Webscraping dan Klasterisasi dengan K-Means

1. Mengambil data dari website pokemondb.net
   Langkah pertama yang dilakukan adalah mengambil data dari website pokemon.db dengan menggunakan teknik scrapping menggunakan library beautifulsoup.

```
import csv
from urllib.request import urlopen, Request
import requests
from bs4 import BeautifulSoup as bs
import pandas as pd

alamat = "https://pokemondb.net/pokedex/all"
safeAdd = Request(alamat, headers={'User-Agent': 'Mozilla/5.0'})
html = urlopen(safeAdd)
data = bs(html, 'html.parser')
```

Membaca data pada class table dan id : pokedex. Selanjutnya akan membaca masing-masing row berdasarkan tag tr, dan terakhir akan membaca data pada tiap-tiap cell secara horizontal dari tiap-tiap row atau baris berdasarkan tag th dan td.

```
table = data.find("table", {"id":"pokedex"})
rows = data.findAll("tr")

row_data = []
for row in rows:
    cell_data = []

    if row.contents[1].get_text() == '501':
      break

    for item in row.findAll(["th","td"]):
      cell_data.append(item.get_text())

    row_data.append(cell_data)

print(row_data)
```

Hasil print row_data

```
print(row_data)
[['#', 'Name', 'Type', 'Total', 'HP', 'Attack', 'Defense', 'Sp. Atk', 'Sp. Def', 'Speed'], ['001', 'Bulbasaur', 'Grass Poison', '318', '45', '49', '49', '65', '65
```

2. Menyimpan data ke dalam file csv
   Mengubah list row_data ke dalam bentuk pandas dataframe dan menyimpannya ke dalam bentuk file excel.

```
df = pd.DataFrame(row_data)
df.columns = df.iloc[0]
df = df[1:]

df.to_csv('data_pokemon.csv',index=False)
```

   Membaca data pada file csv

```
data = pd.read_csv("data_pokemon.csv")
print(data.head())
```

```
   #                   Name          Type  ...  Sp. Atk  Sp. Def  Speed
0  1              Bulbasaur  Grass Poison  ...       65       65     45
1  2                Ivysaur  Grass Poison  ...       80       80     60
2  3               Venusaur  Grass Poison  ...      100      100     80
3  3  Venusaur Mega Venusaur  Grass Poison  ...      122      120     80
4  4             Charmander          Fire  ...       60       50     65

[5 rows x 10 columns]
```

3. Memilih atribut untuk perhitungan pengelompokan kluster, akan digunakan atribut attack dan defense. Selanjutnya dilakukan transformasi logaritmik untuk kedua atribut tersebut agar range nilai defense dan attack sama. Selanjutnya ditambahkan kolom baru untuk menampung nilai hasil transformasi dari kolom attack dan defense.

```
import numpy as np

data["Attack"] = pd.to_numeric(data["Attack"])
data["Defense"] = pd.to_numeric(data["Defense"])

#menambahkan dua kolom tranformasi
data["Alog"] = np.log(data["Attack"])
data["Dlog"] = np.log(data["Defense"])

print(data.head())
```

   Selanjutnya mengambil kolom hasil transformasi data attack dan defense dan mengubahnya ke dalam bentuk array.

```
log_data = data.iloc[:, 10:12]
print(log_data.head())
# mengubahnya menjadi array
log_array = np.array(log_data)
```

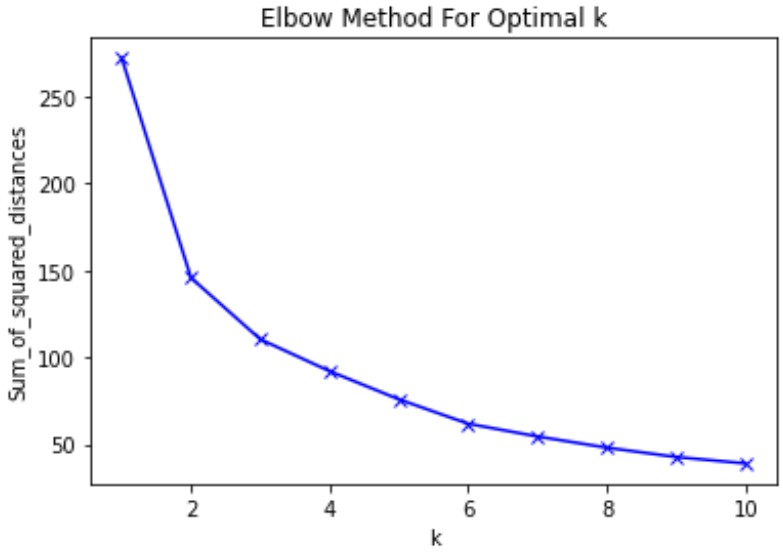
```
print(log_array)
```

Hasil print log_array

```
[[3.8918203  3.8918203 ]
 [4.12713439 4.14313473]
 [4.40671925 4.41884061]
 ...
 [4.14313473 3.80666249]
 [4.53259949 4.00733319]
 [4.81218436 4.17438727]]
```

4. Menentukan K- optimal
   a. Menentukan K-optimal dengan menggunakan Elbow method

```
from sklearn.cluster import KMeans
Sum_of_squared_distances = []
K = range(1,11)
for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(log_array)
    Sum_of_squared_distances.append(km.inertia_)

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()
```



Untuk mengetahui nilai k-optimal dapat digunakan library kneed menggunakan fungsi kneelocator.

