DAFTAR PUSTAKA

Bramer, M. (2020). *Principles of Data Mining*. London: Springer London (Undergraduate Topics in Computer Science). doi:10.1007/978-1-4471-7493-6.

Han, J., Kamber, M., & Pei, J. (2011). *Data Mining Concepts and Techniques, 3rd Edition (The Morgan Kaufmann Series in Data Management Systems)*.

Kementerian Kesehatan Republik Indonesia. (2016). *Buku TB anak 2016*.

Kementerian Kesehatan Republik Indonesia. (2018). *Profil Kesehatan Indonesia 2018.*

Larose, Daniel T., Chantal D. Larose (2015). *Data Mining and Predictive Analytics*. Canada: John Wiley & Sons, Inc., Hoboken, New Jersey.

Latifa, K. T., & Dhita Novika. (2018). *Perlindungan Hukum Terhadap Anak. (Kurnia dan Dhita)*. http://www.kpai.go.id/hukum/undang-undang-uu-

Painho, M. *et al.* (2001). *WebGIS as a teaching tool*. Institute for Statistics and Information Management, New University of Lisbon.

Sharfina, Z., H. B. Santoso. (2016). 'An Indonesian Adaptation of the System UsabilityScale (SUS)'. *International Conference on Advanced Computer Science and Information Systems, ICACSIS 2016, 2017, pp. 145–148*. doi:10.1109/ICACSIS.2016.7872776.

Shofiani, N. (2017). *Segmentasi Supplier Menggunakan Metode K- Means Clustering (Studi Kasus: Ptpn X Pg Meritjan)*. Institut Teknologi Sepuluh Nopember.

Tan, P.-N. *et al.* (2019). *Introduction to data mining, Second edition*. NY NY: Pearson.

Varizi, Daryoush D., *et al*. (2016). 'Exploring user experience and technology acceptance for a fall prevention system: results from a randomized clinical trial and a living lab'. *European Review of Aging and Physical Activity*. doi:10.1186/s11556-016-0165-z

Wakhidah, N. (2010). 'Clustering Menggunakan K-Means Algorithm'. *Jurnal Transformatika, 8(1), p. 33*. doi:10.26623/transformatika.v8i1.45.

LAMPIRAN

Lampiran 1: Kuesioner Penelitian

Kuesioner Tuberkulosis Anak

Yth. Bapak/Ibu Calon Responden Penelitian Di

Tempat

Dr. dr. Bob Wahyudin, Sp.A(K), CIMI dari Klinik Pediatrica Husada dibantu oleh mahasiswa Teknik Informatika Universitas Hasanuddin a.n. Muh. Irzam Kasyfillah dan Muh. Alfarabi Alif P. sedang melaksanakan penelitian terhadap penyakit tuberkulosis pada anak.

Penelitian ini dilatarbelakangi oleh banyaknya kasus tuberkulosis khususnya pada anak di Indonesia. Indonesia berada di urutan ke-3 negara dengan kasus tuberkulosis terbanyak di dunia dengan total kasus sekitar 842 ribu orang. Selain itu, menurut ketua Unit Kerja Koordinasi (UKK) Ikatan Dokter Anak Indonesia (IDAI), diagnosis tuberkulosis pada anak lebih sulit dibanding pada orang dewasa.

Tujuan dari penelitian ini adalah untuk mengetahui variabel (kebiasaan) yang banyak terdapat dalam kasus tuberkulosis anak serta mengetahui hubungan antar variabel tersebut. Manfaat dari penelitian ini adalah dapat membantu penanganan dan pencegahan kasus tuberkulosis anak di masa depan. Sehubungan dengan hal tersebut, peneliti akan melakukan pengumpulan data kepada pasien tuberkulosis anak dengan menggunakan kuesioner.

Adapun kriteria responden yaitu memiliki anak atau kerabat usia 0 - 20 tahun yang sedang atau pernah terkena penyakit tuberkulosis. Kami mengharap partisipasi anda dalam penelitian yang kami lakukan, kami menjamin kerahasiaan dan identitas anda. Informasi yang anda berikan hanya semata-mata digunakan untuk penelitian ini dan tidak digunakan untuk maksud yang lain. Dengan mengisi kuesioner, bapak/ibu telah berperan dan berpartisipasi dalam membantu pencegahan dan penanganan penyakit tuberkulosis anak di masa depan.

