DAFTAR PUSTAKA

Afsahyana, Yohana, Nurhayati, & Isnadiyah. (2022). Survey of Dengue Hemorrhagic Fever Density in Makassar City, South Sulawesi Province. *Pancasakti Journal Of Public Health Science And Research*, *2*(3), 124–131. https://doi.org/10.47650/pjphsr.v2i3.483

Agusta, Y. (2007). K-Means-Penerapan, Permasalahan dan Metode Terkait. *Jurnal Sistem Dan Informatika*, *3*(Pebruari), 47–60. https://yudiagusta.files.wordpress.com/2008/03/k-means.pdf

Arsin, A. (2013). *Epidemiologi Demam Berdarah Dengue (DBD) di Indonesia*. MASAGENA PRESS. https://pdfcoffee.com/epidemiologi-demam-berdarah-dengue-dbd-di-indonesia-pdf-free.html

Dania, I. A. (2016). Gambaran Penyakit dan Vektor Demam Berdarah Dengue (DBD). *Jurnal Warta*, *48*(1), 1–15.

Darman, R., Studi, P., Informasi, S., Teknologi, F., Universitas, I., Manis, L., Manis, L., & Padang, K. (2018). *ANALISIS VISUALISASI DAN PEMETAAN DATA TANAMAN PADI*. *4*(2), 156–162.

Dinas Kesehatan, K. M. (2021). *Jumlah Kasus DBD*. Makassar: Dinas Kesehatan.

Fatmawati, K., & Windarto, A. P. (2018). Data Mining: Penerapan Rapidminer Dengan K-Means Cluster Pada Daerah Terjangkit Demam Berdarah Dengue (DBD) Berdasarkan Provinsi. *Computer Engineering, Science and System Journal*, *3*(2), 173. https://doi.org/10.24114/cess.v3i2.9661

Fitriyani, F. (2016). Implementasi Algoritma Fp-Growth Menggunakan Association Rule Pada Market Basket Analysis. *Jurnal Informatika*, *2*(1). https://doi.org/10.31311/ji.v2i1.85

Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques. In *ata Mining: Concepts and Techniques*. https://doi.org/10.1016/C2009-0-l819-5

Ikhwan, A., Nofriansyah, D., & Sriani. (2021). Penerapan Data Mining dengan Algoritma Fp-Growth untuk Mendukung Strategi Promosi Pendidikan (Studi Kasus Kampus STMIK Triguna Dharma). *Jurnal Ilmiah Saintikom*, *6*(2), 115. https://doi.org/10.30998/string.v6i2.9160

Larose, D. T., & Larose, C. D. (2015). Data Mining and Predictive Analytics. In *Wiley* (Vol. 6, Issue August).

Lestari, Y. D. (2015). Penerapan Data Mining Menggunakan Algoritma FP-Tree Dan FP-Growth Pada Data Transaksi Penjualan Obat Yuyun Dwi Lestari Program Studi Teknik Informatika , Sekolah Tinggi Teknik Harapan Seminar Nasional Teknologi Informasi dan Komunikasi (SNASTIKOM 2015). *Seminar Nasional Teknologi Informasi Dan Komunikasi (SNASTIKOM 2015) ISBN 976-602-19837-9-9*, *Snastikom*, 60–65.

Mudhari, M. A. (2018). SISTEM INFORMASI PEMETAAN KANTOR PEMERINTAH KABUPATEN SITUBONDO BERBASIS WEB. *Informatika, Jurnal Ilmiah*, *3*(2), 235–241.

Rusdiyanto. (2017). SISTEM INFORMASI GOEGRAFIS PEMETAAN FASILITAS UMUM DI KECAMATAN LUBUKLINGGAU UTARA 1 KOTA LUBUKLINGGAU. *JUTIM*, *2*(2), 99–105.

Samuel, D. (2008). Penerapan Stuktur FP-Tree dan Algoritma FP-Growth dalam Optimasi Penentuan Frequent Itemset. *Intitut Teknologi Bandung*, *1*, viii+124. www.infogavamedia@yahoo.com

Shofiani, N. (2017). *Segmentasi Supplier Menggunakan Metode K- Means Clustering (Studi Kasus : Ptpn X Pg Meritjan)* [Skripsi, Institut Teknologi Sepuluh November]. https://repository.its.ac.id/42213/1/5213100051_Undergraduate_Theses.pd

Susanti, S., & Suharyo, S. (2017). Hubungan Lingkungan Fisik Dengan Keberadaan Jentik Aedes Pada Area Bervegetasi Pohon Pisang. *Unnes Journal of Public Health*, *6*(4), 271–276. https://doi.org/10.15294/ujph.v6i4.15236

Trovancia, G., Sorisi, A., & Tuda, J. S. B. (2016). Deteksi transmisi virus dengue pada nyamuk wild Aedes Aegypti betina di Kota Manado. *Jurnal E-Biomedik*, *4*(2). https://doi.org/10.35790/ebm.4.2.2016.14661

Wakhidah, N. (2010). Clustering Menggunakan K-Means Algorithm (K-MEANS ALGORITHM CLUSTERING). *Jurnal Transformatika*, *8*(1), 33. https://doi.org/10.26623/transformatika.v8i1.45

WHO. (2009). Dengue Guidelines For Diagnosis Treatment, Prevention And Control. *WHO*, *41*(1), 29–29. https://doi.org/10.1176/pn.41.1.0029b

LAMPIRAN

Lampiran 1 Surat penugasan meneliti

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK

Poros Malino Km.6Bontomarannu(92172) Gowa, Sulawesi Selatan 92172, Sulawesi Selatan
Telp. (0411) 586015, 586262 Fax (0411) 586015
http://eng.unhas.ac.id, Email : teknik@unhas.ac.id

SURAT PENUGASAN
No. 20073/UN4.7.1/TD.06/2023

Dari : Dekan Fakultas Teknik Universitas Hasanuddin

Kepada : 1. Prof. Dr. Ir. Indrabayu.,ST, MT, M.Bus.Sys — Pemb. I
2. Iqra Aswad, ST., M.T — Pemb. II

I s I : 1. Berdasarkan Surat Ketua Departemen Teknik Informatika Fakultas Teknik Nomor. 982/UN4.7.7./TD.06/2023 tanggal 8 September 2023 tentang usul DOSEN PEMBIMBING MAHASISWA, maka dengan ini kami menugaskan Saudara untuk membimbing penulisan Skripsi/Tugas Akhir mahasiswa Teknik Informatika Fakultas Teknik Universitas Hasanuddin di bawah ini :

N a m a :	No. Stambuk :
Muhammad Wahyudi R Sumara	D121 17 1502

Judul Skripsi/Tugas Akhir :

" Analisis Data Mining Penyakit Demam Berdarah Menggunakan Teknik Clustering dan Association"

2. Surat penugasan pembimbing ini mulai berlaku sejak tanggal ditetapkannya dan berakhir sampai selesainya penulisan Skripsi/Tugas Akhir mahasiswa tersebut.

3. Agar penugasan ini dilaksanakan sebaik-baiknya dengan penuh rasa tanggung jawab.

Ditetapkan di Gowa
Pada tanggal 8 September 2023
a.n. Dekan,
Wakil Dekan Bidang Akademik dan Kemahasiswaan
Fakultas Teknik Unhas

Dr. Amil Ahmad Ilham, ST., M.IT
NIP. 197310101998021001

Tembusan :
1: Dekan FT-UH,
2. Ketua Departemen Teknik Informatika FT-UH,
3. Mahasiswa yang bersangkutan

Lampiran 2 Surat permohonan meneliti Rumah Sakit Bhayangkara

KEPOLISIAN DAERAH SULAWESI SELATAN
BIDANG KEDOKTERAN DAN KESEHATAN
RUMAH SAKIT BHAYANGKARA MAKASSAR

NOTA - DINAS
Nomor : B/ND- 370 /VI/2022/Urdiklit

Kepada : Yth. Kaur Rekam Medis

Dari : Kaur Diklit RS Bhayangkara Makassar

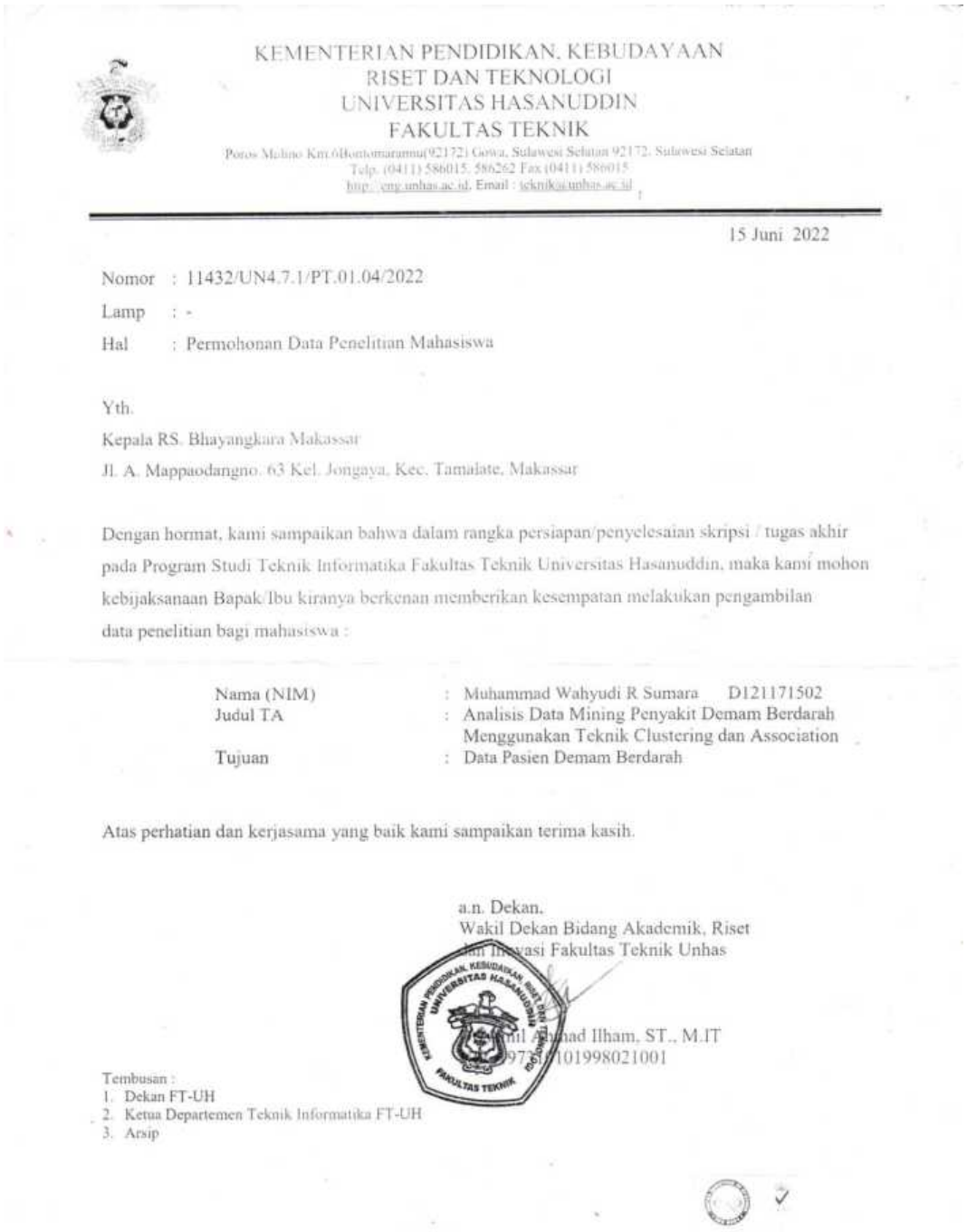
Perihal : Permohanan Pelaksanaan Pengambilan Data Mahasiswa S1 Teknik Informatika An. Muhammad Wahyudi (D121171502) Universitas Hasanuddin (UNHAS MAKASSAR).

