LAPORAN CASE BASED 2

UNSUPERVISED LEARNING

Mata Kuliah: Pembelajaran Mesin

Dosen Pengampu: Bedy Purnama, S.Si, M.T, Doctor of Philosophy

Kode Dosen: BDP

Disusun Oleh :
Johannes Raphael Nandaputra (1301204243)
IF-44-01

PROGRAM STUDI INFORMATIKA

FAKULTAS INFORMATIKA

2022

Tugas ini dikerjakan dengan cara yang

tidak melanggar aturan perkuliahan dan kode etik akademisi

Kata Pengantar

Puji syukur penulis ucapkan kepada Tuhan Yang Maha Esa atas berkat-Nya penulis dapat menyelesaikan Laporan Case Based 2 ini sebagai hasil dari pembelajaran unsupervised learning dimana penulis dapat menjelaskan, mengimplementasikan, menganalisis, dan mendesain teknik pembelajaran mesin supervised learning.

Dalam pengerjaan laporan ini, penulis mengerjakan tugas ini dengan cara yang tidak melanggar aturan perkuliahan dan kode etik akademisi. Meski begitu, penulis merasa masih banyak kekurangan-kekurangan baik pada penulisan maupun materi, mengingat kemampuan yang dimiliki penulis. Untuk itu, kritik dan saran dari semua pihak sangat penulis harapkan demi penyempurnaan laporan ini.

Bandung, 2 Desember 2022

Johanes Raphael Nandaputra

A. Ikhtisar Kumpulan Data yang Dipilih

Data yang digunakan dalam pengerjaan tugas ini adalah data *water_treatment*, dimana kumpulan data ini berasal dari pengukuran harian sensor di instalasi pengolahan air limbah perkotaan. Tujuannya adalah untuk mengklasifikasikan keadaan operasional pembangkit untuk memprediksi kesalahan melalui variabel keadaan pembangkit pada setiap tahapan proses perawatan. Domain ini telah dinyatakan sebagai domain yang tidak terstruktur. Berikut informasi data yang digunakan:

N	Attrib.	Description
1	Q-E	input flow to plant
2	ZN-E	input Zinc to plant
3	PH-E	input pH to plant
4	DBO-E	input Biological demand of oxygen to plant
5	DQO-E	input chemical demand of oxygen to plant
6	SS-E	input suspended solids to plant
7	SSV-E	input volatile suspended solids to plant
8	SED-E	input sediments to plant
9	COND-E	input conductivity to plant
10	PH-P	input pH to primary settler
11	DBO-P	input Biological demand of oxygen to primary settler
12	SS-P	input suspended solids to primary settler
13	SSV-P	input volatile suspended solids to primary settler
14	SED-P	input sediments to primary settler
15	COND-P	input conductivity to primary settler
16	PH-D	input pH to secondary settler
17	DBO-D	input Biological demand of oxygen to secondary settler
18	DQO-D	input chemical demand of oxygen to secondary settler
19	SS-D	input suspended solids to secondary settler
20	SSV-D	input volatile suspended solids to secondary settler
21	SED-D	input sediments to secondary settler
22	COND-D	input conductivity to secondary settler
23	PH-S	output pH
24	DBO-S	output Biological demand of oxygen
25	DQO-S	output chemical demand of oxygen
26	SS-S	output suspended solids
27	SSV-S	output volatile suspended solids
28	SED-S	output sediments

29	COND-S	output conductivity
30	RD-DBO-P	performance input Biological demand of oxygen in primary settler
31	RD-SS-P	performance input suspended solids to primary settler
32	RD-SED-P	performance input sediments to primary settler
33	RD-DBO-S	performance input Biological demand of oxygen to secondary settler
34	RD-DQO-S	performance input chemical demand of oxygen to secondary settler
35	RD-DBO-G	global performance input Biological demand of oxygen
36	RD-DQO-G	global performance input chemical demand of oxygen
37	RD-SS-G	global performance input suspended solids
38	RD-SED-G	global performance input sediments

