Klasterisasi Pembangkit Listrik Energi Terbarukan Pada Negara di Dunia Pada tahun 2020 menggunakan metode Fuzzy Possibility C-Means

Latar Belakang

Akses ke listrik yang andal dan berkelanjutan sangat penting untuk pembangunan ekonomi, kesejahteraan sosial, dan kelestarian lingkungan. kumpulan data sumber pembangkit listrik yang komprehensif, termasuk energi terbarukan dan tidak terbarukan, dikumpulkan untuk beberapa negara. Dengan mengelompokkan negara-negara berdasarkan faktor pembangkit listriknya, penelitian ini berkontribusi pada pemahaman yang lebih mendalam tentang lanskap energi global

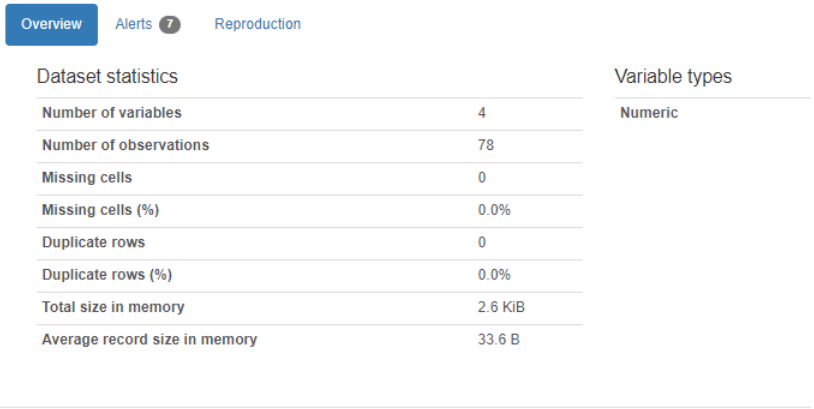
Tujuan

Tujuan pembangunan berkelanjutan, sebagaimana digariskan oleh Perserikatan Bangsa-Bangsa, mencakup berbagai tujuan, termasuk energi yang terjangkau dan bersih (Tujuan 7) dan aksi iklim (Tujuan 13). Mempromosikan Adopsi Energi Terbarukan: Analisis pengelompokan membantu mengidentifikasi negara-negara dengan strategi energi terbarukan yang berhasil dan pangsa sumber energi terbarukan yang tinggi dalam campuran pembangkit listrik mereka dalam mewujudkan energi yang terjangkau, bersih, dan berkelanjutan untuk semua.

Metodologi

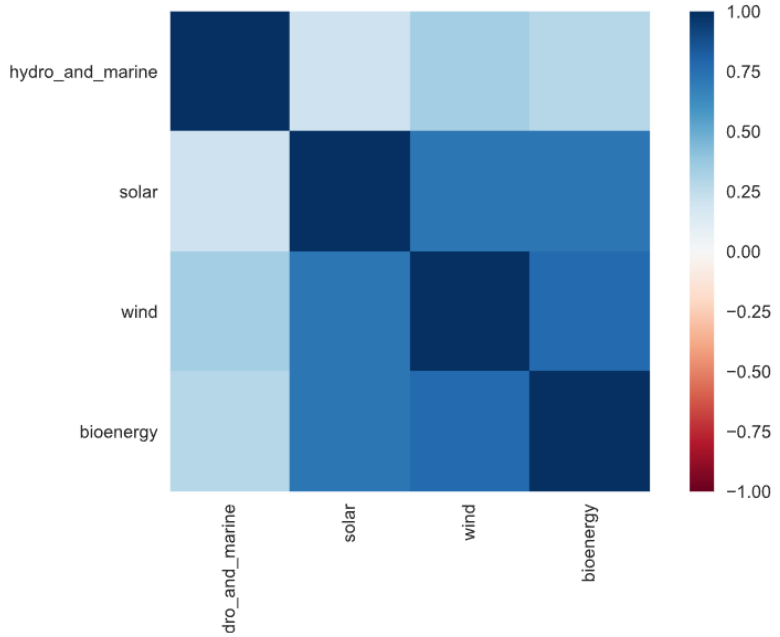

Data Selection
Prepocesing Data
Transformasi Data
Data Mining
Evaluasi

Visualisasi EDA FPCM



Pada data yang diolah dengan overview data cukup baik, tidak ada dayta kosong pada dataset yang akan diolah ini.

Visualisasi EDA FPCM



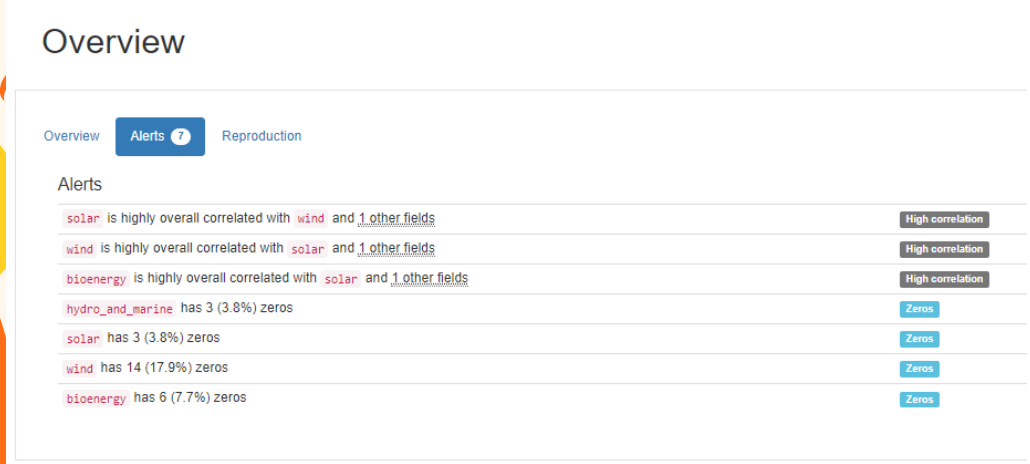
Pada data korelasi pada setiap variable yang ada memliki korelasi yang cukup baik, ditandakan dengan warna semakin menjadi biru gelap, hal ini menandakan bahwa hasil korelasi antar variable cukup baik.

Visualisasi EDA FPCM

	hydro_and_marine	solar	wind	bioenergy
hydro_and_marine	1.000	0.211	0.338	0.282
solar	0.211	1.000	0.726	0.725
wind	0.338	0.726	1.000	0.775
bioenergy	0.282	0.725	0.775	1.000

Pada data korelasi pada setiap variable yang ada memliki korelasi yang cukup baik, dibuktikan dengan data nilai pervariabel, korelasi tertinggi pada hubungan variable bio energy dengan data wind dan korelasi terrendah adalah variable solar dengan hydro and marine.