1. Rujukan :
   a. Undang-Undang Nomor 2 Tahun 2002, tentang Kepollisian Negara Republik Indonesia;
   b. Peraturan Kapolri Nomor 11 tahun 2011 tanggal 30 Juni 2011 tentang Organisasi dan Tata Kerja Rumah Sakit Bhayangkara ;
   c. Rencana Kerja Rumah Sakit Bhayangkara Makassar Tahun 2022
   d. Surat masuk Fak.Tehnik UNHAS Makassar Nomor : 11432/UN4.7.1/PT.01.04/2022, tanggal 15 Juni 2022 perihal Permohonan Pengambilan Data Penelitian Mahasiswa.
   e. Lembaran Disposisi Karumkit Nomor : 542/VI/2022/Rumkit Tanggal 20 Juni 2022 Perihal Perijinan Pengambilan data penelitian mahasiswa.
2. Dalam rangka tertibnya pelaksanaan Penelitian Mahasiswa S1 Teknik Informatika Universitas Hasanuddin, maka dianggap perlu di keluarkan surat pengantar dari Urdiklit Subbag Binfung Rumah Sakit Bhayangkara Makassar, sebagai tanda bahwa Mahasiswa/i yang bersangkutan telah memenuhi syarat Administrasi dan diberikan ijin untuk dapat melakukan Pengambilan data penelitian di Rumah Sakit Bhayangkara,
3. Berkaitan dengan hal tersebut diatas, bersama ini dimohon kepada Penanggung Jawab Rekam Medis kiranya dapat menerima Mahasiswa/i tersebut diatas untuk melakukan Pengambilan data di ruangan Rekam Medis Rumah Sakit Bhayangkara Makassar.
4. Demikian disampaikan, mohon saran untuk pelaksanaan nya.

Makassar, 30 Juni 2022
KAUR DIKLIT

DARDIN, S.Kep Ns, M.Kep
AKP NRP. 69120379

Lampiran 3 Permohonan data penelitian universitas hasanuddin



KEMENTERIAN PENDIDIKAN, KEBUDAYAAN
RISET DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK

Poros Malino Km.6Bontomaranu(921721 Gowa, Sulawesi Selatan 92172, Sulawesi Selatan
Telp. (0411) 586015, 586262 Fax (0411) 586015
http://eng.unhas.ac.id, Email : teknik@unhas.ac.id

15 Juni 2022

Nomor : 11432/UN4.7.1/PT.01.04/2022
Lamp : -
Hal : Permohonan Data Penelitian Mahasiswa

Yth.
Kepala RS. Bhayangkara Makassar
Jl. A. Mappaodangno. 63 Kel. Jongaya, Kec. Tamalate, Makassar

Dengan hormat, kami sampaikan bahwa dalam rangka persiapan/penyelesaian skripsi / tugas akhir pada Program Studi Teknik Informatika Fakultas Teknik Universitas Hasanuddin, maka kami mohon kebijaksanaan Bapak/Ibu kiranya berkenan memberikan kesempatan melakukan pengambilan data penelitian bagi mahasiswa :

Nama (NIM)	:	Muhammad Wahyudi R Sumara D121171502
Judul TA	:	Analisis Data Mining Penyakit Demam Berdarah Menggunakan Teknik Clustering dan Association
Tujuan	:	Data Pasien Demam Berdarah

Atas perhatian dan kerjasama yang baik kami sampaikan terima kasih.

a.n. Dekan,
Wakil Dekan Bidang Akademik, Riset dan Inovasi Fakultas Teknik Unhas

Ir. Muhammad Ilham, ST., M.IT
NIP. 19730101998021001

Tembusan :
1. Dekan FT-UH
2. Ketua Departemen Teknik Informatika FT-UH
3. Arsip

Lampiran 4 Lembar perbaikan skripsi

LEMBAR PERBAIKAN SKRIPSI

"ANALISIS DATA MINING PENYAKIT DEMAM BERDARAH MENGGUNAKAN TEKNIK KLASTERING DAN ASOSIASI"

OLEH:

MUHAMMAD WAHYUDI R SUMARA
D121171502

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana pada tanggal 08 Maret 2024. Telah dilakukan perbaikan penulisan dan isi skripsi berdasarkan usulan dari penguji dan pembimbing skripsi.

Persetujuan perbaikan oleh tim penguji:

	Nama	Tanda Tangan
Ketua	Prof. Dr. Ir. Indrabayu, S.T., M.T., M.Bus.Sys., IPM.ASEAN. Eng.	
Sekretaris	Iqra Aswad, S.T., M.T.	
Anggota	Elly Warni S.T., M.T.	
	Muhammad Alief Fadhal Imran Oemar, S.T., M.Sc.	

Persetujuan perbaikan oleh pembimbing:

Pembimbing	Nama	Tanda Tangan
I	Prof. Dr. Ir. Indrabayu, S.T., M.T., M.Bus.Sys., IPM.ASEAN. Eng.	
II	Iqra Aswad, S.T., M.T.	

Lampiran 5 Kuesioner penelitian/Data Primer

Kuesioner Demam Berdarah

Yth. Bapak/Ibu Calon Responden Penelitian Di

Tempat

Saya dari mahasiswa departemen Teknik Informatika a.n Muhammad Wahyudi Rusli Sumara sedang melaksanakan penelitian dengan judul "Analisis Data Mining Penyakit Demam Berdarah menggunakan Teknik Klasterisasi dan Asosiasi". Penelitian ini di latar belakangi oleh meningkatnya kasus penyakit demam berdarah ini di Kota Makassar pada Tahun 2019-2021 yang dimana penyebab meningkatnya kasus ini antara lain tempat perkembangbiakan yaitu lingkungan-lingkungan yang kotor, genangan air, rumah dekat Sungai atauapun dekat pembuangan tempat sampah, dan juga kepadatan lingkungan rumah.

Sehubungan dengan hal tersebut, peneliti akan melakukan pengumpulan data kepada pasien penyakit demam berdarah yang pernah mengalami ataupun yang sedang lagi dirawat di Rumah Sakit Bhayangkara dengan menggunakan Kuesioner. Tujuan dari penelitian ini adalah mengetahui keterkaitan antar variabel(kebiasaan) dan juga pengelompokkan karakteristik penyakit demam berdarah di Kota Makassar. Manfaat dari penelitian ini adalah dapat membantu dalam menganalisis serta solusi dalam pengelompokkan daerah penyebaran demam berdarah dan permasalahan demam berdarah baik keterkaitan pola terjadinya demam berdarah, sehingga dapat memberikan informasi dasar untuk mengambil tindakan pencegahan kasus demam berdarah di Kota Makassar.

Kami mengharapkan partisipasi anda dalam penelitian yang kami lakukan, kami menjamin kerahasiaan dan identitas anda. Informasi yang anda berikan hanya sematamata digunakan untuk penelitian ini dan tidak digunakan untuk maksud yang lain.

Apabila bapak/ibu ibu bersedia menjadi responden, anda dapat mengisi pertanyaan di bagian berikutnya. Jika ada pertanyaan, silahkan menghubungi di

ıaramwr17d@student.unhas.ac.id

*CATATAN : Kami Menyediakan Go-Pay/OVO/DANA/Pulsa dengan total 500k bagi yang beruntung dan telah mengisi kuesioner.

Terima Kasih.

*Wajib

1. No.Handphone *

Silakan cantumkan Nomor HP yang aktif. Nomor tersebut akan digunakan untuk mengirim Go-Pay/OVO/DANA/Pulsa.

.....

Identitas Pasien

2. Tanggal Lahir *

   .....

   Contoh : 7 Januari 2019

3. Umur (dalam tahun dan bulan) *

   .....

4. Tinggi Badan (dalam cm) *

   .....

5. Berat badan (dalam kg)

   .....

6. Jenis Kelamin *

   ☐ Laki-laki

   ☐ Perempuan

7. Alamat tinggal *

   .....

8. Kelurahan *

   .....

9. Kecamatan *

   .....

ekerjaan Orang Tua *

- ☐ PNS
- ☐ TNI/POLRI
- ☐ Karyawan Swasta
- ☐ Wiraswasta
- ☐ BUMN
- ☐ Yang lain: ....

11. Pendapatan Orang Tua *
    - ☐ < 2.500.000
    - ☐ 2.500.000 – 5.000.000
    - ☐ 5.000.000 – 10.000.000
    - ☐ > 10.000.000

Riwayat Kesehatan

Beberapa pertanyaan di bawah ini berkaitan dengan riwayat-riwayat Kesehatan pasien serta keluarga

12. Apakah ciri seseorang terkena DBD adalah demam tinggi mendadak selama 2 – 7 hari dan terdapat bitnik-bintik merah pada kulitnya? *
    - ☐ Ya
    - ☐ Tidak
13. Apakah anda pernah menderita demam berdarah? *
    - ☐ Ya
    - ☐ Tidak
14. Apakah terdapat anggotan keluarga yang pernah menderita demam berdarah selama kurun waktu 6 bulan? *
    - ☐ Ya
    - ☐ Tidak
    - ☐ Yang lainnya : ....
15. Jika ya, keadaan penderita tersebut saat ini: *
    - ☐ Sehat
    - ☐ Masih menderita sakit akibat komplikasi penyakit DBD
    - ☐ Meninggal dunia

16. Apakah pasien pernah menerima vaksin/imunisasi *dengvaxia* (CYD-TDV) untuk mencegah penyakit DBD? *
    - ☐ Ya
    - ☐ Tidak
17. Apa pasien pernah di opname sebelumnya? *
    - ☐ Ya
    - ☐ Tidak
18. Jika pernah, opnamenya karena penyakit apa saja?

    ....

    .....

Kondisi lingkungan rumah

Pertanyaan di bawah berkaitan dengan kondisi lingkungan rumah responden

19. Berapa luas rumah tempat tinggal pasien? *
    - ☐ < 36 m^2
    - ☐ 36 – 54 m^2
    - ☐ 54 – 120 m^2
    - ☐ 120 m^2
20. Berapa jumlah kamar tidur didalam rumah? *

    ....

    .....
21. Berapa jumlah orang tinggal didalam rumah? *

    .....
22. Apakah disekitar rumah terdapat genangan air atau rumah yang saling berdekatan/padat? *
    - ☐ Ya
    - ☐ Tidak
23. Apakah anda memiliki jendela dan ventilasi serta memasangkannya kawat dirumah anda? *
    - ☐ Ya
    - ☐ Tidak

