PREDIKSI BAWANG MERAH MENGGUNAKAN RANTAI MARKOV DI PROVINSI MALUKU

[1.] *Clorinda Sally* [2.] *Paulus Agung Prastowo* [3.] *Fransiskus Jremiegi Saputra*
[1.] sallyclorinda20@gmail.com [2.] paulusagungprastowo@gmail.com [3.] fransiskusjremiegi@gmail.com
Program Studi Informatika, Fakultas Sains dan Teknologi
Universitas Sanata Dharma

ABSTRAK

Bawang Merah merupakan salah satu bumbu masak dunia dengan varietas tumbuhan berumbi yang dapat hidup di dataran tinggi. Data yang digunakan merupakan data yang diperoleh dari Badan Pusat Statistik (bps.go.id) terkait hasil panen bawang merah di Provinsi Maluku pada periode 2000 - 2021. Langkah awal yang penulis lakukan adalah dengan mengumpulkan dan mengolah data, mendeskripsikan data, memberi predikat, analisis data, melakukan perhitungan menggunakan rumus rantai markov, memperoleh probabilitas dan memprediksi hasil panen untuk beberapa tahun kedepan. Berdasarkan hasil yang diperoleh, penulis menemukan bahwa probabilitas untuk memprediksi hasil panen yang berjumlah sedikit selama tiga tahun berturut - turut yakni 0,682379, probabilitas untuk peluang terjadinya kondisi panen sedikit dalam dua tahun berturut - turut yakni 0,702332 dan mengalami keadaan steady state yang dimulai pada iterasi ke - 49.

Kata Kunci : Bawang Merah, Rantai Markov, Maluku

ABSTRACT

Red Onion is one of the globally recognized cooking ingredients with various bulbous plant varieties that can thrive in highland areas. The data utilized in this study was obtained from the Central Bureau of Statistics (bps.go.id) concerning red onion harvest yields in the Maluku Province during the period of 2000 to 2021. The initial steps undertaken by the researcher involved data collection and processing, data description, categorization, data analysis, computation using the Markov chain formula, obtaining probabilities, and predicting harvest yields for upcoming years. Based on the results obtained, it was found that the probability of predicting a low harvest yield for three consecutive years was 0.682379, while the probability of a consecutive occurrence of low yield conditions for two years in a row was 0.702332. Furthermore, the system reached a steady state starting from the 49th iteration.

Keywords: Red Onion, Markov Chain, Maluku.

PENDAHULUAN

Bawang Merah merupakan salah satu bumbu masak dunia dengan varietas tumbuhan berumbi yang dapat hidup di dataran tinggi. Tanaman ini berasal dari Iran, Pakistan, dan daerah pegunungan di sebelah utaranya yang kemudian menyebar ke berbagai daerah subtropis atau pun tropis. Tanaman ini memiliki nama ilmiah Allium Cepa L yang termasuk dalam kelompok Aggregatum. Memiliki ciri khas dengan bau yang tajam tetapi tidak setajam bau bawang putih dan aromanya gurih serta sedikit pedas. Bawang merah merupakan tanaman dua musim yang tumbuh sebagai tanaman satu musim kecuali untuk produksi benih. Tanaman ini merupakan tanaman hortikultura yang termasuk sayuran rempah yang digunakan sebagai bumbu masakan agar menambah cita rasa dan kenikmatan suatu masakan. Pada kali ini, penulis akan mencoba membahas mengenai hasil prediksi/probabilitas hasil panen bawang merah di Provinsi Maluku dengan menggunakan pendekatan rantai markov.

Rantai Markov adalah rangkaian proses kejadian yang mana probabilitas bersyarat kejadian yang akan datang tergantung pada kejadian yang sekarang. Rantai Markov pertama kali dikemukakan oleh Andrey Andreyevich Markov, matematikawan berkebangsaan Rusia. Publikasinya yang pertama membahas rantai markov adalah pada tahun 1906.

Pada kali ini penulis akan membahas mengenai prediksi hasil panen bawang merah dengan menggunakan metode rantai markov. Data yang digunakan diambil dari Badan Pusat Statistik (bps.go.id) dengan rentang data dari tahun 2000 sampai dengan tahun 2021 terkait hasil panen bawang merah di Provinsi Maluku.