Visualisasi EDA FPCM



Pada data alert ditunjukkan ada data yang memiliki korelasi yang terlalu tinggi, hal ini menjadi sebuah peringatan yang bisa mnjadi kan data ini terlalu bias

Dan juga terdapat data dengan isi data 0, hal ini dikarenakan pad data per negara memiliki data sebenarnya kosong.

Hasil dan Analisis

1. Loading Dataset

```
import pandas as pd
df = pd.read_excel('renewable electricity generation.xlsx')
df = df.drop(["country"], axis = 1)
df
```

	hydro_and_marine	solar	wind	bioenergy
0	29685	1346	9412	2607
1	14764	21033	20396	3351
2	41998	2043	6792	4591
3	777	389	5	4
4	399	176	194	578
...	...	...	...	...
73	4094	462	5476	2701
74	5000	0	16	0
75	64501	8	88	0
76	73496	1660	1803	322
77	3882	34	0	283

78 rows × 4 columns

Melakukan loading dataset pada data rosir dan juga melakukan import library yang diperllukan pada pengolahan nantinya.

2. Melakukan transformasi pa

```
# Standardize the features
scaler = StandardScaler()
X = scaler.fit_transform(df)
# Perform PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
```

Melakukan transformasi pada data dengan menggunakan standar scalar lalu dilakukan pca untuk mendapatkan 2 komponen

3. Fuzzy Possibility C-Means

```
def fpcm(X, c, m, error, max_iters):

    # Initialize cluster centroids randomly
    centroids = np.random.rand(c, X.shape[1])

    # Initialize membership and possibility matrices
    U = np.random.rand(X.shape[0], c)
    P = np.random.rand(X.shape[0], c)

    # Main loop
    for i in range(max_iters):
        # Update membership matrix
        for j in range(X.shape[0]):
            for k in range(c):
                s = 0
                for l in range(c):
                    s += ((np.linalg.norm(X[j] - centroids[k]) / np.linalg.norm(X[j] - centroids[l])) ** (2 / (m - 1)))
                U[j][k] = 1 / s

        # Update possibility matrix
        for j in range(X.shape[0]):
            for k in range(c):
                s = 0
                for l in range(c):
                    s += ((np.linalg.norm(X[j] - centroids[k]) / np.linalg.norm(X[j] - centroids[l])) ** (2 / (m)))
                P[j][k] = 1 / s

        # Update cluster centroids
        old_centroids = centroids.copy()
        for j in range(c):
            centroids[j] = np.sum((U[:,j]**m * P[:,j]**(1-m)).reshape(-1,1) * X, axis=0) / np.sum(U[:,j]**m * P[:,j]**(1-m))

        # Check for convergence
        if np.linalg.norm(centroids - old_centroids) < error:
            break

    # Assign data points to clusters
    clusters = np.argmax(U, axis=1)

    # Evaluate the model using Silhouette score and Davies-Bouldin index
    sil_score = silhouette_score(X, clusters)
    db_index = davies_bouldin_score(X, clusters)

    # Plot the clustering
    plt.figure(figsize=(8, 6))
    cmap = ListedColormap(['r', 'g', 'b', 'y', 'm', 'c'])
    plt.scatter(X[:, 0], X[:, 1], c=clusters, cmap=cmap)
    plt.scatter(centroids[:, 0], centroids[:, 1], marker='*', s=200, color='k')
    plt.title(f'FPCM Clustering\nSilhouette Score: {sil_score:.3f}, Davies-Bouldin Index: {db_index:.3f}')
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.show()

    return clusters, sil_score, db_index
```

3. Fuzzy Possibility C-Means

M=2 yang merupakan nilai default umum yang digunakan dalam FCM. Nilai m yang lebih tinggi menunjukkan ketidakjelasan yang lebih besar, yang memungkinkan titik data menjadi milik beberapa cluster dengan berbagai tingkat keanggotaan.

Dengan k=3 yang merupakan pembagian cluster dengan 3 bagian yaitu rendah, sedang dan tinggi.

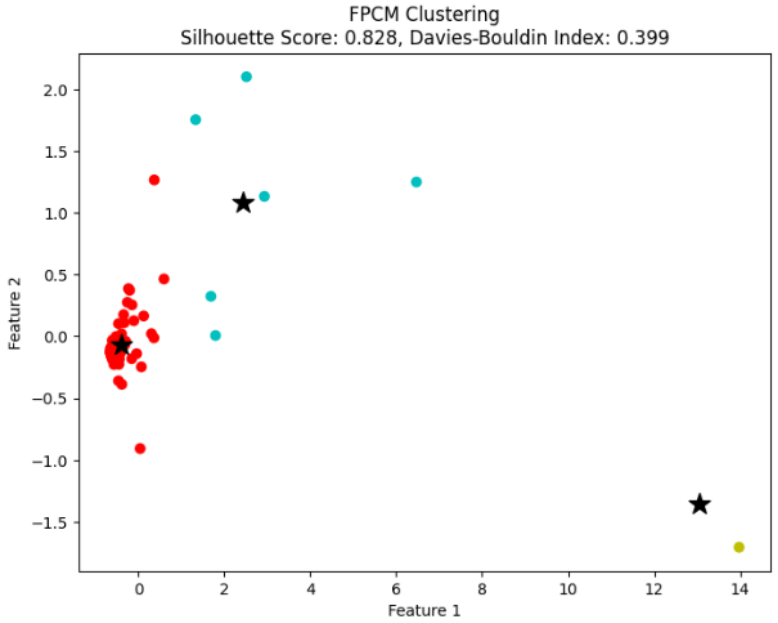
Ol diatur ke 0,0001. Ini adalah tingkat toleransi untuk menentukan konvergensi algoritma. Algoritma akan berhenti jika perubahan fungsi tujuan (yang mengukur kualitas pengelompokan) antara dua iterasi kurang dari ambang batas konvergensi.