Lampiran 6 Source code clustering

```
import numpy as np
import pandas as pd
import re
import matplotlib.pyplot as plt
import plotly.graph_objs as go
import seaborn as sns
from tqdm import tqdm
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
from sklearn.preprocessing import PowerTransformer
from google.colab import drive
drive.mount('/content/gdrive')
from google.colab import drive
drive.mount('/content/gdrive')
df.head()
df = df.drop(['Timestamp',
        'No.Handphone',
        'Tanggal Lahir',
        'Alamat (mohon sertakan nama kelurahan dan
kecamatan)',
        'Apa pasien pernah di opname sebelumnya?'],
        axis=1)
df.head()
df = df.rename(columns = {
    'Umur (dalam tahun dan bulan)': 'umur',
    'Ciri seseorang terkena DBD adalah demam tinggi mendadak
2-7 hari dan terdapat bintik-bintik merah pada kulit.':
'ciri-ciri terkena DBD',
    'Apakah terdapat anggota keluarga yang pernah menderita
demam berdarah selama kurun waktu 6 bulan?': 'riwayat DBD
orang serumah',
    'Jika ya, keadaan penderita tersebut saat ini :':
'penyakit lainnya',
    'Apakah Pasien pernah menerima vaksin/imunisasi
Dengvaxia (CYD-TDV) untuk mencegah penyakit demam
berdarah?': 'riwayat vaksin CYD',
    'Jika pernah, opnamenya karena penyakit apa saja? ':
    'it opname',
    'rapa Luas rumah tempat tinggal pasien?': 'luas

    'rapa jumlah kamar tidur dirumah?': 'jumlah kamar',
```

```
    'berapa jumlah orang yang tinggal di dalam rumah?':
'jumlah orang',
    'Apakah disekitar rumah ada genangan air atau rumah yang
saling berdekatan?': 'genangan air sekitar',
    'Apakah jendela dan ventilasi rumah anda dipasangkan
kawat?' : 'ventilasi',
    'Apakah anda pernah menderita demam berdarah?': 'Pernah
DBD',

})
df['Jenis Kelamin'] = df['Jenis Kelamin'].replace('laki-
Laki', 'Laki-Laki')
df.head()
data_status_gizi_laki['Tahun'] =
data_status_gizi_laki['Tahun'].replace(np.nan, 0)
data_status_gizi_perempuan['Tahun'] =
data_status_gizi_perempuan['Tahun'].replace(np.nan, 0)
df= df.replace(np.nan, 'Tidak Pernah')
df.head()
# 1. TRANSFORMASI KOLOM UMUR

age = df['umur'].str.split(r'(\d+)', expand = True)
age = age.replace([None], '0')
tinggi = df['Tinggi Badan (dalam cm)'].str.split(r'(\d+)',
expand = True)
tinggi = tinggi.replace([None], '0')
berat = df['Berat badan (dalam kg)'].str.split(r'(\d+)',
expand = True)
berat = berat.replace([None], '0')

df['Tahun'] = age[1]
df['Tinggi Badan (dalam cm)'] = tinggi[1]
df['Berat badan (dalam kg)'] = berat[1]
df['Bulan'] = age[3]

df = df.astype({"Tahun": float, "Bulan": float, "Tinggi
Badan (dalam cm)": float, "Berat badan (dalam kg)": float})

df.head()
# df['Jenis Kelamin']

jk = df['Jenis Kelamin'].unique()
    Total JK: {} \n{}\n'.format(len(jk), jk))
NAMBAH KOLOM 'STATUS GIZI'

g Nilai IMT menggunakan rumus (IMT = Berat/Tinggi^2)
```

```
df['IMT'] = np.round(df['Berat badan (dalam
kg)']/(df['Tinggi Badan (dalam cm)'] * df['Tinggi Badan
(dalam cm)'] / 10000), 2)

# Menentukan nilai Status Gizi tiap data

def count_z_score(jenis_kelamin, tahun, bulan, imt):
  def count_z(idx):
      if (imt > status_gender['Median'][idx[0]]):
        z = (imt - status_gender['Median'][idx[0]]) /
(status_gender['+1 SD'][idx[0]] -
status_gender['Median'][idx[0]])
      else:
        z = (imt - status_gender['Median'][idx[0]]) /
(status_gender['Median'][idx[0]] - status_gender['-1
SD'][idx[0]])
      return z

  if jenis_kelamin == 'Laki-Laki':
    status_gender = data_status_gizi_laki.copy()
  else:
    status_gender = data_status_gizi_perempuan.copy()

  if (tahun < 5):
    tahun = tahun * 12
    bulan = tahun + bulan
    index = status_gender.index[status_gender['Bulan'] ==
bulan].tolist()
    z_score = count_z(index)
  else:
    index = status_gender.index[(status_gender['Tahun'] ==
tahun) & (status_gender['Bulan'] == bulan)].tolist()
    if ((tahun == 5) & (bulan == 0)):
      index = status_gender.index[(status_gender['Tahun'] ==
5) & (status_gender['Bulan'] == 1)].tolist()
      index[0] = index[0] - 1
    z_score = count_z(index)
  return z_score

# z_score = np.round(count_z_score('Perempuan', 10, 0,
19.6), 2)
    .(z_score)
    .(df['Jenis Kelamin'][i])
    .(int(df['tahun'][i]))
    .(int(df['bulan'][i]))
```

```
# print(df['IMT'][i])

status_gizi = []
i = 0
for imt in df['IMT']:
  z_score = np.round(count_z_score(df['Jenis Kelamin'][i],
int(df['Tahun'][i]), int(df['Bulan'][i]), df['IMT'][i]), 2)

  if (int(df['Tahun'][i]) < 5):
    if (z_score < -3):
      status = "Gizi buruk"
    elif ((z_score > -3) & (z_score < -2)):
      status = "Gizi kurang"
    elif ((z_score >= -2) & (z_score < 1)):
      status = "Gizi baik"
    elif ((z_score >= 1) & (z_score < 2)):
      status = "Berisiko gizi lebih"
    elif ((z_score >= 2) & (z_score < 3)):
      status = "Gizi lebih"
    elif ((z_score) >= 3):
      status = "Obesitas"
  else:
    if ((z_score >= -3) & (z_score < -2)):
      status = "Gizi kurang"
    elif ((z_score >= -2) & (z_score < 1)):
      status = "Gizi baik"
    elif ((z_score >= 1) & (z_score < 2)):
      status = "Gizi lebih"
    elif ((z_score) >= 2):
      status = "Obesitas"

  # print("IMT = ", imt, ", Z score = ", z_score)
  status_gizi.append(status)
  i += 1

df['Status Gizi'] = status_gizi
df.head()
df['riwayat DBD orang serumah'] = df['riwayat DBD orang
serumah'].replace(['Ada tapi cuma 1 minggu','1 minggu', '4
tahun lalu', ""],
                                                    ['Ya'
, 'Ya', 'Ya','Ya'])
    opname = []
    n df['penyakit opname']:
    st = re.split(', ', i.upper())
    r_opname.append(to_list)
```

```
df['penyakit opname'] = daftar_opname
df.head()
opname = df['penyakit opname'].explode().unique()

for op in opname:
  if op == 'Tidak Pernah':
    continue
  new_col = []
  for row in df['penyakit opname']:
    if op in row:
      new_col.append('Ya')
    else:
      new_col.append('Tidak')
  df[op + ' (opname)'] = new_col

df.head()
data = df.loc[df['Pernah DBD'] == 'Ya'].copy()
data_perkota = data.copy()
data_perkota['Jumlah DBD'] = data_perkota['umur']
data_perkota = data_perkota.groupby('Kab/Kota').agg({
        'Jumlah DBD':'count',
})
data_perkota
data = data.loc[data['Kab/Kota'].isin(['Makassar', 'Gowa',
'Jeneponto', 'Maros', 'Pangkep', 'Takalar'])]
l=data.columns
kolom = []
persentase = []

for i in l:
    res=data[i].value_counts(normalize=True)*100
    if ('opname' in i or 'orang serumah' in i or i ==
'riwayat DBD orang serumah'):
      kolom.append(i)
      split = str(res).split()[0:4]
      split[1] = str(round(float(split[1]), 3)) + '%'
      x = ' '.join(split)
      persentase.append(x)
      if res.iloc[0]>=80:
          del data[i]

l.DataFrame({'Penyakit':kolom,
tase':persentase})

ad()
```

```
data['Kecamatan '] = data['Kecamatan
'].replace(['Rappoccini','Tamalate ', 'TAMALATE',
'Tamalanrea ', 'Tamalanrea Jaya', 'Somba opu', 'KEC.
MAMAJANG', 'Panakukkang','Bontomarannu '],
                                                    ['Rappoc
ini', 'Tamalate', 'Tamalate','Tamalanrea', 'Tamalanrea',
'Somba Opu','Mamajang', 'Panakkukang', 'Bontomarannu' ])
#=============== CREATE DICTIONARY FOR OBJECT AND NUMERICAL
COLUMNS ================#
dict_cat = {}
dict_num = {}

for cat in data.select_dtypes(['object', 'category']):
  if (cat == 'Kecamatan'):
    continue
  dict_cat[cat] = lambda x: x.value_counts().index[0]

for num in data.select_dtypes(['int64','float64']):
  if (num == 'Total'):
    continue
  dict_num[num] = ['mean']

#============ CREATE TABLE FOR EACH KECAMATAN ============#

data['Total'] = data['umur']
data_perkecamatan = data.groupby('Kecamatan ').agg({

        'Total':'count',
        **dict_num,
        **dict_cat

})
data_perkecamatan
riwayat_col = ['riwayat vaksin CYD', 'penyakit lainnya',
'riwayat DBD orang serumah', 'genangan air sekitar']

for k in riwayat_col:
  new_col = []
  for i in data_perkecamatan.index:
    total = 0
    for j in data.index:
      if (i == data['Kecamatan '][j] and data[k][j] ==

        total = total + 1
    _col.append(total)
```

```
  data_perkecamatan[k] = new_col

riwayat_col = ['Status Gizi']
for k in riwayat_col:
  new_col = []
  for i in data_perkecamatan.index:
    total = 0
    for j in data.index:
      if (i == data['Kecamatan '][j] and data[k][j] == 'Gizi
baik'):
        total = total + 1
    new_col.append(total)

  data_perkecamatan['Status Gizi baik'] = new_col

for k in riwayat_col:
  new_col = []
  for i in data_perkecamatan.index:
    total = 0
    for j in data.index:
      if (i == data['Kecamatan '][j] and (data[k][j] ==
'Berisiko gizi lebih' or data[k][j] == 'Gizi lebih' or
data[k][j] == 'Obesitas')):
        total = total + 1
    new_col.append(total)

  data_perkecamatan['Status Gizi lebih'] = new_col

for k in riwayat_col:
  new_col = []
  for i in data_perkecamatan.index:
    total = 0
    for j in data.index:
      if (i == data['Kecamatan '][j] and (data[k][j] ==
'Gizi kurang' or data[k][j] == 'Gizi buruk')):
        total = total + 1
    new_col.append(total)

  data_perkecamatan['Status Gizi kurang'] = new_col
riwayat_col = ['Pendapatan Orang Tua']
for k in riwayat_col:
  new_col = []
      in data_perkecamatan.index:
    al = 0
    j in data.index:
```

```
      if (i == data['Kecamatan '][j] and data[k][j] == '<
2.500.000'):
        total = total + 1
    new_col.append(total)

  data_perkecamatan['Pendapatan ( < 2.500.000 )'] = new_col

for k in riwayat_col:
  new_col = []
  for i in data_perkecamatan.index:
    total = 0
    for j in data.index:
      if (i == data['Kecamatan '][j] and data[k][j] ==
'2.500.000 - 5.000.000'):
        total = total + 1
    new_col.append(total)

  data_perkecamatan['Pendapatan ( 2.500.000 - 5.000.000 )']
= new_col

for k in riwayat_col:
  new_col = []
  for i in data_perkecamatan.index:
    total = 0
    for j in data.index:
      if (i == data['Kecamatan '][j] and data[k][j] ==
'5.000.000 - 10.000.000'):
        total = total + 1
    new_col.append(total)

  data_perkecamatan['Pendapatan ( 5.000.000 - 10.000.000 )']
= new_col

for k in riwayat_col:
  new_col = []
  for i in data_perkecamatan.index:
    total = 0
    for j in data.index:
      if (i == data['Kecamatan '][j] and data[k][j] == '>
10.000.000'):
        total = total + 1
    new_col.append(total)

    perkecamatan['Pendapatan ( > 10.000.000 )'] = new_col

    _col = ['luas rumah']
```

```
for k in riwayat_col:
  new_col = []
  for i in data_perkecamatan.index:
    total = 0
    for j in data.index:
      if (i == data['Kecamatan '][j] and data[k][j] == '< 36
m^2'):
        total = total + 1
    new_col.append(total)

  data_perkecamatan['Luas rumah ( < 36 m^2 )'] = new_col

riwayat_col = ['luas rumah']
for k in riwayat_col:
  new_col = []
  for i in data_perkecamatan.index:
    total = 0
    for j in data.index:
      if (i == data['Kecamatan '][j] and data[k][j] == '36 -
54 m^2'):
        total = total + 1
    new_col.append(total)

  data_perkecamatan['Luas rumah ( 36 - 54 m^2 )'] = new_col

riwayat_col = ['luas rumah']
for k in riwayat_col:
  new_col = []
  for i in data_perkecamatan.index:
    total = 0
    for j in data.index:
      if (i == data['Kecamatan '][j] and data[k][j] == '54 -
120 m^2'):
        total = total + 1
    new_col.append(total)

  data_perkecamatan['Luas rumah ( 54 - 120 m^2 )'] = new_col

riwayat_col = ['luas rumah']
for k in riwayat_col:
  new_col = []
  for i in data_perkecamatan.index:
      al = 0
      j in data.index:
      f (i == data['Kecamatan '][j] and data[k][j] == '>
      '):
```

```
            total = total + 1
        new_col.append(total)

    data_perkecamatan['Luas rumah ( > 120 m^2 )'] = new_col
data_perkecamatan = data_perkecamatan.droplevel(1, axis=1)
data_perkecamatan =
data_perkecamatan.rename(columns={'Total':'Jumlah DBD',
'Total':'Jumlah DBD', 'jumlah kamar tidur':'Jumlah kamar
(mean)', 'jumlah orang dalam rumah':'Jumlah orang di rumah
(mean)', 'vaksin DBD':'Persentase anak telah DBD', 'riwayat
DBD orang serumah':'Persentase kasus dengan riwayat DBD
serumah', 'Status Gizi baik':'Persentase status gizi baik',
'Status Gizi lebih':'Persentase status gizi lebih', 'Status
Gizi kurang':'Persentase status gizi kurang'})
data_perkecamatan.index = data_perkecamatan.index + ' (' +
data_perkecamatan['Kab/Kota'] + ')'
