Analisis Waktu Respons Kepolisian Detroit Menggunakan Pendekatan Pemrograman Fungsional Python

Irhamna Mahdi[1], Salwa Farhanatussaidah[2], Ganiya Syazwa[3], Reynaldi Rahmad[4], Syalaisha Andina Putriansyah[5]

Program Studi Sains Data, Fakultas Sains, Institut Teknologi Sumatera.

Email: irhamna.122450049@student.itera.ac.id, salwa.122450055@student.itera.ac.id, ganiya.122450073@student.itera.ac.id, reynaldi.122450088@student.itera.ac.id, syalaisha.122450121@student.itera.ac.id

1. Pendahuluan

Efisiensi waktu respons pada kepolisian merupakan hal yang penting dalam menilai kinerja layanan darurat untuk menjaga keamanan dan ketertiban masyarakat di kota-kota besar seperti halnya Kota Detroit. Menganalisis waktu respons kepolisian memerlukan pendekatan yang dapat menangani data dalam jumlah besar dan variasi yang besar.

Pendekatan pemrograman fungsional *python* memberikan solusi ideal untuk menyelesaikan permasalahan ini. Pemrograman fungsional berfokus pada penggunaan fungsi dan ekspresi yang tidak secara langsung mengubah keadaan program tersebut.

Dalam konteks ini, untuk mendapatkan informasi mendalam mengenai permasalahan tersebut dilakukan analisis dan pencarian pola-pola dalam waktu respons dan berbagai variabel data yang ada dapat mempengaruhi efisiensi waktu respon kepolisian. Agar dapat memproses dan juga menganalisis data secara efisien, digunakan fungsi pada Python yang mendukung pemrograman fungsional seperti fungsi *reduce*, *map*, *lambda* serta modul *json*.

2. Metode

2.1. Fungsi

2.1.1. Fungsi Reduce

Fungsi reduce dalam python merupakan fungsi bawaan yang digunakan untuk menerapkan suatu fungsi tertentu ke semua elemen dalam sebuah iterable secara urut dengan mengambil hasil operasi sebelumnya dan elemen berikutnya sebagai argumen (Kuraś et al., 2023). Fungsi ini umumnya digunakan untuk melakukan agregasi data atau operasi pemrosesan data yang melibatkan urutan elemen.

2.1.2. Fungsi Map

Fungsi map merupakan fungsi built-in dalam bahasa pemrograman python yang digunakan untuk memproses dan mengubah setiap item yang ada ke

dalam perulangan tanpa menggunakan loop. Cara kerja fungsi ini adalah dengan menerapkan fungsi ke dalam setiap item dan melakukan pembaruan pada variabelnya yang berbentuk list (Ramandhany & Kunang. 2021).

2.1.3. Fungsi Lambda

Fungsi lambda adalah fungsi kecil satu baris yang digunakan untuk menggantikan fungsi sederhana. Sayangnya, fungsi lambda hanya dapat memuat satu ekspresi saja dan tidak dapat digunakan untuk menggantikan fungsi yang kompleks dan rumit (Calvin, A .2022). Penulisan lamda di awal adalah sebuah keharusan agar program mengetahui bahwa fungsi yang dimasukkan menggunakan fungsi lambda sedangkan ekspresi dapat diubah sesuai dengan kebutuhan user.

2.2. Modul

2.2.1. Modul'json'

Modul 'json' dapat digunakan untuk mengubah data dari python ke format 'json' dan sebaliknya. Penggunaan modul ini dapat memungkinkan untuk menyimpan data dalam format yang ringkas dan mudah dibaca, dan juga dapat mengambil serta memanipulasi data yang dikirim atau diterima dalam format json (William, S.2022).

3. Pembahasan

3.1. Model Populasi Detroit

3.1.1. Import Modul

```
import pandas as pd
from functools import reduce
```

Gambar 1. Import modul

Kode pada Gambar 1. merupakan tahap awal yang dilakukan, yaitu mengimport modul pandas untuk membaca data dalam berbagai format, sedangkan reduce digunakan untuk menerapkan fungsi ke item berurutan dari iterable dan menggabungkannya menjadi satu hasil.

3.1.2. Impor Dataset

```
data = pd.read_csv('/content/911_Calls_for_Service_(Last_30_Days).csv')
```

Gambar 2. Membaca file

Variabel data menyimpan DataFrame yang merupakan hasil dari pembacaan file CSV yang diberikan.