METODOLOGI

Data Penelitian

Data yang digunakan pada paper ini merupakan data yang diperoleh dari Badan Pusat Statistik (bps.go.id) terkait hasil panen bawang merah di Provinsi Maluku pada periode 2000 - 2021.

Metode Analisis Data

PEMBAHASAN

Langkah awal kami saat kami mengerjakan project kami ini adalah, pertama kami mengambil data hasil panen bawang merah di provinsi Maluku dari tahun 2000 sampai dengan tahun 2021. Berikut data awal yang kami gunakan

Tahun	Kuantitas
2000	328,00
2001	3303,00
2002	272,20
2003	524,00
2004	1093,00
2005	2079,00
2006	1724,00
2007	594,00
2008	459,00
2009	167,00
2010	398,00
2011	484,00
2012	433,00
2013	470,00
2014	543,00
2015	451,00
2016	304,00
2017	592,00
2018	1042,00
2019	736,00
2020	1106,00
2021	1013,00

Lalu, setelah itu kami menentukan predikat pada setiap data yang kami ambil dari tahun 2000 sampai dengan tahun 2021 tersebut, dengan isian predikat S = yang menyatakan sedikit, lalu C = yang menyatakan cukup, lalu B = yang menyatakan banyak. Perhitungan ini didapat dari menghitung nilai minimal, nilai maksimal, nilai range, dan nilai jangkauan. Berikut rumus yang kami gunakan untuk mencari perhitungannya:

Nilai Minimum = nilai min dari data yang digunakan

Nilai Maksimum = nilai max dari data yang digunakan

Nilai Range = nilai maksimal - nilai minimum

Nilai Jangkauan = range / 3

Dari rumus tersebut maka yang kami dapat adalah:

Nilai Maksimal	167
Nilai Minimum	3303
Range	3136
Jangkauan	1045,33

Lalu, setelah itu kita mencari predikatnya dengan menggunakan nilai Jangkauan yang telah di dapat. Yaitu sebagai berikut:

Prediksi	Range Panen
Sedikit	167 - < 1212,33
Cukup	1212,33 - < 2257,667
Banyak	2257,667 - 3303

Dengan prediksi tersebut, maka kita bisa mengubah data hasil panen yang sebelumnya bernilai angka menjadi berlabel predikat. Data tersebut akan menjadi seperti berikut

Tahun	Kuantitas	Prediksi
2000	328,00	S
2001	3303,00	B
2002	272,20	S
2003	524,00	S
2004	1093,00	S
2005	2079,00	C
2006	1724,00	C
2007	594,00	S
2008	459,00	S
2009	167,00	S
2010	398,00	S
2011	484,00	S
2012	433,00	S
2013	470,00	S
2014	543,00	S
2015	451,00	S
2016	304,00	S
2017	592,00	S
2018	1042,00	S
2019	736,00	S
2020	1106,00	S
2021	1013,00	S

Berikutnya, kami akan menghitung banyaknya transisi antara S-S, S-C, S-B, C-S, C-C, C-B, B-S, B-C, B-B sebagai berikut:

	Sedikit	Cukup	Banyak	Total
Sedikit	16	1	1	18
Cukup	1	1	0	2
Banyak	1	0	0	1

Dari gambar tabel diatas, maka didapat bahwa data S-S ada sebanyak 16, lalu data S-C ada sebanyak 1, lalu berikutnya ada S-B ada sebanyak 1 dan begitu seterusnya sampai dengan B-B ada sebanyak 0. Lalu, berikutnya kita perlu menjumlahkan semua yang telah di data tersebut. Dari gambar tersebut, kita ketahui bahwa jumlah Sedikit ada total sebanyak 18, lalu jumlah Cukup ada total sebanyak 2, dan jumlah Banyak ada total sebanyak 1.

