LAPORAN CASE-BASED 2
PEMBELAJARAN MESIN

Disusun oleh

Nama : Anyelir Belia Azzahra
NIM : 1301200048
Kelas : IF-44-08 [DDR]

Pernyataan : Saya mengerjakan tugas ini dengan cara yang tidak melanggar aturan perkuliahan dan kode etik akademisi

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM

2022/2023

DAFTAR ISI

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Laporan ini dibuat untuk memenuhi tugas besar Case Based-2 mata kuliah Pembelajaran Mesin untuk melakukan beberapa analisis data unsupervised dengan dataset learning on country data (https://www.kaggle.com/datasets/rohan0301/unsupervised-learning-on-country-data). Algoritma unsupervised yang saya gunakan untuk mengimplementasikan dan menganalisis adalah K-Means. K-Means clustering terdapat nilai K yang merupakan cluster yang dibentuk. K merujuk pada jumlah centroid yang dibutuhkan pada dataset. Saya memilih menggunakan algoritma ini karena merupakan metode pengolahan data yang cukup mudah diimplementasikan dan dipahami serta membutuhkan waktu yang relatif cepat.

1.2 Rumusan Masalah

Rumusan masalah pada laporan ini adalah sebagai berikut.

a. Apa algoritma unsupervised yang digunakan?
b. Bagaimana ikhtisar kumpulan data yang dipilih?
c. Bagaimana pra-pemrosesan data yang diusulkan?
d. Bagaimana menerapkan algoritma yang dipilih?
e. Bagaimana evaluasi hasil yang diperoleh?

1.3 Tujuan

Tujuan dibuatnya laporan ini adalah sebagai berikut.

a. Menjelaskan algoritma unsupervised yang digunakan (k-means/dbscan/hierarchical)
b. Menjelaskan ikhtisar kumpulan data yang dipilih
c. Menjelaskan pra-pemrosesan data yang diusulkan
d. Menjelaskan penerapan algoritma yang dipilih
e. Menjelaskan evaluasi hasil yang diperoleh

BAB 2
PEMBAHASAN

2.1 Ikhtisar Data

Berikut ini merupakan ikhtisar data yang dipilih berdasarkan data dari aturan NIM Genap yaitu dataset learning on country data. Penerapannya menggunakan bahasa Python untuk menganalisis data data dan menunjukkan pra-pemrosesan data yang diperlukan. Pada ikhtisar data ini terdapat tabel dan plot yang terstruktur mengenai data learning on country data.

2.1.1 Data yang dipilih

Berikut ini merupakan code untuk menampilkan data yang dipilih dari dataset learning on country. Serta menggunakan head() untuk menampilkan data awal atau data teratas pada dataframe seperti berikut ini.

```
[14] data = pd.read_csv('Country-data.csv', index_col=0)
     #menampilkan dataset
     data.head()
```

1 to 5 of 5 entries Filter

country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553
Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090
Algeria	27.3	38.4	4.17	31.4	12900	16.1	76.5	2.89	4460
Angola	119.0	62.3	2.85	42.9	5900	22.4	60.1	6.16	3530
Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200

Show 25 per page
Like what you see? Visit the data table notebook to learn more about interactive tables.

2.1.2 Menampilkan Info dari Data yang dipilih

Menampilkan info data berupa nama column dan tipe datanya sebagai berikut. Dapat dilihat bahwa tipe data pada dataset tersebut adalah float64 dan int64.

```
#untuk menampilkan info data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 167 entries, Afghanistan to Zambia
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   child_mort  167 non-null    float64
 1   exports     167 non-null    float64
 2   health      167 non-null    float64
 3   imports     167 non-null    float64
 4   income      167 non-null    int64  
 5   inflation   167 non-null    float64
 6   life_expec  167 non-null    float64
 7   total_fer   167 non-null    float64
 8   gdpp        167 non-null    int64  
 9   Cluster     167 non-null    float64
dtypes: float64(8), int64(2)
memory usage: 18.4+ KB
```

2.1.3 Menampilkan Deskripsi dari Data yang dipilih

Menampilkan deskripsi data dari country-data. Deskripsi ini berisi count, min, max, mean, std, dll dari setiap kategori.

```
[22] data.describe()
```

1 to 8 of 8 entries Filter

index	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	Cluster
count	167.0	167.0	167.0	167.0	167.0	167.0	167.0	167.0	167.0	167.0
mean	38.27005988023952	41.108976047904186	6.8156886227544895	46.89021497005987	17144.688622754493	7.781832335329342	70.55568862275449	2.947964071856287	12964.155688622754	1.3712574850299402
std	40.32893145927617	27.412010111424 16	2.7468374978890795	24.209588976108698	19278.067697657672	10.570703901430559	8.893171908900408	1.5138475432630463	18328.704808675564	0.6205536668289433
min	2.6	0.109	1.81	0.0659	609.0	-4.21	32.1	1.15	231.0	0.0
25%	8.25	23.8	4.92	30.2	3355.0	1.81	65.3	1.795	1330.0	1.0
50%	19.3	35.0	6.32	43.3	9960.0	5.39	73.1	2.41	4660.0	1.0
75%	62.1	51.349999999999994	8.600000000000001	58.75	22800.0	10.75	76.8	3.88	14050.0	2.0
max	208.0	200.0	17.9	174.0	125000.0	104.0	82.8	7.49	105000.0	2.0

