Klasterisasi Negara Berdasakan sosio-ekonomi dan kesehatan menggunakan KMeans Dan DBScan

Sebuah Makalah
Ditujukan sebagai
tugas project case based pembelajaran mesin

oleh:
Khalifardy Midarasah
1304211035

Dosen:
SITI SA'ADAH (SSD)

PROGRAM STUDI S1 PJJ TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG

2023

Lembar Pernyatan

Pernyataan:

Saya mengerjakan tugas ini dengan cara yang tidak melanggar aturan perkuliahan dan kode etik akademisi. Jika melakukan plagiarism atau jenis pelanggaran lainnya, maka Tim kami bersedia diberi nilai E untuk Mata Kuliah ini.

Kata Pengantar

Halaman ini adalah halaman yang saya sangat sukai, karena pada karya ilmiah di sinilah tempat paling bebas untuk mengeluarkan segala isi kepala. Bukan hanya kebenaran yang di tulis secara kaku, namun kebenaran yang dilingkupi dengan keindahan. Maka sesungguhnya ilmu tanpa di balut oleh keindahan akan terasa membosankan, walaupun bagi para pencintanya pada kedalaman-kedalaman nya akan di temukan keindahan yang membuat pikiran berbinar-binar takjub.

Sejak dahulu saya yang mempunyai minat yang sama dengan musik, sastra, matematika dan fisika, saya mempunyai impian bagaimana bisa menghasilkan sebuah karya ilmiah bukan hanya sekedar benar secara metodologi namun penyampainanya dan penulisannya juga indah, sehingga esensi dari karya ilmiah bisa dijangkau bukan hanya oleh kalangan akademisi atau ilmuwan akan tetapi juga masyarakat yang lebih luas.

Perjalanan menuju ke sana memang masih panjang, semoga saya masih punya nafas untuk terus berjalan. Masih di berikan rezeki ilmu dan pemikiran untuk bisa lebih bermanfaat. Dan yang terutama api semangat saya masih terus menyala , demi mereka yang tertindas, demi mereka yang termarjinalkan, dilindas-lindas oleh zaman, terasingkan dan dibuat tak berdaya. Tolong kuatkan saya wahai Engkau yang memegang nyawaku , peluk aku duhai kekasih-Nya yang paling utama, aku rindu.

Bekasi, 03 Desember 2023

Penulis

Daftar isi

Daftar Gambar

Lembar Kosong

Bab 1

Pendahuluan

Studi case pada makalah ini mengambil kasus clustering negara-negara berdasarkan kondisi sosio-ekonomi dan kesehatan pada negara-negara tersebut. Pada data set yang diberikan terdapat 10 fitur yang berhubungan dengan kondisi sosio-ekonomi dan kesehatan pada negara-negara, termasuk juga di dalamnya terdapat nama negara. Untuk mengetahui lebih detail bisa dilihat pada gambar 1.1.

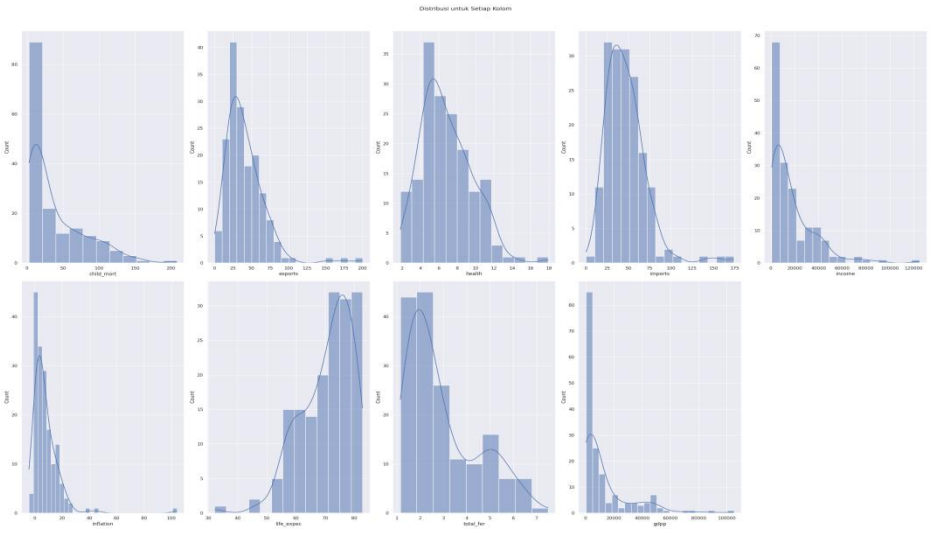
```
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   country     167 non-null    object
 1   child_mort  167 non-null    float64
 2   exports     167 non-null    float64
 3   health      167 non-null    float64
 4   imports     167 non-null    float64
 5   income      167 non-null    int64
 6   inflation   167 non-null    float64
 7   life_expec  167 non-null    float64
 8   total_fer   167 non-null    float64
 9   gdpp        167 non-null    int64
dtypes: float64(7), int64(2), object(1)
memory usage: 13.2+ KB
```

Gambar 1.1. Tabel Fitur

Jika diperhatikan terdapat 10 fitur dengan record sebanyak 167, sehingga dimensi dari datanya adalah 167 x 10 . Tipe-tipe data pada fitur tersebut sebagian besar adalah numerik, hanya pada fitur country atau nama negara yang mempunyai tipe data string.

Kualitas dari data set ini sudah cukup baik, tidak mempunyai data kosong dan semua fitur hampir semua memiliki tipe data yang sama sehingga tidak diperlukan lagi untuk penyesuaian tipe data ataupun mengisi data yang kosong.

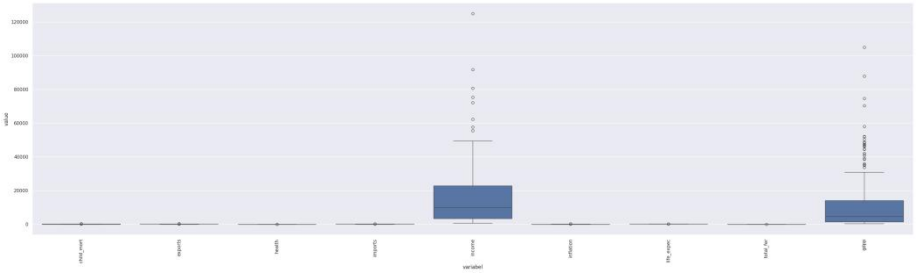
Hal lain yang diperhatikan adalah distribusi data dari masing-masing fitur karena akan menentukan metode yang akan dipakai untuk skalaisasi. bisa lihat pada gambar 1.2 untuk masing-masing distribusi dari fitur.



