Analisis Respons dan Waktu Pengiriman Polisi: Studi Kasus Kota Detroit

Kartini Lovian Simbolon[1], Siti Nur Aarifah[2], Elisabeth Claudia Simanjuntak[3], Pardi Octaviando[4], Ahmad Rizqi[5]
Program Studi Sains Data, Fakultas Sains, Institut Teknologi Sumatera
Lampung Selatan, Indonesia
Email: Kartini.122450003@student.itera.ac.id siti.122450006@student.itera.ac.id elisabeth.122450123@student.itera.ac.id pardi.122450132@student.itera.ac.id ahmad.122450138@student.itera.ac.id

1. Latar Belakang

Kejahatan seperti perampokan, pencurian kendaraan, dan kekerasan menjadi perhatian utama di Detroit. Masyarakat frustrasi karena polisi tidak segera menangani kasus-kasus ini. Lamanya respons disebabkan oleh kekurangan karyawan, sumber daya yang terbatas, dan kemacetan lalu lintas. Hal ini mengganggu kualitas hidup masyarakat, menyebabkan ketakutan, kecemasan, dan rasa tidak aman, dan menghambat pertumbuhan ekonomi dan pariwisata.

Studi kasus ini bertujuan untuk mengevaluasi tanggapan polisi dan lamanya pengiriman di Detroit. Tujuannya adalah untuk menemukan hal-hal yang menyebabkan respons yang lambat dan menawarkan solusi untuk meningkatkannya. Hasil penelitian yang akurat diperoleh melalui penggunaan berbagai teknik seperti analisis data, wawancara, dan survei. Studi menunjukkan bahwa penegakan hukum Detroit lambat. Ini disebabkan oleh hal-hal seperti kemacetan lalu lintas, kekurangan karyawan, dan keterbatasan sumber daya. Untuk meningkatkan waktu respons, studi ini menyarankan penggunaan teknologi baru, peningkatan dana untuk departemen kepolisian, dan perekrutan petugas baru.

Untuk mengatasi kejahatan dan respons lambat di Detroit, para pemangku kepentingan harus bekerja sama. Diharapkan Detroit akan menjadi tempat yang lebih aman dan nyaman bagi semua orang dengan meningkatkan waktu respons polisi.

2. Metode

2.1 *Map*

Pengertian dari tipe data *map* dalam dart adalah objek koleksi yang setiap elemennya berupa pasangan kunci (*key*) dan nilai (*value*). Terdapa asosiasi antara kunci dan nilai pada setiap elemen yang terdapat didalam elemen map. Dalam satu objek map, kunci harus bersifat unik,namun tidak dengan nilai. Hal ini berarti bahwa satu nilai yang sama bisa saja muncul pada beberapa elemen

map. Beberapa bahasa pemrograman lain menamai map dengan istilah yang berbeda. Sebagai contoh, *python* menyebut map dengan istilah *dictionary*, Ruby menyebutnya dengan hash, dan PHP menyebutnya dengan array asosiatif. Dalam dart, map dinyatakan dengan tipe *map*. Objek dari kelas *Map* dibuat dengan menggunakan tanda { } [1].

2.2 *Filter*

Filter adalah teknik yang digunakan untuk memilih elemen-elemen tertentu dari suatu objek berdasarkan kriteria atau kondisi tertentu. *Filter* dapat diterapkan pada berbagai jenis objek seperti *list, tuple, set,* dan *dictionary* menggunakan fungsi-fungsi *built-in* atau menggunakan ekspresi *lambda*. Untuk melakukan *filtering* pada *dictionary*, kita dapat menggunakan list comprehension dengan menentukan kondisi pada pasangan kunci dan nilai *(key-value pairs)* [2].

2.3 *Reduce*

Fungsi *reduce* dapat diimport dari modul *functools*. Fungsi ini, seperti *fungsi map()*, menerima dua argumen yakni sebuah fungsi dan *iterable.* Jika fungsi map menghasilkan *iterable* baru, fungsi *reduce* menghasilkan suatu nilai kumulatif dari operasi fungsi masukan terhadap nilai pada *iterable* masukan [4].