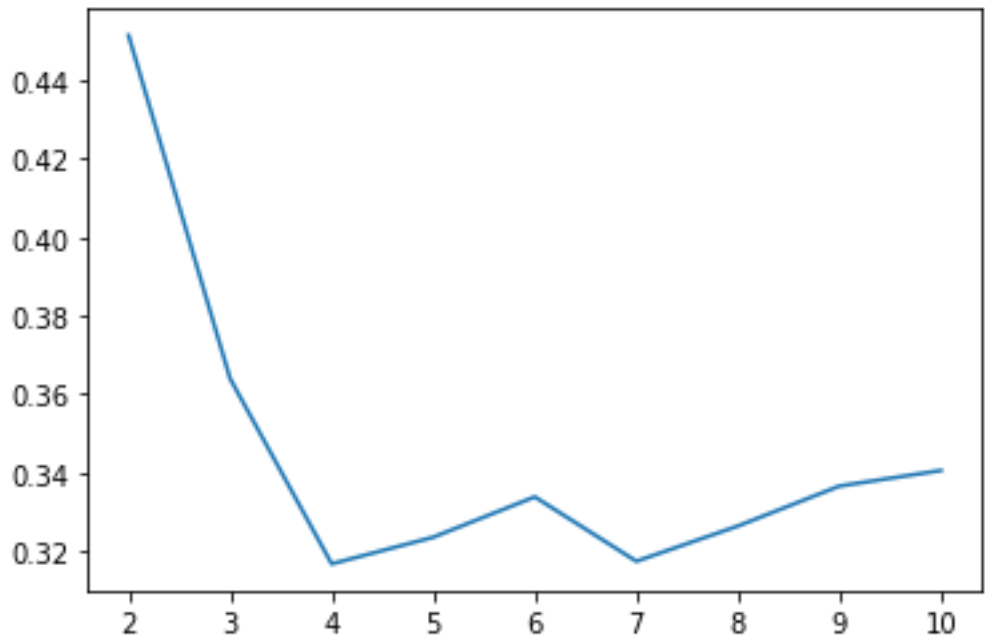
Seteleh di-run didapatkan nilai k-optimal = 3

```
from kneed import KneeLocator
kl = KneeLocator(range(1, 11), Sum_of_squared_distances, curve="convex", direction="decreasing")
kl.elbow
```

```
3
```

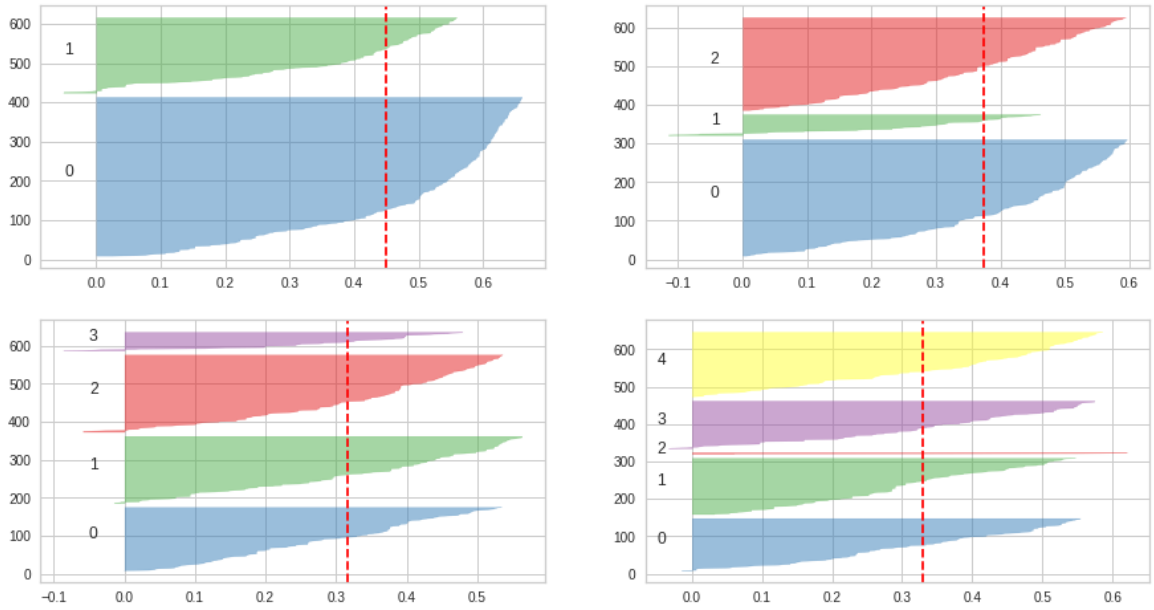
b. Menentukan K-optimal dengan metode silhouette

```
from sklearn.metrics import silhouette_score

data = []
k_list = []

for k in range(2, 11):
    kmeans = KMeans(n_clusters = k).fit(log_array)
    labels = kmeans.labels_
    data.append(silhouette_score(log_array, labels, metric = 'euclid
ean'))
    k_list.append(k)

plt.plot(k_list,data)
plt.show()
```