Hasil Perhitungan tersebut akan kami gunakan untuk membuat perhitungan probabilitas awalnya, yaitu sebagai berikut:

S	C	B	Total
19	2	1	22
0,863636	0,090909	0,045455	1

Mencari Peluang Transisi

Lalu, setelah probabilitas awal terbentuk, maka kami akan melanjutkan untuk mencari matriks satu langkahnya, yaitu sebagai berikut:

	S	C	B
S	0,888889	0,055556	0,055556
C	0,5	0,5	0
B	1	0	0

Dari matriks satu langkah tersebut, kita bisa mencari probabilitas nya sampai dalam keadaan steady state nya. Sebelumnya kami akan menjelaskan mengenai pengertian steady state, steady state merupakan suatu kondisi atau keadaan di mana suatu keadaan tersebut tak berubah dengan seiring berjalannya waktu atau bisa disebut dengan keadaan konstan. Pada kasus kami, untuk mencari steady state tersebut, kami menggunakan rumus :

P X P X P … X Pn

P itu merupakan matriks satu langkahnya.

Kami menemukan pola dalam keadaan steady statenya berada pada probabilitas ke- 20. Jadi, mulai dari probabilitas ke- 20, hasilnya sudah mulai konstan dan sama semua, sehingga kami menyimpulkan bahwa pada keadaan pola ke 20 tersebut, program kami sudah steady state.

Probabilitas steady state:

0,85714286	0,0952381	0,04761905
0,85714286	0,0952381	0,04761905
0,85714286	0,0952381	0,04761905

Pada Rantai Markov, untuk memudahkan mencari peluang transisi hingga transisi ke - n langkah, kita perlu membuat diagram markovnya terlebih dahulu, seperti gambar dibawah ini:

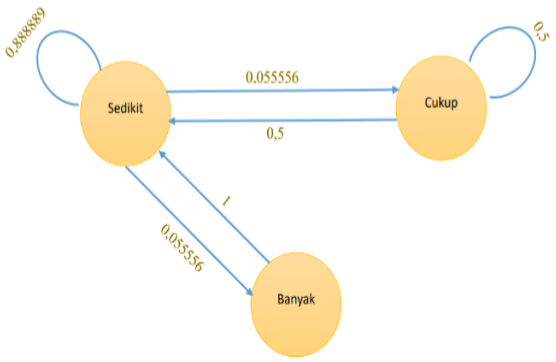

Diagram Markov diatas dibuat berdasarkan matriks transisi satu langkah yang telah kami buat dan telah kami bahas di atas tadi. Dengan demikian, kita bisa menyelesaikan studi kasus yang akan kami jadikan sebagai bahan diskusi. Yaitu,

A. kita bisa menentukan nilai probabilitas yang berjumlah sedikit selama tiga tahun berturut - turut yakni dengan menggunakan rumus sebagai berikut:

= Probabilitas sedikit X probabilitas sedikit sedikit X probabilitas sedikit sedikit

= 0,863636 x 0,888889 x 0,888889

= 0,682379

B. kita bisa menentukan nilai probabilitas jika produksi saat ini sedikit, berapa peluang terjadinya kondisi panen sedikit dalam 2 tahun berturut - turut ?

= Probabilitas sedikit sedikit X probabilitas sedikit sedikit X probabilitas sedikit sedikit

= 0,888889 x 0,888889 x 0,888889

= 0,702332

KESIMPULAN

Berdasarkan hasil penelitian kami, maka kami dapat menyimpulkan bahwa program kami dapat digunakan untuk memprediksi hasil panen probabilitas yang berjumlah sedikit selama tiga tahun berturut - turut sebanyak 0,682379. Penulis juga dapat menemukan probabilitas jika produksi saat ini sedikit, berapa peluang terjadinya kondisi panen sedikit dalam dua tahun berturut - turut, yang menghasilkan 0,702332. Rantai markov ini cukup dapat membantu para petani bawang merah di provinsi Maluku. Di dalam program penelitian kami ini, kami telah mendapatkan keadaan steady state yang dimulai pada iterasi ke - 49. Dimana matriks tersebut adalah sebagai berikut:

	S	C	B
S	0,85714286	0,0952381	0,04761905
C	0,85714286	0,0952381	0,04761905
B	0,85714286	0,0952381	0,04761905

DAFTAR PUSTAKA

Tanah Kaya. (n.d.). Tanaman Bawang Merah. Diakses dari https://tanahkaya.com/tanaman-bawang-merah/