- Info dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 527 entries, 0 to 526
Data columns (total 39 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Date      527 non-null    object
 1   Q-E       527 non-null    object
 2   ZN-E      527 non-null    object
 3   PH-E      527 non-null    float64
 4   DBO-E     527 non-null    object
 5   DQO-E     527 non-null    object
 6   SS-E      527 non-null    object
 7   SSV-E     527 non-null    object
 8   SED-E     527 non-null    object
 9   COND-E    527 non-null    int64
 10  PH-P      527 non-null    float64
 11  DBO-P     527 non-null    object
 12  SS-P      527 non-null    int64
 13  SSV-P     527 non-null    object
 14  SED-P     527 non-null    object
 15  COND-P    527 non-null    int64
 16  PH-D      527 non-null    float64
 17  DBO-D     527 non-null    object
 18  DQO-D     527 non-null    object
 19  SS-D      527 non-null    object
 20  SSV-D     527 non-null    object
 21  SED-D     527 non-null    object
 22  COND-D    527 non-null    int64
 23  PH-S      527 non-null    object
 24  DBO-S     527 non-null    object
 25  DQO-S     527 non-null    object
 26  SS-S      527 non-null    object
 27  SSV-S     527 non-null    object
 28  SED-S     527 non-null    object
 29  COND-S    527 non-null    object
 30  RD-DBO-P  527 non-null    object
 31  RD-SS-P   527 non-null    object
 32  RD-SED-P  527 non-null    object
 33  RD-DBO-S  527 non-null    object
 34  RD-DQO-S  527 non-null    object
 35  RD-DBO-G  527 non-null    object
 36  RD-DQO-G  527 non-null    object
 37  RD-SS-G   527 non-null    object
 38  RD-SED-G  527 non-null    object
dtypes: float64(3), int64(4), object(32)
memory usage: 160.7+ KB