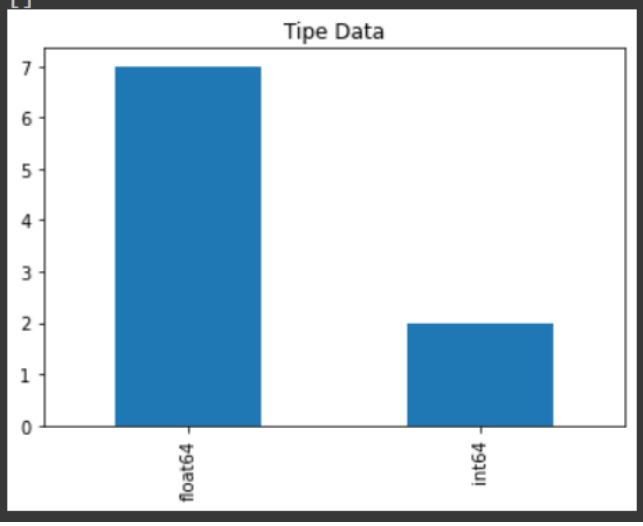
Show 25 per page
Like what you see? Visit the data table notebook to learn more about interactive tables.

2.1.4 Menampilkan Tipe Data

Pada bagian ini berisi code untuk menampilkan tipe data dari dataset Country Data menggunakan plot. Terlihat bahwa tipedata pada dataset tersebut adalah float dan integer.

```
tipe = data.dtypes.value_counts()
tipe = tipe.plot.bar()
tipe.set_title("Tipe Data")
tipe.plot()
```

```
[]
```



2.1.5 Menampilkan Data

Berikut ini terlihat data yang ditampilkan sebagai berikut.

data

1 to 25 of 167 entries Filter

country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	Cluster
Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553	0.0
Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090	1.0
Algeria	27.3	38.4	4.17	31.4	12900	16.1	76.5	2.89	4460	2.0
Angola	119.0	62.3	2.85	42.9	5900	22.4	60.1	6.16	3530	0.0
Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200	1.0
Argentina	14.5	18.9	8.1	16.0	18700	20.9	75.8	2.37	10300	2.0
Armenia	18.1	20.8	4.4	45.3	6700	7.77	73.3	1.69	3220	2.0
Australia	4.8	19.8	8.73	20.9	41400	1.16	82.0	1.93	51900	2.0
Austria	4.3	51.3	11.0	47.8	43200	0.873	80.5	1.44	46900	1.0
Azerbaijan	39.2	54.3	5.88	20.7	16000	13.8	69.1	1.92	5840	2.0
Bahamas	13.8	35.0	7.89	43.7	22900	-0.393	73.8	1.86	28000	1.0
Bahrain	8.6	69.5	4.97	50.9	41100	7.44	76.0	2.16	20700	1.0
Bangladesh	49.4	16.0	3.52	21.8	2440	7.14	70.4	2.33	758	2.0
Barbados	14.2	39.5	7.97	48.7	15300	0.321	76.7	1.78	16000	1.0
Belarus	5.5	51.4	5.61	64.5	16200	15.1	70.4	1.49	6030	1.0
Belgium	4.5	76.4	10.7	74.7	41100	1.88	80.0	1.86	44400	1.0
Belize	18.8	58.2	5.2	57.5	7880	1.14	71.4	2.71	4340	1.0
Benin	111.0	23.8	4.1	37.2	1820	0.885	61.8	5.36	758	0.0
Bhutan	42.7	42.5	5.2	70.7	6420	5.99	72.1	2.38	2180	1.0
Bolivia	46.6	41.2	4.84	34.3	5410	8.78	71.6	3.2	1980	2.0
Bosnia and Herzegovina	6.9	29.7	11.1	51.3	9720	1.4	76.8	1.31	4610	1.0
Botswana	52.5	43.6	8.3	51.3	13300	8.92	57.1	2.88	6350	0.0
Brazil	19.8	10.7	9.01	11.8	14500	8.41	74.2	1.8	11200	2.0
Brunei	10.5	67.4	2.84	28.0	80600	16.7	77.1	1.84	35300	1.0
Bulgaria	10.8	50.2	6.87	53.0	15300	1.11	73.9	1.57	6840	1.0

Show 25 per page 1 2 3 4 5 6 7

2.2 Ringkasan Pra-Pemrosesan Data

Pada bagian ini berisi penjelasan pra-pemrosesan data yang diusulkan, metode/teknik, hasil pendekatan, dan alat analisis yang digunakan.