Pada hasil visualisasi dapat dilihat bahwa k-optimal yang diperoleh adalah 2 (berada pada titik puncak tertinggi). Hail ini berbeda dengan hasil yang didapatkan ketika menggunakan metode elbow.
Dari hasil silhouette tersebtjuga dapat divisualisasikan seperti di bawah ini.

```
from yellowbrick.cluster import SilhouetteVisualizer
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

fig, ax = plt.subplots(2, 2, figsize=(15,8))
for i in [2, 3, 4, 5]:
    '''
    Create KMeans instance for different number of clusters
    '''
    km = KMeans(n_clusters=i, init='k-
means++', n_init=10, max_iter=100, random_state=42)
    q, mod = divmod(i, 2)
    '''
    Create SilhouetteVisualizer instance with KMeans instance
    Fit the visualizer
    '''
    visualizer = SilhouetteVisualizer(km, colors='yellowbrick', ax=a
x[q-1][mod])
    visualizer.fit(log_array)
```



Dari gambar di atas didapatkan gambar yang baik ketika nilai k =2, dimana ukuran dari masing-masing kluster tidak terlalu jauh berbeda dan hanya sedikit data yang salah masuk kluster (bernilai negatif).

5. Melakukan clustering dan visualisasi hasil clustering

   Melakukan clustering dengan algoritma k-means dengan nilai k = 2 menggunakan library sklearn.

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

kmeans = KMeans(n_clusters=2, random_state=500)
kmeans.fit(log_array)
data['kluster'] = kmeans.labels_
print(data.head())
```

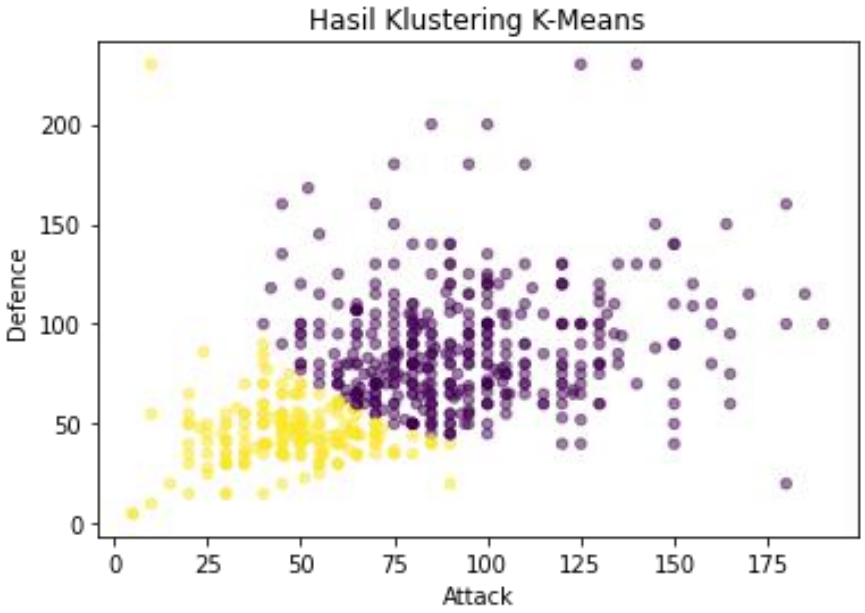
```
   #                  Name          Type  ...      Alog      Dlog  kluster
0  1             Bulbasaur  Grass Poison  ...  3.891820  3.891820        0
1  2               Ivysaur  Grass Poison  ...  4.127134  4.143135        0
2  3               Venusaur  Grass Poison  ...  4.406719  4.418841        1
3  3  Venusaur Mega Venusaur  Grass Poison  ...  4.605170  4.812184        1
4  4            Charmander         Fire   ...  3.951244  3.761200        0

[5 rows x 13 columns]
```

Menyimpan hasil clustering ke dalam file .csv

```
df_clustering = pd.DataFrame(data)
df_clustering

df_csv = df_clustering.to_csv('df_clustering_pokemon.csv')
```

Melakukan visualisasi hasil clustering

```
plt.scatter(data.Attack, data.Defense, s = 20, c = data.kluster, mar
ker = "o", alpha = 0.5)

plt.title("Hasil Klustering K-Means")
plt.xlabel("Attack")
plt.ylabel("Defence")



plt.show()
```


Hasil Klustering K-Means
Defence
Attack
0
50
100
150
200
0
25
50
75
100
125
150
175