Laporan Pembelajaran Mesin (Praktikum)

Disusun oleh :
Kelompok 3

1. Gede Rangga Wira Aditya (082011633048)
2. Muhammad Rahmadhani Ferdiansyah (082011633068)
3. Arya Danu Triatmodjo (082011633069)
4. Mukhamad Ikhsanudin (082011633086)

Kelas I3

PROGRAM STUDI S1 SISTEM INFORMASI

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS AIRLANGGA

SURABAYA

2021/2022

K-MEANS CLUSTERING

(IN PYTHON)

Clustering is a set of techniques used to partition data into groups, or clusters. It is often used as a data analysis technique for discovering interesting patterns in data, such as groups of customers based on their behavior. Clusters are loosely defined as groups of data objects that are more similar to other objects in their cluster than they are to data objects in other clusters.

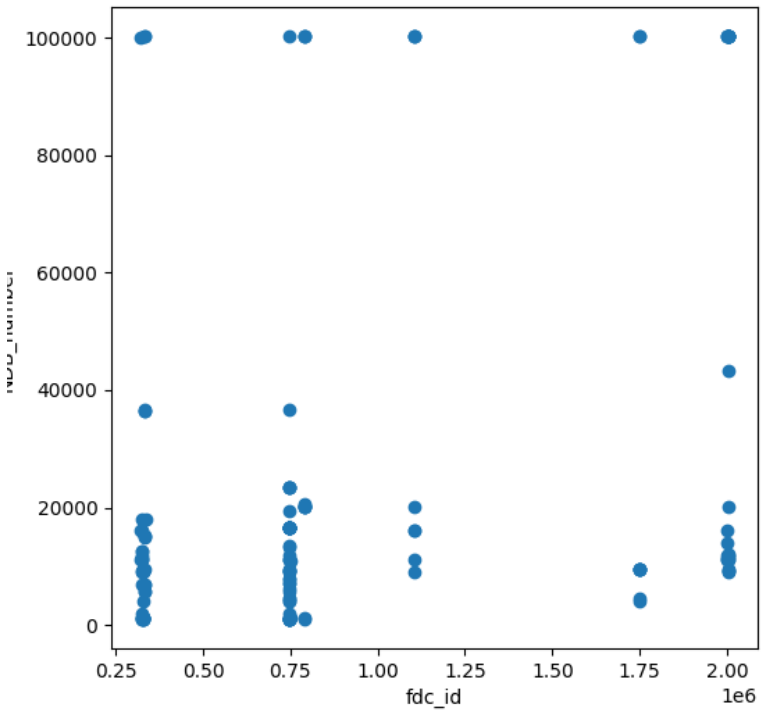
K-Means is one of the clustering methods that requires only a few steps. The first step is to randomly select k centroids, where k is equal to the number of clusters we choose. Centroids are data points representing the center of a cluster. We can use the scikit-learn library which is implemented in sklearn.cluster.KMeans.

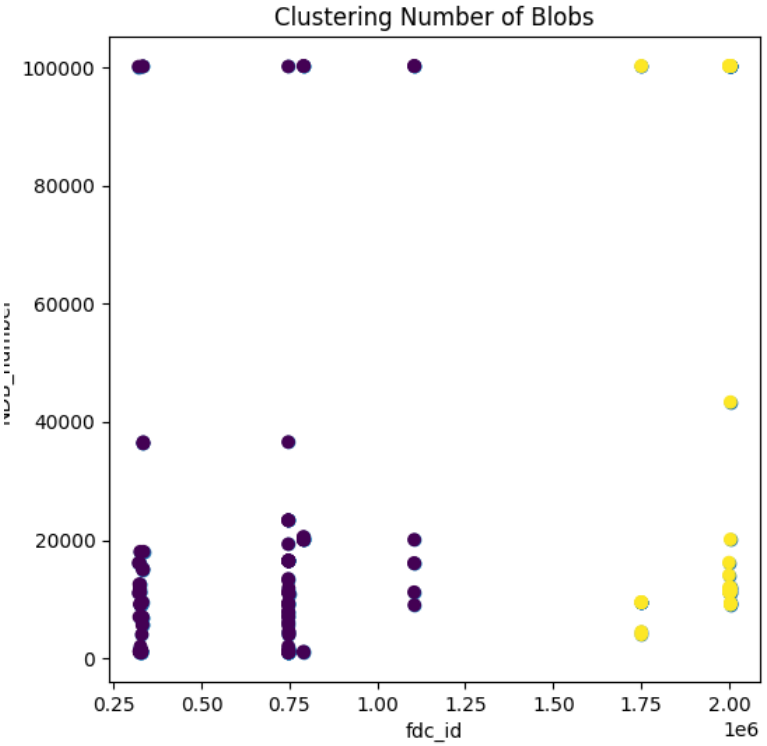
1. **Generate Data**

```
RangeIndex: 159 entries, 0 to 158
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   fdc_id      159 non-null    int64
 1   NDB_number  159 non-null    int64
dtypes: int64(2)
memory usage: 2.6 KB
   fdc_id  NDB_number
0  321358       16158
1  321360      100147
2  321611       11056
3  323121        7022
4  323294       12563
```