Gambar 1.2. Distribusi masing-masing fitur

Bisa dilihat bahwa distribusi dari fitur-fitur pada data set memiliki distribusi yang normal, yang artinya data-data ini akan mudah di modelkan karena sudah memiliki bentuk distribusi yang baku.

Hal selanjutnya yang diperhatikan adalah outliers , perlu diperhatikan untuk data yang memiliki outliers cukup banyak, karena akan mempengaruhi model dari machine learning . Pada kasus makalah ini outliers bisa dilihat dari gambar 1.3 , gambar boxplot.

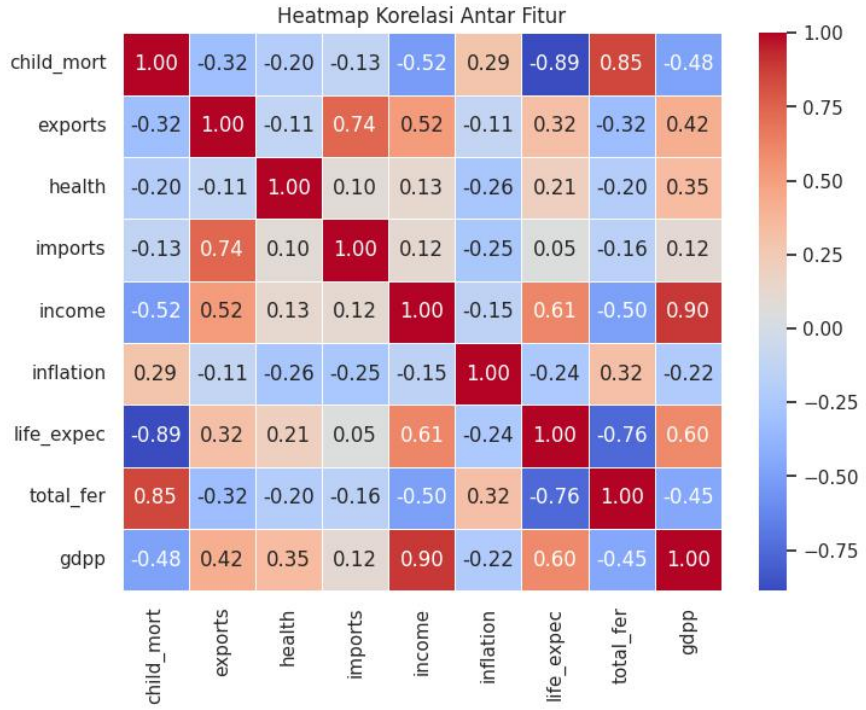


Gambar 1.3. Boxplot masing-masing fitur

Karena data fitur yang relatif sama dengan distribusi normal bisa dilihat pada boxplot gambar 1.3. outliers cukup sedikit sehingga bisa diabaikan atau dibiarkan saja.

Hal lain yang bisa dieksplor adalah korelasi antar fitur, idealnya untuk model machine learning sesama fitur harus salang independen artinya tidak terkait satu sama lain sehingga nilai korelasinya haruslah mendekati nol, namun pada kenyataanya pada kasus dunia real antar fitur masih ada sedikit korelasinya.

Untuk korelasi dari data set pada makalah ini bisa dilihat pada grafik heatm di gambar 1.4.



Gambar 1.4. Heatmap korelasi

Dilihat dari heatmap gambar 1.4 sebagian besar fitur memiliki nilai korelasi yang mendekati nol, kecuali pada diagonalnya karena memang di korelasikan terhadap dirinya sendiri. Sedangkan pada kasus gdpp dan income memiliki korelasi yang sangat kuat karena memang perhitungan gdpp memasukan unsur income sehingga pasti berkorelasi kuat positif, sedangkan pada kasus child mortallity dan life expectaction juga memiliki hubungan kuat yang negatif , karena kematian dan ekpektasi hidup juga berkaitan erat secara terbalik.

Bab 2

Pre-processing

Pada bagian preprocessing ini tidak banyak yang dilakukan, karena data sudah sangat baik, tidak mempumyai data kosong, outliers yang sedikit dan tipe data yang sama. Hal yang perlu dilakukan hanyalah menyamakan skala, penyamaan skala ini dilakukan supaya semua fitur mempunyai skala yang sama sehingga seimbang. Ada beberapa metode yang bisa lakukan untuk penyamaan skala :

1. Minmax
2. Standarisasi
3. logaritmik

Penskalaan dengan minmax artinya membuat rentang skala pada fitur berada pada rentang antara 0 - 1. secara matematis dituliskan

$$\frac{x_i - x_{min}}{x_{max} - x_{min}}$$

data yang ingin diskalakan dikurangi data minimum dari kumpulan data pada suatu fitur dibagi dengan hasil pengurangan data maksimal dengan data minimum. Hasil ini akan membuat range dari fitur berada pada rentang 0-1.

Metode lainnya mengunakan standarisasi, metode ini akan membuat range dari fitur berada pada -1 dan 1 mengikuti skala dari distribusi normal. penskalaan dengan standarasisasi pada intinya adalah membagi jarak data dari rata-rata dengan standar deviasainya, secara matematis ditulis:

$$X' = \frac{X - \mu}{\sigma}$$

dimana :

X’ = data hasil standarisasi

X = data awal

μ = rata-rata

σ = standard deviasi

Selanjutnya standarisasi yang sebenarnya kurang umum dilakukan yaitu standarisasi dengan menggunakan logritma. Ini tidak umum karena banyak asumsi yang diperlukan jika menggunakan skala logaritma. Asumsi yang paling penting adalah data fitur harus positif karena logaritma pada bilangan negati akan menghasilkan bilangan kompleks buka bilangan Real. Namun metode ini dipakai biasanya untuk data-data yang sudah pasti postif dan berdistribusi normal.