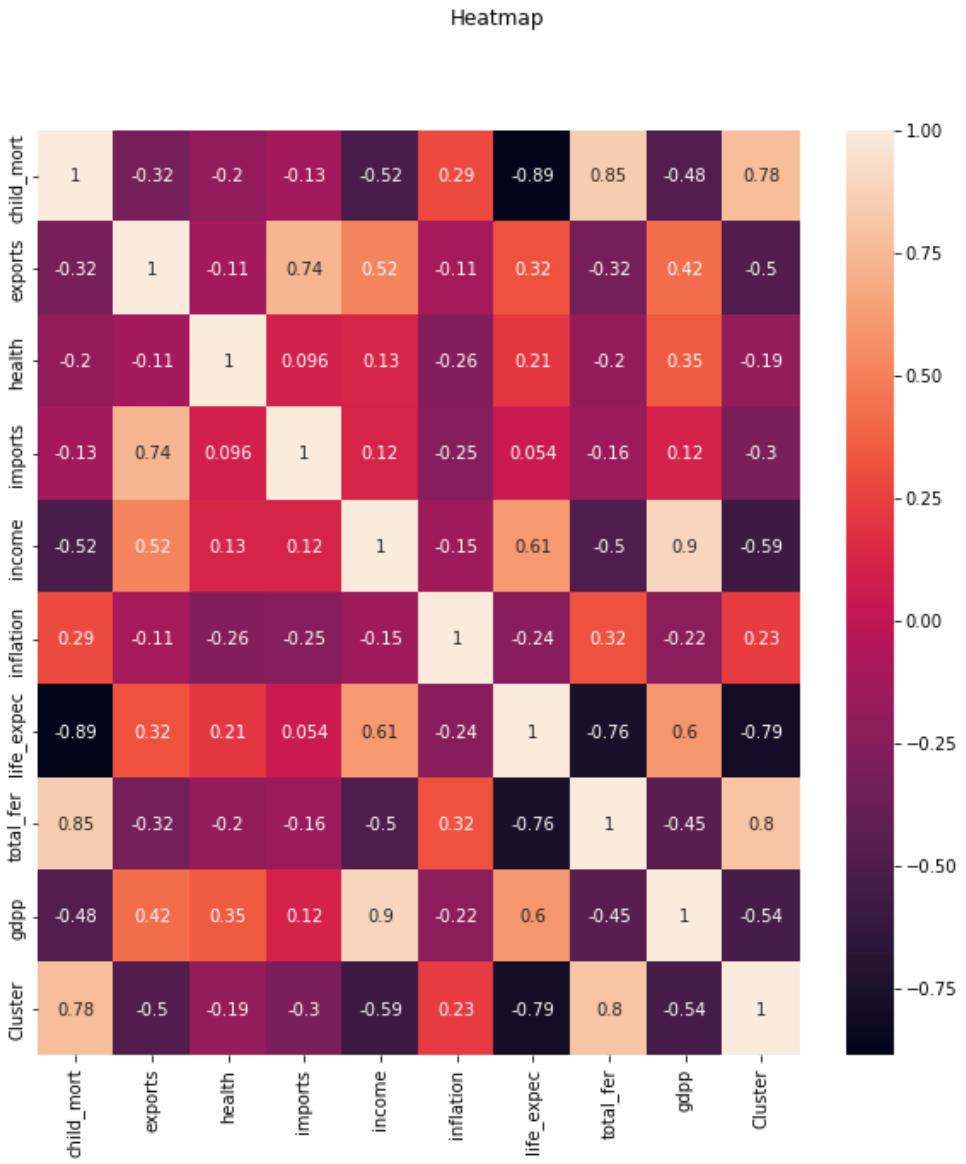
2.2.1 Membuat heatmap atau Mencari Korelasi

Mengeksplorasi data dengan menampilkan korelasi pada data set Country-data menggunakan heatmap. Korelasi digunakan untuk melihat skala korelasi data dari

yang paling besar dan paling kecil. Korelasi yang tidak ada atau kecil bisa di drop karena tidak diperlukan.

```
#membuat heatmap/korelasi
corr = data.corr()
top_corr_features = corr.index
plt.figure(figsize = (10,10))
plt.suptitle("Heatmap")

#plot heat map
data1 = sns.heatmap(data[top_corr_features].corr(), annot=True)
```

Dan menghasilkan heatmap korelasi sebagai berikut.



2.2.2 Data yang digunakan

Berikut ini merupakan data data yang dipilih untuk digunakan pada pemrosesan selanjutnya. Data ini dipilih berdasarkan korelasi yang paling besar pada heatmap di atas (dapat berbeda tergantung pemrosesan yang dilakukan).

```
[4] data12 = data[['exports','imports','child_mort','life_expec','total_fer']]
    print(data12)
```

```
     exports  imports  child_mort  life_expec  total_fer
0       10.0     44.9        90.2        56.2       5.82
1       28.0     48.6        16.6        76.3       1.65
2       38.4     31.4        27.3        76.5       2.89
3       62.3     42.9       119.0        60.1       6.16
4       45.5     58.9        10.3        76.8       2.13
..       ...      ...         ...         ...        ...
162     46.6     52.7        29.2        63.0       3.50
163     28.5     17.6        17.1        75.4       2.47
164     72.0     80.2        23.3        73.1       1.95
165     30.0     34.4        56.3        67.5       4.67
166     37.0     30.9        83.1        52.0       5.40

[167 rows x 5 columns]
```