We use the foundation_food dataset then import the.csv file into the program

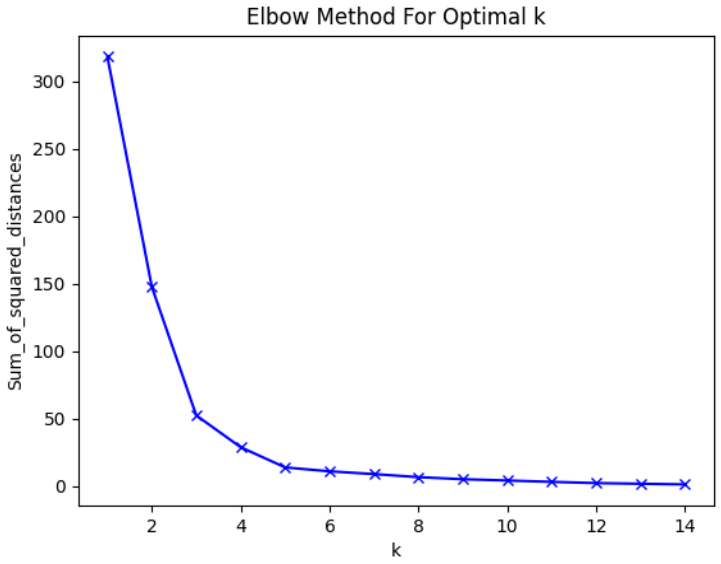
2. **Clustering K-Means**





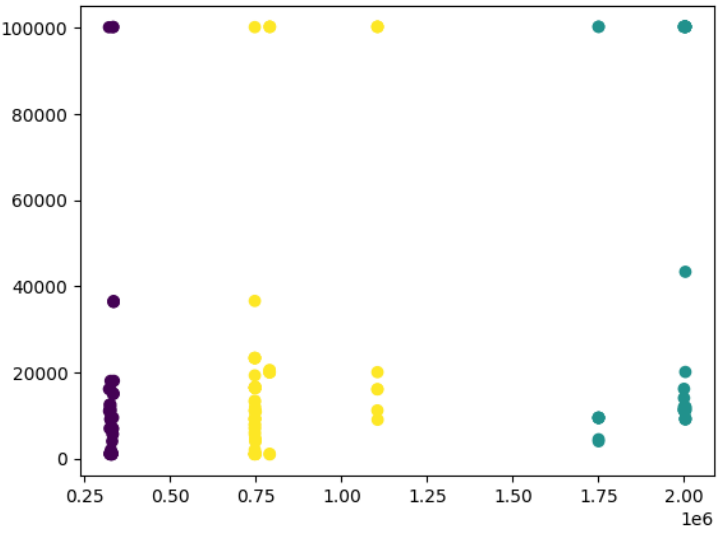
To see the distribution of data before clustering, we can first create a scatter plot. To get optimal clustering results, we need to do data scaling. The purpose of this data scaling is to make the data fall within a range that is not too far away.

3. **Elbow Method**



Then we determine the number of clusters that will be used in k-means using the Elbow method

4. **Clustering with the best K (K=3)**



Define the y_pred variable as K-Means with the desired number of clusters/groups, for example 3 pieces. Then in the y_predicted section, we do the clustering of the distribution of data on nbd_number, then store the results in the y_pred variable.

5. **Evaluation with Confusion matrix**

```
  ============CONFUSION MATRIX============
[[15958  9164]
 [    0     0]]
```

Then, evaluate the result with confusion matrix

6. Syntax

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics.cluster import pair_confusion_matrix
from sklearn.preprocessing import StandardScaler

# Generate Data / Read Data
df = pd.read_csv("foundation_food.csv")
df = df.iloc[:,0:2]
df.info()
print(df.head())

plt.figure(figsize=(6, 6))
plt.scatter(df['fdc_id'],df['NDB_number'] )
plt.xlabel('fdc_id')
plt.ylabel('NDB_number')
plt.show()

scaler = StandardScaler()
scaler.fit(df)
df_scaled = scaler.transform(df)
df_scaled = pd.DataFrame(df_scaled, columns=['fdc_id','NDB_number'])

# Try with k=2
y_pred = KMeans(n_clusters=2).fit_predict(df)
plt.scatter(df['fdc_id'],df['NDB_number'], c=y_pred)
plt.title("Clustering Number of Blobs")
plt.show()

# Trying to find the best number of clusters
Sum_of_squared_distances = []
K = range(1,15)
for k in K:
 km = KMeans(n_clusters=k)
 km = km.fit(df_scaled[['fdc_id','NDB_number']])
 Sum_of_squared_distances.append(km.inertia_)

# Elbow Method
plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```

```
# Clustering with the best k value
y_pred = KMeans(n_clusters=3).fit_predict(df)
plt.scatter(df['fdc_id'],df['NDB_number'], c=y_pred)
plt.show()

# Confusion Matrix
print("\n", "CONFUSION MATRIX".center(40, "="))
print(pair_confusion_matrix(df['NDB_number'],y_pred))
```