Pada studi kasus di makalah ini digunakan metode standarisasi untuk penskalaannya. Alasan utama pemilihan skalasisasi menggunakan standarisasi adalah semua fitur berdistribusi normal dan kedua data memiliki nilai negatif sehingga tidak

memungkinkan menggunakan metode logaritma. Adapun code yang digunakan adalah sebagai berikut.

```
scal = Skalasisasi(data_fitur)
stand = scal.standar_scaler()
```

Skalisasi adalah library yang dibangun sendiri dengan parameter input berupa data fitur dan didalamnya memiliki method standard_scaler() yang mengembalikan dataframe yang sudah di standarisasi. ada pun code library nya adalah sebgai berikut.

```
class Skalasisasi:
"""
kelas untuk skalasisasi
"""
def __init__(self,data):
self.__data = data.copy()
def minmax_scaler(self):
kolom = self.__data.columns
data = self.__data.copy()
for col in kolom:
maksimal = data[col].max()
minimum = data[col].min()
delta = maksimal - minimum
data[col] = data[col].apply(lambda x : abs((x-minimum))/delta)
return data
def standar_scaler(self):
kolom =self.__data.columns
data = self.__data.copy()
for col in kolom:
mean = data[col].mean()
std = data[col].std()
data[col] = data[col].apply(lambda x : (x-mean)/std)
return data
```

Sesudah dilakukan standarisasi selanjutnya menconvert dataframe hasil standarisasi ke bentuk csv ataupun excel. adapun codenya sebagai berikut:

```
stand.to_csv('data_standard.csv',index=False)
```

Sampai sini maka proses preprocessing sudah selesai, karena memang data sudah cukup baik sehingga tidak perlu banyak di olah.

Bab 3

Implementasi Algoritma

Studi kasus pada makalah ini akan memakai dua metode yaitu Kmeans dan DBscan . Pemilihan dua metode tersbut dimaksudkan untuk membandingkan metode mana yang lebih baik dalam clustering pada kasus dimakalah ini.

Hipotesis awal dari percobaan ini, melihat dari kumpulan dataset nya yang cukup rapat dan tidak jelas klusterisasinya, maka seharusnya DBscan akan lebih baik dalam mengklusterisasi dibandingkan dengan Kmeans.

3.1. Kmeans

Seperti namanya pada dasarnya Kmeans adalah mencari nilai rata-rata yang tepat pada suatu kumpulan data yang sesuai sebagai titik pusat pada masing-masing K clustering. Ide dasarnya dari Kmeans jika suatu kumpulan data memiliki atribut yang mirip satu sama lain seharusnya memusat pada titik tengah yang sama.

Secara literal algoritmis Kmeans bekerja seperti ini :

1. Tentukan berapa k cluster yang ingin dibuat
2. Setelah itu menentukan titik tengah permulaan pada masing masing cluster, penentuan titik tengah umunya dilakukan secara acak pada satu record /instance dari data set.
3. lakukan perhitugan jarak setiap data record terhadap setiap titik pusat yang telah ditentukan. Pada jarak yang terpendek ke titik pusat masukan data tersebut kedalam cluster yang berkaitan dengan titik pusat tersebut.
4. hitung rata-rata setiap fitur pada masingt-masing cluster, jika nilai rata-rata masih sama dengan nilai titik pusat sebelumnya maka proses berakhir, jika tidak update nilai titik pusat dengan nilai rata-rata terbaru. lalu kembali ke langkah ketiga sampai nilai titik pusat atau rata-rata sudah tidak berubah.

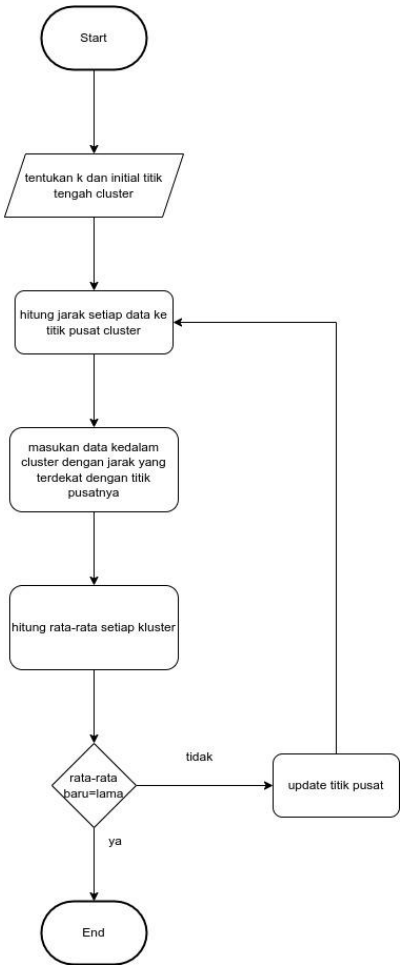
untuk penjabaran secara flowchart bisa dilihat pada gambar 3.1.

Jika diperhatikan penentuan titik pusat pada awal mula initial dari kmeans sangatlah penting. Karena penentuan yang tidak tepat pada titik pusat permulaan akan mempengaruhi langkah-langkah selanjutnya saat update titik pusat. Selain itu jumlah cluster yang tepat juga menentukan seberapa baik algoritma Kmeans ini akan mengclusterisasi data.

Belum ada cara yang eksak untuk menentukan titik pusat permulaan yang optimal, untuk itu perlu dilakukan percobaan berulang-ulang untuk menentukan titik pusat yang terbaik. Namun untuk penentuan berapa cluster yang baik untuk suatu kumpulan data bisa dilakukan evaluasi di awal menggunakan beberapa metode berikut:

1. Hopkins statistik
2. silhoutte skor

3. elbow method



Gambar 3.1. FlowchartKmeans

Hopkins statistik adalah suatu nilai statistik dimana ketika nilai hopkins mendekati 1 maka suatu kumpulan data memiliki cluster hampir tidak ada , sedangkan ketika nilainya mendekati 0 kemungkinan terdapat cluster dalam suatu kumpulan data.

Silhoutte skor adalah suatu nilai yang menunjukan seberapa suatu data berdasarkan anggota cluster yang sama memiliki kemiripan, semakin nilai mendekati 1 menunjukan cluster sudah ter klasterisasai dengan baik, sedangkan ketika mendekati -1 kemungkinan besar banyak data ditempatkan pada klaster yang salah.

Elbow method adalah suatu metode yang berdasarkan pada SSE (sum square error) atau seberapa dekat jarak data-data klaster terhadap titik pusatnya, semakin mendekati nol berarti error nya semakin kecil menunjukan data sudah ditempatkan pada klaster yang tepat. Namun pada dasarnya elbow method merujuk pada suatu titik dimana perbandingan error setelah titik tersebut sudah tidak signifikan lagi.

Dengan mengabbungkan ketiga metode diatas nilai n kluster yang akan dibentuk bisa lebih optimal.