2.4 *Pandas*

Pandas adalah sebuah library pada python yang berlisensi BSD dan *open source* yang digunakan untuk analisis data. Pandas memiliki struktur dasar yaitu *DataFrame*. Pandas memiliki dua tipe struktur data yaitu *Series* dan *DataFrame*. *Series* adalah satu dimensi struktur data *array*. *DataFrame* adalah dua dimensi struktur data atau bisa gabungan dari beberapa *Series* [5].

2.5 JSON

JavaScript Object Notation adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemrograman *JavaScript*, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, *Java, JavaScript, Perl, Python* dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran data[10]. JSON terbuat dari dua struktur:

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table),* daftar berkunci *(keyed list)*, atau associative *array*.
2. Daftar nilai terurutkan (*an ordered list of values*) [3].

3. Hasil dan Pembahasan

3.1 Memasukkan Dataset dengan Pandas

```python
import pandas as pd

# Path ke file Excel
jalur_excel = '/content/911_Calls_for_Service_(Last_30_Days).csv'

# Baca file CSV (bukan Excel) menggunakan pd.read_csv
data_excel = pd.read_csv(jalur_excel)

# Tampilkan lima baris pertama data
print(data_excel.head())
```

```
           X          Y   incident_id agency              incident_address  \
0 -82.993989  42.410293  202306901627   DPD     Glenfield Ave & Gratiot Ave
1 -83.052792  42.415266  202306901629   DPD           Luce St & Fenelon St
2 -83.072103  42.368421  202306901630   DPD   W Baltimore St & Woodward Ave
3 -83.226541  42.429892  202306901632   DPD          Avon Ave & W 7 Mile Rd
4 -83.240073  42.409465  202306901633   DPD     Heyden St & Grand River Ave

   zip_code priority callcode             calldescription  category  ...  \
0     48213        3   W4807   START OF SHIFT INFORMATION  STRTSHFT  ...
1     48212        3   SA               SPECIAL ATTENTION  SPCL ATT  ...
2     48202        3   SA               SPECIAL ATTENTION  SPCL ATT  ...
3     48219        4   372040                UDAA REPORT  UDAAREPT  ...
```

Gambar 3.1.1. Memasukkan Dataset

Untuk mengolah dataset yang berbentuk csv, gunakan library *pandas* untuk membaca dataset yang berbentuk csv lalu dataset dapat ditampilkan. Pada kode ini dataset diberikan nama sebagai jalur_excel jadi setiap pemanggilan jalur_excel maka data akan terpanggil

3.2 Membaca Dataset

```python
import csv
from functools import reduce

# Fungsi untuk membaca file CSV dan memfilter baris yang lengkap
def baca_csv_filter(nama_file):
    with open(nama_file, 'r') as file:
        pembaca = csv.DictReader(file)
        data = filter(lambda x: x['zip_code'] != '' and x['neighborhood'] != '', pembaca)
        return list(data)
```

Gambar 3.2.1. Membaca Dataset

Membuat fungsi baru bernama baca_cvs_filter yang berisikan parameter nama_file. fungsi ini berguna sebagai pemfilter baris baris yang tidak lengkap. Cara kerja fungsi ini dengan membuka file dataset yang dimiliki, lalu data yang dibuka akan di filter dan dikembalikan lagi menjadi *list*

3.3 Analisis Rata-rata Waktu Respons, Waktu Pengiriman, dan Total Waktu Pengiriman Barang di Detroit

```python
# Menghitung total waktu respons rata-rata, waktu pengiriman rata-rata, dan total waktu rata-rata
def hitung_rerata(data):
    total_waktu_respons = 0
    total_waktu_pengiriman = 0
    total_waktu = 0
    jumlah_rekaman = 0

    for baris in data:
        try:
            total_waktu_respons += float(baris['totalresponsetime'])
            total_waktu_pengiriman += float(baris['traveltime'])
            total_waktu += float(baris['totaltime'])
            jumlah_rekaman += 1
        except ValueError:
            pass  # Tidak lakukan apa-apa jika terjadi kesalahan dalam konversi nilai ke float

    if jumlah_rekaman == 0:
        return {
            'neighborhood': 'DETROIT',  # Tambahkan kunci neighborhood dengan nilai 'DETROIT'
            'rerata_waktu_respons': 0,
            'rerata_waktu_pengiriman': 0,
            'rerata_total_waktu': 0
        }
```

