LAPORAN TUGAS BESAR
MATA KULIAH PEMBELAJARAN MESIN
K-MEANS CLUSTERING

Oleh :

Imam Rafiif Arrazaan (1301194152)

Prodi S1 Informatika

Fakultas Informatika

Telkom University 2021

Daftar Isi

1. Rumusan Masalah 3
2. Eksplorasi dan Data Preprocessing 3
   a. Eksplorasi 3
   b. Data Preprocessing 5
3. Pemodelan 8
4. Evaluasi 11
5. Eksperimen 12
6. Hasil 13
7. Kesimpulan 13
Link 13

1. Rumusan Masalah

Dataset yang diberikan untuk dianalisis memiliki beberapa fitur. Data tersebut akan dianalisis menggunakan metode *clustering* dengan algoritma K-Means untuk mengelompokkan pembeli berdasarkan umur dan lama berlangganan. Fitur-fitur tersebut dipilih karena memiliki *variance* yang luas. Dengan demikian, pemilihan dua fitur tersebut mengakibatkan data memiliki persebaran yang beragam dan *cluster* yang dihasilkan akan lebih baik.

2. Eksplorasi dan Data Preprocessing

a. Eksplorasi

Pada bagian eksplorasi data, saya memproses data menggunakan library pandas, dengan begitu kita dapat menggunakan method info dan describe untuk mendapatkan ringkasan dari data kendaraan yang sudah didownload dari google drive. Dapat dilihat hasil eksplorasi data menggunakan method info, sebagai berikut:

```
1 df.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
```

Dari info tersebut, dapat diketahui, bahwa ada sebanyak 285831 record dan 12 fitur pada dataset kendaraan. Tipe data yang digunakan pada dataset kendaraan ada 3 jenis yaitu, float64, int64, dan object. Selain itu, dapat diketahui juga fitur-fitur yang tersedia, sebagai berikut:

- Id
- Jenis_Kelamin
- Umur
- SIM
- Kode_Daerah
- Sudah_Asuransi

- Umur_Kendaraan
- Premi
- Kanal_Penjualan
- Lama_Berlangganan
- Tertarik

Selain menggunakan method info, saya juga menggunakan method describe untuk menampilkan beberapa informasi seperti: jumlah data tiap fiturnya, mean, standar deviasi, nilai minimum, nilai maximum, dan nilai kuartil. Sebelum menggunakan method describe, saya menghilangkan data yang hilang pada fitur-fitur yang bertipe kategorikal atau nominal karena data bertipe kategorikal atau nominal akan menjadi buruk jika dilakukan impute data. Dapat dilihat hasil eksplorasi data menggunakan method info, sebagai berikut:

```
1 df1.describe()
```

	id	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
count	209929.000000	199496.000000	209929.000000	209929.000000	209929.000000	209929.000000	199144.000000	209929.000000	199674.000000	209929.000000
mean	142986.942004	38.830277	0.997852	26.401750	0.459465	0.641788	30504.896597	112.183510	154.277152	0.122056
std	82485.405094	15.537280	0.046301	13.248741	0.498355	0.780725	17121.859687	54.142769	83.793190	0.327351
min	1.000000	20.000000	0.000000	0.000000	0.000000	0.000000	2630.000000	1.000000	10.000000	0.000000
25%	71557.000000	25.000000	1.000000	15.000000	0.000000	0.000000	24370.750000	29.000000	81.000000	0.000000
50%	143114.000000	36.000000	1.000000	28.000000	0.000000	1.000000	31618.000000	139.000000	154.000000	0.000000
75%	214345.000000	49.000000	1.000000	35.000000	1.000000	1.000000	39333.250000	152.000000	227.000000	0.000000
max	285831.000000	85.000000	1.000000	52.000000	1.000000	3.000000	540165.000000	163.000000	299.000000	1.000000

Dari tabel tersebut juga dapat terbukti, bahwa masih ada data yang hilang, karena adanya perbedaan jumlah data pada tiap fiturnya.

b. Data Preprocessing

- Missing Values Handling

```
[ ]  1 df.isna().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
```

```
1 df1 = df.dropna(subset = ['Jenis_Kelamin', 'SIM', 'Kode_Daerah', 'Sudah_Asuransi', 'Kendaraan_Rusak', 'Kanal_Penjualan'])
2 df1.isna().sum()
```

```
id                    0
Jenis_Kelamin         0
Umur              10433
SIM                   0
Kode_Daerah           0
Sudah_Asuransi        0
Umur_Kendaraan    10491
Kendaraan_Rusak       0
Premi             10785
Kanal_Penjualan       0
Lama_Berlangganan 10255
Tertarik              0
dtype: int64
```

Dari gambar di atas, dapat diketahui, ada banyak data yang hilang atau null. Agar proses pemodelan dapat berjalan efektif dan data menjadi berkualitas. Maka cara handle yang saya gunakan adalah menghapus atau menghilangkan data null yang bertipe kategorikal. Dan untuk data numerical, saya menggunakan metode impute mean untuk menjaga keaslian data. Hal ini diperlukan agar pemodelan yang akan dilakukan masih berkualitas.