Apabila bapak/ibu bersedia menjadi responden, silahkan memilih untuk melanjutkan ke bagian berikutnya. Jika ada pertanyaan, silahkan menghubungi kami melalui email:

- kasyfillahmi17d@student.unhas.ac.id
- putramaa17d@student.unhas.ac.id

*CATATAN: Kami menyediakan pulsa dengan total 200k bagi yang beruntung dan telah mengisi kuesioner.

Terima Kasih

* Wajib

1. No. Handphone

   Silahkan cantumkan Nomor HP yang aktif. Nomor tersebut akan digunakan untuk dikirimkan pulsa.

   .....

Identitas Anak

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 (mohon sertakan nama kelurahan dan kecamatan) *

   .....

   .....

8. Pekerjaan Ayah*

   ☐ PNS

   ☐ TNI/POLRI

   ☐ Karyawan Swasta

   ☐ Wiraswasta

   ☐ Yang lain: .....

9. Pekerjaan Ibu*

   ☐ PNS

   ☐ TNI/POLRI

   ☐ Karyawan Swasta

   ☐ Wiraswasta

   ☐ Yang lain: .....

10. Pendapatan Orang Tua*

   ☐ < 2.500.000

   ☐ 2.500.000 – 5.000.000

   ☐ 5.000.001 – 10.000.000

   ☐ > 10.000.000

Riwayat Kesehatan Anak

Pertanyaan di bawah berkaitan dengan riwayat-riwayat kesehatan dan penyakit yang pernah dialami oleh anak.

11. Apakah anak pernah atau sedang dalam pengobatan tuberkulosis? *

   ☐ Ya

   ☐ Tidak

12. Apakah anak pernah mengalami penyakit diabetes? *
    - ☐ Ya
    - ☐ Tidak

13. Apakah anak telah menerima imunisasi BCG (Bacillus Calmette-Guérin, imunisasi untuk mencegah penyakit TB)? *
    - ☐ Ya
    - ☐ Tidak

14. Apakah anak pernah di opname sebelumnya? *
    - ☐ Ya
    - ☐ Tidak

15. Jika pernah, anak di opname karena penyakit apa saja?

    .....

    .....

16. Apakah anak mengkonsumsi ASI secara eksklusif? (ASI eksklusif adalah pemberian ASI tanpa makanan/minuman (susu formula) tambahan hingga berusia 6 bulan) *
    - ☐ Ya
    - ☐ Tidak

Riwayat Penyakit Orang Serumah

Pertanyaan di bawah berkaitan dengan riwayat-riwayat penyakit yang pernah dialami oleh keluarga dan orang-orang yang tinggal serumah dengan anak.

17. Apakah ada riwayat penyakit tuberkulosis dalam orang serumah? *
    - ☐ Ya
    - ☐ Tidak

18. Apakah ada riwayat penyakit diabetes dalam keluarga (orang tua)? *

☐ Ya

☐ Tidak

19. Apakah ada riwayat penyakit lainnya selain tuberkulosis, diabetes dalam orang serumah? *

☐ Ya

☐ Tidak

20. Jika ada, penyakit apa saja?

.....

.....

21. Apakah ada yang pernah atau sedang mengkonsumsi obat tuberkulosis dalam orang serumah? *

☐ Ya

☐ Tidak

Kondisi Rumah

Pertanyaan dibawah berkaitan dengan situasi dan kondisi rumah tempat tinggal anak.

22. Berapa luas rumah tempat anak tinggal? *

☐ < 36 m^2

☐ 36-54 m^2

☐ 54-120 m^2

☐ > 120 m^2

23. Berapa jumlah kamar tidur dalam rumah? *

.....

24. Berapa jumlah orang yang tinggal dalam satu rumah? *

.....