2.2.3 Mengecek Data yang Error

Berikut ini merupakan code untuk mengecek data yang dipilih apakah data tersebut terdapat error atau tidak. Pada data yang dipilih ternyata tidak memiliki data error.

```
data12.isna().sum()
```

```
exports       0
imports       0
child_mort    0
life_expec    0
total_fer     0
dtype: int64
```

2.2.4 Mengecek Data yang Null

Berikut ini merupakan code untuk mengecek apakah data yang dipilih terdapat data bernilai null atau tidak. Terlihat pada gambar berikut, data yang bernilai null tidak ada.

```
[37] #mengecek null -> tidak ada data yang null
     data12.isnull().sum()

     exports       0
     imports       0
     child_mort    0
     life_expec    0
     total_fer     0
     dtype: int64
```

Apabila ada data yang null dapat didrop sebagai berikut.

```
[11] data12.dropna().sum()

     exports        6865.1990
     imports        7830.6659
     child_mort     6391.1000
     life_expec    11782.8000
     total_fer       492.3100
     dtype: float64
```

2.2.5 Mengecek Data yang Duplikat

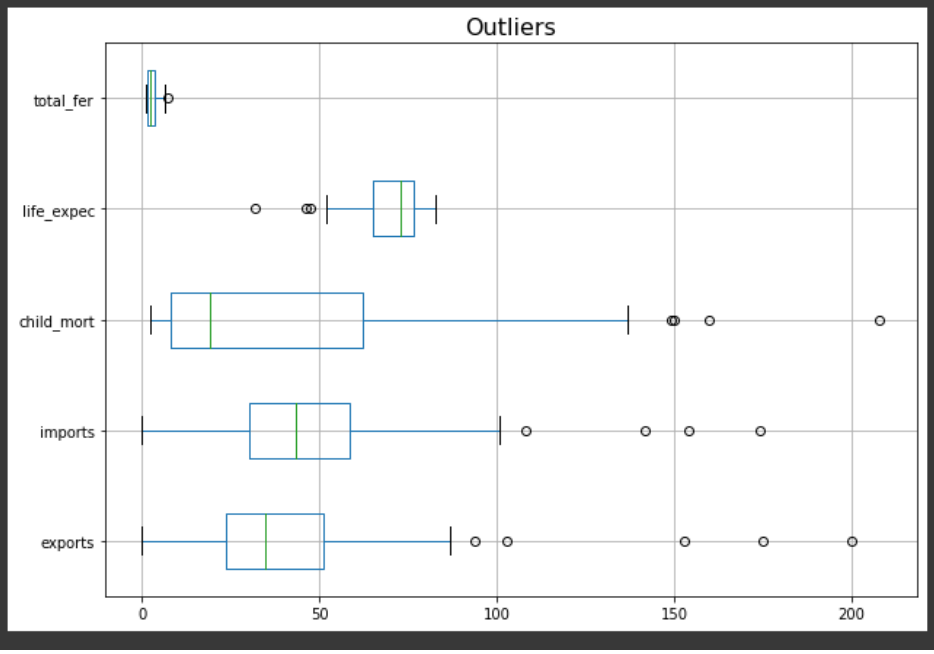
Pada code berikut ini untuk mengecek apakah data yang dipilih ada yang terduplikat atau tidak. Dan terlihat bahwa datanya tidak ada yang duplicate.

```
#mengecek data duplikat -> tidak ada data yang duplikat
data12.duplicated().sum()

0
```

2.2.6 Mengecek Outlier

Code berikut digunakan untuk mengecek outlier data yang dipilih dengan menampilkan plot dan bloxpot nya.

```
[44] #mengecek outliers
     cont=data12.dtypes[(data12.dtypes!='object')].index #assign data12 per index
     plt.figure(figsize=(10,7)) #bikin plot
     data12[cont].boxplot(vert=0) #bikin bloxpot nya
     plt.title('Outliers',fontsize=16)
     plt.show()
```



2.2.7 Menghilangkan Outlier

Outlier dihilangkan karena data tersebut menyimpang terlalu jauh dari data yang lainnya dalam suatu pencilan dan menghasilkan data yang samar, sehingga perlu dihilangkan.

2.2.7.1 Menghitung Jarak Interquartile

Fungsi ini digunakan untuk menentukan lower range dan upper range seperti berikut. Lower range dengan rumus Q1 - (1,5 * IQR), sedangkan upper range = Q3 + (1,5 * IQR). Dimana Q1 : Quartile 1, dan Q3 : Quartile 3, serta IQR : jangkauan interkuartil.

```
[15] #menghitung jarak interquartile
     def interquartile(col):
         sorted(col)
         Q1,Q3=np.percentile(col,[25,75])
         IQR=Q3-Q1
         lower_range= Q1-(1.5 * IQR)
         upper_range= Q3+(1.5 * IQR)
         return lower_range, upper_range
```