Informasi Insan. (2021, Desember). Markov Chain (Rantai Markov): Pengertian, Perhitungan, Contoh. Diakses dari https://informasains.com/edu/post/2021/12/markov-chain-rantai-markov-pengertian-perhitungan-contoh/

Badan Pusat Statistik Maluku. (n.d.). Hortikultura. Diakses dari https://maluku.bps.go.id/subject/55/hortikultura.html

LAMPIRAN

1. CAPTURE PROGRAM

Program dapat diakses melalui:
https://github.com/SallyClorinda/Markov-Chain/blob/main/python_markov%20chain/main.py

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

# Baca Data Excel
Data = pd.read_excel('D:\AGUNG\K U L I A H\stokastik\Data.xlsx');

# Ekstrak data kolom kuantitas
df_data = Data["Kuantitas"]
print(df_data)

print("\n")

# # Drop missing values and reset index
# df_clean = df_data.dropna().reset_index(drop=True)

# Mencari nilai minimal, maksimal, range & jangkauan
nilai_min = df_data.min()
nilai_max = df_data.max()
nilai_range = nilai_max - nilai_min
nilai_jangkauan = nilai_range / 3

print("Min =", nilai_min)
print("Max =", nilai_max)
print("Range = ", nilai_range)
print("Jangkauan = ", nilai_jangkauan)

print("\n")

# Mencari nilai prediksi
Prediksi = []

for i in range (len(df_data)):
    if df_data[i] < 1212.33 :
        # Data['Prediksi'] = Prediksi.append('S')
        Prediksi.append('S')
```

```python
    elif df_data[i] >= 1212.33 and df_data[i] < 2257.667 :
        # Data['Prediksi'] = Prediksi.append('C')
        Prediksi.append('C')
    elif df_data[i] >= 2257.667 :
        # Data['Prediksi'] = Prediksi.append('B')
        Prediksi.append('B')
print("Prediksi = ", Prediksi)

# Menghitung Jumlah Prediksi
jumlah_s, jumlah_c, jumlah_b = 0, 0, 0

for i in range(len(Prediksi)):
    Data['Prediksi'] = Prediksi
    if(Prediksi[i] == "S"):
        jumlah_s += 1
    elif(Prediksi[i] == "C"):
        jumlah_c += 1
    elif(Prediksi[i] == "B"):
        jumlah_b += 1

print("Jumlah S = ", jumlah_s)
print("Jumlah C = ", jumlah_c)
print("Jumlah B = ", jumlah_b)

total = jumlah_b + jumlah_c + jumlah_s
print("Total = ", total)

print("\n")

# Menghitung probabilitas awal
probabilitas_s = jumlah_s / total
probabilitas_c = jumlah_c / total
probabilitas_b = jumlah_b / total

print("Probabilitas S = ", probabilitas_s)
```

```
    print("Probabilitas S = ", probabilitas_s)
    print("Probabilitas C = ", probabilitas_c)
    print("Probabilitas B = ", probabilitas_b)

    print("\n")

    # Menghitung transisi
    transis_ss, transis_sc, transis_sb = 0, 0, 0
    transis_cs, transis_cc, transis_cb = 0, 0, 0
    transis_bs, transis_bc, transis_bb = 0, 0, 0

    for i in range(len(Prediksi) - 1):
        if Prediksi[i] == "S" and Prediksi[i + 1] == "S":
            transis_ss += 1
        elif Prediksi[i] == "S" and Prediksi[i + 1] == "C":
            transis_sc += 1
        elif Prediksi[i] == "S" and Prediksi[i + 1] == "B":
            transis_sb += 1
        elif Prediksi[i] == "C" and Prediksi[i + 1] == "S":
            transis_cs += 1
        elif Prediksi[i] == "C" and Prediksi[i + 1] == "C":
            transis_cc += 1
        elif Prediksi[i] == "C" and Prediksi[i + 1] == "B":
            transis_cb += 1
        elif Prediksi[i] == "B" and Prediksi[i + 1] == "S":
            transis_bs += 1
        elif Prediksi[i] == "B" and Prediksi[i + 1] == "C":
            transis_bc += 1
        elif Prediksi[i] == "B" and Prediksi[i + 1] == "B":
            transis_bb += 1

    print("Transisi SS = ", transis_ss)
    print("Transisi SC = ", transis_sc)
    print("Transisi SB = ", transis_sb)
```