3.2. DBScan

DBscan atau *Density-based Clustering Based on connecte regions with high density,* Adalah metode machine learning yang membuat kluster berdasarkan kepadatan pada suatu region. Semisal ada objek data *i* maka kepadatan pada data i tersebut diukur berdasarkan kepada tetangga-tetangga terdekat pada radius dan jumlah tertentu.

Pada DBscan memiliki dua parameter penting yaitu ε yang merupakan radius data ke-i di sekitarnya atau jangkauan dan minPts yang berarti jumlah minimal anggota pada radius ε sehingga bisa disebut satu kluster.

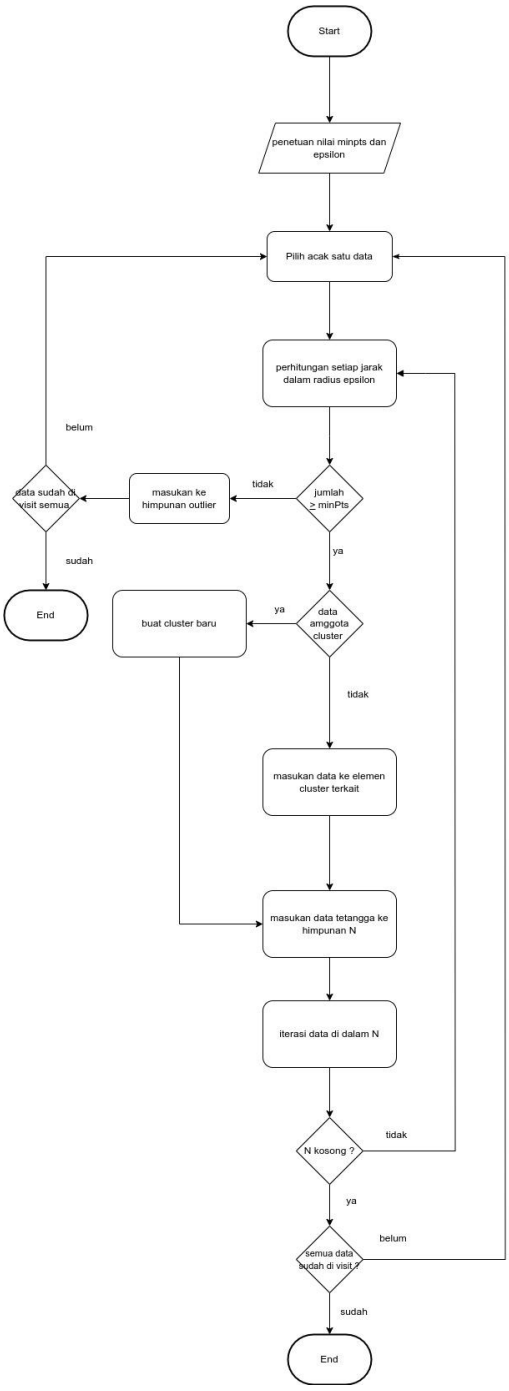
Secara literal algoritmis DBscan bekerja seperti ini :

1. tentukan ε dan minpts
2. pilih secara acak satu data pada kumpulan data
3. lalu tandai data yang sudah terpilih tersebut dengan visited, lalu hitung jarak data tersebut ke data yang belum ditandai.
4. hitung berapa data lain yang masih berada dalam radius ε , jika jumlahnya ≥ minPts, bentuk cluster baru lalu dengan elemen pertama adalah data yang terpilih sedangkan data yang berada dalam radius masukan ke dalam kumpulan data N, jika ternyata jumlahnya ≤ minPts, masukan data terpilih kedalam data outliers.
5. Lakukan pengecekan satu persatu pada kumpulan data N , dengan setiap selesai pengecekan tandai data tersebut dengan visited, ulangi langkah ke -4 dengan tambahan konstrain jika ε dan min pts terpenuhi dan data-i tidak termasuk cluster manapun, baru buat cluster baru, jika termasuk salah satu cluster , tidak ada cluster yang terbentuk.
6. setelah kumpulan data N habis dan masib terdapat data yang belum di visit maka langkah ke dua akan diulangi. Namun jika sudah semua di kunjungi maka DBscan sudah selesai mengklusterisasi.

Penjabaran dengan flow chart bisa dilihat pada gambar 3.2.

Seperti halnya Kmeans penentuan titik tengah awal sangat krusial, DBScan penentuan ε dan minPts menentukan performansi Kmeans. Sayangnya penentual ε dan minPts yang optimal secara eksak belum ditemukan caranya. Maka harus dilakukan eksperimen secara berulang-ulang. Jika beruntung ε dan minPts yang optimal akan ditemukan pada percobaan pertama . Jika tidak harus bersabar untuk terus mencoba sampai di dapatkan hasil yang baik.

Pada makalah ini penentuan ε awal dan minpts di dasari pada rata-rata jarak antar data. Dari rata-rata jarak tersbut akan diinisiasi sebagai ε awal sedangkan minpts sendiri dilakukan percobaan beberapa kali sampai di dapatkan hasil yang sekiranya baik.



Gambar 3.2. Flowchart DBScan

3.3. Implementasi

Di bagian ini akan dijelaskan bagaimana implemetasi metode DBScan dan Kmeans ke dalam bahasa python. Untuk implementasi modelnya sendiri code dibangun dari nol, tidak menggunakan library langsung seperti scikit-learn , hal ini

dilakukan supaya metode lebih bisa dipahami lebih baik , jika dibandingkan penggunaan langsung library.

3.3.1. Implementasi Kmeans

Code implementasi untuk Kmeans dibangun dalam satu folder bernama utils yang didalamnya terdapat file python bernama unsupervised.py, adapun code implementasinya adalah sebagai berikut :

```
import numpy as np

import seaborn as sns

from .matematika.distance import euclidian_distance,kuadrat_jarak
```

Pada awal line code berisi library yang dibutuhkan seperti numpy dan seaborn juga matematika. Untuk library matematika sendiri juga di bangun sendiri pada folder yang sama dengan nama berbeda. FUngsi numpy untuk mengolah matrix atau vektor sedangkan matematika dibangun untuk menghitung jarak secara euclidean

```
class Kmeans:

def
__init__(self,cluster:int=2,inital_centroid=None,limit_loop:int=300,konvergen=None,r
andom_state:int=2):

self.__cluster = cluster

self.__inital_centroid=inital_centroid

self.__limit_loop=limit_loop

self.__historycentroid = {}

self.__konvergen = konvergen

self.__random_state = random_state

self.__cluster_fit = None

self.__inersia = None

self.__x = None

self.__label = None

@property

def inersia(self):

#ini hanya sebuah dekorator

pass
```

Kmeans dipackage kedalam kelas sebagai object dimana mempunyai konstruktor berupa atribut cluster untuk berapa banyak cluster yang ingin dibentuk, intial_Centroid jika sudah mempunyai asumsi awal untuk initial_centroid, limit_loop untuk membatasi looping dan konvergen untuk menentukan pada titik berapa dia konvergen dan random state sebagai bilangan integer acak supaya ketika di random mempunyai keacakan yang sama.