Sebelum dilakukan impute mean kepada data numerical yang hilang atau null, diperlukan melakukan label encoding kepada fitur Umur_Kendaraan agar bisa dicari nilai mean nya lalu dilakukan impute mean.

```
[ ]  1 labelencoder = LabelEncoder()
     2 df1['Umur_Kendaraan'] = labelencoder.fit_transform(df1['Umur_Kendaraan'])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

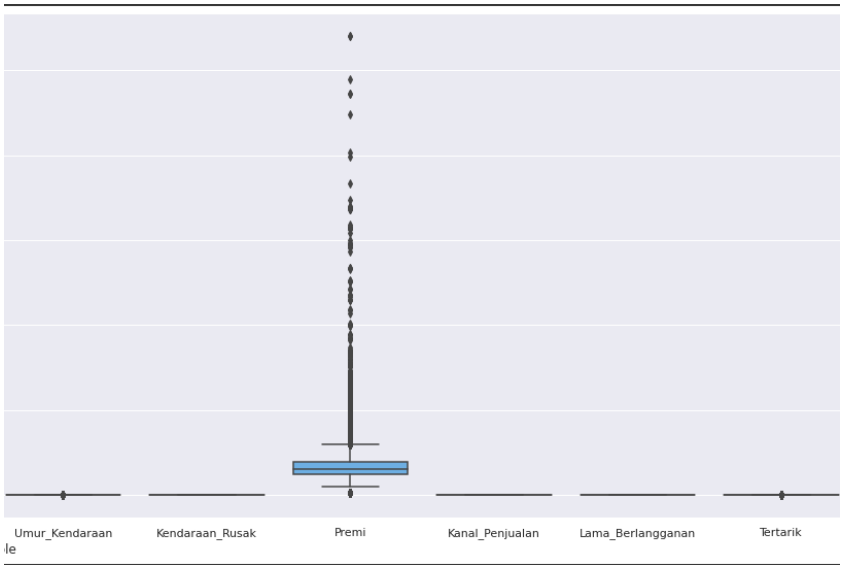
```
[ ]  1 df1.fillna(df1.mean(), inplace=True)

/usr/local/lib/python3.7/dist-packages/pandas/core/series.py:4536: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  downcast=downcast,
```

```
1 df1.isna().sum()
```

```
id                  0
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 melakukan label encoding kepada fitur Umur_Kendaraan, dapat dilakukan impute mean kepada semua fitur yang numerical.

Dari gambar di atas dapat diketahui bahwa tidak ada lagi record yang tidak memiliki nilai.



- Outliers Handling

Outliers handling dilakukan agar persebaran data tidak terlalu extreme yang dapat mengganggu atau memperburuk hasil pemodelan. Pertama-tama saya membentuk box plot untuk melihat apakah ada outliers di dataset kendaraan. Dan hasilnya ada beberapa outliers pada fitur premi. Saya menggunakan algoritma zscore sebagai algoritma outliers handling.

```
1 from scipy import stats
2 z = np.abs(stats.zscore(df1))
3 print(z)
```

```
[[1.73347373 1.08522316 0.58300134 ... 0.73539987 0.70088899 0.37285953]
 [1.7334616  0.92146946 0.60541255 ... 1.53637706 0.04555575 0.37285953]
 [1.73343736 1.08522316 1.26564249 ... 0.2182474  1.11694015 0.37285953]
 ...
 [1.73172937 1.08522316 1.0451623  ... 0.73539987 0.87765807 0.37285953]
 [1.73174149 0.92146946 1.92587242 ... 0.2182474  1.41607722 0.37285953]
 [1.73175361 0.92146946 0.40734356 ... 1.59178626 1.34943933 0.37285953]]
```

```
[ ] 1 threshold = 3
    2 print(np.where(z > 3))
```

```
(array([     4,     34,     46, ..., 209866, 209869, 209870]), array([6, 6, 6, ..., 6, 6, 3]))
```

```
[ ] 1 df2 = df1[(z < 3).all(axis=1)]
```

```
[ ] 1 print(df1.shape)
    2 print(df2.shape)
```

```
(209929, 12)
(197896, 12)
```

Dapat dilihat bahwa setelah dilakukan missing values handling dan outliers handling, banyak data yang asalnya 285831 records menjadi 197896 records.