data_perkecamatan = data_perkecamatan.drop(['Pendapatan
Orang Tua', 'luas rumah', 'Kab/Kota', 'Kelurahan'], axis=1)
data_perkecamatan.info()
data_perkecamatan['Pendapatan ( < 2.500.000 )'] =
(data_perkecamatan['Pendapatan ( < 2.500.000
)']/data_perkecamatan['Jumlah DBD']) * 100
data_perkecamatan['Pendapatan ( 2.500.000 - 5.000.000 )'] =
(data_perkecamatan['Pendapatan ( 2.500.000 - 5.000.000
)']/data_perkecamatan['Jumlah DBD']) * 100
data_perkecamatan['Pendapatan ( 5.000.000 - 10.000.000 )'] =
(data_perkecamatan['Pendapatan ( 5.000.000 - 10.000.000
)']/data_perkecamatan['Jumlah DBD']) * 100
data_perkecamatan['Pendapatan ( > 10.000.000 )'] =
(data_perkecamatan['Pendapatan ( > 10.000.000
)']/data_perkecamatan['Jumlah DBD']) * 100

data_perkecamatan['Luas rumah ( < 36 m^2 )'] =
(data_perkecamatan['Luas rumah ( < 36 m^2
)']/data_perkecamatan['Jumlah DBD']) * 100
data_perkecamatan['Luas rumah ( 36 - 54 m^2 )'] =
(data_perkecamatan['Luas rumah ( 36 - 54 m^2
)']/data_perkecamatan['Jumlah DBD']) * 100
data_perkecamatan['Luas rumah ( 54 - 120 m^2 )'] =
(data_perkecamatan['Luas rumah ( 54 - 120 m^2
)']/data_perkecamatan['Jumlah DBD']) * 100
data_perkecamatan['Luas rumah ( > 120 m^2 )'] =
    erkecamatan['Luas rumah ( > 120 m^2
    a_perkecamatan['Jumlah DBD']) * 100
```

```
data_perkecamatan['riwayat vaksin CYD'] =
(data_perkecamatan['riwayat vaksin
CYD']/data_perkecamatan['Jumlah DBD']) * 100
data_perkecamatan['Persentase kasus dengan riwayat DBD
serumah'] = (data_perkecamatan['Persentase kasus dengan
riwayat DBD serumah']/data_perkecamatan['Jumlah DBD']) * 100
data_perkecamatan['genangan air sekitar'] =
(data_perkecamatan['genangan air
sekitar']/data_perkecamatan['Jumlah DBD']) * 100

data_perkecamatan['Persentase status gizi baik'] =
(data_perkecamatan['Persentase status gizi
baik']/data_perkecamatan['Jumlah DBD']) * 100
data_perkecamatan['Persentase status gizi lebih'] =
(data_perkecamatan['Persentase status gizi
lebih']/data_perkecamatan['Jumlah DBD']) * 100
data_perkecamatan['Persentase status gizi kurang'] =
(data_perkecamatan['Persentase status gizi
kurang']/data_perkecamatan['Jumlah DBD']) * 100
data_perkecamatan
data_perkecamatan.info()
data1 = data_perkecamatan.copy()
cluster1 = data1[['Jumlah DBD', 'Tahun']]

# ============= DATA TRANSFORMATION ============== #
# preprocessing numerical

scaler = StandardScaler()
Num_features =
cluster1.select_dtypes(include=['int64','float64']).columns
scaler.fit(cluster1[Num_features])
cluster1[Num_features] =
scaler.transform(cluster1[Num_features])

#OPTIONAL: Elbow plot with inertia
#Elbow method to choose the optimal number of clusters
# # ============= Function for plotting elbow curve
=============== #
def plot_elbow_curve_k_means(start, end, data):
  sse = {}
  score = {}

  : in (range(start, end+1)):
  ans = KMeans(n_clusters=k, random_state=9)
  ds = kmeans.fit_predict(data)
```

```
    sse[k] = kmeans.inertia_ # Inertia: Sum of distances of
samples to their closest cluster center
    score[k] = (round(silhouette_score(data, preds),4))
    print("For n_clusters = {}, silhouette score is
{}".format(k, round(silhouette_score(data, preds), 4)))
    print("For n_clusters = {}, SSE is {}".format(k,
round(kmeans.inertia_, 4)))

  sns.set_theme(style="whitegrid", palette="bright",
font_scale=1.2)

  plt.figure(figsize=(10, 5))
  ax = sns.lineplot(x=list(sse.keys()),
y=list(sse.values()), marker="o", dashes=False,
color="blue")
  ax.set_title('Elbow curve', fontsize=18)
  ax.set_xlabel('No of clusters', fontsize=14)
  ax.set_ylabel('SSE / Inertia', fontsize=14)
  ax.set(xlim=(start-0.1, end+0.1))
  plt.plot()

  plt.figure(figsize=(10, 5))
  ax = sns.lineplot(x=list(score.keys()),
y=list(score.values()), marker="o", dashes=False,
color="blue")
  ax.set_title('Graph Silhouette', fontsize=18)
  ax.set_xlabel('No of clusters', fontsize=14)
  ax.set_ylabel('Silhouette Score', fontsize=14)
  ax.set(xlim=(start-0.1, end+0.1))
  plt.plot()

# Plotting elbow curve for k=2 to k=10
plot_elbow_curve_k_means(2,10,cluster1)
#Actual Clustering
kmeans = KMeans(n_clusters=4, random_state=9)
preds = kmeans.fit_predict(cluster1)

pd.Series(kmeans.labels_).value_counts()
#new column for cluster labels associated with each subject
cluster1['labels'] = kmeans.labels_
cluster1['Cluster'] = cluster1['labels'].map({0:'First',
1:'Second',2:'Third',3: 'Fourth'})

    the cluster
1['Cluster'] = cluster1['Cluster'].astype('category')
```

```
cluster1['Cluster'] =
cluster1['Cluster'].cat.reorder_categories(['First','Second'
,'Third', 'Fourth'])

#============== INVERSE TRANSFORMATION FOR NUMERIC DATA
================#

cluster1[Num_features] =
scaler.inverse_transform(cluster1[Num_features])

#=============== CREATE DICTIONARY FOR OBJECT AND NUMERICAL
COLUMNS ================#
dict_cat = {}
dict_num = {}

for cat in cluster1.select_dtypes('object'):
  dict_cat[cat] = lambda x: x.value_counts().index[0]

for num in cluster1.select_dtypes(['int64','float64']):
  dict_num[num] = ['mean']

#============ CREATE TABLE FOR EACH CLUSTER ============#

cluster1.rename(columns = {'labels':'Total'}, inplace =
True)
data_percluster = cluster1.groupby('Cluster').agg({

        'Total':'count',
        **dict_num,
        **dict_cat

}).T

data_percluster
cluster1[cluster1['Cluster']== 'First']
cluster1[cluster1['Cluster']== 'Second']
cluster1[cluster1['Cluster']== 'Third']
cluster1[cluster1['Cluster']== 'Fourth']
cluster1
cluster1_test = data1[['Jumlah DBD', 'Tahun']]

# ============= DATA TRANSFORMATION ============== #
     ocessing numerical

     = StandardScaler()
```

```
Num_features =
cluster1_test.select_dtypes(include=['int64','float64']).col
umns
scaler.fit(cluster1_test[Num_features])
cluster1_test[Num_features] =
scaler.transform(cluster1_test[Num_features])
cluster1_test
# Getting the values and plotting it
f1 = cluster1_test['Jumlah DBD'].values
f2 = cluster1_test['Tahun'].values
X = np.array(list(zip(f1, f2)))
plt.scatter(f1,f2 ,c='black', s=7)

# Number of clusters
k = 4
# X coordinates of random centroids
C_x = np.random.uniform(np.min(X), np.max(X), size=k)
# Y coordinates of random centroids
C_y = np.random.uniform(np.min(X), np.max(X), size=k)
C = np.array(list(zip(C_x, C_y)), dtype=np.float32)
print(C)
# Euclidean Distance Caculator
def dist(a, b, ax=1):
    return np.linalg.norm(a - b, axis=ax)

# Cluster Lables(0, 1, 2)
clusters = np.zeros(len(X))

# Assigning each value to its closest cluster
# jarak = []
for i in range(len(X)):
  distances = dist(X[i], C)
  cluster = np.argmin(distances)
  clusters[i] = cluster
  print(distances)
from copy import deepcopy
# To store the value of centroids when it updates
C_old = np.zeros(C.shape)
# Cluster Lables(0, 1, 2)
clusters = np.zeros(len(X))
# Error func. - Distance between new centroids and old
centroids
    : dist(C, C_old, None)
    will run till the error becomes zero
    0
    rror != 0:
```

```
        # Assigning each value to its closest cluster
        for i in range(len(X)):
            distances = dist(X[i], C)
            cluster = np.argmin(distances)
            clusters[i] = cluster

        # Storing the old centroid values
        C_old = deepcopy(C)
        # Finding the new centroids by taking the average value
        for i in range(k):
            points = [X[j] for j in range(len(X)) if clusters[j]
== i]
            C[i] = np.mean(points, axis=0)
        error = dist(C, C_old, None)
        iter += 1
        # print('error : ', error)
        print('iteration :', iter)
        # print('previous centroid : \n', C_old)
        print('centroid : \n', C, '\n')

        if iter == 5:
          break
cluster2 = data1[['Persentase status gizi baik', 'Persentase
status gizi lebih', 'Persentase status gizi kurang']]

# ============= DATA TRANSFORMATION ============== #
# preprocessing numerical

scaler = StandardScaler()
Num_features =
cluster2.select_dtypes(include=['int64','float64']).columns
scaler.fit(cluster2[Num_features])
cluster2[Num_features] =
scaler.transform(cluster2[Num_features])
#OPTIONAL: Elbow plot with inertia
#Elbow method to choose the optimal number of clusters
# # ============= Function for plotting elbow curve
=============== #
def plot_elbow_curve_k_means(start, end, data):
  sse = {}
  score = {}

    : in (range(start, end+1)):
    ans = KMeans(n_clusters=k, random_state=9)
    ds = kmeans.fit_predict(data)
```

```
    sse[k] = kmeans.inertia_ # Inertia: Sum of distances of
samples to their closest cluster center
    score[k] = (round(silhouette_score(data, preds),4))
    print("For n_clusters = {}, silhouette score is
{}".format(k, round(silhouette_score(data, preds), 4)))
    print("For n_clusters = {}, SSE is {}".format(k,
round(kmeans.inertia_, 4)))

  sns.set_theme(style="whitegrid", palette="bright",
font_scale=1.2)

  plt.figure(figsize=(10, 5))
  ax = sns.lineplot(x=list(sse.keys()),
y=list(sse.values()), marker="o", dashes=False,
color="blue")
  ax.set_title('Elbow curve', fontsize=18)
  ax.set_xlabel('No of clusters', fontsize=14)
  ax.set_ylabel('SSE / Inertia', fontsize=14)
  ax.set(xlim=(start-0.1, end+0.1))
  plt.plot()

  plt.figure(figsize=(10, 5))
  ax = sns.lineplot(x=list(score.keys()),
y=list(score.values()), marker="o", dashes=False,
color="blue")
  ax.set_title('Graph Silhouette', fontsize=18)
  ax.set_xlabel('No of clusters', fontsize=14)
  ax.set_ylabel('Silhouette Score', fontsize=14)
  ax.set(xlim=(start-0.1, end+0.1))
  plt.plot()

# Plotting elbow curve for k=2 to k=10
plot_elbow_curve_k_means(2,10,cluster2)
#Actual Clustering
kmeans = KMeans(n_clusters=3, random_state=9)
preds = kmeans.fit_predict(cluster2)

pd.Series(kmeans.labels_).value_counts()
#new column for cluster labels associated with each subject
cluster2['labels'] = kmeans.labels_
cluster2['Cluster'] = cluster2['labels'].map({0:'First',
1:'Second',2:'Third'})

    the cluster
2['Cluster'] = cluster2['Cluster'].astype('category')
```

```
cluster2['Cluster'] =
cluster2['Cluster'].cat.reorder_categories(['First','Second'
,'Third'])

#============== INVERSE TRANSFORMATION FOR NUMERIC DATA
================#

cluster2[Num_features] =
scaler.inverse_transform(cluster2[Num_features])
#============== CREATE DICTIONARY FOR OBJECT AND NUMERICAL
COLUMNS ================#
dict_cat = {}
dict_num = {}

for cat in cluster2.select_dtypes('object'):
  dict_cat[cat] = lambda x: x.value_counts().index[0]

for num in cluster2.select_dtypes(['int64','float64']):
  # dict_num[num] = ['mean', 'median', 'min', 'max']
  dict_num[num] = ['mean']

#============ CREATE TABLE FOR EACH CLUSTER ============#

cluster2.rename(columns = {'labels':'Total'}, inplace =
True)
data_percluster = cluster2.groupby('Cluster').agg({

        'Total':'count',
        **dict_num,
        **dict_cat

}).T

data_percluster
cluster2[cluster2['Cluster']== 'First']
cluster2[cluster2['Cluster']== 'Second']
cluster2[cluster2['Cluster']== 'Third']
cluster2

cluster3 = data1[['Pendapatan ( < 2.500.000 )', 'Pendapatan
( 2.500.000 - 5.000.000 )', 'Pendapatan ( 5.000.000 -
10.000.000 )', 'Pendapatan ( > 10.000.000 )']]

        ======== DATA TRANSFORMATION ============== #
        ocessing numerical
```

```
scaler = StandardScaler()
Num_features =
cluster3.select_dtypes(include=['int64','float64']).columns
scaler.fit(cluster3[Num_features])
cluster3[Num_features] =
scaler.transform(cluster3[Num_features])
#OPTIONAL: Elbow plot with inertia
#Elbow method to choose the optimal number of clusters
# # ============= Function for plotting elbow curve
============== #
def plot_elbow_curve_k_means(start, end, data):
  sse = {}
  score = {}

  for k in (range(start, end+1)):
    kmeans = KMeans(n_clusters=k, random_state=9)
    preds = kmeans.fit_predict(data)
    sse[k] = kmeans.inertia_ # Inertia: Sum of distances of
samples to their closest cluster center
    score[k] = (round(silhouette_score(data, preds),4))
    print("For n_clusters = {}, silhouette score is
{}".format(k, round(silhouette_score(data, preds), 4)))
    print("For n_clusters = {}, SSE is {}".format(k,
round(kmeans.inertia_, 4)))

  sns.set_theme(style="whitegrid", palette="bright",
font_scale=1.2)

  plt.figure(figsize=(10, 5))
  ax = sns.lineplot(x=list(sse.keys()),
y=list(sse.values()), marker="o", dashes=False,
color="blue")