P=2 yang berarti bahwa semua fitur berbobot sama. Menetapkan p ke nilai yang lebih besar dari 2 akan memberi bobot lebih pada fitur yang lebih besar, sementara menyetel p ke nilai kurang dari 2 akan memberi bobot lebih pada fitur yang lebih kecil.

```
# Set the number of clusters
k = 3
# Set the fuzziness parameter
m = 2
# Set the maximum number of iterations
max_iter = 100
# Set the convergence threshold
tol = 0.0001

clusters, sil_score, db_index = fpcm(X_pca, k, m, tol, max_iter)
```

Visualisasi FPCM



Pada hasil visualisasi FPCM dengan diambil per tiga kluster yang dibuat ditandakan dengan tiga warna yang berbeda, terlihat bahwa pada cluster berwarna merah mengalami penumpukkan data yang sangat banyak. Dan pada pada cluster berwarna bitu mengalami perpencara titik data yang cukup berpisah dan data cluster pada warna kuning menglami data yang cukup berbeda pada data lainnya. Pada cluster berwarna kuning hanya memliki satu data saja.

Visualisasi Fuzzy Set

```
silhouette_fpcm = silhouette_score(X_pca, clusters)
```

```
print("Silhouette score untuk FPCM: ", silhouette_fpcm)
```

```
Silhouette score untuk FPCM:  0.8276270019576544
```

Pada hasil silhoute yang dihasilkan pada metode FPCM pada dataset ini menghasilkan skor yang sangat baik yaitu sebesar 0,82

Partitioning scheme used by fuzziffying

```
#Add the cluster labels to the original dataset
df['Hasil Cluster FPCM'] = clusters

df
```

	hydro_and_marine	solar	wind	bioenergy	Hasil Cluster FPCM
0	29685	1346	9412	2607	0
1	14764	21033	20396	3351	0
2	41998	2043	6792	4591	0
3	777	389	5	4	0
4	399	176	194	578	0
...	...	...	...	...	...
73	4094	462	5476	2701	0
74	5000	0	16	0	0
75	64501	8	88	0	0
76	73496	1660	1803	322	0
77	3882	34	0	283	0

78 rows × 5 columns

Menaruh hasil cluster ditaruh pada dataset yang sama untuk nantinya dibandingkan pada data manual.

FPCM Manual Mathematic

```
import pandas as pd
df = pd.read_excel('renewable electricity generation.xlsx')
df = df.drop(["country"], axis = 1)
df
```

	hydro_and_marine	solar	wind	bioenergy
0	29685	1346	9412	2607
1	14764	21033	20396	3351
2	41998	2043	6792	4591
3	777	389	5	4
4	399	176	194	578
...	...	...	...	...
73	4094	462	5476	2701
74	5000	0	16	0
75	64501	8	88	0
76	73496	1660	1803	322
77	3882	34	0	283

78 rows × 4 columns

Melakukan import pada pengolahan FPCM dengan algoritma manual, dengan dilkukan drop pada variable country.

FPCM Manual Algoritma

```python
import numpy as np
import pandas as pd

def normalisasi(array_matriks):
    hasil = array_matriks / sum(array_matriks)
    return hasil


def fcm(dataframe, jumlah_cluster = 3, w = 2, maxIter = 100, error_threshold = 0.01):
    jumlah_data, jumlah_fitur = dataframe.shape

    matriks_myu = np.absolute(np.random.randn(jumlah_data, jumlah_fitur))
    v = np.zeros((jumlah_cluster, jumlah_fitur))

    df_myu = pd.read_excel("renewable_electricity_generation_model.xlsx")
    jumlah_baris_myu, jumlah_kolom_myu = df_myu.shape
    for index_baris in range(jumlah_baris_myu):
        for index_kolom in range(jumlah_kolom_myu):
            matriks_myu[index_baris][index_kolom] = df_myu.iloc[index_baris][index_kolom]

    jumlah_fungsi_objektif_sebelumnya = 0

    lebih_error = True
    iter_sekarang = 1

    while lebih_error and iter_sekarang <= maxIter:
        d_kuadrat = np.zeros((jumlah_data, jumlah_fitur))
        matriks_myu_baru = np.zeros((jumlah_data, jumlah_cluster))
        gabungan_myu_kuadrat = pow(matriks_myu, 2)

        # cari pusat cluster
        for indeks_cluster in range(jumlah_cluster):
            myu_kuadrat = []
            for indeks_data in range(jumlah_data):
                myu_kuadrat.append(
                    pow(matriks_myu[indeks_data][indeks_cluster], w))
            jumlah_myu_kuadrat = sum(myu_kuadrat)
            for indeks_fitur in range(jumlah_fitur):
                jumlah_myukuadrat_kalifitur = 0
                for indeks_data in range(jumlah_data):
                    jumlah_myukuadrat_kalifitur += myu_kuadrat[indeks_data] * \
                        dataframe.iloc[indeks_data][indeks_fitur]
                v[indeks_cluster][indeks_fitur] = jumlah_myukuadrat_kalifitur / \
                    jumlah_myu_kuadrat
```

Pada tahap awal pada normalisasi, pada fungsi ini bertujuan untuk dilakukan normalisasi jika diperlukan jika data memerlukan normalisasi, pada percobaan ini data tidak dibutuhkan normalisasi dikarenakan memiliki data dengan sifat data yang sama.

Pertama menentukan matrik pada data yang akan diolah dengan jumlah clusyer yaitu 3. pada matrik yang dibuat dengan besar berdsarkan jumlah data record dengan fitur yang ada,

Melakukan pencarian pust cluster dengan dilakukan looping dari tiap titik data yang ada

FPCM Manual Algoritma

```
# cari myu baru
for indeks_data in range(jumlah_data):
    for indeks_cluster in range(jumlah_cluster):
        for indeks_fitur in range(jumlah_fitur):
            d_kuadrat[indeks_data][indeks_cluster] += pow(
                dataframe.iloc[indeks_data][indeks_fitur] - v[indeks_cluster][indeks_fitur], 2)

for indeks_data in range(jumlah_data):
    for indeks_cluster in range(jumlah_cluster):
        jumlah_d_kuadrat_i_bagi_d_kuadrat_j = 0
        for j in range (jumlah_cluster):
            jumlah_d_kuadrat_i_bagi_d_kuadrat_j += pow(
                d_kuadrat[indeks_data][indeks_cluster] / d_kuadrat[indeks_data][j], 1 / (w - 1))
        matriks_myu_baru[indeks_data][indeks_cluster] = 1 / jumlah_d_kuadrat_i_bagi_d_kuadrat_j

# cari fungsi objektif
jumlah_fungsi_objektif = 0
for indeks_data in range(jumlah_data):
    for indeks_cluster in range(jumlah_cluster):
            jumlah_fungsi_objektif += d_kuadrat[indeks_data][indeks_cluster] * gabungan_myu_kuadrat[indeks_data][indeks_cluster]
```

Pencarian my baru dengan melakukan indeks data dengan range jumlah fitur data yang ada, dilakukan perhitungan matematika untuk menghasilkan nilai matriks my Mencari fungsi objektif dengan menjumlahkan beberapa gabungan myukuadrta dengan dilakukannya looping pada data.