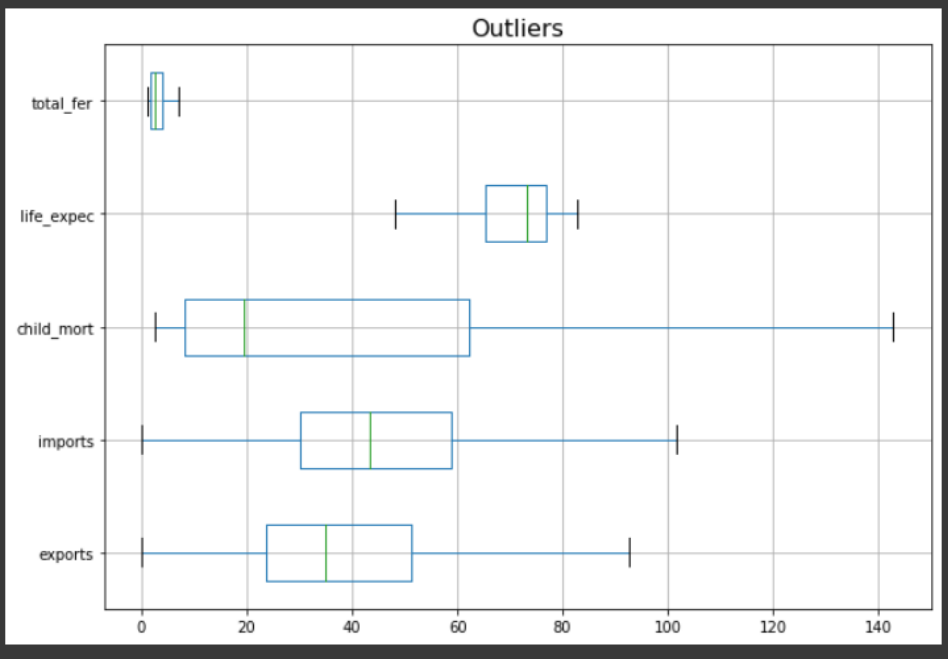
```
[16] for column in data12[cont].columns:
         print(column)
         lr,ur=interquartile(data12[column])
         data12[column]=np.where(data12[column]>ur,ur,data12[column])
         data12[column]=np.where(data12[column]<lr,lr,data12[column])
```

```
exports
imports
child_mort
life_expec
total_fer
```

2.2.7.2 Mengecek Outlier Setelah Handling Outlier

Pada code berikut ini terlihat hasil setelah handling adalah semua outlier telah berhasil dihilangkan. Sehingga sudah dapat diolah lebih lanjut.

```
#mengecek outliers setelah handling outliers
cont=data12.dtypes[(data12.dtypes!='object')].index #assign data12 per index
plt.figure(figsize=(10,7)) #bikin plot
data12[cont].boxplot(vert=0) #bikin bloxpot nya
plt.title('Outliers',fontsize=16)
plt.show()
```



2.2.8 Scaling

Scaling digunakan untuk menormalisasikan data yang ada atau penyeragaman skala antar atribut. Sebelum discaling, data per atributnya memiliki jarak yang jauh, sehingga perlu diadakannya scaling.

```
#Scalling
#untuk penyeragaman skala antar atribut (Normalisasi)
scale = StandardScaler()
scale.fit(data12)
temp =scale.transform(data12)
data_baru = pd.DataFrame(temp, index=data12.index, columns=data12.columns)

data_baru
```

Hasil setelah scaling adalah sebagai berikut. Terlihat jarak antar atribut sudah dekat.