3.1.3. Filter Data Kosong

```
filtered_data = data.dropna(subset=['zip_code', 'neighborhood'])
```

Gambar 3. Filter data kosong

Pada Gambar 3. metode .dropna() digunakan untuk menghapus baris yang bernilai null atau NaN pada kolom 'zip_code' atau 'neighborhood' dari

DataFrame data, kemudian hasilnya disimpan ke dalam Variabel filtered_data.

3.1.4. Operasi perhitungan waktu kepolisian detroit

```
# Ubah menjadi daftar dictionary
filtered_dict = filtered_data.to_dict(orient='records')

# Filter data untuk kepolisian Detroit
detroit_data = filter(lambda x: 'Detroit' in x['neighborhood'], filtered_dict)

# Fungsi menghitung total waktu respons, waktu pengiriman, dan total waktu rata-rata
def calculate_averages(acc, curr):
    acc['total_response_time'] += curr['totalresponsetime']
    acc['total_dispatch_time'] += curr['dispatchtime']
    acc['total_time_on_scene'] += curr['time_on_scene']
    acc['count'] += 1
    return acc

# Inisialisasi nilai awal
initial_values = {'total_response_time': 0, 'total_dispatch_time': 0, 'total_time_on_scene': 0, 'count': 0}

# Hitung total waktu respons, waktu pengiriman, dan total waktu rata-rata
totals = reduce(calculate_averages, detroit_data, initial_values)

# Hitung rata-rata
average_response_time = totals['total_response_time'] / totals['count']
average_dispatch_time = totals['total_dispatch_time'] / totals['count']
average_time_on_scene = totals['total_time_on_scene'] / totals['count']
```

Gambar 4. Operasi perhitungan waktu kepolisian detroit

Kode pada Gambar 4. Merupakan proses mengolah dataset yang berkaitan dengan Detroit. Berikut penjelasan bagian-bagian kode pada Gambar 4:

- Konversi: variabel filtered_dict menyimpan hasil dari konversi objek filtered_data dari tipe dataframe menjadi sebuah dictionary.
- Pengelompokan data: mendefinisikan detroit_data sebagai hasil dari filter data, dimana baris neighborhood hanya yang mengandung kata “Detroit”.
- Fungsi Perhitungan Rata-Rata: Fungsi calculate_averages(acc, curr) menghitung total dan rata-rata waktu respons, waktu pengiriman, dan total waktu di tempat. Fungsi ini menggunakan reduce() untuk mengakumulasi total dari setiap kategori waktu.
- Inisialisasi Nilai Awal: Sebuah dictionary initial_values dibuat dengan nilai awal untuk total waktu respons, waktu pengiriman, dan total waktu di tempat.
- Perhitungan Total dan Rata-Rata: Total dan rata-rata waktu dihitung menggunakan data yang telah difilter dan diakumulasi.

3.2. Model Neighborhood Samples

3.2.1. Fungsi calculate_neighborhood_averages

```python
# Fungsi menghitung total waktu respons, waktu pengiriman, dan total waktu rata-rata untuk setiap neighborhood
def calculate_neighborhood_averages(acc, curr):
    neighborhood = curr[0]['neighborhood']
    total_response_time = sum(map(lambda x: x['totalresponsetime'], curr))
    total_dispatch_time = sum(map(lambda x: x['dispatchtime'], curr))
    total_time_on_scene = sum(map(lambda x: x['time_on_scene'], curr))
    count = len(curr)

    acc.append({
        'neighborhood': neighborhood,
        'total_response_time': total_response_time,
        'total_dispatch_time': total_dispatch_time,
        'total_time_on_scene': total_time_on_scene,
        'count': count,
        'average_response_time': total_response_time / count,
        'average_dispatch_time': total_dispatch_time / count,
        'average_time_on_scene': total_time_on_scene / count
    })
    return acc
```

Gambar 5. Operasi perhitungan waktu neighborhood

Pembuatan fungsi calculate_neighborhood_averages bertujuan untuk menghitung total dan rata-rata waktu respons, waktu pengiriman, dan total waktu di tempat untuk setiap ‘neighborhood’. Fungsi ini bekerja dengan mengambil list baris data untuk satu ‘neighborhood’, menghitung total dan rata-rata waktu respons, waktu pengiriman, dan total waktu di tempat, kemudian menambahkan hasilnya ke dalam list acc.