Gambar 3.3.1. Rata-rata Waktu *Respons*, Waktu Pengiriman, dan Total Waktu Pengiriman Barang di Detroit

```python
    return {
        'neighborhood': 'DETROIT',  # Tambahkan kunci neighborhood dengan nilai 'DETROIT'
        'rerata_waktu_respons': total_waktu_respons / jumlah_rekaman,
        'rerata_waktu_pengiriman': total_waktu_pengiriman / jumlah_rekaman,
        'rerata_total_waktu': total_waktu / jumlah_rekaman
    }



# Path file CSV
path_csv = "/content/911_Calls_for_Service_(Last_30_Days).csv"

# Baca file CSV dan filter data
data_terfilter = baca_csv_filter(path_csv)

# Hitung rata-rata waktu respons, waktu pengiriman, dan total waktu
hasil_part1 = hitung_rerata(data_terfilter)
print(hasil_part1)
```

```
{'neighborhood': 'DETROIT', 'rerata_waktu_respons': 8.336226605030683, 'rerata_waktu_pengiriman': 2.333406625428829
```

Gambar 3.3.2. Rata-rata Waktu *Respons*, Waktu Pengiriman, dan Total Waktu Pengiriman Barang di Detroit

Fungsi hitung_rerata(data) merupakan bagian kode yang bertugas menghitung rata-rata waktu *respons*, waktu pengiriman, dan total waktu dari data yang sudah melewati proses filtrasi. Melalui iterasi, fungsi tersebut mengambil nilai-nilai waktu dari setiap baris data dan mengakumulasikannya untuk setiap kategori yang bersangkutan. Selain itu, fungsi ini juga menghitung jumlah rekaman untuk digunakan dalam perhitungan rata-rata berikutnya. Jika data tidak ditemukan, fungsi akan mengembalikan sebuah dictionary dengan nilai rata-rata waktu diatur ke 0, sementara 'neighborhood' akan ditambahkan dengan nilai 'DETROIT' untuk menandakan bahwa data tersebut mewakili seluruh kawasan Detroit. Pada bagian yang mencakup path_csv, kode membaca dan memproses data dari file CSV yang telah disediakan, menggunakan fungsi baca_csv_filter(nama_file) untuk membaca dan memfilter baris-baris yang lengkap. Data

yang telah difilter kemudian disimpan dalam variabel data_terfilter. Selanjutnya, data tersebut diproses untuk menghitung rata-rata waktu dengan memanggil fungsi hitung_rerata(data), dan hasil perhitungannya disimpan dalam variabel hasil_part1. Terakhir, hasil perhitungan tersebut dicetak menggunakan perintah print(hasil_part1), yang menampilkan rata-rata waktu *respons*, waktu pengiriman, dan total waktu untuk seluruh wilayah Detroit dalam 30 hari terakhir. Dengan demikian, potongan kode tersebut menciptakan suatu alur logika yang utuh, mulai dari pembacaan dan pemrosesan data dari file CSV hingga pencetakan hasil perhitungan rata-rata waktu.

```
# Mengelompokkan data berdasarkan daerah (neighborhood)
def kelompokkan_berdasarkan_daerah(data):
    data_terkelompok = {}
    for baris in data:
        daerah = baris['neighborhood']
        if daerah not in data_terkelompok:
            data_terkelompok[daerah] = []
        data_terkelompok[daerah].append(baris)
    return data_terkelompok

# Menghitung rata-rata waktu respons, waktu perjalanan, dan total waktu untuk setiap daerah
def hitung_rata_rata_daerah(data):
    rata_rata = {}
    for daerah, catatan in data.items():
        if not catatan:
            rata_rata[daerah] = {
                'rata_rata_waktu_respons': 0,
                'rata_rata_waktu_perjalanan': 0,
                'rata_rata_waktu_total': 0,
                'populasi': 0
            }
```

Gambar 3.3.3. Rata-rata Waktu *Respons*, Waktu Pengiriman, dan Total Waktu Pengiriman Barang di *neighborhood*.