1 to 25 of 167 entries Filter

country	exports	imports	child_mort	life_expec	total_fer
Afghanistan	-1.3911068421944566	-0.04744449056008515	1.3698019371510812	-1.7022250384322908	1.9152760192689342
Albania	-0.5435473131936275	0.1350208036751986	-0.5504642151233057	0.6633212517023314	-0.862779263729723
Algeria	-0.0538462519931484	-0.7131962397969307	-0.27129508700732824	0.6868590257533522	-0.03669088221453472
Angola	1.0715244559579524	-0.1460743793359141	2.1212104315193194	-1.2432384448240805	2.1417841238779376
Antigua and Barbuda	0.28046889555717874	0.6429647308707176	-0.7148348232663578	0.722165686780307	-0.5430031160464244
Argentina	-0.9720357417440467	-1.4726463833708137	-0.6052544178376564	0.6044768166243556	-0.38311504220477494
Armenia	-0.882571124793959	-0.027718512804919423	-0.5113283560416265	0.3102546412344772	-0.8361312514227814
Australia	-0.9296577652940051	-1.2310031558700327	-0.8583329732325145	1.3341478115912546	-0.6762431775811322
Austria	0.5535714104574457	0.09556884816486678	-0.8713782595930742	1.1576145063573275	-1.0026813283411662
Azerbaijan	0.6948313319575584	-1.2408661447476157	0.03918272837399249	-0.18403861342051897	-0.6829051806578675
Bahamas	-0.21394082969330497	-0.1066224238558232	-0.6235178187424399	0.3690990763124528 7	-0.7228771991182797
Bahrain	1.4105482675582843	0.2484451757674017 4	-0.7591887968922607	0.6280145906555462	-0.523017106816218
Bangladesh	-1.1085869991941801	-1.1866197059209096	0.3053065701294101	-0.03104308221778082	-0.4097630545117165
Barbados	-0.002050947443097665	0.1399522981139901	-0.6130815896539922	0.7103967997647126	-0.7761732237321629
Belarus	0.5582800745074504	0.9191284194430388	-0.8400695723277308	-0.03104308221778082	-0.9693713129574892
Belgium	1.7354460870086024	1.4221408521997667	-0.8661601450488502	1.0987700712793518	-0.7228771991182797
Belize	0.8784692299077639	0.5739238087276374	-0.493064955136843	0.08664578793817056	-0.15660693759577182
Benin	-0.7413112032938208	-0.4271695623470264 7	1.9124858497503645	-1.0431673655589637	1.6088238777391064
Bhutan	0.13920897405704052	1.2248810746481087	0.13049973289791034	0.1690279970473352	-0.3764530391280397
Bolivia	0.07799634140698078	-0.5701829010719788	0.2322529665102759	0.1101835619693595	0.16983121316426236
Bosnia and Herzegovina	-0.4635000243435491 4	0.2681711535225675	-0.8035427705181638	0.722165686780307	-1.089287368338726
Botswana	0.19100427860709127	0.2681711535225675	0.3861873455648802	-1.5963050552919347	-0.04335288529127026
Brazil	-1.3581461938444244	-1.6797691498000547	-0.4669743824157264	0.4161746243748341	-0.7628492175786922
Brunei	1.311666322508188	-0.8808670507158398	-0.7096167087221339	0.7574723478270922	-0.7362012052717506
Bulgaria	0.5017761059073954	0.3520065589820222	-0.7017895369057982	0.380867963328049	-0.9160752883436061

2.2.9 Feature Selection

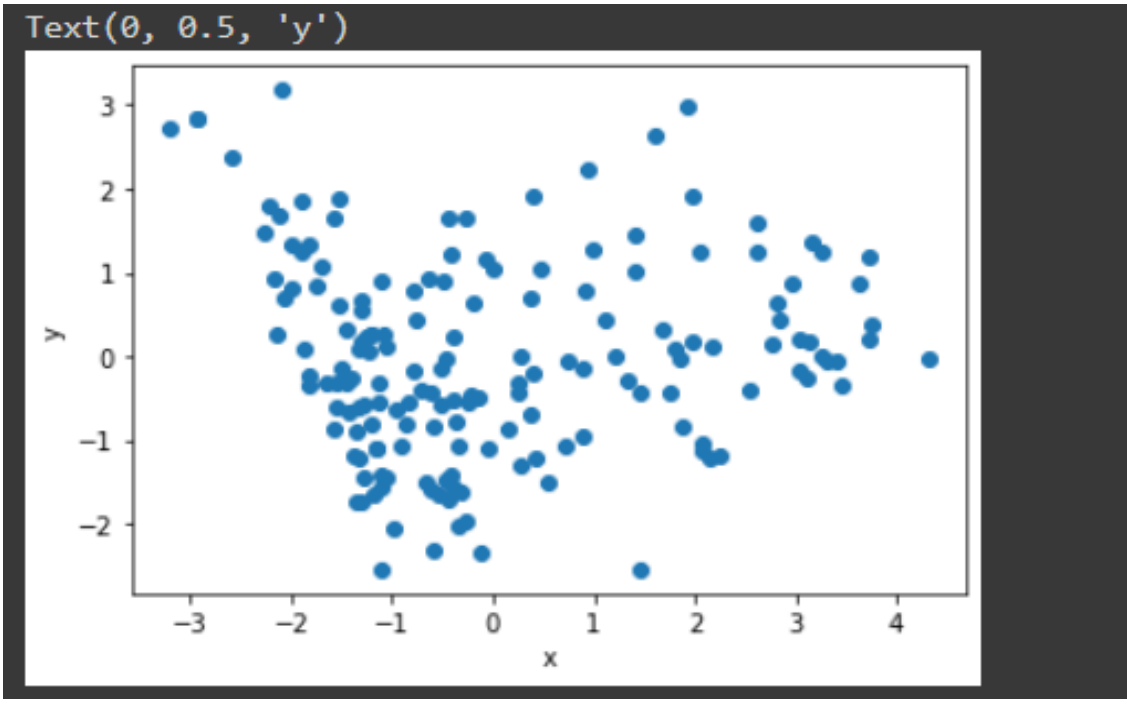
Dilakukan feature selection dimana akan memilih beberapa value yang penting untuk proses clustering. Disini saya menggunakan bantuan PCA dalam proses feature selection. Digunakan untuk dekomposisi kolom yang dipakai dan menampilkan penyebaran data.

```
#PCA
#dekomposisi kolom dipakai, menampilkan penyebaran data
test = PCA(n_components=2)
data_cluster = test.fit_transform(data_baru)
data_cluster = pd.DataFrame(data = data_cluster, columns = ['x', 'y'])
data_cluster

arrayScal = np.array(list(zip(data_cluster['x'], data_cluster['y'])))
print(arrayScal)

x=arrayScal[:,0]
y=arrayScal[:,1]
sns.scatterplot(x, y, s=5, color='grey')

plt.scatter(data_cluster['x'], data_cluster['y'])
plt.xlabel('x')
plt.ylabel('y')
```