25. Bagaimana sistem ventilasi di rumah Anda? *

- ☐ Setiap ruangan terdapat ventilasi
- ☐ Hanya ruangan depan/belakang
- ☐ Ada ventilasi namun selalu tertutup
- ☐ Ada ventilasi namun kadang terbuka/tertutup

Lampiran 2: Kuesioner Kepuasan *User*

Untuk melakukan asesmen atau evaluasi terhadap penggunaan website tuberkulosis anak, silahkan mengisi kuesioner berikut ini.. Silakan memutuskan penilaian secara spontan. Jangan berpikir terlalu lama tentang keputusan Anda untuk meyakinkan bahwa Anda memberikan impresi yang orisinal. Silahkan putuskan evaluasi Anda atas setiap item. Pendapat Anda sangat penting. Mohon diperhatikan: tidak ada jawaban salah atau benar.

Keterangan:		
STS : Sangat Tidak Setuju	**TS** : Tidak Setuju	**RG** : Ragu-ragu
ST : Setuju	**SS** : Sangat Setuju	

	STS	**TS**	**RG**	**ST**	**SS**
1. Saya berpikir akan menggunakan sistem ini lagi.					
	1	2	3	4	5
2. Saya merasa sistem ini rumit untuk digunakan.					
	1	2	3	4	5
3. Saya merasa sistem ini mudah digunakan.					
	1	2	3	4	5
4. Saya membutuhkan bantuan dari orang lain atau teknisi dalam menggunakan sistem ini.					
	1	2	3	4	5
5. Saya merasa fitur-fitur sistem ini berjalan dengan semestinya.					
	1	2	3	4	5
6. Saya merasa ada banyak hal yang tidak konsisten (tidak serasi pada sistem ini).					
	1	2	3	4	5
7. Saya merasa orang lain akan memahami cara menggunakan sistem ini dengan cepat.					
	1	2	3	4	5