Manual FPCM

```
        #print(jumlah_fungsi_objektif)
        #print(matriks_myu_baru)
        error = abs(jumlah_fungsi_objektif - jumlah_fungsi_objektif_sebelumnya)
        print('Iterasi ke {}, Hasil Fungsi Objektif = {}, Error = {}'.format(iter_sekarang, jumlah_fungsi_objektif, error))

        iter_sekarang += 1

        lebih_error = error > error_threshold
        matriks_myu = matriks_myu_baru
        jumlah_fungsi_objektif_sebelumnya = jumlah_fungsi_objektif

    print("Hasil akhir matriks FCM yang akan digunakan pada perhitungan FPCM")
    print(matriks_myu)
    return(matriks_myu, v)
```

Menghitung jumlah error yang ada dengan dilakukan looping periterasi yang ada. Program akan melakukan looping trs menerus hingga mendaptkan hasil yang optimum.

Manual FPCM

```
def fpcm(dataframe, myu, v, jumlah_cluster = 3, w = 2, eta = 2, maxIter = 100, error_threshold = 0.01):
    jumlah_data, jumlah_fitur = dataframe.shape
    jumlah_fungsi_objektif_sebelumnya = 0
    matriks_t = np.zeros((jumlah_data, jumlah_cluster))
    d_kuadrat_lama = np.zeros((jumlah_data, jumlah_fitur))
    lebih_error = True
    iter_sekarang = 1

    #cari t baru dari myu fcm dan v fcm
    for indeks_data in range(jumlah_data):
        for indeks_cluster in range(jumlah_cluster):
            for indeks_fitur in range(jumlah_fitur):
                d_kuadrat_lama[indeks_data][indeks_cluster] += pow(
                    dataframe.iloc[indeks_data][indeks_fitur] - v[indeks_cluster][indeks_fitur], 2)
            d_kuadrat_lama[indeks_data][indeks_cluster] = 1 / d_kuadrat_lama[indeks_data][indeks_cluster]

    jumlah_t_cluster = []
    for indeks_cluster in range(jumlah_cluster):
        jumlah = 0
        for indeks_data in range(jumlah_data):
            jumlah += d_kuadrat_lama[indeks_data][indeks_cluster]
        jumlah_t_cluster.append(jumlah)
        for indeks_data in range(jumlah_data):
            matriks_t[indeks_data][indeks_cluster] = pow(d_kuadrat_lama[indeks_data][indeks_cluster] / jumlah_t_cluster[indeks_cluster], 1 / (eta - 1))
```

Pada perhitungan algoritma FPCM ditentukan dengan jumlah data dan fungsi objektif dan matriks yang ada

Mencari t baru dengan my fcm dan v fcm yang dihasilkan dari variable d kuadrat lama dan indeks-indeksnya.

Melakukan klasterisasi dari jumalh cluster yang telah ditentukan

Manual FPCM

```
while lebih_error and iter_sekarang <= maxIter:
    d_kuadrat_miu = np.zeros((jumlah_data, jumlah_fitur))
    d_kuadrat_t = np.zeros((jumlah_data, jumlah_fitur))
    matriks_myu_baru = np.zeros((jumlah_data, jumlah_cluster))
    matriks_t_baru = np.zeros((jumlah_data, jumlah_cluster))
    gabungan_myu_kuadrat = pow(myu, 2)
    gabungan_t_kuadrat = pow(matriks_t, 2)

    #cari pusat cluster (v)
    for indeks_cluster in range(jumlah_cluster):
        myu_kuadrat = []
        t_kuadrat = []
        for indeks_data in range(jumlah_data):
            myu_kuadrat.append(
                pow(myu[indeks_data][indeks_cluster], w))
            t_kuadrat.append(
                pow(matriks_t[indeks_data][indeks_cluster], eta)
            )
        jumlah_myu_t_kuadrat = sum(myu_kuadrat) + sum(t_kuadrat)
        for indeks_fitur in range(jumlah_fitur):
            jumlah_myukuadrat_kalifitur = 0
            for indeks_data in range(jumlah_data):
                jumlah_myukuadrat_kalifitur += (myu_kuadrat[indeks_data] + t_kuadrat[indeks_data]) * \
                    dataframe.iloc[indeks_data][indeks_fitur]
            v[indeks_cluster][indeks_fitur] = jumlah_myukuadrat_kalifitur / \
                jumlah_myu_t_kuadrat
    # print(v)
```

Jika erro lebih tinggi maka akan dilakukan perulanagan hingga mendaptkan hasil yang optimum pada data.
Mencari pusat cluster pada setiap cluster yang ada dengan berdasarkan nilai my kuadrta yang ada dibantu dengan bantuan ‘pow’.
Menjumlahkan my t kuadrat [ada semua nilai my kuadrat dan dilakukan looping.

Manual FPCM

```
#cari miu baru
for indeks_data in range(jumlah_data):
    for indeks_cluster in range(jumlah_cluster):
        for indeks_fitur in range(jumlah_fitur):
            d_kuadrat_miu[indeks_data][indeks_cluster] += pow(
                dataframe.iloc[indeks_data][indeks_fitur] - v[indeks_cluster][indeks_fitur], 2)

for indeks_data in range(jumlah_data):
    for indeks_cluster in range(jumlah_cluster):
        jumlah_d_kuadrat_i_bagi_d_kuadrat_j = 0
        for j in range (jumlah_cluster):
            jumlah_d_kuadrat_i_bagi_d_kuadrat_j += pow(
                d_kuadrat_miu[indeks_data][indeks_cluster] / d_kuadrat_miu[indeks_data][j], 1 / (w - 1))
        matriks_myu_baru[indeks_data][indeks_cluster] = 1 / jumlah_d_kuadrat_i_bagi_d_kuadrat_j
```

Mencari miu baru dengan indeks data dan indeks fitur yang ada
Melakukan looping perualangan pada matriks my baru dengan nilai matematika yang telah dirtentukan sebelumnya