```
    print("Transisi CS = ", transis_cs)
    print("Transisi CC = ", transis_cc)
    print("Transisi CB = ", transis_cb)

    print("Transisi BS = ", transis_bs)
    print("Transisi BC = ", transis_bc)
    print("Transisi BB = ", transis_bb)

    total_transisi = transis_ss + transis_sc + transis_sb + transis_cs + transis_cc + transis_cb + transis_bs + transis_bc + transis_bb
    print("Total Transisi = ", total_transisi)

    print("\n")

    # Menghitung probabilitas setiap transisi
    total_scb = transis_ss + transis_sc + transis_sb
    total_csb = transis_cs + transis_cc + transis_cb
    total_bsc = transis_bs + transis_bc + transis_bb

    # print(total_scb)
    # print(total_csb)
    # print(total_bsc)

    probabilitas_ss = transis_ss / total_scb
    probabilitas_sc = transis_sc / total_scb
    probabilitas_sb = transis_sb / total_scb

    probabilitas_cs = transis_cs / total_csb
    probabilitas_cc = transis_cc / total_csb
    probabilitas_cb = transis_cb / total_csb

    probabilitas_bs = transis_bs / total_bsc
    probabilitas_bc = transis_bc / total_bsc
    probabilitas_bb = transis_bb / total_bsc

    print("probabilitas Transisi SS = ", probabilitas_ss)
    print("probabilitas Transisi SC = ", probabilitas_sc)
```

```
    print("probabilitas Transisi SC = ", probabilitas_sc)
    print("probabilitas Transisi SB = ", probabilitas_sb)

    print("probabilitas Transisi CS = ", probabilitas_cs)
    print("probabilitas Transisi CC = ", probabilitas_cc)
    print("probabilitas Transisi CB = ", probabilitas_cb)

    print("probabilitas Transisi BS = ", probabilitas_bs)
    print("probabilitas Transisi BC = ", probabilitas_bc)
    print("probabilitas Transisi BB = ", probabilitas_bb)

    print("\n")

    # Membuat hasil dalam bentuk matrix
    matrix_transisi = np.array([[probabilitas_ss, probabilitas_sc, probabilitas_sb],
                                [probabilitas_cs, probabilitas_cc, probabilitas_cb],
                                [probabilitas_bs, probabilitas_bc, probabilitas_bb]])
    print("Matrix Transisi: ")
    print(matrix_transisi)

    print("\n")


    # Menghitung probabilitas panen dalam jumlah sedikit selama 3 tahun berturut-turut
    prediksi_a = probabilitas_s * probabilitas_ss * probabilitas_ss
    print("Probabilitas Jumlah (S) Selama 3 Tahun Berturut-Turut = ", prediksi_a)

    # Menghitung probabilitas apabila saat ini sedikit maka peluang panen kondisi sedikit selama 2 tahun berturut-turut
    prediksi_b = probabilitas_ss * probabilitas_ss * probabilitas_ss
    print("Probabilitas Jumlah Saat Ini (S) Selama 2 Tahun Berturut-Turut = ", prediksi_b)

    # Menghitung probabilitas apabila saat ini banyak maka peluang panen kondisi banyak selama 3 tahun berturut-turut
    prediksi_c = probabilitas_bb * probabilitas_bb * probabilitas_bb * probabilitas_bb
    print("Probabilitas Jumlah Saat Ini (S) Selama 2 Tahun Berturut-Turut = ", prediksi_c)

    print("\n")
```

```
# Steady State
# Definisikan matriks awal
P_current = np.copy(matrix_transisi)
P_previous = np.zeros_like(matrix_transisi)

# Inisialisasi counter iterasi
iteration = 1
# Perulangan hingga mencapai konvergensi atau keadaan di mana matriks saat ini memiliki semua elemen yang sama dengan matriks sebelumnya
while True:
    P_previous = np.copy(P_current)
    P_current = np.dot(P_current, matrix_transisi)
    iteration += 1