8. Saya merasa sistem ini membingungkan.

1	2	3	4	5

9. Saya merasa tidak ada hambatan dalam menggunakan sistem ini.

1	2	3	4	5

10. Saya perlu membiasakan diri terlebih dahulu sebelum menggunakan sistem ini.

1	2	3	4	5

Lampiran 3: Source Code

```
import numpy as np
import pandas as pd
from geopy.geocoders import Nominatim
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
# ============= READ ALL REQUIRED DATA ============== #
df_original = pd.read_csv(dataset)
data_status_gizi_laki =
pd.read_csv('./csv/status_gizi_laki.csv',header=1)
data_status_gizi_perempuan =
pd.read_csv('./csv/status_gizi_perempuan.csv',header=1)
data = df_original.copy()
data = data.rename(columns={'Alamat (mohon sertakan nama kelurahan
dan kecamatan)': 'Alamat lengkap',
                           'Apakah anak pernah atau sedang dalam
pengobatan tuberkulosis?': 'pernah/sedang TB',
                           'Apakah anak pernah mengalami penyakit
diabetes?': 'riwayat diabetes anak',
                           'Apakah anak telah menerima imunisasi
BCG (Bacillus Calmette-Guérin, imunisasi untuk mencegah penyakit
TB)?': 'riwayat vaksin BCG',
                           'Apakah anak pernah di opname
sebelumnya?': 'riwayat opname',
                           'Jika pernah, anak diopname karena
penyakit apa saja?': 'daftar penyakit opname',
                           'Apakah anak mengkonsumsi ASI secara
eksklusif? (ASI Eksklusif adalah pemberian ASI tanpa
makanan/minuman (susu formula) tambahan hingga berusia 6 bulan)':
'ASI eksklusif',
                           'Apakah ada riwayat penyakit
tuberkulosis dalam orang serumah?': 'riwayat TB orang serumah',
                           'Apakah ada riwayat penyakit diabetes
dalam keluarga (orang tua)?': 'riwayat diabetes keluarga',
                           'Apakah ada riwayat penyakit lainnya
selain tuberkulosis, diabetes dalam orang  serumah?': 'riwayat
penyakit lain orang serumah',
                           'Jika ada, penyakit apa saja?':
'daftar penyakit lain orang serumah',
                           'Berapa luas rumah tempat anak
tinggal?': 'luas rumah',
                           'Berapa jumlah kamar tidur dalam
rumah?': 'jumlah kamar tidur',
                           'Berapa jumlah orang yang tinggal
dalam satu rumah?': 'jumlah orang dalam rumah',
                           'Bagaimana sistem ventilasi di rumah
Anda? ': 'sistem ventilasi'})

# ====  PRE PROCESSING  ================= #
# DATA CLEANING, TRANSFORMATION, SELECTION
# HANDLE NAN VALUES
data['daftar penyakit opname'] = data['daftar penyakit
```

```
opname'].replace(np.nan, 'Tidak Pernah')
data['daftar penyakit lain orang serumah'] = data['daftar penyakit
lain orang serumah'].replace(np.nan, 'Tidak Ada')
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)
# ADD NEW COLUMN FOR 'TAHUN' AND 'BULAN'
age = data['Umur'].str.split(r'(\d+)', expand=True)
age = age.replace([None], '0')
data['Tahun'] = age[1]
data['Bulan'] = age[3]
data = data.astype({"Tahun": float, "Bulan": float})
# CALCULATE THE IMT FOR STATUS GIZI
data['IMT'] = np.round(
    data['Berat badan (dalam kg)'] / (data['Tinggi badan (dalam
cm)'] * data['Tinggi badan (dalam cm)'] / 10000), 2)
# == DATA DAFTAR OPNAME DAN DAFTAR PENYAKIT LAIN ORANG SERUMAH ==#
daftar_opname = []
for i in data['daftar penyakit opname']:
    to_list = re.split(', ', i.upper())
    daftar_opname.append(to_list)
data['daftar opname'] = daftar_opname
daftar_penyakit = []
for i in data['daftar penyakit lain orang serumah']:
    to_list = re.split(', ', i.upper())
    daftar_penyakit.append(to_list)
data['daftar penyakit lain'] = daftar_penyakit
# ===== DETERMINE THE STATUS GIZI OF EACH 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
status_gizi = []
i = 0
for imt in data['IMT']:
    z_score = np.round(
        count_z_score(data['Jenis Kelamin'][i],
int(data['Tahun'][i]), int(data['Bulan'][i]), data['IMT'][i]), 2)

    if (int(data['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"