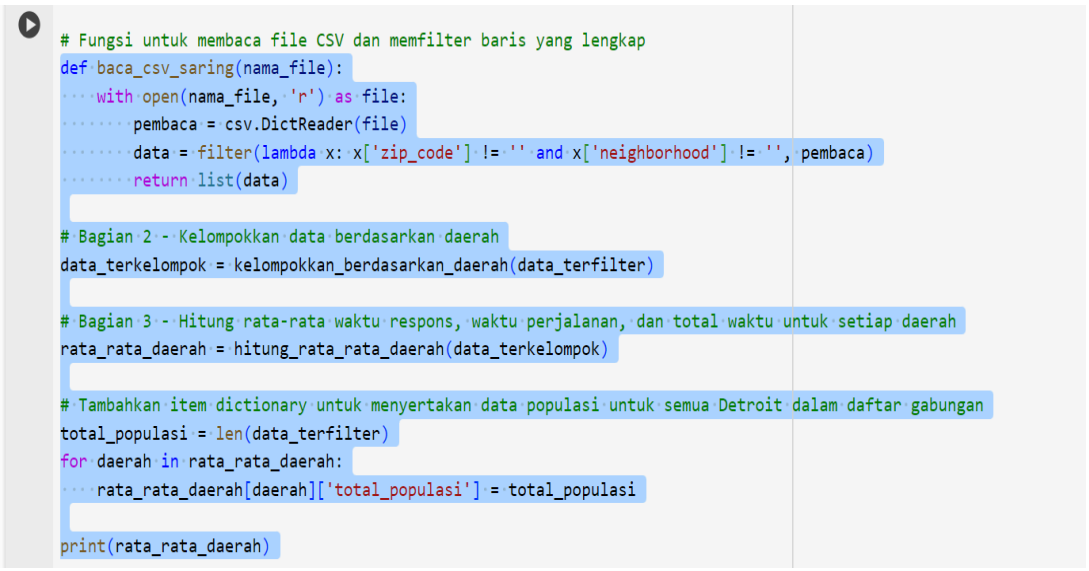
```
else:
    waktu_respons = [float(baris['totalresponsetime']) for baris in catatan if baris['totalresponsetime'].strip()]
    waktu_perjalanan = [float(baris['traveltime']) for baris in catatan if baris['traveltime'].strip()]
    waktu_total = [float(baris['totaltime']) for baris in catatan if baris['totaltime'].strip()]

    jumlah_catatan = len(waktu_respons)
    if jumlah_catatan > 0:
        total_waktu_respons = sum(waktu_respons)
        total_waktu_perjalanan = sum(waktu_perjalanan)
        total_waktu = sum(waktu_total)
        rata_rata[daerah] = {
            'rata_rata_waktu_respons': total_waktu_respons / jumlah_catatan,
            'rata_rata_waktu_perjalanan': total_waktu_perjalanan / jumlah_catatan,
            'rata_rata_waktu_total': total_waktu / jumlah_catatan,
            'populasi': len(catatan)  # Menambahkan populasi setiap daerah
        }
    else:
        rata_rata[daerah] = {
            'rata_rata_waktu_respons': 0,
            'rata_rata_waktu_perjalanan': 0,
            'rata_rata_waktu_total': 0,
            'populasi': len(catatan)
        }
```

Gambar 3.3.4. Rata-rata Waktu *Respons*, Waktu Pengiriman, dan Total Waktu Pengiriman Barang di *neighborhood*.

Potongan kode tersebut merupakan bagian dari fungsi hitung_rata_rata_daerah(data), yang bertujuan untuk menghitung rata-rata waktu *respons*, waktu perjalanan, dan total waktu untuk setiap daerah yang telah dikelompokkan sebelumnya berdasarkan daerah (*neighborhood*). Dalam potongan kode tersebut, setiap daerah dan catatan yang terkait diperiksa untuk memastikan apakah ada data yang tersedia. Jika daerah tidak memiliki catatan yang terkait, sebuah entri dibuat dalam dictionary yang menyatakan rata-rata waktu *respons*, waktu perjalanan, dan waktu total untuk daerah tersebut, dengan nilai-nilai tersebut diatur ke 0. Hal ini memastikan bahwa perhitungan rata-rata tetap dilakukan bahkan jika tidak ada data yang tersedia untuk daerah tertentu.