```

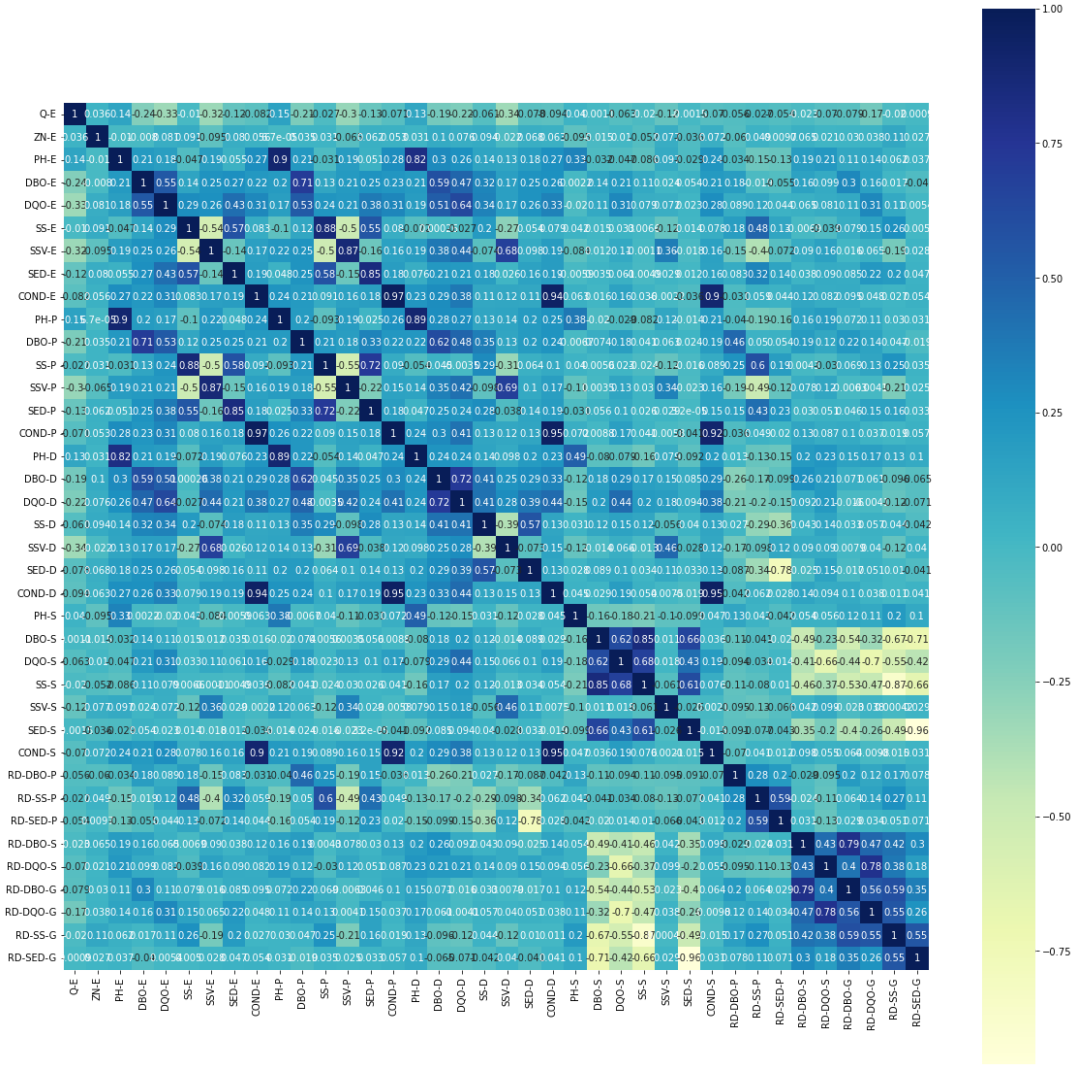
- 5 data teratas

	Date	Q-E	ZN-E	PH-E	DBO-E	DQO-E	SS-E	SSV-E	SED-E	COND-E	...	COND-S	RD-DBO-P	RD-SS-P	RD-SED-P	RD-DBO-S	RD-DQO-S	RD-DBO-G	RD-DQO-G	RD-SS-G	RD-SED-G
0	D-1/3/90	44101	1.50	7.8	?	407	166	66.3	4.5	2110	...	2000	?	58.8	95.5	?	70.0	?	79.4	87.3	99.6
1	D-2/3/90	39024	3.00	7.7	?	443	214	69.2	6.5	2660	...	2590	?	60.7	94.8	?	80.8	?	79.5	92.1	100
2	D-4/3/90	32229	5.00	7.6	?	528	186	69.9	3.4	1666	...	1888	?	58.2	95.6	?	52.9	?	75.8	88.7	98.5
3	D-5/3/90	35023	3.50	7.9	205	588	192	65.6	4.5	2430	...	1840	33.1	64.2	95.3	87.3	72.3	90.2	82.3	89.6	100
4	D-6/3/90	36924	1.50	8.0	242	496	176	64.8	4.0	2110	...	2120	?	62.7	95.6	?	71.0	92.1	78.2	87.5	99.5

- 5 data terakhir

	Date	Q-E	ZN-E	PH-E	DBO-E	DQO-E	SS-E	SSV-E	SED-E	COND-E	...	COND-S	RD-DBO-P	RD-SS-P	RD-SED-P	RD-DBO-S	RD-DQO-S	RD-DBO-G	RD-DQO-G	RD-SS-G	RD-SED-G
522	D-26/8/91	32723	0.16	7.7	93	252	176	56.8	2.3	894	...	942	?	62.3	93.3	69.8	75.9	79.6	78.6	96.6	99.6
523	D-27/8/91	33535	0.32	7.8	192	346	172	68.6	4.0	988	...	950	?	58.3	97.8	83.0	59.1	91.1	74.6	90.7	100
524	D-28/8/91	32922	0.30	7.4	139	367	180	64.4	3.0	1060	...	1136	?	65.0	97.1	76.2	66.4	82.0	77.1	88.9	99
525	D-29/8/91	32190	0.30	7.3	200	545	258	65.1	4.0	1260	...	1326	39.8	65.9	97.1	81.7	70.9	89.5	87.0	89.5	99.8
526	D-30/8/91	30488	0.21	7.5	152	300	132	69.7	?	1073	...	1224	?	69.5	?	81.7	76.4	?	81.7	86.4	?

- Korelasi antara Atribut



B. Ringkasan Pra-Pemrosesan Data yang Diusulkan

1. Data Cleaning

Saat dataset tersebut diperiksa, missing value pada dataset tersebut berbentuk simbol ‘?’ yang mengharuskan untuk mengubah datum tersebut menjadi NaN dan setelah itu diperiksa jumlah missing value tiap kolom yang ada.