    status_gizi.append(status)
    i += 1
data['Status Gizi'] = status_gizi
data = data.drop(['Timestamp', 'Umur', 'Tanggal Lahir', 'Alamat
lengkap',
                  'Apakah ada yang pernah atau sedang mengkonsumsi
obat tuberkulosis dalam orang serumah?', 'Bulan',
                  'IMT'], axis=1)
data_positive = data.loc[data['pernah/sedang TB'] == 'Ya'].copy()
# == EXPAND COLUMN FOR EACH DAFTAR OPNAME DAN PENYAKIT LAIN ==== #
opname = data_positive["daftar opname"].explode().unique()
penyakit_lain = data_positive["daftar penyakit
lain"].explode().unique()
for op in opname:
    if op == 'TIDAK PERNAH':
        continue
    new_col = []
    for row in data_positive['daftar opname']:
        if op in row:
            new_col.append('Ya')
        else:
            new_col.append('Tidak')
    data_positive[op + ' (opname)'] = new_col
for peny in penyakit_lain:
```

```
    if peny == 'TIDAK ADA':
        continue
    new_col = []
    for row in data_positive['daftar penyakit lain']:
        if peny in row:
            new_col.append('Ya')
        else:
            new_col.append('Tidak')
    data_positive[peny + ' (org serumah)'] = new_col
data_positive['Tahun'] = data_positive['Tahun'].astype('float')
data_positive = data_positive.drop(['daftar opname', 'riwayat
opname', 'pernah/sedang TB', 'daftar penyakit lain',
                                    'daftar penyakit lain orang
serumah', 'daftar penyakit opname',
                                    'riwayat penyakit lain orang
serumah'], axis=1)
data_positive =
data_positive.loc[data_positive['Kab/Kota'].isin(['Makassar',
'Gowa'])]
data_positive['Total'] = data_positive['Tahun']
l = data_positive.columns
for i in l:
    res = data_positive[i].value_counts(normalize=True) * 100
    if res.iloc[0] >= 96:
        del data_positive[i]
# ===== CREATE DICTIONARY FOR OBJECT AND NUMERICAL COLUMNS ======#
dict_cat = {}
dict_num = {}
for cat in data_positive.select_dtypes(['object', 'category']):
    if (cat == 'Kecamatan'):
        continue
    dict_cat[cat] = lambda x: x.value_counts().index[0]
for num in data_positive.select_dtypes(['int64', 'float64']):
    if (num == 'Total'):
        continue
    dict_num[num] = ['mean']
# ============ CREATE TABLE FOR EACH KECAMATAN ============#
data_perkecamatan = data_positive.groupby('Kecamatan').agg({
    'Total': 'count',
    **dict_num,
    **dict_cat
})
riwayat_col = ['riwayat diabetes anak', 'riwayat vaksin BCG', 'ASI
eksklusif', 'riwayat TB orang serumah',
               'riwayat diabetes keluarga', 'TIPOID (opname)',
'DEMAM (opname)', 'SESAK (opname)',
               'HIPERTENSI (org serumah)', 'ASMA (org serumah)',
'ASAM URAT (org serumah)']
for k in riwayat_col:
    new_col = []
    for i in data_perkecamatan.index:
        total = 0
        for j in data_positive.index:
            if (i == data_positive['Kecamatan'][j] and
data_positive[k][j] == 'Ya'):
                total = total + 1
```

```
        new_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_positive.index:
            if (i == data_positive['Kecamatan'][j] and
data_positive[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_positive.index:
            if (i == data_positive['Kecamatan'][j] and (
                    data_positive[k][j] == 'Berisiko gizi lebih'
or data_positive[k][j] == 'Gizi lebih' or
                    data_positive[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_positive.index:
            if (i == data_positive['Kecamatan'][j] and (
                    data_positive[k][j] == 'Gizi kurang' or
data_positive[k][j] == 'Gizi buruk')):
                total = total + 1
        new_col.append(total)
    data_perkecamatan['Status Gizi kurang'] = new_col
riwayat_col = ['sistem ventilasi']
for k in riwayat_col:
    new_col = []
    for i in data_perkecamatan.index:
        total = 0
        for j in data_positive.index:
            if (i == data_positive['Kecamatan'][j] and