Hasil yang ditampilkan sebagai berikut. Berikut ini merupakan hasil dekomposisi kolom

```
[[ 3.13047778e+00  1.64852551e-01]
 [-9.51800577e-01 -6.34081488e-01]
 [-3.80755280e-01 -7.82445614e-01]
 [ 2.62466165e+00  1.59221561e+00]
 [-1.26753152e+00  2.24634010e-01]
 [-2.63990027e-01 -1.97278463e+00]
 [-5.89961029e-01 -8.49782988e-01]
 [-9.97572863e-01 -2.05220670e+00]
 [-1.81748696e+00 -2.32564121e-01]
 [-2.52786218e-01 -5.62959903e-01]
 [-8.29646358e-01 -5.35049602e-01]
 [-1.53595399e+00  6.13159913e-01]
 [ 5.47431634e-01 -1.51554460e+00]
 [-1.14522737e+00 -3.24274158e-01]
 [-1.31188360e+00  6.81844418e-01]
 [-2.26617409e+00  1.48748676e+00]
 [-7.94241775e-01  7.89149499e-01]
 [ 2.77407434e+00  1.51704969e-01]
 [-4.83874913e-01  8.94696182e-01]
 [ 2.37261682e-01 -3.26109374e-01]
 [-1.29260414e+00 -5.91097428e-01]
 [ 9.10793782e-01  7.74031459e-01]
 [-1.26407078e-01 -2.33928218e+00]
 [-1.45347640e+00 -3.28278104e-01]
 [-1.30171825e+00  1.66426199e-01]
 [ 3.40560458e+00 -5.09873799e-02]
 [ 3.31055171e+00 -4.82815198e-02]
 [ 4.77311244e-03  1.05117216e+00]
 [ 3.03462203e+00 -1.69411113e-01]
 [-1.28451996e+00 -1.44236310e+00]
 [-4.05523539e-01  2.45915507e-01]
 [ 4.30507247e+00 -1.53693362e-02]
 [ 3.73346459e+00  1.18736383e+00]
 [-1.15329834e+00 -1.08734385e+00]
 [-6.23887432e-01 -1.57815216e+00]
 [-3.38892687e-01 -2.02801021e+00]
 [ 1.96437810e+00  1.71440723e-01]
 [ 3.14786296e+00  1.35685204e+00]
```

Dan berikut ini tampilan penyebaran data



BAB 3
PENERAPAN ALGORITMA K-MEANS

Algoritma unsupervised learning yang digunakan dalam tugas ini adalah K-Means. Algoritma ini dipilih karena merupakan algoritma yang paling mudah dipahami dan prosesnya relatif cepat.

Langkah pertama adalah menghitung jarak euclidean dengan membuat fungsi euclid

```
#menghitung jarak euclidean
def euclid(a, b, ax=1):
    kurang = a-b
    result_euclid = np.linalg.norm(kurang, axis=ax)
    return result_euclid
```

Kemudian membuat fungsi untuk menampilkan nilai centroid. Pada tugas kali ini K yang saya pakai adalah K = 3. Karena sebelumnya saya experimen nilai K yang lain, tetapi menghasilkan nilai yang kurang optimal. Kemudian nilai centroid ditampilkan

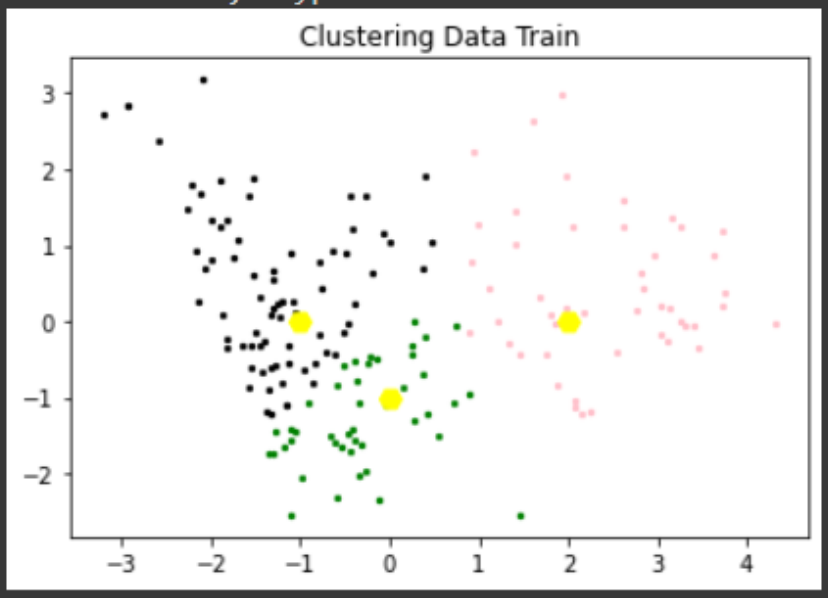
```
#menampilkan nilai centroid
k = 3 #banyaknya centroid
def centro(arrayScal,k):
  min = np.min(arrayScal)
  max = np.max(arrayScal)

#untuk mendapatkan nilai centroid
  centro1 = np.random.randint(min, max, size=k)
  centro2 = np.random.randint(min, max, size=k)
  centroid = np.array(list(zip(centro1, centro2)))
  return centroid
```