    # Cetak matriks setelah setiap iterasi
    print("Iterasi", iteration-1)
    print(P_current)
    print()

    # Periksa konvergensi
    if np.array_equal(P_current, P_previous):
        break
# Cetak hasil matriks distribusi stasioner yang konvergen
print("Distribusi Stasioner (Konvergen) setelah", iteration-1, "iterasi:")
print(P_previous)
print("\n")
# Lanjutkan iterasi setelah mencapai matriks steady state
max_iterations = 6  # Jumlah iterasi tambahan yang diinginkan
for i in range(max_iterations):
    P_current = np.dot(P_current, matrix_transisi)
    iteration += 1

    # Cetak matriks setelah setiap iterasi tambahan
    print("Iterasi", iteration-1)
    print(P_current)
    print()
```

2. CAPTURE OUTPUT PROGRAM

```
In [9]: runfile('D:/a/uts 2 stokasik.py', wdir='D:/a')
0       328.0
1      3303.0
2       272.2
3       524.0
4      1093.0
5      2079.0
6      1724.0
7       549.0
8       459.0
9       167.0
10      398.0
11      484.0
12      433.0
13      470.0
14      543.0
15      451.0
16      304.0
17      592.0
18     1042.0
19      736.0
20     1106.0
21     1013.0
Name: Kuantitas, dtype: float64
```

```
Min = 167.0
Max = 3303.0
Range =  3136.0
Jangkauan =  1045.3333333333333


Prediksi =  ['S', 'B', 'S', 'S', 'S', 'C', 'C', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S',
'S', 'S', 'S', 'S', 'S', 'S']
Jumlah S =  19
Jumlah C =  2
Jumlah B =  1
Total =  22


Probabilitas S =  0.8636363636363636
Probabilitas C =  0.09090909090909091
Probabilitas B =  0.045454545454545456


Transisi SS =  16
Transisi SC =  1
Transisi SB =  1
Transisi CS =  1
Transisi CC =  1
Transisi CB =  0
Transisi BS =  1
```

```
Transisi BS =  1
Transisi BC =  0
Transisi BB =  0
Total Transisi =  21


probabilitas Transisi SS =  0.8888888888888888
probabilitas Transisi SC =  0.05555555555555555
probabilitas Transisi SB =  0.05555555555555555
probabilitas Transisi CS =  0.5
probabilitas Transisi CC =  0.5
probabilitas Transisi CB =  0.0
probabilitas Transisi BS =  1.0
probabilitas Transisi BC =  0.0
probabilitas Transisi BB =  0.0


Matrix Transisi:
[[0.88888889 0.05555556 0.05555556]
 [0.5        0.5        0.        ]
 [1.         0.         0.        ]]


Probabilitas Jumlah (S) Selama 3 Tahun Berturut-Turut =  0.6823793490460156
Probabilitas Jumlah Saat Ini (S) Selama 2 Tahun Berturut-Turut =  0.7023319615912208
Probabilitas Jumlah Saat Ini (S) Selama 2 Tahun Berturut-Turut =  0.0
```

```
Iterasi 1
[[0.87345679 0.07716049 0.04938272]
 [0.69444444 0.27777778 0.02777778]
 [0.88888889 0.05555556 0.05555556]]

Iterasi 2
[[0.864369   0.08710562 0.04852538]
 [0.78395062 0.17746914 0.03858025]
 [0.87345679 0.07716049 0.04938272]]

Iterasi 3
[[0.86040619 0.09157331 0.0480205 ]
 [0.82415981 0.13228738 0.04355281]
 [0.864369   0.08710562 0.04852538]]

Iterasi 4
[[0.85861266 0.093587   0.04780034]
 [0.842283   0.11193035 0.04578666]
 [0.86040619 0.09157331 0.0480205 ]]

Iterasi 5
[[0.85780509 0.0944942  0.0477007 ]
 [0.85044783 0.10275867 0.0467935 ]
 [0.85861266 0.093587   0.04780034]]
```