Selain itu terdapat atribut inersia untuk menyimpan insersia dari model, label untuk menyimpan hasil prediksi dan cluster_fit untuk menyimpan hasil nama kluster dan anggotanya.

```
@property
def label(self):
#inin hanya sbeuah dekorator
pass
def sethistory_centroid(self,loop,list_centroid):
self.__historycentroid[loop]=list_centroid
def history_centroid(self)->dict:
return self.__historycentroid
def fit_predict(self,x:np.ndarray)->np.ndarray:
self.__x = x
self.fit(x)
predict = [None for _ in range(len(x))]
for key3 in self.__cluster_fit.keys():
for index in self.__cluster_fit[key3]:
predict[index] = key3
self.__label = np.array(predict)
return np.array(predict)
```

Setelah itu terdapat beberapa method namun yang paling penting diatas adalah method fit_predict dimana akan mengembalikan hasil prediksi dari training, berupa array numpy. DIdalam method tersebut terdapat method fit yang akan dijelaskan dibawah.

```
def fit(self,x:np.ndarray):
self.__x = x
np.random.seed(self.__random_state)
if self.__inital_centroid== None:
self.__inital_centroid= {}
for i in range(self.__cluster):
#centroid = np.random.uniform(x.min(),high = x.max(),size=len(x[0]) )
centroid = x[np.random.choice(x.shape[0], 1, replace=False)]
self.__inital_centroid[i] = list(centroid)
loop = 0
konv = 0
while loop < self.__limit_loop:
cluster={}
```

```
for i in range(self.__cluster):
cluster[i]=[]
self.sethistory_centroid(loop,self.__inital_centroid)
for index,value in enumerate(x):
cls = None
jrk = None
for key in self.__inital_centroid.keys():
jarak = euclidian_distance(value,np.array(self.__inital_centroid[key]))
if jrk == None or jarak <= jrk:
jrk = jarak
cls = key
cluster[cls].append(index)
update_centroid = {}
for key2 in cluster.keys():
if len(cluster[key2])!=0:
init = x[cluster[key2][0]].copy()
for rec in cluster[key2][1:]:
init += x[rec].copy()
init = init/len(cluster[key2])
else:
init = list( x[np.random.choice(x.shape[0], 1, replace=False)])
update_centroid[key2] = list(init)
if konv == self.__konvergen:
break
if update_centroid == self.__inital_centroid:
konv +=1
else:
self.__inital_centroid = update_centroid
loop += 1
self.__cluster_fit = cluster
```

Selanjutnya method fit ini adalah sebuah prosedur dimana algoritma dari kmeans dijalankan. Hasil dari train yang dilakukan akan disimapn pada self.__cluster_fit. utnuk penjelasan percodenya bisa dilihat di link github yang ada di lampiran.

```
@inersia.getter
def inersia_(self):
```

```
total = 0

for key in self.__cluster_fit.keys():

centroid = self.__inital_centroid[key]

for rec in self.__cluster_fit[key]:

total += kuadrat_jarak(self.__x[rec],np.array(centroid))

self.__inersia = total

return self.__inersia
```

Method inersia ini akan mengembalikan nilai inersia yaitu suatu nilai dari kuadrat jarak dari data terhadap pusat clusternya.

```
@label.getter

def label__(self):

return self.__label
```

Method ini hanya untuk mengembalikan hasil prediksi train dari Kmeans

Code-code diatas adalah code Kmeans yang dibangun untuk pengimplementasian membangun model machine learning berbasis Kmeans. Adapun implementasi code dalam membangun model nya adalah sebagai berikut:

```
import pandas as pd
import numpy as np
import sys
sys.path.append('/home/khalifardy/Dokumen/ruang_kerja/code/kuliah/ML/klasterisasi_ne
gara/')
from utils.unsupervised import Kmeans
from utils.evaluasi import elbow_method,hopkins_statistik,visual_silhoutte
from utils.EDA import visual_data
```

code diatas adalah untuk mengimport library-library yang dibutuhkan, pandas, numpy, sys , Kmeans, elbow_method, hopkins_statistik, visual_silhoutte dan visual_data. library sys digunakan hanya untuk supaya sistem bekerja pada direktori yang dinginkan. Kmeans library yang tadi dibangun, elbow_method dibangun juga sendiri untuk menampilkan grafik, hopkins_statistik library yang dibangun untuk mendapatkan nilai hopkins dan library visual_silhoutte dibangun untuk menampilkan grafik silhoutte score.

```
df =
pd.read_csv('/home/khalifardy/Dokumen/ruang_kerja/code/kuliah/ML/klasterisasi_negara
/pre_processing/data_standard.csv')
```

code diatas untuk membuka file data yang sudah distandarisasi

```
x = df.to_numpy()
```

code ini untuk merubah bentuk dataframe pandas dari dataset kedalam bentuk array numpy. ini dilakukan karena Kmeans hanya menerima inputan beruapa array numpy.

```
hopkins_statistik(x,2)
visual_silhoutte(Kmeans,2,21,x)
elbow_method(Kmeans,2,21,x)
```

code-code diatas untuk mendapatkan nilai hopkins dan juga visualisasi dari sillhoutte dan elbow.

```
k = Kmeans(3,random_state=22)
k.fit_predict(x)
```

code ini adalah implementasi dari kmeans dengan cluster berjumlah tiga lalau di train sekaligus di prediksi dengan method fit_predict yang mempunyai inputan x yang berupa array numpy.

```
visual_data("Kmeans",x,k.label__,2)
```

code diatas ini untuk visualisasi dari hasil klustering Kmeans

3.3.2. Implementasi DBScan

Code implementasi dari DBScan juga dibangun pada folder dan nama file yang sama dengan Kmeans. Dengan code nya sebagai berikut:

```
class DBscan:
def __init__(self,eps:float, minpts:int):
self.__eps = eps
self.__minpts = minpts
self.__cluster = {}
self.__label = None
self.__jarak = []
@property
def label(self):
#ini hanya dekorator
pass
@property
def jarak(self):
#ini hanya dekorator
pass
```