Populasi daerah juga dihitung dan disertakan dalam hasil akhir. Dengan demikian, potongan kode tersebut mengimplementasikan penanganan yang sesuai untuk kasus di mana tidak ada data yang tersedia untuk sebuah daerah, memastikan konsistensi dalam perhitungan rata-rata dan inklusi populasi dalam analisis data.Kode melakukan pengumpulan data dengan iterasi melalui setiap daerah (*neighborhood*) untuk mengambil nilai waktu *respons*, waktu perjalanan, dan waktu total dari catatan yang terkait. Ini dilakukan menggunakan *list* comprehension yang memastikan nilai-nilai tersebut diambil dari setiap baris catatan yang tidak hanya berisi spasi setelah di-*strip*.

Setelah nilai-nilai waktu berhasil dikumpulkan, kode menghitung rata-rata waktu *respons*, waktu perjalanan, dan waktu total. Jika ada data yang tersedia (jumlah catatan lebih dari 0), maka rata-rata waktu dihitung dan dimasukkan ke dalam *dictionary* untuk daerah yang bersangkutan. Namun, jika tidak ada data yang tersedia (jumlah catatan adalah 0), maka rata-rata untuk semua waktu diatur ke 0 untuk mencegah pembagian oleh 0. Dalam kasus di mana tidak ada data yang tersedia untuk sebuah daerah, sebuah *dictionary* dibuat dengan nilai rata-rata waktu 0 untuk waktu respons, waktu perjalanan, dan waktu total, memastikan jumlah populasi daerah tetap disertakan dalam hasil akhir. Setelah semua proses pengumpulan dan perhitungan selesai, dictionary yang berisi rata-rata waktu dan populasi untuk setiap daerah dikembalikan sebagai hasil akhir.



```
# Fungsi untuk membaca file CSV dan memfilter baris yang lengkap
def baca_csv_saring(nama_file):
    with open(nama_file, 'r') as file:
        pembaca = csv.DictReader(file)
        data = filter(lambda x: x['zip_code'] != '' and x['neighborhood'] != '', pembaca)
        return list(data)

# Bagian 2 - Kelompokkan data berdasarkan daerah
data_terkelompok = kelompokkan_berdasarkan_daerah(data_terfilter)

# Bagian 3 - Hitung rata-rata waktu respons, waktu perjalanan, dan total waktu untuk setiap daerah
rata_rata_daerah = hitung_rata_rata_daerah(data_terkelompok)

# Tambahkan item dictionary untuk menyertakan data populasi untuk semua Detroit dalam daftar gabungan
total_populasi = len(data_terfilter)
for daerah in rata_rata_daerah:
    rata_rata_daerah[daerah]['total_populasi'] = total_populasi

print(rata_rata_daerah)
```

Gambar 3.3.5. Perbandingan Waktu Respons, Waktu Perjalanan, dan Total Waktu antara Daerah di Detroit

Potongan kode ini mencakup serangkaian langkah untuk menganalisis data yang berasal dari file CSV. Pertama-tama, fungsi baca_csv_saring(nama_file) digunakan untuk membaca file CSV dan melakukan filter terhadap baris-baris yang lengkap, yaitu yang memiliki nilai tidak kosong untuk kolom *'zip_code'* dan *'neighborhood'*. Setelah data difilter, langkah berikutnya adalah mengelompokkan data berdasarkan daerah (*neighborhood*) menggunakan fungsi kelompokkan_berdasarkan_daerah(). Hasil dari pengelompokkan ini disimpan dalam variabel data_terkelompok. Selanjutnya, dilakukan perhitungan rata-rata waktu respons, waktu perjalanan, dan total waktu untuk setiap daerah dengan memanggil fungsi hitung_rata_rata_daerah(), dan hasilnya disimpan dalam variabel rata_rata_daerah.