```
Iterasi 6
[[0.85744122 0.09490294 0.04765584]
 [0.85412646 0.09862644 0.0472471 ]
 [0.85780509 0.0944942  0.0477007 ]]

Iterasi 7
[[0.85727728 0.09508709 0.04763562]
 [0.85578384 0.09676469 0.04745147]
 [0.85744122 0.09490294 0.04765584]]

Iterasi 8
[[0.85720342 0.09517006 0.04762652]
 [0.85653056 0.09592589 0.04754355]
 [0.85727728 0.09508709 0.04763562]]

Iterasi 9
[[0.85717014 0.09520744 0.04762241]
 [0.85686699 0.09554798 0.04758503]
 [0.85720342 0.09517006 0.04762652]]

Iterasi 10
[[0.85715515 0.09522429 0.04762056]
 [0.85701857 0.09537771 0.04760372]
 [0.85717014 0.09520744 0.04762241]]
```

```
Iterasi 11
[[0.8571484  0.09523187 0.04761973]
 [0.85708686 0.095301   0.04761214]
 [0.85715515 0.09522429 0.04762056]]

Iterasi 12
[[0.85714535 0.09523529 0.04761936]
 [0.85711763 0.09526644 0.04761594]
 [0.8571484  0.09523187 0.04761973]]

Iterasi 13
[[0.85714398 0.09523683 0.04761919]
 [0.85713149 0.09525086 0.04761765]
 [0.85714535 0.09523529 0.04761936]]

Iterasi 14
[[0.85714336 0.09523753 0.04761911]
 [0.85713774 0.09524385 0.04761842]
 [0.85714398 0.09523683 0.04761919]]

Iterasi 15
[[0.85714309 0.09523784 0.04761908]
 [0.85714055 0.09524069 0.04761876]
 [0.85714336 0.09523753 0.04761911]]
```

```
Iterasi 16
[[0.85714296 0.09523798 0.04761906]
 [0.85714182 0.09523926 0.04761892]
 [0.85714309 0.09523784 0.04761908]]

Iterasi 17
[[0.8571429  0.09523804 0.04761905]
 [0.85714239 0.09523862 0.04761899]
 [0.85714296 0.09523798 0.04761906]]

Iterasi 18
[[0.85714288 0.09523807 0.04761905]
 [0.85714265 0.09523833 0.04761902]
 [0.8571429  0.09523804 0.04761905]]

Iterasi 19
[[0.85714287 0.09523808 0.04761905]
 [0.85714276 0.0952382  0.04761904]
 [0.85714288 0.09523807 0.04761905]]

Iterasi 20
[[0.85714286 0.09523809 0.04761905]
 [0.85714281 0.09523814 0.04761904]
 [0.85714287 0.09523808 0.04761905]]
```

```
Iterasi 21
[[0.85714286 0.09523809 0.04761905]
 [0.85714284 0.09523812 0.04761905]
 [0.85714286 0.09523809 0.04761905]]

Iterasi 22
[[0.85714286 0.09523809 0.04761905]
 [0.85714285 0.09523811 0.04761905]
 [0.85714286 0.09523809 0.04761905]]

Iterasi 23
[[0.85714286 0.09523809 0.04761905]
 [0.85714285 0.0952381  0.04761905]
 [0.85714286 0.09523809 0.04761905]]

Iterasi 24
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.09523809 0.04761905]]

Iterasi 25
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]
```

```
Iterasi 26
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 27
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 28
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 29
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 30
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]
```

```
Iterasi 31
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 32
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 33
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 34
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 35
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]
```

```
Iterasi 41
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 42
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 43
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 44
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 45
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]
```

```
Iterasi 46
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 47
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 48
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 49
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Distribusi Stasioner (Konvergen) setelah 49 iterasi:
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]
```

```
Distribusi Stasioner (Konvergen) setelah 49 iterasi:
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]


Iterasi 50
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 51
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 52
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 53
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]
```

```
Iterasi 53
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 54
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]

Iterasi 55
[[0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]
 [0.85714286 0.0952381  0.04761905]]
```

3. **LINK PRESENTASI**

   https://youtu.be/99Ig1WolbbM