- Normalization

Sebelum dilakukan pemodelan, saya melakukan normalisasi data agar skala dari data yang diambil tidak terlalu besar. Saya menggunakan Standard Scaler untuk melakukan normalisasi data

```
[ ] 1 from sklearn.preprocessing import StandardScaler
    2 sc_X = StandardScaler()
    3 dfn = sc_X.fit_transform(df3)
```

3. Pemodelan

Metode pemodelan yang digunakan adalah k-Means clustering dimana metode tersebut menggunakan konsep centroid. Saya membentuk model K-Means dengan membuat class K-Means yang berisi atribut dan method yang akan digunakan untuk *clustering* ini.

```python
class KMeans:
    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 indeks sampel untuk setiap cluster
        self.clusters = [[] for _ in range(self.K)]
        # pusat (vektor fitur rata-rata) untuk setiap cluster
        self.centroids = []

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

        # initialize
        random_sample_idxs = np.random.choice(self.n_samples, self.K, replace=False)
        self.centroids = [self.X[idx] for idx in random_sample_idxs]

        # Optimize clusters
        for _ in range(self.max_iters):
            # Assign samples to closest centroids (create clusters)
            self.clusters = self._create_clusters(self.centroids)

            if self.plot_steps:
                self.plot()

            # Tetapkan sampel ke centroid terdekat (buat cluster)
            centroids_old = self.centroids
            self.centroids = self._get_centroids(self.clusters)

            # periksa apakah cluster telah berubah
            if self._is_converged(centroids_old, self.centroids):
                break

            if self.plot_steps:
                self.plot()

        # Klasifikasikan sampel sebagai indeks cluster-nya
        return self._get_cluster_labels(self.clusters)
```

```python
    def _get_cluster_labels(self, clusters):
        # setiap sampel akan mendapatkan label dari cluster yang ditugaskan
        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):
        # Tetapkan 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):
        # 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):
        # tetapkan 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 setiap centroid lama dan baru, untuk semua centroid
        distances = [
            euclidean_distance(centroids_old[i], centroids[i]) for i in range(self.K)
        ]
        return sum(distances) == 0
```

Berikut adalah gambaran singkat algoritma dari k-Means,

Algorithm 8.1 Basic K-means algorithm.

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.

Berdasarkan gambar di atas, hal pertama yang dilakukan adalah menginisialisasi titik K sebagai centroid. Inisialisasi centroid ini dilakukan dengan mengambil secara acak titik yang ada di sampel acak.

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

    # initialize
    random_sample_idxs = np.random.choice(self.n_samples, self.K, replace=False)
    self.centroids = [self.X[idx] for idx in random_sample_idxs]
```

Berikutnya adalah mulai melakukan iterasi menetapkan setiap titik yang terbentuk kepada titik centroid yang paling dekat dengan titik tersebut. Pada proses pencarian titik terdekat dengan titik centroid, saya menggunakan metode Euclidean distance.

```
1 np.random.seed(42)
2
3 def euclidean_distance(x1, x2):
4     return np.sqrt(np.sum((x1 - x2) ** 2))
```

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

Selanjutnya adalah tahapan recompute centroid. Centroid akan terus berubah hingga mendapatkan titik yang stabil, proses ini menggunakan mean dari jarak antara titik-titik data terhadap centroidnya sebagai algoritma untuk mendapatkan centroid yang baru.