Untuk memperkaya analisis, jumlah populasi untuk seluruh Detroit ditambahkan ke setiap daerah dalam dictionary rata_rata_daerah. Langkah terakhir adalah mencetak hasil analisis yang mencakup rata-rata waktu respons, waktu perjalanan, dan total waktu untuk setiap daerah, serta populasi total Detroit. Dengan demikian, potongan kode ini menunjukkan proses lengkap dari membaca, mengelompokkan, menghitung rata-rata, dan menganalisis data dari file CSV untuk mendapatkan pemahaman yang lebih baik tentang waktu layanan untuk setiap daerah di Detroit.

```python
import json

# Bagian 3: Buat file Output JSON
def simpan_ke_json(data_part1, data_daerah, nama_file):
    with open(nama_file, 'w') as berkas_json:
        json.dump({'hasil_part1': data_part1, 'rata_rata_daerah': data_daerah}, berkas_json, indent=4)

# Path untuk menyimpan file JSON
path_output_json = "output.json"

# Simpan data ke dalam file JSON
simpan_ke_json(hasil_part1, rata_rata_daerah, path_output_json)

print("Data telah disimpan dalam file JSON:", path_output_json)
```

```
Data telah disimpan dalam file JSON: output.json
```

Gambar 3.3.6. Penyimpanan dan Pemulihan Data: Menggunakan Format JSON untuk Analisis Data

Gambar 3.3.6. Penyimpanan dan Pemulihan Data: Menggunakan Format JSON untuk Analisis Data

Potongan kode ini bertujuan untuk menyimpan hasil analisis data ke dalam format file JSON. Fungsi simpan_ke_json(data_part1, data_daerah, nama_file) digunakan untuk menulis data ke dalam file JSON dengan format yang terstruktur. Setelah fungsi didefinisikan, path_output_json ditentukan sebagai lokasi tempat menyimpan file JSON. Kemudian, fungsi simpan_ke_json() dipanggil dengan parameter hasil analisis yang telah disiapkan sebelumnya, yaitu hasil_part1 dan rata_rata_daerah, serta path_output_json sebagai lokasi penyimpanan file JSON. Setelah penyimpanan selesai, pesan "Data telah disimpan dalam file JSON:" disertai dengan path_output_json dicetak untuk memberi tahu pengguna bahwa proses penyimpanan telah berhasil dilakukan. Dengan demikian, potongan kode ini menambahkan langkah terakhir dalam proses analisis data, yaitu menyimpan hasil analisis ke dalam file JSON untuk referensi dan penggunaan selanjutnya.

Potongan kode ini membaca isi dari file JSON yang telah disimpan sebelumnya dengan nama "*output*.json". Dengan menggunakan blok with open("output.json", "r") as file, file JSON dibuka dalam mode baca ("r") dan diakses dengan nama file. Selanjutnya, isi file dibaca menggunakan metode .read() dan disimpan dalam variabel content. Terakhir, isi dari file JSON tersebut dicetak menggunakan perintah print(content), sehingga pengguna dapat melihat data yang telah disimpan dalam format JSON.

4. Kesimpulan

Berdasarkan pembahasan diatas dapat disimpulkan :

1. kita dapat menggunakan modul pandas untuk membaca dataset csv
2. dengan menggunakan *neighborhood* kita dapat mengelompokan data populasi yang hilang dengan dibantu memakai fungsi filter dan fungsi lambda untuk menghitung waktu respons rata-rata,waktu pengiriman rata2,dan total waktu rata-rata untuk kepolisian detroit
3. dengan *neighborhood* juga tidak hanya mengelompokan data populasi saja melainkan dapat mengelompokan samples lalu disimpan ke dalam bentuk list dictionary yang dibantu menggunakan filter dan fungsi lambda
4. hasil analisis dapat disimpan kedalam format list dengan menggunakan modul json dimana tidak hanya disimpan namun dapat pembacaan kembali

5. Daftar Pustaka

[1] T. Suryana and T. I. Unikom, “Belajar Pemrograman LIST dan MAP Constructor (())”.

[2] M. Nugroho, “Bab iii landasan teori 3.1.,” *http://e-journal.uajy.ac.id/7244/4/3TF03686.pdf*, no. C, pp. 15–48, 2003.

[3] Y. Herdiana, “Aplikasi Rumus Matematika Sma Berbasis Mobile,” *J. Ilm. Komput. dan Inform. (KOMPUTA)*, vol. 3, no. 1, pp. 112–121, 2018.

[4] J. D. Beazley and B. K. Jones, "Python Cookbook," O'Reilly Media, 2013.

[5] Wes McKinney, "Python for Data Analysis," O'Reilly Media, 2017.

6. Lampiran

Lampiran berupa kode Google Collab yang telah kami kerjakan secara kelompok :
Kelompok_4_Modul_7.ipynb - Colab (google.com)