DBScan dipackage dalam satu object class yang mempunyai aatribut eps dan minipts. Sedabngkan beberapa atribut lain seperti self.__cluster untuk menyimpan cluster dan anggotanya, label untuk menyimpan hasil prediksi, dan jarak untuk menyimpan jarak setiap data terhadap data lainnya.

```
def fit(self,x:np.ndarray):
cluster = 1
visited = []
unvisited = [i for i in range(len(x))]
noise = []
N = []
while len(unvisited) != 0:
init = np.random.choice(unvisited, 1, replace=False)[0]
visited.append(init)
unvisited.remove(init)
elemen = []
for index,value in enumerate(x):
if index != init :
jarak = euclidian_distance(x[init],value)
self.__jarak.append(jarak)
if jarak <= self.__eps:
N.append(index)
if len(N) >= self.__minpts:
elemen.append(init)
else:
N = []
noise.append(init)
while len(N) >0:
if N[0] not in visited:
visited.append(N[0])
```

```
unvisited.remove(N[0])
el2 = []
for idx2,value in enumerate(x):
if N[0] != idx2:
jarak = euclidian_distance(x[N[0]],value)
self.__jarak.append(jarak)
if jarak <= self.__eps:
el2.append(idx2)
if len(el2)>= self.__minpts:
for j in el2:
if j not in visited and j not in N and j not in noise:
N.append(j)
found = False
for lst in self.__cluster.values():
if N[0] in lst:
found = True
break
if not found :
elemen.append(N[0])

N.pop(0)
if len(elemen) > 0:
self.__cluster[cluster] = elemen
cluster += 1
self.__cluster[-1] = noise
def fit_predict(self,x:np.ndarray):
self.fit(x.copy())
predict = [None for _ in range(len(x))]
for key in self.__cluster.keys():
for index in self.__cluster[key]:
predict[index] = key
self.__label = np.array(predict)
return self.__label
```

Pada method fit adalah method utama dimana algoritma dari DBscan di implementasikan sedangkan fit_predict di dalam nya terdapat method fit yang akan mengembalikan nilai array hasil prediksi.

```
@label.getter
def label_(self):
return self.__label
@jarak.getter
def jarak_(self):
return self.__jarak
```

code diatas adalah method-method yangakan mengembalikan nilai label hasil prediksi dan list dari jarak setiap iterasi.

Selanjutnya adalah code untuk mengimplementasikan library yang sudah dibangun untuk mebuat model DBscan:

```
import pandas as pd
import numpy as np
import sys
sys.path.append('/home/khalifardy/Dokumen/ruang_kerja/code/kuliah/ML/klasterisasi_ne
gara/')
from utils.unsupervised import DBscan
from utils.evaluasi import
elbow_method,hopkins_statistik,visual_silhoutte,visual_silhoutte_dbms
from utils.EDA import visual_data
from statistics import mode,mean,median
```

Library yang digunakan dengan Kmeans kurang lebih sama hanya ada pergantian untuk Kmeans menjadi DBscan dan penambahan library bawaaan python yaitu statistic yang nanti berfungsi untuk menghitung mode, mean dan media pada sebuah list.

```
df =
pd.read_csv('/home/khalifardy/Dokumen/ruang_kerja/code/kuliah/ML/klasterisasi_negara
/pre_processing/data_standard.csv')
x = df.to_numpy()
```

Seperti halnya Kmeans code-code diatas untuk membuka file data terstandarisasi lalu merubahnya kedalam bentuk array numpy

```
d = DBscan(3,3)
d.fit_predict(x)
```

code diatas adalah pengimplementasian DBscan dengan epsilon = 3 dan minpts = 3. Lalu train dan diprediksi.

```
print(min(d.jarak_))
print(max(d.jarak_))
print(mode(d.jarak_))
print(median(d.jarak_))
print(mean(d.jarak_))
```

Code ini untuk menampilkan nilai minimum, maximum, modus, median dan mean dari list jarak yang sudah terisi pada train sebelumnya. Fungsi menggunakan semua hitungan statistik diatas supaya ada dasar ketika menentukan epsilon berapa yang optimal supaya DBscan mempunyai performanis yang baik.

```
visual_data("DBscan",x,d.label_,2)
```

code diatas untuk visualisasi data hasil klastering dari DBscan

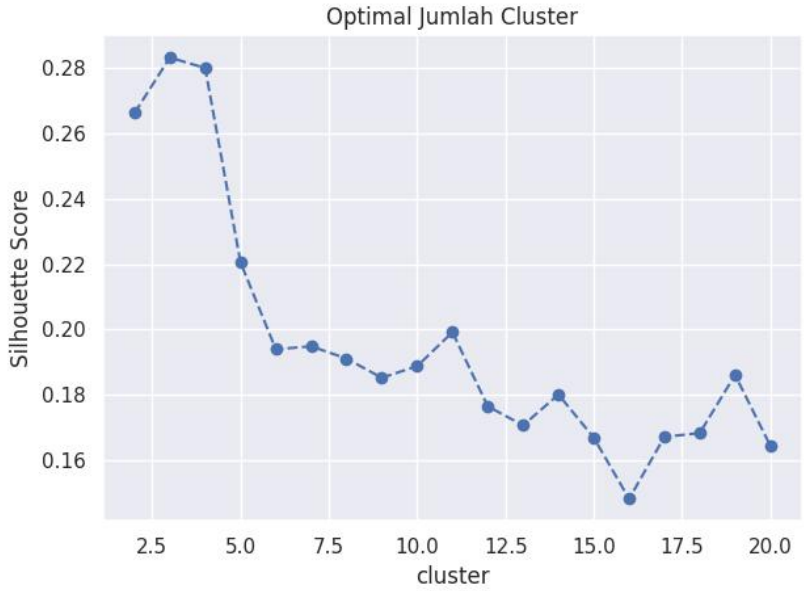
Link github : *https://github.com/khalifardy/klasterisasi_negara*