Manual FPCM

```
#cari t baru
for indeks_data in range(jumlah_data):
    for indeks_cluster in range(jumlah_cluster):
        for indeks_fitur in range(jumlah_fitur):
            d_kuadrat_t[indeks_data][indeks_cluster] += pow(
                dataframe.iloc[indeks_data][indeks_fitur] - v[indeks_cluster][indeks_fitur], 2)
        d_kuadrat_t[indeks_data][indeks_cluster] = 1 / d_kuadrat_t[indeks_data][indeks_cluster]

jumlah_t_cluster = []
for indeks_cluster in range(jumlah_cluster):
    jumlah = 0
    for indeks_data in range(jumlah_data):
        jumlah += d_kuadrat_t[indeks_data][indeks_cluster]
    jumlah_t_cluster.append(jumlah)
    for indeks_data in range(jumlah_data):
        matriks_t_baru[indeks_data][indeks_cluster] = pow(d_kuadrat_t[indeks_data][indeks_cluster] / jumlah_t_cluster[indeks_cluster], 1 / (eta - 1))
```

Mencari t baru dengan melakukan looping berulangan dengan nilai amtematika yang ada. Menetukan jumlah t cluster dengan indeks cluster dan jumlah cluster yang ada., akan dilkuak [erhitungan matematika dengan bantuan ‘pow’

Manual FPCM

```
# cari fungsi objektif
jumlah_fungsi_objektif = 0
for indeks_data in range(jumlah_data):
    for indeks_cluster in range(jumlah_cluster):
        jumlah_fungsi_objektif += d_kuadrat_miu[indeks_data][indeks_cluster] *(gabungan_myu_kuadrat[indeks_data][indeks_cluster] + gabungan_t_kuadrat[indeks_data][indeks_cluster])
error = abs(jumlah_fungsi_objektif - jumlah_fungsi_objektif_sebelumnya)
print('Iterasi ke {}, Hasil Fungsi Objektif = {}, Error = {}'.format(iter_sekarang, jumlah_fungsi_objektif, error))

iter_sekarang += 1
# lebih_error = abs(jumlah_fungsi_objektif - jumlah_fungsi_objektif_sebelumnya)
lebih_error = error > error_threshold
myu = matriks_myu_baru
matriks_t = matriks_t_baru
jumlah_fungsi_objektif_sebelumnya = jumlah_fungsi_objektif
```

Mencari fungsi objektif yang ada dan membandingkan nilai error yang ada.

Manual FPCM

```
iter_sekarang += 1
# lebih_error = abs(jumlah_fungsi_objektif - jumlah_fungsi_objektif_sebelumnya)
lebih_error = error > error_threshold
myu = matriks_myu_baru
matriks_t = matriks_t_baru
jumlah_fungsi_objektif_sebelumnya = jumlah_fungsi_objektif
```

Dengan deklor error tracehold yang ditentukan untuk menentukan jumlah fungsi objektif yang ada.

Manual FPCM

```
path = "renewable_electricity_generation_model.xlsx"
df = pd.read_excel(path)
(myu_fcm, v_fcm) = fcm(df)
fpcm(df, myu_fcm, v_fcm)
```

Melakukan pengolahan pada data

```
Iterasi ke 1, Hasil Fungsi Objektif = 5.352610574732582e+23, Error = 5.352610574732582e+23
Iterasi ke 2, Hasil Fungsi Objektif = 1096502073262.8167, Error = 5.352610574721617e+23
Iterasi ke 3, Hasil Fungsi Objektif = 700094146891.5486, Error = 396407926371.26807
Iterasi ke 4, Hasil Fungsi Objektif = 622973180766.5104, Error = 77120966125.03821
Iterasi ke 5, Hasil Fungsi Objektif = 563633155427.5969, Error = 59340025338.91345
Iterasi ke 6, Hasil Fungsi Objektif = 512630228747.5557, Error = 51002926680.0412
Iterasi ke 7, Hasil Fungsi Objektif = 408184463981.34467, Error = 104445764766.21106
Iterasi ke 8, Hasil Fungsi Objektif = 218242152537.80405, Error = 189942311443.54062
Iterasi ke 9, Hasil Fungsi Objektif = 176363113752.81485, Error = 41879038784.9892
Iterasi ke 10, Hasil Fungsi Objektif = 166595984334.87164, Error = 9767129417.943207
Iterasi ke 11, Hasil Fungsi Objektif = 157838309869.81122, Error = 8757674465.060425
Iterasi ke 12, Hasil Fungsi Objektif = 153039192427.35834, Error = 4799117442.452881
Iterasi ke 13, Hasil Fungsi Objektif = 150814234503.43607, Error = 2224957923.9222717
Iterasi ke 14, Hasil Fungsi Objektif = 149719703472.0804, Error = 1094531031.3556519
Iterasi ke 15, Hasil Fungsi Objektif = 149144341102.27603, Error = 575362369.8043823
Iterasi ke 16, Hasil Fungsi Objektif = 148832231579.69147, Error = 312109522.5845642
Iterasi ke 17, Hasil Fungsi Objektif = 148661722209.71613, Error = 170509369.9753418
Iterasi ke 18, Hasil Fungsi Objektif = 148568973584.3602, Error = 92748625.35592651
Iterasi ke 19, Hasil Fungsi Objektif = 148518927386.81335, Error = 50046197.54684448
Iterasi ke 20, Hasil Fungsi Objektif = 148492143062.39822, Error = 26784324.415130615
Iterasi ke 21, Hasil Fungsi Objektif = 148477908376.9162, Error = 14234685.482025146
Iterasi ke 22, Hasil Fungsi Objektif = 148470385221.81342, Error = 7523155.102783203
Iterasi ke 23, Hasil Fungsi Objektif = 148466426035.35147, Error = 3959186.46194458
Iterasi ke 24, Hasil Fungsi Objektif = 148464349073.98816, Error = 2076961.3633117676
Iterasi ke 25, Hasil Fungsi Objektif = 148463262083.51697, Error = 1086990.4711914062
Iterasi ke 26, Hasil Fungsi Objektif = 148462694187.70612, Error = 567895.8108520508
Iterasi ke 27, Hasil Fungsi Objektif = 148462397869.01862, Error = 296318.6875
Iterasi ke 28, Hasil Fungsi Objektif = 148462243398.323, Error = 154470.69561767578
Iterasi ke 29, Hasil Fungsi Objektif = 148462162927.29147, Error = 80471.0315246582
Iterasi ke 30, Hasil Fungsi Objektif = 148462121026.73715, Error = 41900.55432128906
Iterasi ke 31, Hasil Fungsi Objektif = 148462099217.25693, Error = 21809.480224609375
Iterasi ke 32, Hasil Fungsi Objektif = 148462087868.2237, Error = 11349.033233642578
Iterasi ke 33, Hasil Fungsi Objektif = 148462081963.61145, Error = 5904.612243652344
Iterasi ke 34, Hasil Fungsi Objektif = 148462078892.00644, Error = 3071.605010986328
Iterasi ke 35, Hasil Fungsi Objektif = 148462077294.30005, Error = 1597.7063903808594
Iterasi ke 36, Hasil Fungsi Objektif = 148462076463.3056, Error = 830.9944458007812
Iterasi ke 37, Hasil Fungsi Objektif = 148462076031.11316, Error = 432.19244384765625
Iterasi ke 38, Hasil Fungsi Objektif = 148462075806.3422, Error = 224.77096557617188
Iterasi ke 39, Hasil Fungsi Objektif = 148462075689.44824, Error = 116.89395141601562
Iterasi ke 40, Hasil Fungsi Objektif = 148462075628.6578, Error = 60.790435791015625
Iterasi ke 41, Hasil Fungsi Objektif = 148462075597.0442, Error = 31.613616943359375
Iterasi ke 42, Hasil Fungsi Objektif = 148462075580.60403, Error = 16.440155029296875
Iterasi ke 43, Hasil Fungsi Objektif = 148462075572.0547, Error = 8.549346923828125
Iterasi ke 44, Hasil Fungsi Objektif = 148462075567.60873, Error = 4.445953369140625
Iterasi ke 45, Hasil Fungsi Objektif = 148462075565.2967, Error = 2.312042236328125
Iterasi ke 46, Hasil Fungsi Objektif = 148462075564.0944, Error = 1.202301025390625
Iterasi ke 47, Hasil Fungsi Objektif = 148462075563.46902, Error = 0.6253662109375
Iterasi ke 48, Hasil Fungsi Objektif = 148462075563.1441, Error = 0.324920654296875
Iterasi ke 49, Hasil Fungsi Objektif = 148462075562.975, Error = 0.16909790039062
Iterasi ke 50, Hasil Fungsi Objektif = 148462075562.8871, Error = 0.087921142578125
Iterasi ke 51, Hasil Fungsi Objektif = 148462075562.84134, Error = 0.045745849609375
Iterasi ke 52, Hasil Fungsi Objektif = 148462075562.81757, Error = 0.023773193359375
Iterasi ke 53, Hasil Fungsi Objektif = 148462075562.80505, Error = 0.01251220703125
Iterasi ke 54, Hasil Fungsi Objektif = 148462075562.7988, Error = 0.006256103515625
Hasil akhir matriks FCM yang akan digunakan pada perhitungan FPCM
```