```
Date         0
Q-E         18
ZN-E         3
PH-E         0
DBO-E       23
DQO-E        6
SS-E         1
SSV-E       11
SED-E       25
COND-E       0
PH-P         0
DBO-P       40
SS-P         0
SSV-P       11
SED-P       24
COND-P       0
PH-D         0
DBO-D       28
DQO-D        9
SS-D         2
SSV-D       13
SED-D       25
COND-D       0
PH-S         1
DBO-S       23
DQO-S       18
SS-S         5
SSV-S       17
SED-S       28
COND-S       1
RD-DBO-P    62
RD-SS-P      4
RD-SED-P    27
RD-DBO-S    40
RD-DQO-S    26
RD-DBO-G    36
RD-DQO-G    25
RD-SS-G      8
RD-SED-G    31
dtype: int64
```

Setelah itu, tiap tipe data object akan diubah menjadi numeric dan mengisi data missing value (NaN) dengan mean.

```
#ubah data tipe object menjadi numeric dan mengisi data NaN dengan mean
cols = plant.columns[plant.dtypes.eq('object')]
plant[cols] = plant[cols].apply(pd.to_numeric, errors='coerce')
plant = plant.fillna(plant.mean())
```

2. Redundant Data

Pada pengecekan data yang redundan, diketahui bahwa tidak ada data yang duplikat.

```
#cek duplikat
plant.duplicated().sum()
```

```
0
```

3. Feature Scaling

Pada feature scaling, akan dilakukan normalisasi data yang bertujuan untuk meng-scalingkan data-data yang ada yang berguna juga untuk meningkatkan performanya.

