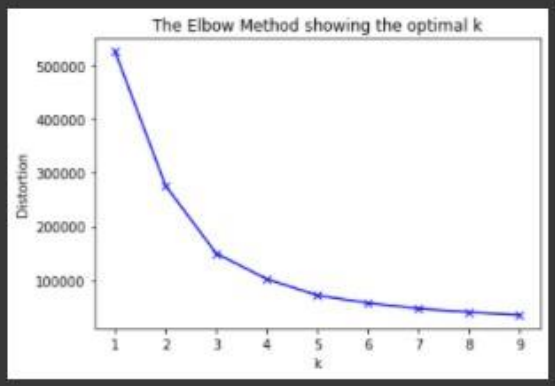
```
label, centroid = k_means(df,3,300)
label_centre = ['centroid 1', 'centroid 2', 'centroid 3', 'centroid 4']
colors=['r', 'g', 'b', 'y', 'teal']
j = 0

u_labels = np.unique(label)

for i in u_labels:
    plt.scatter(df[label == i , 0] , df[label == i , 1] , c = colors[i], linewidths=1 ,alpha=0.5, edgecolors= 'k', label = i )

for centre in centroid:
    plt.scatter(centroid[: , 0] , centroid[: , 1] , s=200, c = 'violet',linewidths=1, edgecolors= 'k', label = label_centre[j])
    j += 1

plt.show()
```



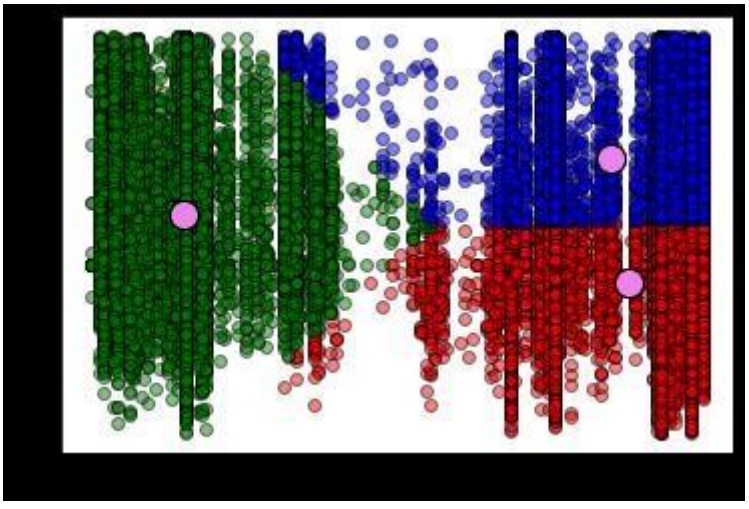
K = 3

```
[39] from sklearn.preprocessing import StandardScaler

     normalize = StandardScaler()
     dataWithOutlier = normalize.fit_transform(dataWithOutlier)
```

```
[40] from sklearn.cluster import KMeans #HANYA UNTUK ELBOW METHOD
     distortions = []
     K = range(1,10)
     for k in K:
         kmeanModel = KMeans(n_clusters=k, max_iter=300)
         kmeanModel.fit(dataWithOutlier)
         distortions.append(kmeanModel.inertia_)

     plt.plot(K, distortions, 'bx-')
     plt.xlabel('k')
     plt.ylabel('Distortion')
     plt.title('The Elbow Method showing the optimal k')
     plt.show()
```




```
label, centroid = k_means(dataWithOutlier,3,300)
label_centre = ['centroid 1', 'centroid 2', 'centroid 3', 'centroid 4']
colors=['r', 'g', 'b', 'y', 'teal']
j = 0

u_labels = np.unique(label)

for i in u_labels:
    plt.scatter(df[label == i , 0] , df[label == i , 1] , c = colors[i], linewidths=1 ,alpha=0.5, edgecolors= 'k', label = i )

for centre in centroid:
    plt.scatter(centroid[: , 0] , centroid[: , 1] , s=200, c = 'violet',linewidths=1, edgecolors= 'k', label = label_centre[j])
    j += 1

plt.show()
```



K = 4

```
[42] label, centroid = k_means(dataWithOutlier,4,300)
     label_centre = ['centroid 1', 'centroid 2', 'centroid 3', 'centroid 4']
     colors=['r', 'g', 'b', 'y', 'teal']
     j = 0

     u_labels = np.unique(label)

     for i in u_labels:
         plt.scatter(df[label == i , 0] , df[label == i , 1] , c = colors[i], linewidths=1 ,alpha=0.5, edgecolors= 'k', label = i )

     for centre in centroid:
         plt.scatter(centroid[: , 0] , centroid[: , 1] , s=200, c = 'violet',linewidths=1, edgecolors= 'k', label = label_centre[j])
         j += 1

     plt.show()
```