Manual FPCM

Perhitungan dengan loopingan iterasi yang berlakukan dengan fungsi objektif dan error hingga menghasilkan nilai optimum

```
Hasil akhir matriks FCM yang akan digunakan pada perhitungan FPCM
[[9.21343508e-05 2.13246047e-03 9.97775405e-01]
 [2.27081676e-04 5.07590065e-03 9.94697018e-01]
 [3.30243770e-04 8.07613265e-03 9.91593624e-01]
 [1.69147830e-04 3.35559913e-03 9.96475253e-01]
 [1.72453783e-04 3.41798979e-03 9.96409556e-01]
 [1.49978177e-04 3.06677303e-03 9.96783249e-01]
 [6.02029618e-05 1.24989301e-03 9.98689904e-01]
 [1.36833248e-04 2.74049389e-03 9.97122673e-01]
 [1.16643989e-04 2.35280677e-03 9.97530549e-01]
 [1.54596733e-02 8.72369289e-01 1.12171038e-01]
 [1.21001224e-04 2.43388896e-03 9.97445110e-01]
 [1.26121870e-04 2.53515818e-03 9.97338720e-01]
 [2.13178964e-05 4.74041706e-04 9.99504640e-01]
 [9.99995410e-01 2.83195056e-06 1.75756379e-06]
 [6.03387593e-04 1.49754914e-02 9.84421121e-01]
 [7.04079106e-05 1.44721629e-03 9.98482376e-01]
 [9.14847339e-05 1.86122877e-03 9.98047286e-01]
 [1.31348187e-04 2.63779744e-03 9.97230854e-01]
 [5.33367960e-05 1.11434951e-03 9.98832314e-01]
 [1.84526909e-04 3.78679859e-03 9.96028675e-01]
 [1.52859052e-04 3.04678583e-03 9.96800355e-01]
 [7.20124092e-05 1.58470449e-03 9.98343283e-01]
 [1.45146577e-04 2.89822205e-03 9.96956631e-01]
 [1.70714645e-04 3.38576305e-03 9.96443522e-01]
 [3.76887746e-05 8.19559579e-04 9.99142752e-01]
 [1.67461985e-03 5.06365717e-02 9.47688808e-01]
 [7.98849884e-03 2.06672424e-01 7.85339077e-01]
 [8.95421564e-05 1.82646555e-03 9.98083992e-01]
 [9.65834315e-05 1.98173585e-03 9.97921681e-01]
```