```
plant = ((plant - plant.min()) / (plant.max() - plant.min())) * 9 + 1
```

	Q-E	ZN-E	PH-E	DBO-E	DQO-E	SS-E	SSV-E	SED-E	COND-E	PH-P	...	COND-S	RD-DBO-P	RD-SS-P	RD-SED-P	RD-DBO-S	RD-DQO-S	RD-DBO-G	RD-DQO-G	RD-SS-G	RD-SED-G
count	527.000000	527.000000	527.000000	527.000000	527.000000	527.000000	527.000000	527.000000	527.000000	527.000000	...	527.000000	527.000000	527.000000	527.000000	527.000000	527.000000	527.000000	527.000000	527.000000	527.000000
mean	5.888751	1.608730	5.550285	4.487539	4.410563	1.609950	7.040932	2.060237	3.888168	4.974858	...	3.236412	5.412385	6.274985	9.078958	8.829277	7.265789	9.071354	7.690864	8.945797	9.870701
std	1.173137	0.738770	1.230875	1.342057	1.252408	0.639949	1.539872	0.677009	1.378085	1.701936	...	1.067596	1.592922	1.264141	0.849508	0.880530	1.068570	0.788709	0.989558	0.824126	0.611862
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	5.121964	1.215569	4.500000	3.587224	3.569186	1.339267	6.346100	1.707865	2.919349	4.000000	...	2.520661	4.502548	5.490088	8.917660	8.657803	6.849057	8.930233	7.279468	8.818182	9.886792
50%	5.706782	1.377246	5.500000	4.405405	4.338372	1.461780	7.367688	2.036517	3.634742	4.750000	...	3.066116	5.412385	6.352423	9.346696	8.990751	7.405660	9.174419	7.787072	9.121212	9.957547
75%	6.584458	1.781437	6.500000	5.223587	5.118023	1.678534	8.069638	2.289326	4.563009	6.250000	...	3.778237	6.394268	7.095815	9.580715	9.271676	7.943396	9.453488	8.300380	9.353535	10.000000
max	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	...	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000

C. Menerapkan Algoritma yang Dipilih

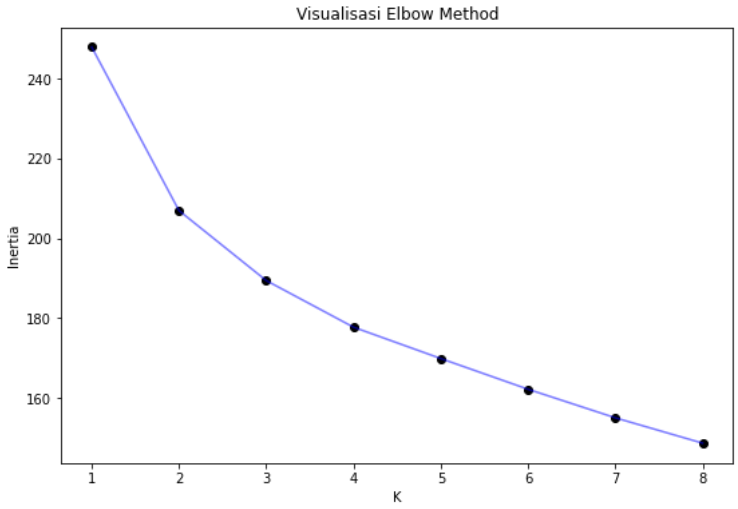
Algoritma yang digunakan pada tugas ini adalah algoritma K-Means Clustering yang merupakan bagian dari Unsupervised Learning.

1. Mencari Cluster Optimum

Saya mencari cluster yang optimum dengan *Elbow Method*. Metode Elbow merupakan salah satu metode untuk menentukan jumlah cluster yang tepat melalui persentase hasil perbandingan antara jumlah cluster yang akan membentuk siku pada suatu titik.

```
#elbow method menentukan nilai k
elbow = []
for k in range(1, 9):
  km = KMeans(n_clusters=k, init='k-means++', random_state=0)
  km.fit(dataset)
  elbow.append(km.inertia_)
```

```
#visualisasi elbow method
plt.figure(1, figsize=(9,6))
plt.plot(range(1,9), elbow, 'o', color='black')
plt.plot(range(1,9), elbow, '-', alpha = 0.5, color = 'blue')
plt.title('Visualisasi Elbow Method'), plt.xlabel('K'), plt.ylabel('Inertia')
plt.show()
```



Dari hasil visualisasi elbow method yang telah dilakukan, cluster optimum yang saya ambil adalah 3. (K = 3)

2. Inisiasi Random Centroids

Setelah saya mencari cluster yang optimal menggunakan metode elbow, saya akan menginisiasi centroids secara random berdasarkan dataset dan cluster.