Kemudian menghitung nilai panjang arrayScal, dan memanggil fungsi centroid sebagai berikut.

```
#mengambil panjang dari arrayscal
arrCluster = np.zeros(len(arrayScal))
#mendapatkan nilai centroid
centroid = centro(arrayScal,k)
#array centroid dengan cara mengambil shaping dari nilai centroid
arrC = np.zeros(centroid.shape)
#list kosong untuk variable baru
titik = []
temp = []

#memanggil fungsi euclid
mark = euclid(centroid, arrC, None)
```

```
while mark != 0:
    for i in range(len(arrayScal)):
        jarak = euclid(arrayScal[i], centroid)
        cluster = np.argmin(jarak)
        arrCluster[i] = cluster
    arrC = deepcopy(centroid) #memanggil nilai centroid


    for i in range(k):
        titik = [arrayScal[j] for j in range(len(arrayScal)) if arrCluster[j] == i]
        centroid[i] = np.mean(titik, axis=0)
        temp.append(arrCluster)

    mark = euclid(centroid, arrC, None)

fig, ax = plt.subplots()
warna = ['pink', 'black', 'green', 'gold', 'silver', 'navy', 'red']

#menampilkan plot
for i in range(k):
        titik = np.array([arrayScal[j] for j in range(len(arrayScal)) if arrCluster[j] == i])
        x = titik[:, 0]
        y = titik[:, 1]
        ax.scatter(x, y, s=5, c=warna[i])

clusterx = centroid[:, 0]
clustery = centroid[:, 1]
ax.scatter(clusterx, clustery, marker='H', s=100, color='yellow')
plt.title("Clustering Data Train")
plt.show

print(centroid)

data["Cluster"] = arrCluster
data
#menampilkan value nya
data['Cluster'].value_counts()
```

```
[[ 2  0]
 [-1  0]
 [ 0 -1]]
1.0    76
0.0    47
2.0    44
Name: Cluster, dtype: int64
```

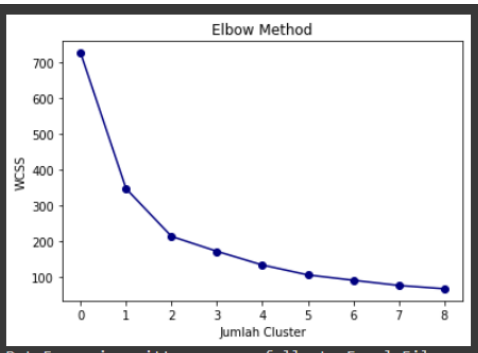


BAB 4
EVALUASI

Setelah itu, akan ditambahkan column baru pada dataframe sebelum scalling, yaitu column cluster, kemudian akan menjadi output berupa excel yang dimana akan diurutkan berdasarkan cluster

```
"""# EVAL"""

data_cluster



from sklearn.cluster import KMeans
wcss=[]
for n in range(1,10):
    km = KMeans(n_clusters= n, init='k-means++', random_state=50)
    km.fit(data_cluster)
    wcss.append(km.inertia_)
plt.plot(wcss,color= 'navy', marker='o')
plt.title('Elbow Method')
plt.xlabel('Jumlah Cluster')
plt.ylabel('WCSS')
plt.show()
```

Setelah proses clustering, akan muncul permasalahan baru,yaitu K berapa yang menghasilkan kluster terbaik. Untuk itu kita akan menggunakan elbow method untuk menentukan K terbaik. Disini kita akan melihat nilai WCSS terhadap jumlah cluster. Biasanya semakin besar kluster, nilai WCSS semakin kecil. Tujuan dari elbow method ini adalah melihat pada K berapa terjadi perubahan nilai yang signifikan. Artinya jika divisualisasikan, nilai K terbaik adalah saat membentuk sebuah siku




```
DataFrame is written successfully to Excel File.
```

evaluasi : berdasarkan elbow method jumlah cluster yang diambil dan optimal adalah 2. karena k = 2 merupakan nilai k yang paling nilai eror nya paling minimum dan tidak signifikan berbeda dengan nilai k lainnya

Dapat dilihat dari visualisasi, siku terbentuk saat K =2, artinya nilai K optimal untuk proses clustering pada dataset ini adalah 2

BAB 5
LAMPIRAN DAN REFERENSI

5.1 Lampiran

5.1.1 Source Code

https://colab.research.google.com/drive/1bt0HkIOve8FzjZAxnHpxoi0AcgW6ffT3?usp=sharing

5.1.2 Video Presentasi

https://youtu.be/f5Oa5lpagXw

5.2 Referensi

https://github.com/GOWTHAMJEEVANANTHAM/Unsupervised-Learning-on-Country-Data