  ax.set_title('Elbow curve', fontsize=18)
  ax.set_xlabel('No of clusters', fontsize=14)
  ax.set_ylabel('SSE / Inertia', fontsize=14)
  ax.set(xlim=(start-0.1, end+0.1))
  plt.plot()

  plt.figure(figsize=(10, 5))
  ax = sns.lineplot(x=list(score.keys()),
y=list(score.values()), marker="o", dashes=False,
color="blue")
    t_title('Graph Silhouette', fontsize=18)
    t_xlabel('No of clusters', fontsize=14)
    t_ylabel('Silhouette Score', fontsize=14)
    t(xlim=(start-0.1, end+0.1))
```

```
  plt.plot()

# Plotting elbow curve for k=2 to k=10
plot_elbow_curve_k_means(2,10,cluster3)
#Actual Clustering
kmeans = KMeans(n_clusters=5, random_state=9)
preds = kmeans.fit_predict(cluster3)

pd.Series(kmeans.labels_).value_counts()
#new column for cluster labels associated with each subject
cluster3['labels'] = kmeans.labels_
cluster3['Cluster'] = cluster3['labels'].map({0:'First',
1:'Second',2:'Third',3:'Fourth', 4:'Fifth'})

# Order the cluster
cluster3['Cluster'] = cluster3['Cluster'].astype('category')
cluster3['Cluster'] =
cluster3['Cluster'].cat.reorder_categories(['First','Second'
,'Third', 'Fourth', 'Fifth'])

#============== INVERSE TRANSFORMATION FOR NUMERIC DATA
================#

cluster3[Num_features] =
scaler.inverse_transform(cluster3[Num_features])
cluster3[cluster3['Cluster']== 'First']
cluster3[cluster3['Cluster']== 'Second']
cluster3[cluster3['Cluster']== 'Third']
cluster3[cluster3['Cluster']== 'Fourth']
cluster3[cluster3['Cluster']== 'Fifth']
#=============== CREATE DICTIONARY FOR OBJECT AND NUMERICAL
COLUMNS ================#
dict_cat = {}
dict_num = {}

for cat in cluster3.select_dtypes('object'):
  dict_cat[cat] = lambda x: x.value_counts().index[0]

for num in cluster3.select_dtypes(['int64','float64']):
  # dict_num[num] = ['mean', 'median', 'min', 'max']
  dict_num[num] = ['mean']
```


```
======= CREATE TABLE FOR EACH CLUSTER ============#

3.rename(columns = {'labels':'Total'}, inplace =
```

```
data_percluster = cluster3.groupby('Cluster').agg({

        'Total':'count',
        **dict_num,
        **dict_cat

}).T

data_percluster

cluster4 = data1[['Luas rumah ( < 36 m^2 )', 'Luas rumah (
36 - 54 m^2 )', 'Luas rumah ( 54 - 120 m^2 )', 'Luas rumah (
> 120 m^2 )']]

# ============= DATA TRANSFORMATION ============== #
# preprocessing numerical

scaler = StandardScaler()
Num_features =
cluster4.select_dtypes(include=['int64','float64']).columns
scaler.fit(cluster4[Num_features])
cluster4[Num_features] =
scaler.transform(cluster4[Num_features])
#OPTIONAL: Elbow plot with inertia
#Elbow method to choose the optimal number of clusters
# # ============= Function for plotting elbow curve
=============== #
def plot_elbow_curve_k_means(start, end, data):
  sse = {}
  score = {}

  for k in (range(start, end+1)):
    kmeans = KMeans(n_clusters=k, random_state=9)
    preds = kmeans.fit_predict(data)
    sse[k] = kmeans.inertia_ # Inertia: Sum of distances of
samples to their closest cluster center
    score[k] = (round(silhouette_score(data, preds),4))
    print("For n_clusters = {}, silhouette score is
{}".format(k, round(silhouette_score(data, preds), 4)))
    print("For n_clusters = {}, SSE is {}".format(k,
round(kmeans.inertia_, 4)))
```

```
et_theme(style="whitegrid", palette="bright",
ale=1.2)

igure(figsize=(10, 7))
```

```
  ax = sns.lineplot(x=list(sse.keys()),
y=list(sse.values()), marker="o", dashes=False,
color="blue")
  ax.set_title('Elbow curve', fontsize=18)
  ax.set_xlabel('No of clusters', fontsize=14)
  ax.set_ylabel('SSE / Inertia', fontsize=14)
  ax.set(xlim=(start-0.1, end+0.1))
  plt.plot()

  plt.figure(figsize=(10, 7))
  ax = sns.lineplot(x=list(score.keys()),
y=list(score.values()), marker="o", dashes=False,
color="blue")
  ax.set_title('Graph Silhouette', fontsize=18)
  ax.set_xlabel('No of clusters', fontsize=14)
  ax.set_ylabel('Silhouette Score', fontsize=14)
  ax.set(xlim=(start-0.1, end+0.1))
  plt.plot()

# Plotting elbow curve for k=2 to k=10
plot_elbow_curve_k_means(2,10,cluster4)
#Actual Clustering
kmeans = KMeans(n_clusters=4,random_state=9)
preds = kmeans.fit_predict(cluster4)

pd.Series(kmeans.labels_).value_counts()
#new column for cluster labels associated with each subject
cluster4['labels'] = kmeans.labels_
cluster4['Cluster'] = cluster4['labels'].map({0:'First',
1:'Second',2:'Third',3:'Fourth'})

# Order the cluster
cluster4['Cluster'] = cluster4['Cluster'].astype('category')
cluster4['Cluster'] =
cluster4['Cluster'].cat.reorder_categories(['First','Second'
,'Third','Fourth'])

#============== INVERSE TRANSFORMATION FOR NUMERIC DATA
================#

cluster4[Num_features] =
scaler.inverse_transform(cluster4[Num_features])
    ========= CREATE DICTIONARY FOR OBJECT AND NUMERICAL
  ================#
  t = {}
  m = {}
```

```
for cat in cluster4.select_dtypes('object'):
  dict_cat[cat] = lambda x: x.value_counts().index[0]

for num in cluster4.select_dtypes(['int64','float64']):
  # dict_num[num] = ['mean', 'median', 'min', 'max']
  dict_num[num] = ['mean']

#============ CREATE TABLE FOR EACH CLUSTER ============#

cluster4.rename(columns = {'labels':'Total'}, inplace =
True)
data_percluster = cluster4.groupby('Cluster').agg({

        'Total':'count',
        **dict_num,
        **dict_cat

}).T

data_percluster
cluster4[cluster4['Cluster']== 'First']
cluster4[cluster4['Cluster']== 'Second']
cluster4[cluster4['Cluster']== 'Third']
cluster4[cluster4['Cluster']== 'Fourth']

cluster5 = data1[['Jumlah DBD', 'jumlah kamar', 'jumlah
orang', 'genangan air sekitar']]

# ============= DATA TRANSFORMATION ============== #
# preprocessing numerical

scaler = StandardScaler()
Num_features =
cluster5.select_dtypes(include=['int64','float64']).columns
scaler.fit(cluster5[Num_features])
cluster5[Num_features] =
scaler.transform(cluster5[Num_features])
#OPTIONAL: Elbow plot with inertia
#Elbow method to choose the optimal number of clusters
# # ============= Function for plotting elbow curve
=============== #
    t_elbow_curve_k_means(start, end, data):
    : {}
    : = {}
```

```
  for k in (range(start, end+1)):
    kmeans = KMeans(n_clusters=k, random_state=9)
    preds = kmeans.fit_predict(data)
    sse[k] = kmeans.inertia_ # Inertia: Sum of distances of
samples to their closest cluster center
    score[k] = (round(silhouette_score(data, preds),4))
    print("For n_clusters = {}, silhouette score is
{}".format(k, round(silhouette_score(data, preds), 4)))
    print("For n_clusters = {}, SSE is {}".format(k,
round(kmeans.inertia_, 4)))

  sns.set_theme(style="whitegrid", palette="bright",
font_scale=1.2)

  plt.figure(figsize=(10, 7))
  ax = sns.lineplot(x=list(sse.keys()),
y=list(sse.values()), marker="o", dashes=False,
color="blue")
  ax.set_title('Elbow curve', fontsize=18)
  ax.set_xlabel('No of clusters', fontsize=14)
  ax.set_ylabel('SSE / Inertia', fontsize=14)
  ax.set(xlim=(start-0.1, end+0.1))
  plt.plot()

  plt.figure(figsize=(10, 7))
  ax = sns.lineplot(x=list(score.keys()),
y=list(score.values()), marker="o", dashes=False,
color="blue")
  ax.set_title('Graph Silhouette', fontsize=18)
  ax.set_xlabel('No of clusters', fontsize=14)
  ax.set_ylabel('Silhouette Score', fontsize=14)
  ax.set(xlim=(start-0.1, end+0.1))
  plt.plot()

# Plotting elbow curve for k=2 to k=10
plot_elbow_curve_k_means(2,10,cluster5)
#Actual Clustering
kmeans = KMeans(n_clusters=6,random_state=9)
preds = kmeans.fit_predict(cluster5)

pd.Series(kmeans.labels_).value_counts()
#new column for cluster labels associated with each subject
    5['labels'] = kmeans.labels_
    5['Cluster'] = cluster5['labels'].map({0:'First',
nd',2:'Third',3:'Fourth', 4:'Fifth', 5:'Sixth'})
```

```
# Order the cluster
cluster5['Cluster'] = cluster5['Cluster'].astype('category')
cluster5['Cluster'] =
cluster5['Cluster'].cat.reorder_categories(['First','Second'
,'Third','Fourth','Fifth','Sixth'])

#============== INVERSE TRANSFORMATION FOR NUMERIC DATA
================#

cluster5[Num_features] =
scaler.inverse_transform(cluster5[Num_features])
#=============== CREATE DICTIONARY FOR OBJECT AND NUMERICAL
COLUMNS ================#
dict_cat = {}
dict_num = {}

for cat in cluster5.select_dtypes('object'):
  dict_cat[cat] = lambda x: x.value_counts().index[0]

for num in cluster5.select_dtypes(['int64','float64']):
  # dict_num[num] = ['mean', 'median', 'min', 'max']
  dict_num[num] = ['mean']

#============ CREATE TABLE FOR EACH CLUSTER ============#

cluster5.rename(columns = {'labels':'Total'}, inplace =
True)
data_percluster = cluster5.groupby('Cluster').agg({

        'Total':'count',
        **dict_num,
        **dict_cat

}).T

data_percluster
cluster5[cluster5['Cluster']== 'First']
cluster5[cluster5['Cluster']== 'Second']
cluster5[cluster5['Cluster']== 'Third']
cluster5[cluster5['Cluster']== 'Fourth']
cluster5[cluster5['Cluster']== 'Fifth']
cluster5[cluster5['Cluster']== 'Sixth']

6 = data1[['riwayat vaksin CYD', 'Persentase kasus
riwayat DBD serumah']]
```

```
# ============= DATA TRANSFORMATION ============== #
# preprocessing numerical

scaler = StandardScaler()
Num_features =
cluster6.select_dtypes(include=['int64','float64']).columns
scaler.fit(cluster6[Num_features])
cluster6[Num_features] =
scaler.transform(cluster6[Num_features])
#OPTIONAL: Elbow plot with inertia
#Elbow method to choose the optimal number of clusters
# # ============= Function for plotting elbow curve
=============== #
def plot_elbow_curve_k_means(start, end, data):
  sse = {}
  score = {}

  for k in (range(start, end+1)):
    kmeans = KMeans(n_clusters=k, random_state=9)
    preds = kmeans.fit_predict(data)
    sse[k] = kmeans.inertia_ # Inertia: Sum of distances of
samples to their closest cluster center
    score[k] = (round(silhouette_score(data, preds),4))
    print("For n_clusters = {}, silhouette score is
{}".format(k, round(silhouette_score(data, preds), 4)))
    print("For n_clusters = {}, SSE is {}".format(k,
round(kmeans.inertia_, 4)))

  sns.set_theme(style="whitegrid", palette="bright",
font_scale=1.2)

  plt.figure(figsize=(10, 7))
  ax = sns.lineplot(x=list(sse.keys()),
y=list(sse.values()), marker="o", dashes=False,
color="blue")
  ax.set_title('Elbow curve', fontsize=18)
  ax.set_xlabel('No of clusters', fontsize=14)
  ax.set_ylabel('SSE / Inertia', fontsize=14)
  ax.set(xlim=(start-0.1, end+0.1))
  plt.plot()

  plt.figure(figsize=(10, 7))
     sns.lineplot(x=list(score.keys()),
     score.values()), marker="o", dashes=False,
     blue")
    t_title('Graph Silhouette', fontsize=18)
```

```
  ax.set_xlabel('No of clusters', fontsize=14)
  ax.set_ylabel('Silhouette Score', fontsize=14)
  ax.set(xlim=(start-0.1, end+0.1))
  plt.plot()

# Plotting elbow curve for k=2 to k=10
plot_elbow_curve_k_means(2,10,cluster6)
#Actual Clustering
kmeans = KMeans(n_clusters=4,random_state=9)
preds = kmeans.fit_predict(cluster6)

pd.Series(kmeans.labels_).value_counts()
#new column for cluster labels associated with each subject
cluster6['labels'] = kmeans.labels_
cluster6['Cluster'] = cluster6['labels'].map({0:'First',
1:'Second',2:'Third',3:'Fourth'})

# Order the cluster
cluster6['Cluster'] = cluster6['Cluster'].astype('category')
cluster6['Cluster'] =
cluster6['Cluster'].cat.reorder_categories(['First','Second'
,'Third','Fourth'])

#============== INVERSE TRANSFORMATION FOR NUMERIC DATA
================#

cluster6[Num_features] =
scaler.inverse_transform(cluster6[Num_features])
#=============== CREATE DICTIONARY FOR OBJECT AND NUMERICAL
COLUMNS ================#
dict_cat = {}
dict_num = {}

for cat in cluster6.select_dtypes('object'):
  dict_cat[cat] = lambda x: x.value_counts().index[0]

for num in cluster6.select_dtypes(['int64','float64']):
  # dict_num[num] = ['mean', 'median', 'min', 'max']
  dict_num[num] = ['mean']

#============ CREATE TABLE FOR EACH CLUSTER ============#

6.rename(columns = {'labels':'Total'}, inplace =

rcluster = cluster6.groupby('Cluster').agg({
```

```
        'Total':'count',
        **dict_num,
        **dict_cat

}).T

data_percluster
cluster6[cluster6['Cluster']== 'First']
cluster6[cluster6['Cluster']== 'Second']
cluster6[cluster6['Cluster']== 'Third']
cluster6[cluster6['Cluster']== 'Fourth']
join_table = pd.concat([cluster1,cluster2, cluster3,
cluster4, cluster5, cluster6], axis=1)
join_table
```