Bab 4

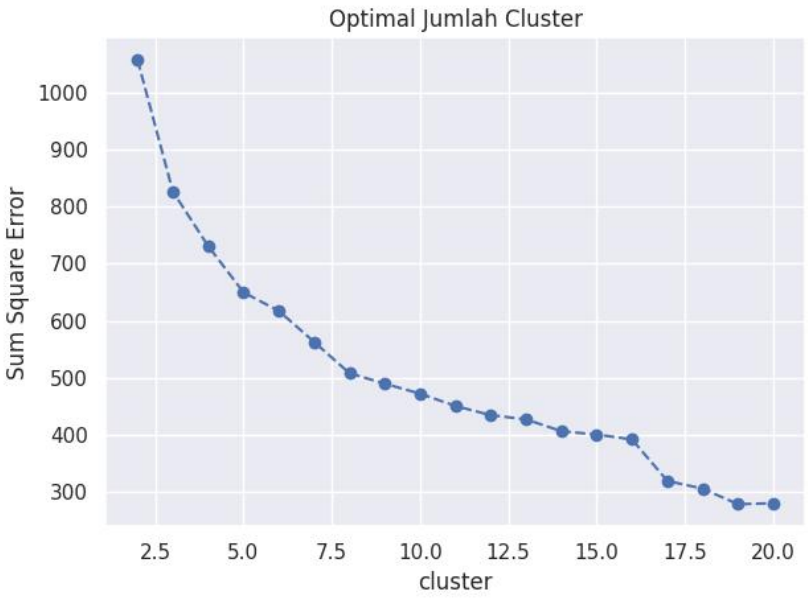
Evaluasi hasil

4.1. Hasil Kmeans

Sebelum menetapkan cluster yang ingin dibuat pada Kmeans, terlebih dahulu dilakukan evaluasi menggunakan silhoutte score dan elbow method dengan rentang kluster dari 2 sampai dengan 20 . Hasil graifknya bisa dilihat pada gambar 4.1 dan 4.2.



Gambar 4.1. grafik Silhoutte

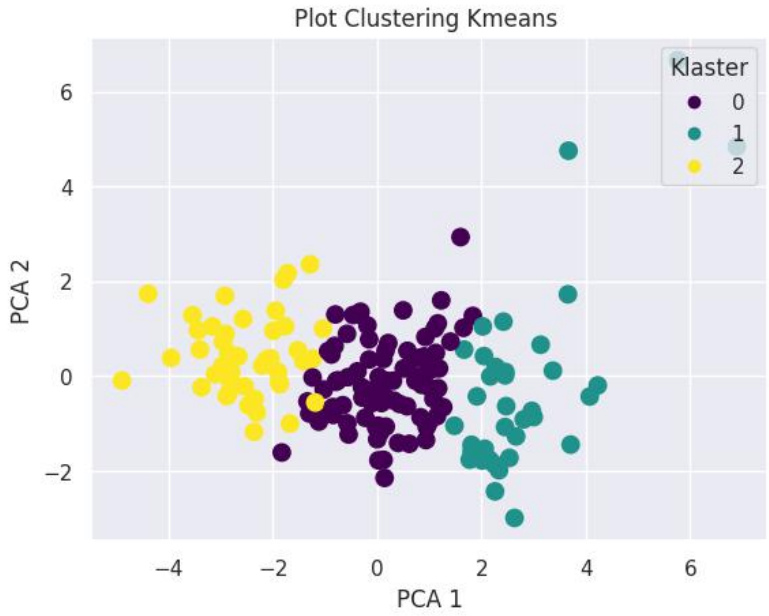


Gambar 4.2. Elbow grafik

Jika diperhatikan pada silhoutte score yang tinggi berada dititk 3 kurang lebih dengan nilai silhoutte nya di sekitar 0.30 sedangkan titik elbow juga berada pada

cluster sama dengan 3. Maka berdasarkan pada hal tersebut maka cluster yang akan dibuat sebanyak 3.

Hasil visualisasi dari jumlah kluster sama dengan 3 bisa dilihat pada gambar 4.3.



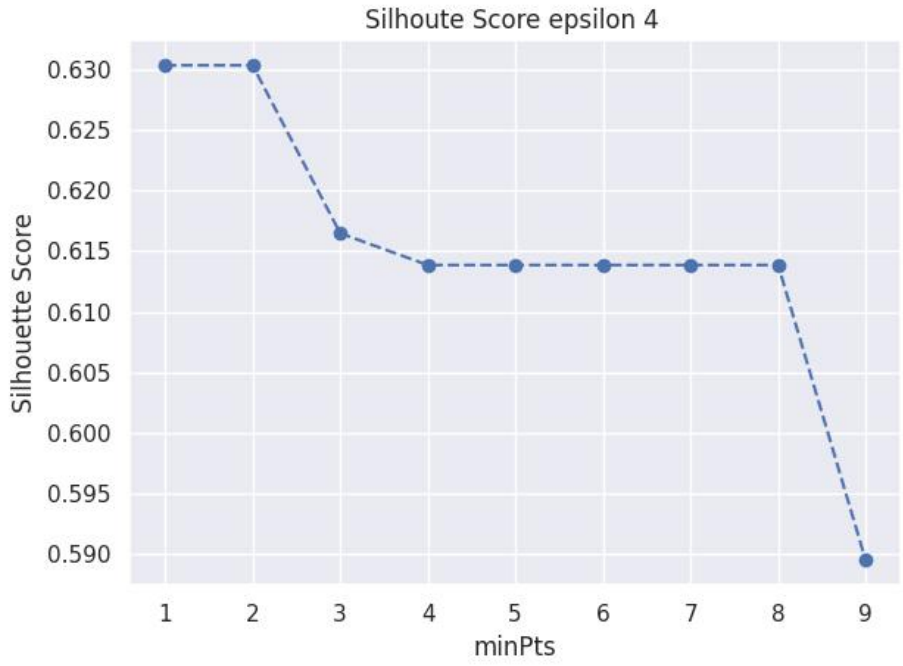
Gambar4.3.Visualisasi hasil clustering k = 3

Jika dilihat dari hasil visualisasi clustering ,Kmeans sudah cukup baik untuk mengklasterisasi data . terlihat bagaimana data-data terluar yang sejajar dengan clusternya bersesuaian satu sama lain. Walaupun sebenarnya jika kita berbicara tentang kepadatan kumpulan data ini harusnya berada dalam satu klaster. Namun pada kasus-kasus tertentu Kmeans dengan pembagian kluster seperti digambar 4.3. sudah cukup baik, karena tidak banyak data yang tumpang tindih dan berpusat pada klasternya masing-masing.