data_positive[k][j] == 'Setiap ruangan terdapat ventilasi'):
                total = total + 1
        new_col.append(total)
    data_perkecamatan['Jumlah rumah dgn ventilasi di setiap
ruangan'] = new_col
riwayat_col = ['Pendapatan Orang Tua']
for k in riwayat_col:
    new_col = []
    for i in data_perkecamatan.index:
        total = 0
        for j in data_positive.index:
            if (i == data_positive['Kecamatan'][j] and
data_positive[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_positive.index:
            if (i == data_positive['Kecamatan'][j] and
data_positive[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_positive.index:
            if (i == data_positive['Kecamatan'][j] and
data_positive[k][j] == '5.000.001 - 10.000.000'):
                total = total + 1
        new_col.append(total)
    data_perkecamatan['Pendapatan ( 5.000.001 - 10.000.000 )'] =
new_col
for k in riwayat_col:
    new_col = []
    for i in data_perkecamatan.index:
        total = 0
        for j in data_positive.index:
            if (i == data_positive['Kecamatan'][j] and
data_positive[k][j] == '> 10.000.000'):
                total = total + 1
        new_col.append(total)
    data_perkecamatan['Pendapatan ( > 10.000.000 )'] = 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_positive.index:
            if (i == data_positive['Kecamatan'][j] and
data_positive[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_positive.index:
            if (i == data_positive['Kecamatan'][j] and
data_positive[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_positive.index:
            if (i == data_positive['Kecamatan'][j] and
data_positive[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:
        total = 0
        for j in data_positive.index:
            if (i == data_positive['Kecamatan'][j] and
data_positive[k][j] == '> 120 m^2'):
                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 TB'})
data = data_perkecamatan.copy()
data = data.rename(
    columns={'Status Gizi lebih': 'Persentase anak gizi lebih',
'Status Gizi kurang': 'Persentase anak gizi kurang',
             'Status Gizi baik': 'Persentase anak gizi baik',
'Total': 'Jumlah TB',
             'Tinggi badan (dalam cm)': 'Tinggi anak (mean)',
'Berat badan (dalam kg)': 'Berat anak (mean)',
             'jumlah kamar tidur': 'Jumlah kamar (mean)',
             'jumlah orang dalam rumah': 'Jumlah orang di rumah
(mean)', 'Tahun': 'Usia (mean)',
             'riwayat diabetes anak': 'Persentase anak menderita
diabetes',
             'riwayat vaksin BCG': 'Persentase anak telah BCG',
             'ASI eksklusif': 'Persentase anak dengan ASI
eksklusif',
             'riwayat TB orang serumah': 'Persentase kasus dengan
riwayat TB serumah',
             'riwayat diabetes keluarga': 'Persentase kasus dengan
keluarga menderita diabetes'})
data.index = data.index + ' (' + data['Kab/Kota'] + ')'
# =========== CHANGE TO PERCENTAGE FORMAT =========== #
data['Persentase anak gizi baik'] = (data['Persentase anak gizi
baik'] / data['Jumlah TB']) * 100
data['Persentase anak gizi lebih'] = (data['Persentase anak gizi
lebih'] / data['Jumlah TB']) * 100
data['Persentase anak gizi kurang'] = (data['Persentase anak gizi
kurang'] / data['Jumlah TB']) * 100

data['Pendapatan ( < 2.500.000 )'] = (data['Pendapatan ( <
```

```
2.500.000 )'] / data['Jumlah TB']) * 100
data['Pendapatan ( 2.500.000 - 5.000.000 )'] = (data['Pendapatan (
2.500.000 - 5.000.000 )'] / data[
    'Jumlah TB']) * 100
data['Pendapatan ( 5.000.001 - 10.000.000 )'] = (data['Pendapatan
( 5.000.001 - 10.000.000 )'] / data[
    'Jumlah TB']) * 100
data['Pendapatan ( > 10.000.000 )'] = (data['Pendapatan ( >
10.000.000 )'] / data['Jumlah TB']) * 100

data['Luas rumah ( < 36 m^2 )'] = (data['Luas rumah ( < 36 m^2 )']
/ data['Jumlah TB']) * 100
data['Luas rumah ( 36 - 54 m^2 )'] = (data['Luas rumah ( 36 - 54
m^2 )'] / data['Jumlah TB']) * 100
data['Luas rumah ( 54 - 120 m^2 )'] = (data['Luas rumah ( 54 - 120