```
def random_centroids(data, k):
  centroids = []
  for i in range(k):
    centroid = data.apply(lambda x: float(x.sample()))
    centroids.append(centroid)
  return pd.concat(centroids, axis=1)
```

```
centroids = random_centroids(plant, 3)
centroids
```

	0	1	2
Q-E	4.903400	5.013851	4.371290
ZN-E	2.751497	1.511976	3.667665
PH-E	3.000000	4.500000	6.000000
DBO-E	6.395577	3.277641	5.931204
DQO-E	4.286047	3.720930	4.338372
SS-E	1.310995	1.273298	1.405236
SSV-E	5.913649	5.963788	6.891365
SED-E	2.238764	1.783708	1.606742
COND-E	5.379604	2.985653	3.366033
PH-P	4.000000	4.000000	4.750000
DBO-P	4.061856	5.917526	4.451546
SS-P	2.790932	1.464736	1.385390
SSV-P	6.197917	6.739583	5.250000
SED-P	2.100000	2.000000	2.600000
COND-P	2.633122	6.077655	3.410460
PH-D	5.846154	5.153846	6.538462
DBO-D	6.386100	6.768340	3.154440
DQO-D	8.141531	4.863109	2.941995
SS-D	5.384615	1.507692	2.153846
SSV-D	7.462406	7.304511	7.383459
SED-D	1.514286	2.542857	2.285714
COND-D	4.352843	4.083218	3.796117
PH-S	4.333333	2.000000	3.000000
DBO-S	1.369085	1.369085	1.369085
DQO-S	3.348974	2.979472	5.064516
SS-S	1.271552	1.543103	1.543103
SSV-S	8.588983	7.584746	9.097458
SED-S	1.000000	1.025714	1.128571
COND-S	2.264463	2.958678	2.641873
RD-DBO-P	5.412385	7.535032	8.360510
RD-SS-P	3.150881	6.243392	7.373348
RD-SED-P	9.249187	7.913326	9.078958
RD-DBO-S	8.220809	9.261272	9.635838
RD-DQO-S	8.037736	7.018868	7.349057
RD-DBO-G	9.071354	9.069767	9.430233
RD-DQO-G	8.596958	7.000000	6.532319
RD-SS-G	9.030303	8.474747	8.747475
RD-SED-G	9.858491	10.000000	9.943396

3. Mencari Labels

Setelah random centroids terinisiasi, akan dicari cluster labels untuk tiap data point. Akan dipilih label dengan jarak yang terdekat atau minimum dari tiap cluster yang sudah diketahui.

```
def get_labels(data, centroids):
  distances = centroids.apply(lambda x: np.sqrt(((data - x) ** 2).sum(axis=1)))
  return distances.idxmin(axis=1)
```

Dimana distance nya sebagai berikut:

```
distances = centroids.apply(lambda x: np.sqrt(((plant - x) ** 2).sum(axis=1)))
distances
```

	0	1	2
0	8.821907	6.522532	8.735540
1	10.411877	10.052056	12.664353
2	10.068549	4.885690	7.401210
3	9.229028	7.558304	9.866029
4	8.909305	7.285415	8.696799
...	...	...	...
522	12.933456	7.530917	7.895209
523	11.595297	5.350999	6.038139
524	12.336519	5.906221	7.267995
525	12.040459	5.793832	8.172497
526	13.385378	6.191438	7.601802

Dengan labels tiap row sebagai berikut:

```
labels = get_labels(data, centroids)
labels
```

```
0      1
1      1
2      1
3      1
4      1
      ..
522    1
523    1
524    1
525    1
526    1
Length: 527, dtype: int64
```

```
labels.value_counts()
```

```
1    399
2    102
0     26
dtype: int64
```

4. Update Centroids

Centroid yang lama akan diganti dengan centroid baru dan akan berhenti jika centroid lama sama dengan centroid baru berdasarkan clusternya.

```
def new_centroids(data, labels, k):
  return data.groupby(labels).apply(lambda x: np.exp(np.log(x).mean())).T
```

5. Visualisasi Plot Cluster

Pada visualisasi cluster tiap iterasi, akan dibantu dengan library PCA dan clear_output yang dimana akan memvisualisasikan scatter plot.