Manual FPCM

Hasil akhir matriks yang ditentukan yang telah dilakukan looping hingga menetukan hasil yang optimum

```
[1.26097884e-04 2.53492717e-03 9.97338975e-01]]
Iterasi ke 1, Hasil Fungsi Objektif = 149493949131.13043, Error = 149493949131.13043
Iterasi ke 2, Hasil Fungsi Objektif = 149488322173.1879, Error = 5626957.9425354
Iterasi ke 3, Hasil Fungsi Objektif = 149485487862.71597, Error = 2834310.471923828
Iterasi ke 4, Hasil Fungsi Objektif = 149484022701.0079, Error = 1465161.7080688477
Iterasi ke 5, Hasil Fungsi Objektif = 149483258921.99142, Error = 763779.0164794922
Iterasi ke 6, Hasil Fungsi Objektif = 149482859475.43555, Error = 399446.55587768555
Iterasi ke 7, Hasil Fungsi Objektif = 149482650249.7689, Error = 209225.66665649414
Iterasi ke 8, Hasil Fungsi Objektif = 149482540562.85693, Error = 109686.91195678711
Iterasi ke 9, Hasil Fungsi Objektif = 149482483025.95276, Error = 57536.90417480469
Iterasi ke 10, Hasil Fungsi Objektif = 149482452832.40567, Error = 30193.547088623047
Iterasi ke 11, Hasil Fungsi Objektif = 149482436983.1891, Error = 15849.216583251953
Iterasi ke 12, Hasil Fungsi Objektif = 149482428661.8643, Error = 8321.324798583984
Iterasi ke 13, Hasil Fungsi Objektif = 149482424292.25052, Error = 4369.61376953125
Iterasi ke 14, Hasil Fungsi Objektif = 149482421997.46866, Error = 2294.7818603515625
Iterasi ke 15, Hasil Fungsi Objektif = 149482420792.22665, Error = 1205.2420043945312
Iterasi ke 16, Hasil Fungsi Objektif = 149482420159.185, Error = 633.0416564941406
Iterasi ke 17, Hasil Fungsi Objektif = 149482419826.67218, Error = 332.5128173828125
Iterasi ke 18, Hasil Fungsi Objektif = 149482419652.0105, Error = 174.66168212890625
Iterasi ke 19, Hasil Fungsi Objektif = 149482419560.26248, Error = 91.74801635742188
Iterasi ke 20, Hasil Fungsi Objektif = 149482419512.06747, Error = 48.19500732421875
Iterasi ke 21, Hasil Fungsi Objektif = 149482419486.75046, Error = 25.3170166015625
Iterasi ke 22, Hasil Fungsi Objektif = 149482419473.45108, Error = 13.29937744140625
Iterasi ke 23, Hasil Fungsi Objektif = 149482419466.46487, Error = 6.9862060546875
Iterasi ke 24, Hasil Fungsi Objektif = 149482419462.795, Error = 3.66986083984375
Iterasi ke 25, Hasil Fungsi Objektif = 149482419460.8671, Error = 1.92791748046875
Iterasi ke 26, Hasil Fungsi Objektif = 149482419459.85428, Error = 1.0128173828125
Iterasi ke 27, Hasil Fungsi Objektif = 149482419459.32227, Error = 0.532012939453125
Iterasi ke 28, Hasil Fungsi Objektif = 149482419459.04276, Error = 0.279510498046875
Iterasi ke 29, Hasil Fungsi Objektif = 149482419458.89603, Error = 0.146728515625
Iterasi ke 30, Hasil Fungsi Objektif = 149482419458.81885, Error = 0.077178955078125
Iterasi ke 31, Hasil Fungsi Objektif = 149482419458.77832, Error = 0.04052734375
Iterasi ke 32, Hasil Fungsi Objektif = 149482419458.7571, Error = 0.021209716796875
Iterasi ke 33, Hasil Fungsi Objektif = 149482419458.74588, Error = 0.01123046875
Iterasi ke 34, Hasil Fungsi Objektif = 149482419458.74014, Error = 0.0057373046875
```

Manual FPCM

Menghasilkan nilai periterasi dengan oping hingga hasil optimum dengan nilai fungsi objektif dan error yang ada,

Model Result

```
from pyFTS.benchmarks import Measures

rows = []
for file in models:
    try:
        model = Util.load_obj(file)

        row = [model.shortname, model.order,len(model)]

        if model.is_multivariate:
            rmse,_,_ = Measures.get_point_statistics(test_mv, model)
            row.append(rmse/np.mean(test_mv['MaxTemp']))
        else:
            rmse,_,_ = Measures.get_point_statistics(test, model)
            row.append(rmse/np.mean(test))

        rows.append(row)
    except:
        pass
pd.DataFrame(rows,columns=["Model","Order","Size","RMSE"]).sort_values(["RMSE","Size"])
```

✓ 2.3s

```
c:\Users\hagan\AppData\Local\Programs\Python\Python39\lib\site-packages\pyFTS\benchmarks\Measures.py:44: Runt
  return np.sqrt(np.nanmean((targets - forecasts) ** 2))
c:\Users\hagan\AppData\Local\Programs\Python\Python39\lib\site-packages\pyFTS\benchmarks\Measures.py:71: Runt
  return np.nanmean(np.abs(np.divide(np.subtract(targets, forecasts), targets))) * 100
c:\Users\hagan\AppData\Local\Programs\Python\Python39\lib\site-packages\pyFTS\benchmarks\Measures.py:119: Run
  return np.sqrt(np.divide(np.sum(y), np.sum(naive)))
```

	Model	Order	Size	RMSE
0	HOFTS3	3	854	0.107534
1	WHOFTS3	3	854	0.107534
2	PWFTS3	3	854	0.593223
3	ClusteredMVFTS2	2	5478	NaN
4	ClusteredMVFTS2	2	5478	NaN

Pda hasil output yang dihasilkan menunjukkan urutan pada model berdasarkan order-order terntu.
Model "HOFTS3" dan "WHOFTS3" memiliki urutan, ukuran, dan nilai RMSE yang sama, menunjukkan bahwa model tersebut serupa atau identik. Model "PWFTS3" memiliki nilai RMSE yang lebih tinggi dibandingkan dengan model lain, yang menunjukkan bahwa model tersebut mungkin memiliki kecocokan yang lebih buruk dengan data atau akurasi prediksi yang lebih rendah.
Output "NaN" (Bukan Angka) di kolom "RMSE" untuk model "ClusteredMVFTS2" menunjukkan bahwa nilai RMSE hilang atau tidak tersedia untuk model tertentu tersebut.