4.2. Hasil DBScan

Pada DBscan pada mulanya untuk menentukan epsilon atau radius yang optimal . Di cari rata-rata dari jarak setiap data ke data yang lainnya. Hal ini dilakukan supaya epsilon yang dipilih tidak terlalu kecil atau terlalu jauh. Ketika dilakukan pengukuran di dapatkan bahwa rata-rata jarak nya adalah **3.78975403066861 .** Dari sini ditentukan untuk permulaan epsilon berada pada jarak 4 , sengaja di bulatkan untuk memudahkan.

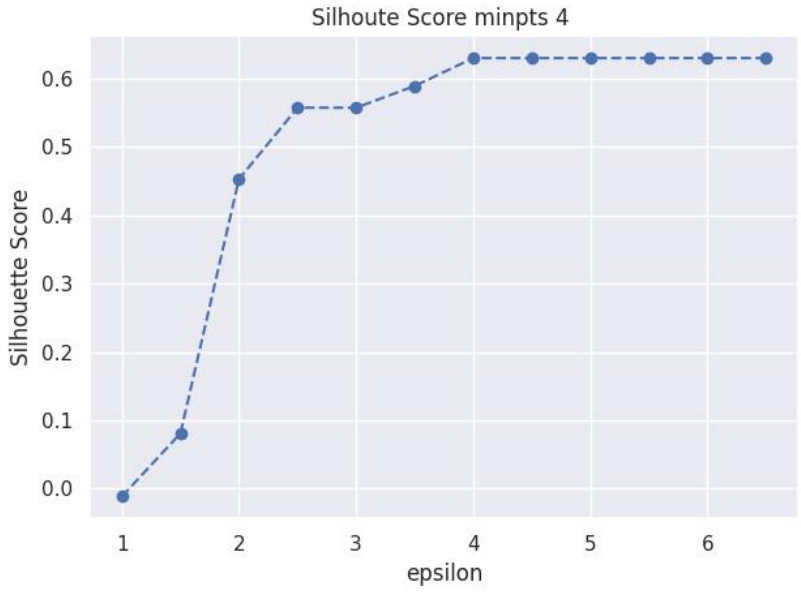
Lalu untuk menentukan minPts digunakan silhoutte score untuk menentukan minpts permulaan yang baik. minPts direntang 2 - 9. Hasilnya bisa dilihat pada gambar 4.4.



Gambar 4.4. Silhoutte score epsilon 4

JIka diperhatikan bahwa pada minPts 1 dan minPts 2 memiliki nilai silhoutte yang tinggi. Dikarenakan pertimbangan satu terlalu sedikit untuk dibilang padat, maka dipilhlah minPts = 2.

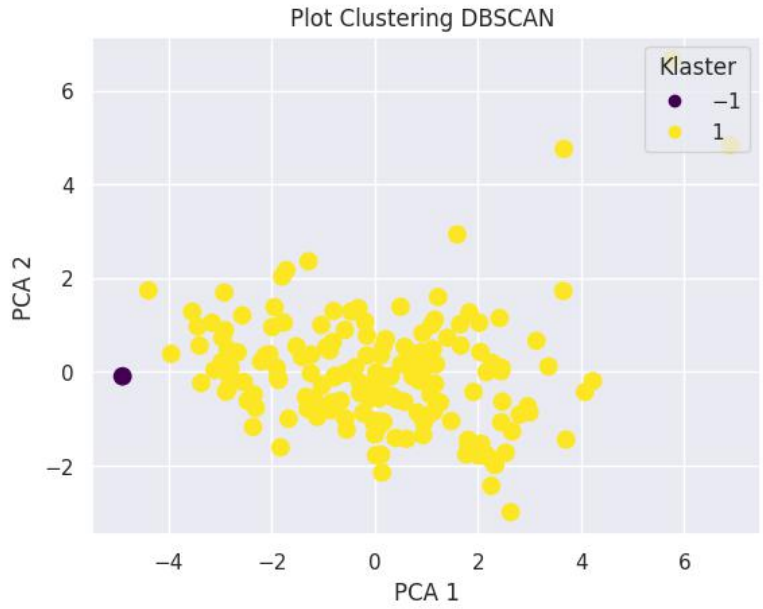
Lalu kemudian dengan minPts=2 juga di lihat kembali silhoutte scorenya dengan epsilon yang berubah-rubah dengan harapan akan menemukan silhoutte score yang lebih tinggi. epsilon yang diuji ada di rentang 1-7. dengan hasilnya bisa dilihat pada gambar 4.5.



Gambar 4.5. Silhoute score minpts = 2

Jika diperhatikan nilai epsilon yang mempunyai silhoutte score yang tinggi ada di $\varepsilon \geq 4$.

Berdasarkan uji silhoutte score sebelumnya maka akan dipilih **mintPts = 2** dan **ε = 4.** Sehingga hasil implementasi DBscan nya bisa dilihat pada Gambar 4.6.



Gambar 4.6. visualisasi clustering DBscan

Dari hasil visualisasi terlihat bahwa hanya ada satu cluster dengan outliers. Karena memang secara kepadatan data cukup rapat sehingga cluster yang terbentuk hanya satu dengan adanya outliers.

4.3. Kesimpulan

Dilandaskan pada hasil visualisasi Kmeans dengan parameter cluster = 3 dam DBscan dengan ε = 4 dan minpts = 2. Didapatkan bahwa jika sudut pandangnya adalah pola yang terbentuk pada data ketika di visualisasikan maka DBScan lebih baik dalam mengclusterisasi karena clusternya memang hanya ada satu . Namun dikarenakan pada studi kasus ini diminta untuk mengkategorikan negara. Maka Kmeans lebih tepat dalam menklasterisasikan, karena terdapat lebih dari satu klaster dengan data yang tersebar cukup merata dan baik .

Daftar Pustaka

Prof. Dr. Suyanto, S.T, M.sc.2022. Machine learning tingkat dasar dan lanjut. Bandung. Penerbit Informatika

Dr. Suyanto. 2019. Data Mining untuk klasifikasi dan klasterisasi data. Bandung. Penerbit Informatika

PPT Pembelajaran Mesin Universitas Telkom SSD-ADF-ATW-SUY

Lampiran

link github : *https://github.com/khalifardy/klasterisasi_negara*

linkdrive: :
https://www.dropbox.com/scl/fo/1itlgv883gzkl6rvhnq0l/h?rlkey=19j3gpj2mlrffjhja3a0niy7m&dl=0