```
from sklearn.decomposition import PCA
from IPython.display import clear_output

def plot_clusters(data, labels, centroids, iteration):
  pca = PCA(n_components=2)
  data_2d = pca.fit_transform(data)
  centroids_2d = pca.fit_transform(centroids.T)
  clear_output(wait=True)
  plt.title(f'Iteration {iteration}')
  plt.scatter(x=data_2d[:,0], y=data_2d[:,1], c=labels)
  plt.scatter(x=centroids_2d[:,0], y=centroids_2d[:,1])
  plt.show()
```

D. Evaluasi Hasil

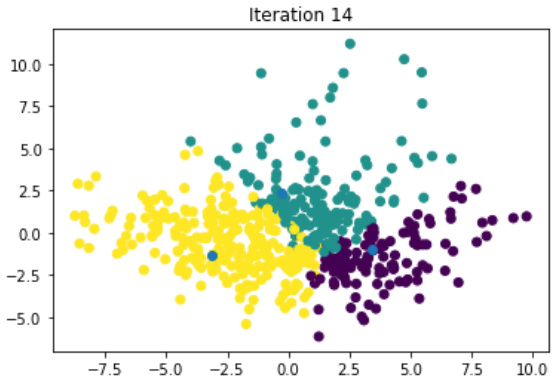
Algoritma yang sudah diterapkan sebelumnya akan dijalan berdasarkan algoritma utama dibawah dengan iterasi maksimalnya adalah 100, cluster yang didapatkan adalah 3, menginisiasi random centroid ke variabel centroids, membuat dataframe baru ke variabel old_centroids, dan assign 1 ke variabel iterasi.

Setelah itu akan memasuki perulangan yang akan berhenti ketika iterasi melebihi maksimal iterasi atau centroid lama sama dengan centroid baru. Dengan variabel old_centroids akan diassign dengan centroids baru, mencari label, mendapat centroid baru dan memplot cluster tiap iterasi.

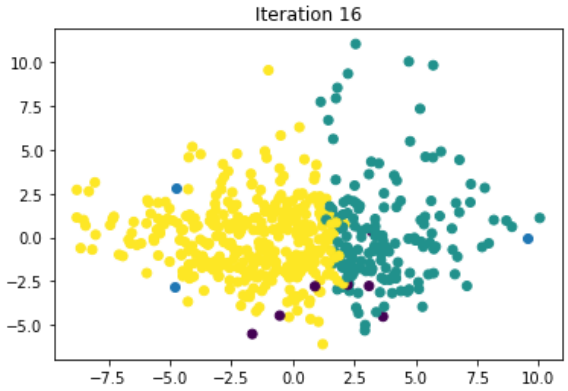
```
max_iterations = 100
k = 3

centroids = random_centroids(data, k)
old_centroids = pd.DataFrame()
iteration = 1

while iteration < max_iterations and not centroids.equals(old_centroids):
  old_centroids = centroids

  labels = get_labels(data, centroids)
  centroids = new_centroids(data, labels, k)
  plot_clusters(data, labels, centroids, iteration)