3.2.2. Memisahkan Data Berdasarkan Neighborhood

```python
# Bagi list dictionary menjadi list dictionary yang lebih kecil yang dipisahkan oleh neighborhood
neighborhood_data = map(lambda neighborhood: list(filter(lambda x: x['neighborhood'] == neighborhood, filtered_dict)),
                        set(map(lambda x: x['neighborhood'], filtered_dict)))
```

Gambar 6. Dictionary neighborhood

Kode pada Gambar 6 merupakan proses pengelompokan data berdasarkan nilai pada kolom ‘neighborhood’. Berikut proses yang dilakukan kode pada gambar 6:

a. Membuat set unik dari ‘neighborhood’ yang berisi semua nilai unik dari kolom ‘neighborhood’ pada objek filtered_dict.
b. Membuat list yang berisi baris-baris data untuk setiap ‘neighborhood’ dengan menggunakan fungsi filter() dalam memilih baris-baris pada objek filtered_dict yang memiliki nilai ‘neighborhood’ yang sama.
c. Pengelompokkan data dilakukan dengan menggunakan fungsi map untuk memasangkan baris-baris data untuk setiap ‘neigborhood’ dengan set unik dari ‘neighborhood’.

3.2.3. Menambahkan Data Populasi untuk Seluruh Detroit

```
# Hitung total waktu respons, waktu pengiriman, dan total waktu rata-rata untuk setiap neighborhood
neighborhood_averages = reduce(calculate_neighborhood_averages, neighborhood_data, [])

# Tambahkan item dictionary untuk menyertakan data populasi untuk semua Detroit
total_detroit_population = {
    'neighborhood': 'All Detroit',
    'total_response_time': totals['total_response_time'],
    'total_dispatch_time': totals['total_dispatch_time'],
    'total_time_on_scene': totals['total_time_on_scene'],
    'count': totals['count'],
    'average_response_time': average_response_time,
    'average_dispatch_time': average_dispatch_time,
    'average_time_on_scene': average_time_on_scene
}
neighborhood_averages.append(total_detroit_population)
```

Gambar 7. Operasi perhitungan waktu neighborhood

Kode pada Gambar 7. melakukan proses perhitungan total dan rata-rata waktu respons, waktu pengiriman, dan total waktu di tempat untuk setiap 'neighborhood' dan juga untuk seluruh Detroit, kemudian hasil dari perhitungan disimpan dalam list neighborhood_averages.

3.3. File Output JSON

```
import json

# Tulis data ke dalam file JSON
with open('output.json', 'w') as f:
    json.dump(neighborhood_averages, f, indent=4)
```

Gambar 8. Konversi format data menjadi JSON

Kode pada Gambar 8. akan mengambil data dari variabel neighborhood_averages dan mengubahnya menjadi format JSON dan menulisnya ke file 'output.json'. Konversi yang dilakukan menggunakan modul json.

4. Kesimpulan

Analisis data kepolisian detroit menggunakan bahasa pemrograman yaitu pyhton yang mencakup fungsi reduce, map, dan lambda serta menggunakan modul json untuk memproses dan mengagregasi data dengan efisien. Reduce digunakan untuk menggabungkan elemen iterable yaitu menghitung total dan rata - rata waktu respons, map digunakan untuk mengubah setiap item yang terdapat pada iterable tanpa menggunakan loop, dan lambda digunakan untuk menyediakan fungsi satu baris untuk mengoperasikan secara sederhana. Modul json digunakan untuk mengonversi data antara format pyhton dan json, hal ini memungkinkan penyimpanan dan manipulasi data dengan efisien. Proses dimulai dengan mengimpor modul dan data csv kedalam dataframe, kemudian meghapus baris dengan nilai null, kemudian data dikelompokkan berdasarkan neighboorhood, dihitung total dan rata rata waktu respon dengan menggunakan fungsi calculate_averages. Hasil perhitungan setiap neighboorhood kemudian dikonversi ke format json dan disimpan didalam file output.json, menunjukkan keefisienan dalam penggunakan pyhton untuk analisis data.

5. Daftar Pustaka

Kuraś, P., Strzalka, D., Kowal, B., & Mazurek, J. (2023). REDUCE – A Python Module for Reducing Inconsistency in Pairwise Comparison Matrices. Advances in Science and Technology Research Journal, 17(4), 227–234. https://doi.org/10.12913/22998624/170187

Ramandhany, D., & Kunang, Y. N. (2021). Visualisasi Heat Map Data Kecelakaan Di Kota Palembang. Bina Darma Conference on Computer Science, 304–311.

William, S., Wilda,N,. & Mahgrisy, S,.(2022) The Development of Telegram Bot API to Maximize The Dissemination Process of Islamic Knowledge in 4.0 Era. Jurnal Teknik Informatika Vol. 15 No. 1,

Calvin, A. Melisa, M. Sandra, O.(2022) Implementasi IoT dengan ESP 32 Untuk Pemantauan Kondisi Suhu Secara Jarak Jauh Menggunakan MQTT Pada AWS. Jurnal Elektro Vol.15 No.2 Oktober 2022