Lampiran 7 Frequent Itemset

support	itemsets
0.911764706	({'Ya berGejala'})
0.686274510	({'Keluarga Tidak DBD'})
0.686274510	({'Tidak Pernah Ada Penderita'})
0.637254902	({'Ya Pernah DBD'})
0.637254902	({'Remaja akhir'})
0.627450980	({'Normal'})
0.578431373	({'pernhan Opname'})
0.490196078	({'Laki-Laki'})
0.480392157	({'DBD'})
0.441176471	({'Tidak lokasi rumah normal'})
0.401960784	({'Gaji2'})
0.382352941	({'Tidak berkawat'})
0.313725490	({'5 Penguni'})
0.225490196	({'Pernah Vaksin'})
0.225490196	({'Tipes'})
0.156862745	({'Luas1'})
0.156862745	({'2 Kamar'})
0.127450980	({'Gowa'})
0.117647059	({'Karyawan Swasta'})
0.784313725	({'Makassar'})
0.774509804	({'Tidak Pernah Vaksin'})
0.617647059	({'Jendela/Ventilasi berkawat'})
2549	({'Keluarga Terkena DBD'})
1372	({'Sehat'})
6274	({'Makassar - Tamalate'})

0.13725490 2	({'Gaji1'})
0.55882352 9	({'Ya ada genangan/lokasi dempet'})
0.42156862 7	({'Tidak Pernah Opname'})
0.39215686 3	({'Tidak Pernah'})
0.36274509 8	({'BelumPernah DBD'})
0.50980392 2	({'Perempuan'})
0.33333333 3	({'PNS'})
0.23529411 8	({'Luas2'})
0.21568627 5	({'4 Penguni'})
0.17647058 8	({'Kurus'})
0.18627451	({'Dewasa awal'})
0.13725490 2	({'Kelebihan Berat Badan'})
0.48039215 7	({'Luas3'})
0.35294117 6	({'3 Kamar'})
0.34313725 5	({'Wiraswasta'})
0.10784313 7	({'3 Penguni'})
0.17647058 8	({'6 Penguni'})
0.20588235 3	({'Makassar - Rappocini'})
0.21568627 5	({'4 Kamar'})
0.20588235 3	({'Gaji4'})
0.12745098	({'Luas4'})
0.17647058 8	({'5 Kamar'})
0.25490196 1	({'Gaji3'})
5098	({'Keluarga Tidak DBD', 'Ya berGejala'})
3137 3	({'Makassar', 'Keluarga Tidak DBD'})

0.549019608	({'Tidak Pernah Vaksin', 'Keluarga Tidak DBD'})
0.529411765	({'Makassar', 'Keluarga Tidak DBD', 'Ya berGejala'})
0.490196078	({'Tidak Pernah Vaksin', 'Keluarga Tidak DBD', 'Ya berGejala'})
0.460784314	({'Makassar', 'Tidak Pernah Vaksin', 'Keluarga Tidak DBD'})
0.411764706	({'Makassar', 'Tidak Pernah Vaksin', 'Keluarga Tidak DBD', 'Ya berGejala'})
0.686274510	({'Tidak Pernah Ada Penderita', 'Keluarga Tidak DBD'})
0.627450980	({'Tidak Pernah Ada Penderita', 'Ya berGejala'})
0.578431373	({'Tidak Pernah Ada Penderita', 'Makassar'})
0.549019608	({'Tidak Pernah Ada Penderita', 'Tidak Pernah Vaksin'})
0.627450980	({'Tidak Pernah Ada Penderita', 'Keluarga Tidak DBD', 'Ya berGejala'})
0.578431373	({'Tidak Pernah Ada Penderita', 'Makassar', 'Keluarga Tidak DBD'})
0.529411765	({'Tidak Pernah Ada Penderita', 'Makassar', 'Ya berGejala'})
0.529411765	({'Tidak Pernah Ada Penderita', 'Makassar', 'Keluarga Tidak DBD', 'Ya berGejala'})
0.549019608	({'Tidak Pernah Ada Penderita', 'Tidak Pernah Vaksin', 'Keluarga Tidak DBD'})
0.490196078	({'Tidak Pernah Ada Penderita', 'Tidak Pernah Vaksin', 'Ya berGejala'})
0.460784314	({'Tidak Pernah Ada Penderita', 'Makassar', 'Tidak Pernah Vaksin'})
0.490196078	({'Tidak Pernah Ada Penderita', 'Tidak Pernah Vaksin', 'Keluarga Tidak DBD', 'Ya berGejala'})
0.460784314	({'Tidak Pernah Ada Penderita', 'Makassar', 'Tidak Pernah Vaksin', 'Keluarga Tidak DBD'})
0.411764706	({'Tidak Pernah Ada Penderita', 'Makassar', 'Tidak Pernah Vaksin', 'Ya berGejala'})
0.411764706	({'Tidak Pernah Vaksin', 'Ya berGejala', 'Keluarga Tidak DBD', 'Tidak Pernah Ada Penderita', 'Makassar'})
0.607843137	({'Ya Pernah DBD', 'Ya berGejala'})
0.421568627	({'Tidak Pernah Ada Penderita', 'Ya Pernah DBD'})
0.421568627	({'Ya Pernah DBD', 'Keluarga Tidak DBD'})
9215	({'Makassar', 'Ya Pernah DBD'})
3823	({'Tidak Pernah Vaksin', 'Ya Pernah DBD'})