m^2 )'] / data['Jumlah TB']) * 100
data['Luas rumah ( > 120 m^2 )'] = (data['Luas rumah ( > 120 m^2
)'] / data['Jumlah TB']) * 100

data['Persentase anak menderita diabetes'] = (data['Persentase
anak menderita diabetes'] / data['Jumlah TB']) * 100
data['Persentase anak telah BCG'] = (data['Persentase anak telah
BCG'] / data['Jumlah TB']) * 100
data['Persentase anak dengan ASI eksklusif'] = (data['Persentase
anak dengan ASI eksklusif'] / data[
    'Jumlah TB']) * 100
data['Persentase kasus dengan riwayat TB serumah'] =
(data['Persentase kasus dengan riwayat TB serumah'] / data[
    'Jumlah TB']) * 100
data['Persentase kasus dengan keluarga menderita diabetes'] =
(data[

'Persentase kasus dengan keluarga menderita diabetes'] /

data['Jumlah TB']) * 100
# ===== CLUSTER 1 ======= #
cluster1 = data[['Jumlah TB', 'Usia (mean)']]
# ============= 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])
# Actual Clustering
kmeans = KMeans(n_clusters=3, 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['Segment'] = cluster1['labels'].map({0: 'First', 1:
'Second', 2: 'Third'})
# Order the cluster
cluster1['Segment'] = cluster1['Segment'].astype('category')
cluster1['Segment'] =
cluster1['Segment'].cat.reorder_categories(['First', 'Second',
```

```
'Third'])
# === 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_percluster1 = cluster1.groupby('Segment').agg({
    'Total': 'count',
    **dict_num,
    **dict_cat
}).T
# ======  COUNT LOGITUDE AND LATITUDE FOR ALL KECAMATAN =======#
alamat = []
for i in cluster1.index:
    splitted = i.split(' (')
    alamat.append(splitted[0] + ' ' + splitted[1][:-1])
geolocator = Nominatim(user_agent="tes")
coord = []
for i in range(0, len(alamat)):
    loc = alamat[i]
    location = geolocator.geocode(loc, timeout=None)
    if location != None:
        m = 0.0025
        coord.append([[location.latitude, location.longitude],
[location.latitude + m, location.longitude + m],
                      [location.latitude + m, location.longitude -
m],
                      [location.latitude - m, location.longitude +
m],
                      [location.latitude - m, location.longitude -
m]])
        print(loc)
coord1 = [i[0] for i in coord]
coord2 = [i[1] for i in coord]
coord3 = [i[2] for i in coord]
coord4 = [i[3] for i in coord]
coord5 = [i[4] for i in coord]
cluster1['coord'] = coord1
cluster1 = cluster1.T
data_percluster1.to_json(r'./result/cluster1_result.json')
cluster1.to_json(r'./result/cluster1_df.json')
# ====== CLUSTER 2 ======== #
cluster2 = data[['Persentase anak gizi baik', 'Persentase anak
gizi lebih', 'Persentase anak gizi kurang']]
# ============= DATA TRANSFORMATION ============== #
scaler = StandardScaler()
Num_features = cluster2.select_dtypes(include=['int64',
'float64']).columns
scaler.fit(cluster2[Num_features])
```

```
cluster2[Num_features] = scaler.transform(cluster2[Num_features])
# 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['Segment'] = cluster2['labels'].map({0: 'First', 1:
'Second', 2: 'Third'})
# Order the cluster
cluster2['Segment'] = cluster2['Segment'].astype('category')
cluster2['Segment'] = cluster2['Segment'].
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']
# ============ CREATE TABLE FOR EACH CLUSTER ============#
cluster2.rename(columns={'labels': 'Total'}, inplace=True)
data_percluster2 = cluster2.groupby('Segment').agg({
    'Total': 'count',
    **dict_num,
    **dict_cat
}).T
cluster2['coord'] = coord2
cluster2 = cluster2.T
data_percluster2.to_json(r'./result/cluster2_result.json')
cluster2.to_json(r'./result/cluster2_df.json')
# ==== CLUSTER 3 ==== #
cluster3 = data[
    ['Pendapatan ( < 2.500.000 )', 'Pendapatan ( 2.500.000 -
5.000.000 )', 'Pendapatan ( 5.000.001 - 10.000.000 )',
     'Pendapatan ( > 10.000.000 )']]
# ============= DATA TRANSFORMATION ============== #
scaler = StandardScaler()
Num_features = cluster3.select_dtypes(include=['int64',
'float64']).columns
scaler.fit(cluster3[Num_features])