```
65    def _get_centroids(self, clusters):
66        # tetapkan nilai rata-rata cluster ke centroid
67        centroids = np.zeros((self.K, self.n_features))
68        for cluster_idx, cluster in enumerate(clusters):
69            cluster_mean = np.mean(self.X[cluster], axis=0)
70            centroids[cluster_idx] = cluster_mean
71        return centroids
```

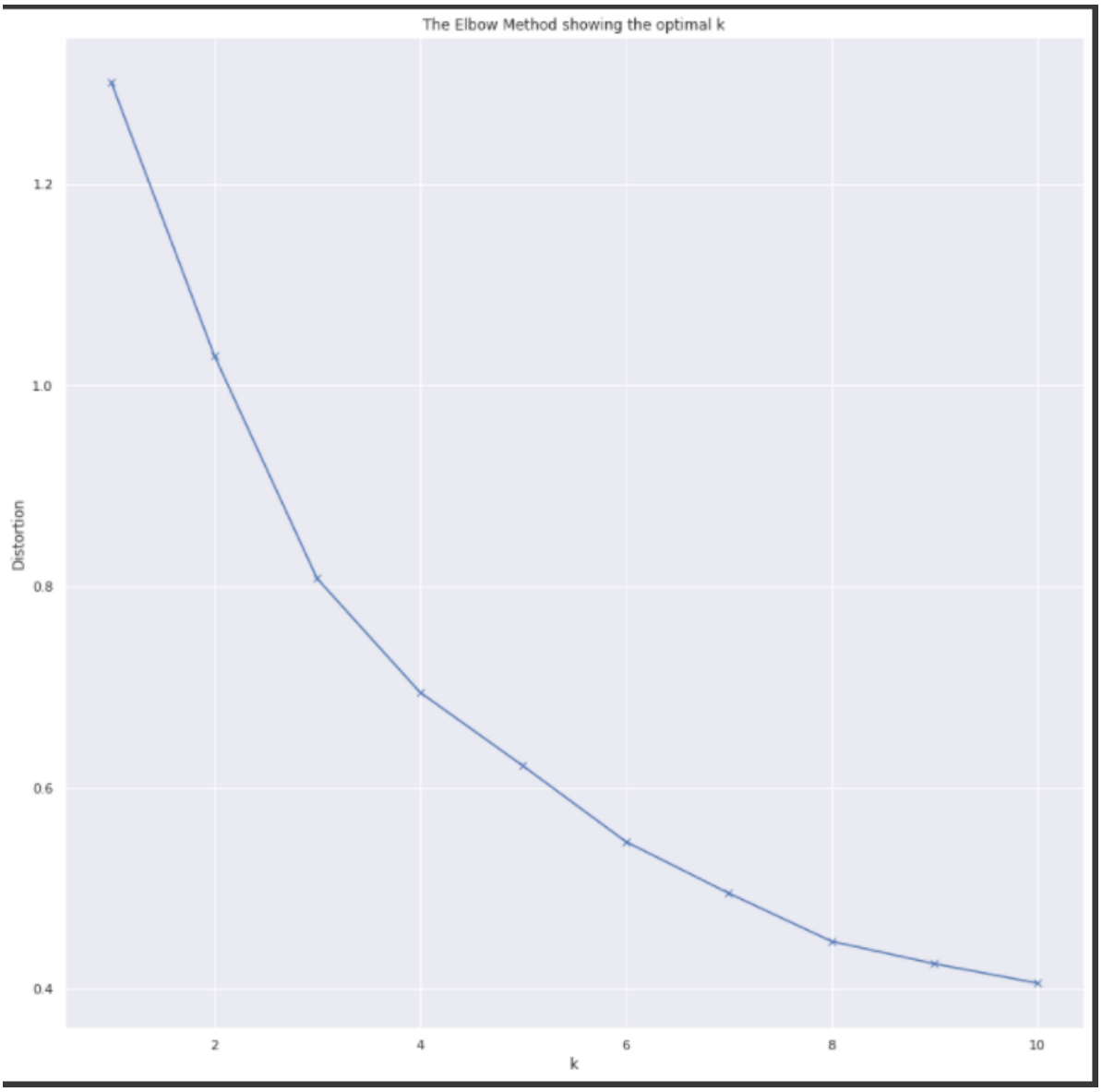
Tahap-tahap tersebut dilakukan untuk centroid setiap klaster. Penetapan klaster dilakukan hingga semua titik centroid tidak berubah. Namun hal itu akan membutuhkan banyak waktu, sehingga saya membatasi jumlah iterasi, maksimal sebanyak 150 kali. *Clustering* juga akan berhenti jika semua posisi centroidnya tidak berubah.

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

4. Evaluasi

Metode evaluasi yang saya gunakan adalah Elbow Method. Elbow Method biasanya digunakan untuk menentukan jumlah K yang optimal. Berikut adalah hasil Elbow Method.