0.41176470 6	({'Tidak Pernah Ada Penderita', 'Ya Pernah DBD', 'Ya berGejala'})
0.31372549	({'Tidak Pernah Ada Penderita', 'Tidak Pernah Vaksin', 'Ya Pernah DBD'})
0.33333333 3	({'Tidak Pernah Ada Penderita', 'Makassar', 'Ya Pernah DBD'})
0.30392156 9	({'Tidak Pernah Ada Penderita', 'Tidak Pernah Vaksin', 'Ya Pernah DBD', 'Ya berGejala'})
0.24509803 9	({'Tidak Pernah Ada Penderita', 'Makassar', 'Tidak Pernah Vaksin', 'Ya Pernah DBD'})
0.23529411 8	({'Tidak Pernah Vaksin', 'Ya berGejala', 'Ya Pernah DBD', 'Tidak Pernah Ada Penderita', 'Makassar'})
0.32352941 2	({'Tidak Pernah Ada Penderita', 'Makassar', 'Ya Pernah DBD', 'Ya berGejala'})
0.42156862 7	({'Tidak Pernah Ada Penderita', 'Ya Pernah DBD', 'Keluarga Tidak DBD'})
0.41176470 6	({'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Ya berGejala'})
0.31372549	({'Tidak Pernah Vaksin', 'Ya Pernah DBD', 'Keluarga Tidak DBD'})
0.33333333 3	({'Makassar', 'Ya Pernah DBD', 'Keluarga Tidak DBD'})
0.41176470 6	({'Tidak Pernah Ada Penderita', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Ya berGejala'})
0.31372549	({'Tidak Pernah Ada Penderita', 'Tidak Pernah Vaksin', 'Ya Pernah DBD', 'Keluarga Tidak DBD'})
0.30392156 9	({'Tidak Pernah Vaksin', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Ya berGejala'})
0.24509803 9	({'Makassar', 'Tidak Pernah Vaksin', 'Ya Pernah DBD', 'Keluarga Tidak DBD'})
0.30392156 9	({'Tidak Pernah Vaksin', 'Ya berGejala', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Tidak Pernah Ada Penderita'})
0.24509803 9	({'Tidak Pernah Vaksin', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Tidak Pernah Ada Penderita', 'Makassar'})
0.23529411 8	({'Tidak Pernah Vaksin', 'Ya berGejala', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Makassar'})
0.23529411 8	({'Tidak Pernah Vaksin', 'Ya berGejala', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Tidak Pernah Ada Penderita', 'Makassar'})
0.33333333 3	({'Tidak Pernah Ada Penderita', 'Makassar', 'Ya Pernah DBD', 'Keluarga Tidak DBD'})
0.32352941 2	({'Makassar', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Ya berGejala'})
[illegible]2941 2	({'Ya berGejala', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Tidak Pernah Ada Penderita', 'Makassar'})
[illegible]3823 5	({'Makassar', 'Ya Pernah DBD', 'Ya berGejala'})

0.450980392	({'Tidak Pernah Vaksin', 'Ya Pernah DBD', 'Ya berGejala'})
0.362745098	({'Makassar', 'Tidak Pernah Vaksin', 'Ya Pernah DBD'})
0.352941176	({'Makassar', 'Tidak Pernah Vaksin', 'Ya Pernah DBD', 'Ya berGejala'})
0.588235294	({'Remaja akhir', 'Ya berGejala'})
0.450980392	({'Tidak Pernah Ada Penderita', 'Remaja akhir'})
0.450980392	({'Keluarga Tidak DBD', 'Remaja akhir'})
0.392156863	({'Ya Pernah DBD', 'Remaja akhir'})
0.539215686	({'Makassar', 'Remaja akhir'})
0.509803922	({'Tidak Pernah Vaksin', 'Remaja akhir'})
0.421568627	({'Tidak Pernah Ada Penderita', 'Remaja akhir', 'Ya berGejala'})
0.401960784	({'Tidak Pernah Ada Penderita', 'Makassar', 'Remaja akhir'})
0.352941176	({'Tidak Pernah Ada Penderita', 'Tidak Pernah Vaksin', 'Remaja akhir'})
0.372549020	({'Tidak Pernah Ada Penderita', 'Makassar', 'Remaja akhir', 'Ya berGejala'})
0.323529412	({'Tidak Pernah Ada Penderita', 'Makassar', 'Tidak Pernah Vaksin', 'Remaja akhir'})
0.323529412	({'Tidak Pernah Ada Penderita', 'Tidak Pernah Vaksin', 'Remaja akhir', 'Ya berGejala'})
0.294117647	({'Tidak Pernah Vaksin', 'Ya berGejala', 'Remaja akhir', 'Tidak Pernah Ada Penderita', 'Makassar'})
0.450980392	({'Tidak Pernah Ada Penderita', 'Keluarga Tidak DBD', 'Remaja akhir'})
0.421568627	({'Keluarga Tidak DBD', 'Remaja akhir', 'Ya berGejala'})
0.401960784	({'Makassar', 'Keluarga Tidak DBD', 'Remaja akhir'})
0.352941176	({'Tidak Pernah Vaksin', 'Keluarga Tidak DBD', 'Remaja akhir'})
0.421568627	({'Tidak Pernah Ada Penderita', 'Keluarga Tidak DBD', 'Remaja akhir', 'Ya berGejala'})
0.401960784	({'Tidak Pernah Ada Penderita', 'Makassar', 'Keluarga Tidak DBD', 'Remaja akhir'})
4902	({'Makassar', 'Keluarga Tidak DBD', 'Remaja akhir', 'Ya berGejala'})
4902	({'Ya berGejala', 'Remaja akhir', 'Keluarga Tidak DBD', 'Tidak Pernah Ada Penderita', 'Makassar'})

0.35294117 6	({'Tidak Pernah Ada Penderita', 'Tidak Pernah Vaksin', 'Keluarga Tidak DBD', 'Remaja akhir'})
0.32352941 2	({'Makassar', 'Tidak Pernah Vaksin', 'Keluarga Tidak DBD', 'Remaja akhir'})
0.32352941 2	({'Keluarga Tidak DBD', 'Tidak Pernah Vaksin', 'Remaja akhir', 'Ya berGejala'})
0.32352941 2	({'Tidak Pernah Vaksin', 'Remaja akhir', 'Keluarga Tidak DBD', 'Tidak Pernah Ada Penderita', 'Makassar'})
0.32352941 2	({'Tidak Pernah Vaksin', 'Ya berGejala', 'Remaja akhir', 'Keluarga Tidak DBD', 'Tidak Pernah Ada Penderita'})
0.29411764 7	({'Tidak Pernah Vaksin', 'Ya berGejala', 'Remaja akhir', 'Keluarga Tidak DBD', 'Makassar'})
0.29411764 7	({'Tidak Pernah Vaksin', 'Ya berGejala', 'Remaja akhir', 'Keluarga Tidak DBD', 'Tidak Pernah Ada Penderita', 'Makassar'})
0.37254902	({'Ya Pernah DBD', 'Remaja akhir', 'Ya berGejala'})
0.24509803 9	({'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Remaja akhir'})
0.24509803 9	({'Tidak Pernah Ada Penderita', 'Ya Pernah DBD', 'Remaja akhir'})
0.30392156 9	({'Makassar', 'Ya Pernah DBD', 'Remaja akhir'})
0.28431372 5	({'Tidak Pernah Vaksin', 'Ya Pernah DBD', 'Remaja akhir'})
0.24509803 9	({'Keluarga Tidak DBD', 'Ya Pernah DBD', 'Remaja akhir', 'Ya berGejala'})
0.20588235 3	({'Makassar', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Remaja akhir'})
0.16666666 7	({'Tidak Pernah Vaksin', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Remaja akhir'})
0.20588235 3	({'Ya berGejala', 'Remaja akhir', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Makassar'})
0.16666666 7	({'Tidak Pernah Vaksin', 'Ya berGejala', 'Remaja akhir', 'Ya Pernah DBD', 'Keluarga Tidak DBD'})
0.14705882 4	({'Tidak Pernah Vaksin', 'Remaja akhir', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Makassar'})
0.14705882 4	({'Tidak Pernah Vaksin', 'Ya berGejala', 'Remaja akhir', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Makassar'})
0.24509803 9	({'Tidak Pernah Ada Penderita', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Remaja akhir'})
0.24509803 9	({'Tidak Pernah Ada Penderita', 'Ya Pernah DBD', 'Remaja akhir', 'Ya berGejala'})
0.20588235 3	({'Tidak Pernah Ada Penderita', 'Makassar', 'Ya Pernah DBD', 'Remaja akhir'})
5666 7	({'Tidak Pernah Ada Penderita', 'Tidak Pernah Vaksin', 'Ya Pernah DBD', 'Remaja akhir'})
9803 9	({'Ya berGejala', 'Remaja akhir', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Tidak Pernah Ada Penderita'})

0.20588235 3	({'Ya berGejala', 'Remaja akhir', 'Ya Pernah DBD', 'Tidak Pernah Ada Penderita', 'Makassar'})
0.20588235 3	({'Remaja akhir', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Tidak Pernah Ada Penderita', 'Makassar'})
0.20588235 3	({'Ya berGejala', 'Remaja akhir', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Tidak Pernah Ada Penderita', 'Makassar'})
0.16666666 7	({'Tidak Pernah Vaksin', 'Ya berGejala', 'Remaja akhir', 'Ya Pernah DBD', 'Tidak Pernah Ada Penderita'})
0.16666666 7	({'Tidak Pernah Vaksin', 'Remaja akhir', 'Ya Pernah DBD', 'Keluarga Tidak DBD', 'Tidak Pernah Ada Penderita'})