cluster3[Num_features] = scaler.transform(cluster3[Num_features])
# 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['Segment'] = cluster3['labels'].map({0: 'First', 1:
'Second', 2: 'Third', 3: 'Fourth', 4: 'Fifth'})
# Order the cluster
cluster3['Segment'] = cluster3['Segment'].astype('category')
cluster3['Segment'] =
```

```
cluster3['Segment'].cat.reorder_categories(['First', 'Second',
'Third', 'Fourth', 'Fifth'])
# ======= INVERSE TRANSFORMATION FOR NUMERIC DATA ==========#
cluster3[Num_features] =
scaler.inverse_transform(cluster3[Num_features])
# == 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']
# ============ CREATE TABLE FOR EACH CLUSTER ============#
cluster3.rename(columns={'labels': 'Total'}, inplace=True)
data_percluster3 = cluster3.groupby('Segment').agg({
    'Total': 'count',
    **dict_num,
    **dict_cat
}).T
cluster3['coord'] = coord3
cluster3 = cluster3.T
data_percluster3.to_json(r'./result/cluster3_result.json')
cluster3.to_json(r'./result/cluster3_df.json')
# ==== CLUSTER 4 ==== #
cluster4 = data[['Luas rumah ( < 36 m^2 )', 'Luas rumah ( 36 - 54
m^2 )', 'Luas rumah ( 54 - 120 m^2 )',
                 'Luas rumah ( > 120 m^2 )']]
# ============= DATA TRANSFORMATION ============== #
scaler = StandardScaler()
Num_features = cluster4.select_dtypes(include=['int64',
'float64']).columns
scaler.fit(cluster4[Num_features])
cluster4[Num_features] = scaler.transform(cluster4[Num_features])
# Actual Clustering
kmeans = KMeans(n_clusters=5, 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['Segment'] = cluster4['labels'].map({0: 'First', 1:
'Second', 2: 'Third', 3: 'Fourth', 4: 'Fifth'})
# Order the cluster
cluster4['Segment'] = cluster4['Segment'].astype('category')
cluster4['Segment'] =
cluster4['Segment'].cat.reorder_categories(['First', 'Second',
'Third', 'Fourth', 'Fifth'])
# ===== INVERSE TRANSFORMATION FOR NUMERIC DATA ==========#
cluster4[Num_features] =
scaler.inverse_transform(cluster4[Num_features])
# ==== CREATE DICTIONARY FOR OBJECT AND NUMERICAL COLUMNS ======#
dict_cat = {}
dict_num = {}
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']
```

```
# ============ CREATE TABLE FOR EACH CLUSTER ============#
cluster4.rename(columns={'labels': 'Total'}, inplace=True)
data_percluster4 = cluster4.groupby('Segment').agg({
    'Total': 'count',
    **dict_num,
    **dict_cat
}).T
cluster4['coord'] = coord4
cluster4 = cluster4.T
data_percluster4.to_json(r'./result/cluster4_result.json')
cluster4.to_json(r'./result/cluster4_df.json')
# ==== CLUSTER 5 ==== #
cluster5 = data[['Persentase anak telah BCG', 'Persentase anak
dengan ASI eksklusif',
                 'Persentase kasus dengan riwayat TB serumah',
'Persentase anak menderita diabetes',
                 'Persentase kasus dengan keluarga menderita
diabetes']]
# ============= DATA TRANSFORMATION ============== #
scaler = StandardScaler()
Num_features = cluster5.select_dtypes(include=['int64',
'float64']).columns
scaler.fit(cluster5[Num_features])
cluster5[Num_features] = scaler.transform(cluster5[Num_features])
# Actual Clustering
kmeans = KMeans(n_clusters=4, random_state=9)
preds = kmeans.fit_predict(cluster5)
pd.Series(kmeans.labels_).value_counts()
cluster5['labels'] = kmeans.labels_
cluster5['Segment'] = cluster5['labels'].map({0: 'First', 1:
'Second', 2: 'Third', 3: 'Fourth'})
cluster5['Segment'] = cluster5['Segment'].astype('category')
cluster5['Segment'] =
cluster5['Segment'].cat.reorder_categories(['First', 'Second',
'Third', 'Fourth'])
# ========= 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']
# ============ CREATE TABLE FOR EACH CLUSTER ============#
cluster5.rename(columns={'labels': 'Total'}, inplace=True)
data_percluster5 = cluster5.groupby('Segment').agg({
    'Total': 'count',
    **dict_num,
    **dict_cat
}).T
cluster5['coord'] = coord5
cluster5 = cluster5.T
data_percluster5.to_json(r'./result/cluster5_result.json')
cluster5.to_json(r'./result/cluster5_df.json')
```