  iteration += 1
```

Setelah beberapa percobaan menghasilkan cluster sebagai berikut:

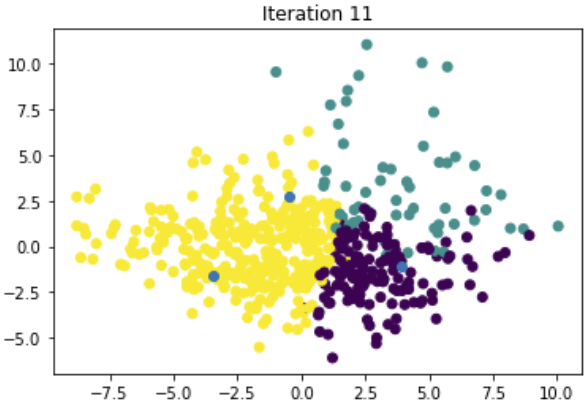
1. Percobaan 1



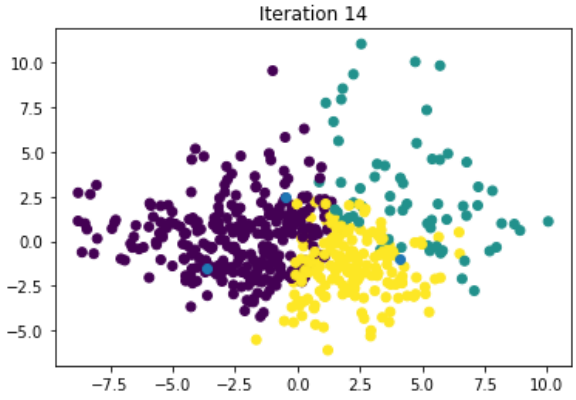
2. Percobaan 2



3. Percobaan 3



4. Percobaan 4



Dengan hasil centroid tiap cluster di percobaan terakhir (4) adalah sebagai berikut:

	0	1	2
Q-E	5.712288	6.725165	5.542885
ZN-E	1.538218	1.574759	1.448779
PH-E	6.028061	5.229268	4.579319
DBO-E	4.963572	3.212816	3.757169
DQO-E	4.859531	3.102687	3.771054
SS-E	1.547045	1.952394	1.418127
SSV-E	7.494457	3.902547	7.131981
SED-E	2.122167	1.895750	1.827248
COND-E	4.253628	2.704112	3.194180
PH-P	5.603205	4.436477	3.496620
DBO-P	4.726552	3.227059	3.381690
SS-P	1.799020	2.220517	1.534752
SSV-P	6.831426	4.188076	6.675822
SED-P	1.879578	1.688359	1.537612
COND-P	4.436888	2.790102	3.277284
PH-D	6.493897	5.612959	4.765792
DBO-D	4.903375	2.930244	3.599662
DQO-D	5.746002	3.022590	4.208078
SS-D	3.229335	2.748775	2.525475
SSV-D	7.118921	5.077977	7.098681
SED-D	2.106449	1.700931	1.771170
COND-D	4.854061	3.572193	4.048520
PH-S	3.393546	3.446257	3.133734
DBO-S	1.476183	1.309237	1.459922
DQO-S	3.081138	2.424152	2.902050
SS-S	1.575547	1.404098	1.611521
SSV-S	7.548844	6.522100	7.404516
SED-S	1.055289	1.034131	1.100773
COND-S	3.507542	2.364322	2.733233
RD-DBO-P	5.289711	5.975381	4.631436
RD-SS-P	6.079819	6.923290	5.933378
RD-SED-P	8.961310	9.162782	9.056911
RD-DBO-S	9.026179	8.663141	8.323773
RD-DQO-S	7.479341	6.884041	6.770841
RD-DBO-G	9.211536	9.065030	8.652222
RD-DQO-G	7.844342	7.512221	7.258619
RD-SS-G	8.969911	9.371800	8.561523
RD-SED-G	9.925883	9.944248	9.609135

E. Link Hasil Pembuatan

1. Laporan

   https://docs.google.com/document/d/1AasfobmdwUpizQkvBdZTI0axpfdEfqbA9rHJwWpdYdk/edit?usp=sharing

2. Slide

   https://www.canva.com/design/DAFTyVpacpg/CVic5miV8lsMTgMsDB8uqA/view?utm_content=DAFTyVpacpg&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

3. Presentasi

   https://youtu.be/sN1Zd0BKUP8

4. Code

   https://colab.research.google.com/drive/17iqmn8U63rC1O5VIv1cYPoFqGUOTKbu5?usp=sharing

REFERENSI

Slide Perkuliahan Pembelajaran Mesin Telkom University

https://archive.ics.uci.edu/ml/datasets/Water+Treatment+Plant

https://www.youtube.com/watch?v=lX-3nGHDhQg&t=1379s&ab_channel=Dataquest

https://www.bradleysawler.com/engineering/ml-clustering-of-a-waste-water-treatment-plant/

https://anderfernandez.com/en/blog/kmeans-algorithm-python/