Lampiran 8 Source code asosiasi

a. Preprocessing

```
import pandas as pd
import numpy as np
import plotly as plt
import mlxtend
!pip install -U mlxtend
df = dataset.drop(['Timestamp',
         'No.Handphone',
         'Tanggal Lahir',
         'Alamat (mohon sertakan nama kelurahan dan
kecamatan)',
         'Jika ya, keadaan penderita tersebut saat ini* :'],
        axis=1)
df.rename(columns={'Umur (dalam tahun dan bulan)': 'Umur',
                   'Tinggi Badan (dalam cm)': 'Tinggi',
                   'Berat badan (dalam kg)':'Berat',
                   'Apakah ciri seseorang terkena DBD adalah
demam tinggi mendadak selama 2-7 hari dan terdapat bintik-
bintik merah pada kulitnya?' : 'Demam 2 -7 hari dan Bintik
Merah',
                   'Apakah anda pernah menderita demam
berdarah?' : 'Pasien Pernah DBD',
                   'Apakah terdapat anggota keluarga yang
pernah menderita demam berdarah selama kurun waktu 6
bulan?':'Ada Keluarga DBD 6 bulan terakhir',
                   'Jika ya, keadaan penderita tersebut saat
ini':'Keadaan saat ini',
                   'Apakah Pasien pernah menerima
vaksin/imunisasi Dengvaxia (CYD-TDV) untuk mencegah penyakit
demam berdarah?': 'Pernah Vaksin',
                   'Apa pasien pernah di opname
sebelumnya?':'Apakah pernhan Opname',
                   'Jika pernah, opnamenya karena penyakit
apa saja?':'Penyakit Opname',
                   'Berapa Luas rumah tempat tinggal
pasien?':'Luas Rumah',
                   'berapa jumlah kamar tidur dirumah?':
'Jumlah Kamar',
                   'berapa jumlah orang yang tinggal di
dalam rumah?' : 'Jumlah Penghuni',
                   'Apakah disekitar rumah ada genangan air
mah yang saling berdekatan?': 'rumah ada genangan air
au lokasi dempet',
```

```
                    'Berapa jumlah kamar tidur dirumah?' :
'Jumlah Kamar',
                    'Berapa jumlah orang yang tinggal dalam
rumah?' : 'Jumlah Penghuni',
                    'Apakah jendela dan ventilasi rumah anda
dipasangkan kawat?':'Jendela/Vetilasi Berkawat'},
inplace=True)
dataset.replace(np.nan, 'Tidak Ada', inplace = True)
df.head()
#Ubah Nilai Berat dan Tinggi Menjadi Numerik
df['Berat'] = df['Berat'].str.extract('(\d+)', expand=False)
df['Tinggi'] = df['Tinggi'].str.extract('(\d+)',
expand=False)
df['Berat'] = pd.to_numeric(df['Berat'], errors='coerce')
df['Tinggi'] = pd.to_numeric(df['Tinggi'], errors='coerce')
#Hitung Nilai IMT berdasarkan Berat dan Tinggi
df = df.assign(IMT = df['Berat'] / (df['Tinggi']/100)**2)
#Ubah Umur jadi float

# Mengambil jumlah tahun dari string umur
df['tahun'] = df['Umur'].str.extract('(\d+)').astype(float)

# Mengambil jumlah bulan dari string umur
df['bulan'] = df['Umur'].str.extract('(\d+)
bulan').astype(float)
df['bulan'] = df['bulan'].replace(np.nan, 0)

# Mengubah tahun dan bulan menjadi tahun dalam desimal
df['umur_desimal'] = df['tahun'] + (df['bulan'] / 12)
#Kategorisasi Umur
df['kategori_umur'] = np.where(df['umur_desimal']<=4.99,
'Balita',
                        np.where(df['umur_desimal']<=11.99,
'Anak-Anak',
                        np.where(df['umur_desimal']<=16.99,
'Remaja awal',
                        np.where(df['umur_desimal']<=25.99,
'Remaja akhir',
                        np.where(df['umur_desimal']<=35.99,
'Dewasa awal',
                        np.where(df['umur_desimal']<=45.99,
'Dewasa akhir',
                        np.where(df['umur_desimal']<=55.99,
    akhir',
                        np.where(df['umur_desimal']<=65.99,
    akhir',
```

```
                        'Manula')))))))))
#Kategorisasi IMT
df['kategori_IMT'] = np.where(df['IMT']<18.5, 'Kurus',
                      np.where(df['IMT']<25, 'Normal',
                      np.where(df['IMT']<30, 'Kelebihan
Berat Badan',
                      'Obesitas')))
#Pre-Processing tahap awal selesai ( Drop kolom yang tidak
diperlukan)
df_pre1 = df.drop(['Umur',
         'Tinggi',
         'Berat',
         'IMT',
         'tahun',
         'bulan',
         'umur_desimal',
         'Kelurahan'],
        axis=1)

#Kelurahan di drop Karena banyak yang salah isi (diisi nilai
kecamatan)
#Perbaiki Kategorisasi di beberapa kolom
df_pre1['Pendapatan Orang Tua'] = df_pre1['Pendapatan Orang
Tua'].replace(['< 2.500.000','2.500.000 -
5.000.000','5.000.000 - 10.000.000','> 10.000.000' ],
                                                        ['
Gaji1', 'Gaji2','Gaji3', 'Gaji4'])
df_pre1['Demam 2 -7 hari dan Bintik Merah'] = df_pre1['Demam
2 -7 hari dan Bintik Merah'].replace(['Ya', 'Tidak'],
                                                        ['
Ya berGejala', 'Tidak berGejala'])
df_pre1['Pasien Pernah DBD'] = df_pre1['Pasien Pernah
DBD'].replace(['Ya','Tidak'],
                                                    ['Ya
Pernah DBD','BelumPernah DBD'])
df_pre1['Ada Keluarga DBD 6 bulan terakhir'] = df_pre1['Ada
Keluarga DBD 6 bulan terakhir'].replace(['Ya','Tidak', 'Ada
tapi cuma 1 minggu','1 minggu', '4 tahun lalu'],
                                                    ['Kel
uarga Terkena DBD','Keluarga Tidak DBD', 'Keluarga Terkena
DBD', 'Keluarga Terkena DBD', 'Keluarga Terkena DBD'])
df_pre1['rumah ada genangan air atau atau lokasi dempet'] =
    ['rumah ada genangan air atau atau lokasi
    ].replace(['Ya', 'Tidak'],
```

```
                    ['Ya ada genangan/lokasi dempet', 'Tidak
lokasi rumah normal'])
df_pre1['Jendela/Vetilasi Berkawat'] =
df_pre1['Jendela/Vetilasi Berkawat'].replace(['Ya','Tidak'],

                        ['Jendela/Ventilasi berkawat',
'Tidak berkawat'])
df_pre1['Pernah Vaksin'] = df_pre1['Pernah
Vaksin'].replace(['Ya','Tidak'],

['Pernah Vaksin', 'Tidak Pernah Vaksin'])
df_pre1['Apakah pernhan Opname'] = df_pre1['Apakah pernhan
Opname'].replace(['Ya','Tidak'],

['pernhan Opname', 'Tidak Pernah Opname'])
df_pre1['Luas Rumah'] = df_pre1['Luas Rumah'].replace(['< 36
m^2','36 - 54 m^2', '54 - 120 m^2', '> 120 m^2'],

['Luas1', 'Luas2', 'Luas3', 'Luas4'])
#Perbaiki Kategorisasi di beberapa kolom
df_pre1['Jumlah Kamar'] = df_pre1['Jumlah
Kamar'].astype(str)
df_pre1['Jumlah Kamar'] = df_pre1['Jumlah Kamar'] + '
Kamar'
df_pre1['Jumlah Penghuni'] = df_pre1['Jumlah
Penghuni'].astype(str)
df_pre1['Jumlah Penghuni'] = df_pre1['Jumlah Penghuni']
+ ' Penghuni'
#Perbaikan Data Jenis Kelamin
df_pre1['Jenis Kelamin'] = df_pre1['Jenis
Kelamin'].replace(['laki-Laki'],

    ['Laki-Laki'])
freqKecamatan = df_pre1['Kecamatan'].value_counts()

print(freqKecamatan)
df_pre1['Kecamatan'] =
df_pre1['Kecamatan'].replace(['Rappoccini','Tamalate ',
'TAMALATE', 'Tamalanrea ', 'Tamalanrea Jaya', 'Somba
    'KEC. MAMAJANG', 'Panakukkang','Bontomarannu '],
                                                    ['R
    ni', 'Tamalate', 'Tamalate','Tamalanrea',
```

```
'Tamalanrea', 'Somba Opu','Mamajang', 'Panakukang',
'Bontomarannu' ])
freqKecamatan = df_pre1['Kecamatan'].value_counts()

print(freqKecamatan)
df_pre1['Pekerjaan Orang tua'] = df_pre1['Pekerjaan
Orang tua'].replace(['Pensiunan ASN','Pensiunan PNS',
'Ibu rumah tangga ', 'IRT', '-'],

                ['Pensiunan', 'Pensiunan', 'Ibu Rumah
Tangga','Ibu Rumah Tangga', 'Tanpa Keterangan'])
import pandas as pd

# create example dataframe

# split values in col1
df_pre1['Jika pernah, riwayat dan opnamenya karena
penyakit apa saja? '] = df_pre1['Jika pernah, riwayat
dan opnamenya karena penyakit apa saja?
'].str.split(',')

# create a new column for each value in col1_split
# for i in range(len(df_pre1['Jika pernah, riwayat dan
opnamenya karena penyakit apa saja?
'].str.len().max())):
#     df_pre1[f'Penyakit{i+1}'] = df_pre1['Jika pernah,
riwayat dan opnamenya karena penyakit apa saja?
'].apply(lambda x: x[i] if len(x) > i else None)
for i in range(df_pre1['Jika pernah, riwayat dan
opnamenya karena penyakit apa saja?
'].str.len().max()):
    df_pre1[f'Penyakit{i+1}'] = df_pre1['Jika pernah,
riwayat dan opnamenya karena penyakit apa saja?
'].apply(lambda x: x[i] if len(x) > i else None)
# drop the temporary col1_split column
df_pre1 = df_pre1.drop('Jika pernah, riwayat dan
opnamenya karena penyakit apa saja? ', axis=1)
```

```
2 = pd.DataFrame(transaction, columns=["items"])
1 to Each Item For Making Countable Table, to be
ɔ perform Group By
2["incident_count"] = 1
ete NaN Items from Dataset
```

```
indexNames = df_pre2[df_pre2['items'] == "nan" ].index
df_pre2.drop(indexNames , inplace=True)
# Making a New Appropriate Pandas DataFrame for
Visualizations
df_table =
df_pre2.groupby("items").sum().sort_values("incident_co
unt", ascending=False).reset_index()
#  Initial Visualizations
df_table.head(20).style.background_gradient(cmap='Blues
')
# importing required module
import plotly.express as px
# to have a same origin
df_table["all"] = "Top 110 items"
# creating tree map using plotly
fig = px.treemap(df_table.head(110), path=['all',
"items"], values='incident_count',
                 color=df_table["incident_count"].head
(110), hover_data=['items'],
                 color_continuous_scale='Blues',
                )
# ploting the treemap
fig.show()
```

b. Algoritma FP-Growth

```
import pandas as pd

# Membaca file csv
df3 = df_pre1

# Mengonversi dataframe menjadi list
data = df3.values.tolist()
!pip install anytree
from anytree import Node, RenderTree

# Definisi dataset
dataset = cleaned_list
```

```
ng support count untuk setiap item
ɔunts = {}
ans in dataset:
r item in trans:
```

```
            if item in item_counts:
                item_counts[item] += 1
            else:
                item_counts[item] = 1

    # Buat root node
    root = Node('Root', count=0)

    # Tambahkan setiap transaksi ke FP-Tree
    for trans in dataset:
        curr_node = root
        for item in sorted(trans, key=lambda x:
    item_counts[x], reverse=True):
            found_child = False
            for child in curr_node.children:
                if child.name == item:
                    child.count += 1
                    curr_node = child
                    found_child = True
                    break
            if not found_child:
                new_node = Node(item, parent=curr_node,
    count=1)
                curr_node = new_node

    # Tampilkan FP-Tree dalam bentuk tree diagram
    for pre, fill, node in RenderTree(root):
        print(f"{pre}{node.name} ({node.count})")
    df_pre1
    import numpy as np
    import pandas as pd
    from mlxtend.preprocessing import TransactionEncoder

    # One-Hot Encoding
    # Transform Every Transaction to Separate List & Gather
    Them into Numpy Array
    transaction = []
    for i in range(df_pre1.shape[0]):
        ans = [str(df_pre1.values[i, j]) for j in
        df_pre1.shape[1]) if str(df_pre1.values[i, j]) !=
        ]
        ansaction.append(trans)
```

```
# creating the numpy array of the transactions
transaction = np.array(transaction)

# initializing the transactionEncoder
te = TransactionEncoder()
te_ary = te.fit(transaction).transform(transaction)
dataset = pd.DataFrame(te_ary, columns=te.columns_)

# dataset after encoded
dataset.head(300)
#Importing Libraries
from mlxtend.frequent_patterns import fpgrowth
#running the fpgrowth algorithm
res1=fpgrowth(dataset,min_support=0.09,
use_colnames=True)
# printing top 10
res1.head(400)

res1.to_csv('Frequent Itemset.csv')
# importing required module
from mlxtend.frequent_patterns import association_rules
# creating asssociation rules
res=association_rules(res1, metric="lift",
min_threshold=1.0)
# printing association rules
res
# importing required module
from mlxtend.frequent_patterns import association_rules
# creating asssociation rules
res=association_rules(res1, metric="confidence",
min_threshold=0.8)
# printing association rules
res
res[ (res['antecedent_len'] >= 2) &
       (res['confidence'] > 0.75) &
       (res['lift'] > 1.2) ]
```

Source code digunakan pada penelitian ini dapat dilihat pada hub.com/yudisumara/Skripsi-Yudi-Asosiasi-dan-Clustering.git

Lampiran 9 Aturan asosiasi berdasarkan kondisi rumah

LINGKUNGAN RUMAH

```
[92] res[ (res['antecedents'] == {'Jendela/Ventilasi berkawat'})]
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:
`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen during thetransform in `pr
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric	antecedent_len
24295	(Jendela/Ventilasi berkawat)	(Ya berGejala)	0.617647	0.911765	0.529412	0.857143	0.940092	-0.033737	0.617647	-0.142857	1
24296	(Jendela/Ventilasi berkawat)	(Makassar)	0.617647	0.784314	0.500000	0.809524	1.032143	0.015571	1.132353	0.081448	1

```
[93] res[ (res['antecedents'] == {'Tidak berkawat'})]
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:
`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen during thetransform in `pr
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric	antecedent_len
19842	(Tidak berkawat)	(Ya berGejala)	0.382353	0.911765	0.382353	1.0	1.096774	0.033737	inf	0.142857	1

```
[94] res[ (res['antecedents'] == {'Tidak lokasi rumah normal'})]
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:
`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen during thetransform in `pr
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric	antecedent_len
9896	(Tidak lokasi rumah normal)	(Ya berGejala)	0.441176	0.911765	0.392157	0.888889	0.974910	-0.010092	0.794118	-0.044025	1
9897	(Tidak lokasi rumah normal)	(Makassar)	0.441176	0.784314	0.401961	0.911111	1.161667	0.055940	2.426471	0.249037	1
9898	(Tidak lokasi rumah normal)	(Tidak Pernah Vaksin)	0.441176	0.774510	0.382353	0.866667	1.118987	0.040657	1.691176	0.190283	1

✓ 0 d selesai pada 14.31