```
from scipy.spatial.distance import cdist
distortions = []

for i in range(1,11):
    k = KMeans(K=i, max_iters=150, plot_steps=False)
    y_pred = k.predict(dfn)
    print(k.centroids)
    distortions.append(sum(np.min(cdist(dfn, k.centroids), axis=1)) / dfn.shape[0])
```



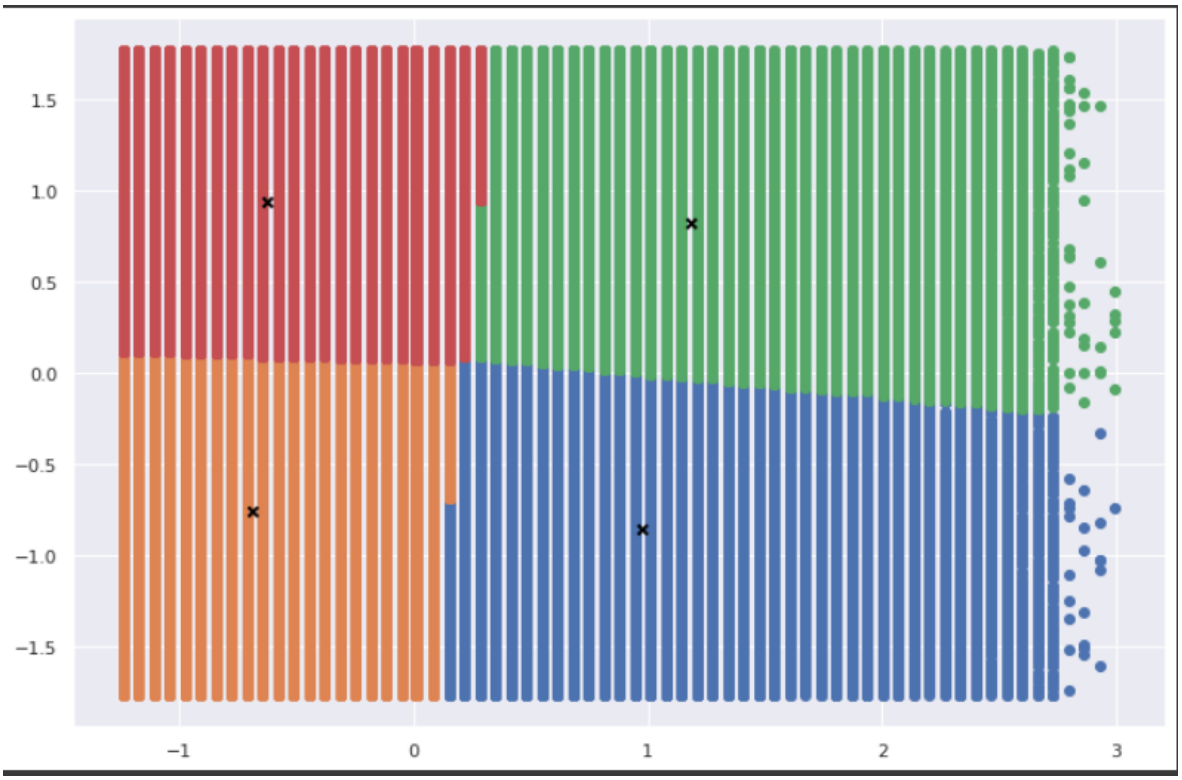
Dari gambar di atas, dapat dilihat, hasil nilai Distortion dari tiap klaster dan centroid-nya. Padagrafik di atas, nilai K mulai stabil pada K = 4, sehingga nilai K yang paling optimum adalah 4.

5. Eksperimen

Pada percobaan ini, saya hanya melakukan 2 kali run program untuk melihat perbedaan hasil clustering pertama dan kedua. Saya juga menggunakan teknik PCA, PCA atau yang bisa disebut sebagai Principal Component Analysis adalah sebuah metode yang mereduksi dimensi suatu data. Dengan menggunakan PCA, hasil clustering akan beragam. Dari hasil tersebut dapat diketahui bahwa dengan teknik preprocessing dan pemilihan fitur yang berbeda dapat menghasilkan hasil clustering yang berbeda juga.

6. Hasil

Hasil yang didapatkan setelah menjalankan program sebanyak 2 kali dengan nilai K 4, yaitu:



7. Kesimpulan

Kesimpulan yang didapatkan dari percobaan clustering ini adalah, sebagai berikut:

- Teknik preprocessing akan berdampak pada hasil clustering.
- Pemilihan fitur yang ada juga akan berdampak signifikan kepada hasil *clustering.*
- Penggunaan algoritma yang berbeda akan berdampak kepada waktu yang dibutuhkan untuk menjalankan program.
- Banyak data yang ada juga akan berpengaruh kepada waktu yang dibutuhkan untuk menjalankan program.
- K-Means Clustering cocok digunakan untuk data dengan bentuk persebaran yang luas dan beragam.

Link:

- Google Colaboratory :https://colab.research.google.com/drive/1tnZXgVtJLV2-EFQRYGKUVWXDighu8rss?usp=sharing

- Video Presentasi : https://youtu.be/1hOYDHRLoRk