Model Result

	country	hydro_and_marine	solar	wind	bioenergy	Hasil Cluster FPCM	Hasil Cluster manual
1	country	hydro_and_marine	solar	wind	bioenergy	Hasil Cluster FPCM	Hasil Cluster manual
2	argentina	29685	1346	9412	2607	0	0
3	australia	14764	21033	20396	3351	0	0
4	austria	41998	2043	6792	4591	0	0
5	bangladesh	777	389	5	4	0	0
6	belarus	399	176	194	578	0	0
7	belgium	267	5105	12764	5275	0	0
8	bhutan	11370	0	1	0	0	0
9	bolivia	2939	250	64	327	0	0
10	bosnia	4580	45	267	13	0	0
11	brazil	396382	10759	57050	58744	1	1
12	bulgaria	2820	1481	1477	1699	0	0
13	cambodia	3850	316	0	79	0	0
14	chile	21721	8141	5602	4495	0	0
15	china	1321717	261659	467037	98978	2	2
16	colombia	49862	206	10	1712	0	0
17	costa rica	8294	78	1459	210	0	0
18	croatia	5667	96	1721	978	0	0
19	czechia	2144	2287	699	5215	0	0
20	north korea	12800	53	1	0	0	0
21	denmark	17	1181	16330	5923	0	0
22	dominica republic	1293	391	1102	291	0	0
23	ecuador	24323	38	77	471	0	0
24	el-savador	2071	909	14	815	0	0
25	estonia	30	123	844	1852	0	0
26	finland	15883	218	7938	11570	0	0
27	france	62544	13398	39792	8845	0	0
28	germany	18322	38641	132102	50858	1	1
29	ghana	7293	107	0	20	0	0
30	greece	3444	4447	9310	454	0	0
31	hungaria	244	2459	655	2155	0	0
32	india	159729	53666	63522	21987	1	1
33	indonesia	19445	176	8	12382	0	0
34	iran	13330	584	587	22	0	0
35	iraq	4963	337	0	0	0	0
36	ireland	933	64	11549	937	0	0
37	italy	47552	24954	18762	19634	0	0
38	israel	0	4163	204	156	0	0
39	japan	78807	79087	8970	27995	1	1
40	netherlands	46	8765	15339	8848	0	0
41	luxemburg	92	161	351	372	0	0
42	laos	29813	43	0	46	0	0
43	malaysia	25906	471	0	2541	0	0
44	mexico	26817	13528	19701	2291	0	0
45	myanmar	12896	79	0	261	0	0
46	nigeria	8226	42	0	21	0	0
47	new zealand	24266	160	2405	784	0	0
48	pakistan	39287	1032	2457	2839	0	0
49	peru	30031	838	1814	585	0	0
50	philippines	6470	1389	1026	1366	0	0
51	poland	2118	1958	15800	8384	0	0
52	portugal	12083	1733	12299	3786	0	0

Model Result

54	russian	212587	1249	1401	393	0	1
55	romania	15381	1733	6945	547	0	0
56	serbia	9034	13	976	192	0	0
57	singapore	0	360	0	1822	0	0
58	slovakia	4517	663	4	1637	0	0
59	slovenia	4934	368	6	274	0	0
60	south africa	1212	7587	300	432	0	0
61	spain	30507	20667	56444	6138	0	0
62	sri lanka	4958	442	332	57	0	0
63	sudan	11008	28	0	104	0	0
64	sweden	72389	1051	27526	11177	0	0
65	switzerland	37869	2599	145	1980	0	0
66	thailand	4540	4939	3522	30692	0	0
67	tajikistan	19169	0	0	0	0	0
68	turkey	78094	10953	24828	4445	0	0
69	uganda	3993	122	0	268	0	0
70	ukraine	5729	7141	3511	755	0	0
71	uea	0	5485	0	1	0	0
72	united state	287140	119329	341818	60269	1	1
73	united kingdom	5363	13158	75369	40186	1	0
74	tanzania	3224	42	0	151	0	0
75	uruguay	4094	462	5476	2701	0	0
76	uzbekistan	5000	0	16	0	0	0
77	venezuela	64501	8	88	0	0	0
78	vietnam	73496	1660	1803	322	0	0
79	zimbabwe	3882	34	0	283	0	0

Pada pengelompokkan negara dengan pembangkit energy terbarukan menghasilkan 3 kelompok yang berbeda. Pada kluster pertama pada pemodelan dengan perhitungan manual menghasilkan 74 negara, kluster 2 dengan 4 negara, dan kluster 3 dengan 3 negara.

Pada perhitungan dengan program library menghasilkan kluster pertama 72 negara, kluster keuda 6 negara dan cluster tiga 1 negara

China Result

Generation in 2020	GWh	%
Non-renewable	5 630 632	72
Renewable	2 149 534	28
Hydro and marine	1 321 717	17
Solar	261 659	3
Wind	467 037	6
Bioenergy	98 978	1
Geothermal	144	0
Total	7 780 166	100

Per capita electricity generation (kWh)

Pada negara china, negara ini masuk pada kluster 3 yang hanya dia sendiri anggota didalamnya, terlihat bahwa pada pembangkit listrik terbrarukan sangat tinggi, dibandingkan negara lain Negra in memilik pembangkit yang sangat tinggi. Namun pada data itu, ternyata hanya menghasilkan 28% dari sumber pembangkit energi.

US Result

Generation in 2020	GWh	%
Non-renewable	3 432 657	81
Renewable	827 387	19
Hydro and marine	287 140	7
Solar	119 329	3
Wind	341 818	8
Bioenergy	60 269	1
Geothermal	18 831	0
Total	4 260 044	100

Pada negara US, menghasilkan pembangkit listrik yang cukup tinggi, negara ini memasuki cluster 2. pada negara us hanya menghasilkan 19% pada total sumber energy yang ada.

Indonesia Result

Generation in 2020	GWh	%
Non-renewable	238 315	83
Renewable	47 583	17
Hydro and marine	19 455	7
Solar	176	0
Wind	8	0
Bioenergy	12 382	4
Geothermal	15 563	5
Total	285 899	100

Pada negara Indonesia sendiri, menghasilkan energy energi terbarukan yang cukup, dengan tingkat peersentase sebesar 17% dari total energy yang ada.

Switzerland Result

Generation in 2020	GWh	%
Non-renewable	**29 044**	**41**
Renewable	**42 593**	**59**
Hydro and marine	37 869	53
Solar	2 599	4
Wind	145	0
Bioenergy	1 980	3
Geothermal	0	0
Total	**71 637**	**100**

Pada negara ini, menghasilkan persentasi energy terbarukan dengan persentasi tertinggi yaitu 59%, dengan menghasilkan 42000 GWH.

KESIMPULAN

Pada pengelompokkan negara dengan pembangkit energy terbarukan menghasilkan 3 kelompok yang berbeda. Pada kluster pertama pada pemodelan dengan perhitungan manual menghasilkan 74 negara, kluster 2 dengan 4 negara, dan kluster 3 dengan 3 negara. Pada perhitungan dengan program library menghasilkan kluster pertama 72 negara, kluster keuda 6 negara dan cluster tiga 1 negara.

Pada pengolompokkan kali ini hanya menghasilkan negara mana yang menghasilkan energy terbarukan tertinggi tetapi cukup disayangkan pada tiap negara tersebut tidak berkorelasi dengan persentasi antara pembangkit energy tidak terbarukan dengan energy terbarukan

Harapannya pada setiap negara dengan penghasil energy terbarukan tertinggi dengan persentase antara lebih tinggi dari energy tidak terbarukan menjadi sebuah contoh untuk negara lain agar berlomba-lomba beralih menjadi negara dengan memproduksi energy terbarukan untuk menjadikan energy bersih dan terjangkau yang akan memengeruhi iklim yang sehat.

Thanks!

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**