Memodelkan Populasi Kepolisian Kota Detroit terkait Informasi Panggilan Layanan Darurat 911 dalam 30 Hari Terakhir

Deva Anjani Khayyuninafsyah (122450014), Nabiilah Putri Karnaia (122450029),
Chevando Daffa Pramanda (122450095), Novelia Adinda (122450104),
Rafly Prabu Darmawan (122450140)

Program Studi Sains Data, Fakultas Sains, Institut Teknologi Sumatera
Jl. Terusan Ryacudu, Way Huwi, Kec. Jatiagung, Kabupaten Lampung Selatan, Lampung 35365

Email:
deva.122450014@student.itera.ac.id, nabiilah.122450029@student.itera.ac.id,
chevando.122450095@student.itera.ac.id, novelia.122450104@student.itera.ac.id,
rafly.122450140@student.itera.ac.id

1. Pendahuluan

Sebagai salah satu kota terbesar di Amerika Serikat, Detroit dihadapkan pada tantangan yang sangat rumit dalam menjaga ketertiban dan keamanan masyarakat. Banyak faktor yang mempengaruhi tingkat kejahatan dan tanggapan polisi. Untuk itu, perlu pemahaman terkait susunan, penyebaran, dan perilaku polisi sebagai kunci utama dalam merencanakan strategi kepolisian yang efektif dalam mengatasi permasalahan tersebut.

Berdasarkan uraian di atas, langkah yang sangat penting dalam menganalisis keamanan kota secara menyeluruh salah satunya adalah membuat model populasi polisi Detroit. Di samping itu, memodelkan Neighborhood Samples juga merupakan langkah penting dalam upaya memahami dinamika keamanan di tingkat lokal. Dalam melakukannya, digunakan dataset layanan panggilan darurat 911 dalam 30 hari terakhir di kepolisian kota Detroit. Untuk menyimpan informasi tentang populasi polisi Detroit, sampel lingkungan, dan data keamanan lainnya dapat digunakan pengumpulan dan analisis data dengan format file JSON (JavaScript Object Notation).

Dengan demikian, diharapkan langkah-langkah tersebut mampu memudahkan pertukaran data yang efisien antara pihak-pihak terkait sehingga dapat menghasilkan pemahaman yang lebih mendalam tentang dinamika keamanan kota dan merumuskan strategi peningkatan keamanan yang lebih efektif.

2. Metode

Dalam permasalahan kali ini akan dilakukan pemodelan populasi polisi Detroit untuk menghitung total waktu respon rata-rata, waktu pengiriman rata-rata, dan total waktu rata-rata untuk Kepolisian Detroit. Tidak hanya itu, dilakukan pula pemodelan Neighborhood Samples untuk memisahkan data berdasarkan data neighborhood tersebut. Selain itu, hasilnya akan diubah dan disimpan dalam bentuk file JSON.

Untuk melakukan proses pada data tersebut diterapkan beberapa fungsi. Fungsi-fungsi tersebut antara lain fungsi adalah fungsi *map()*, *filter(), reduce()*, dan fungsi lambda yang dijelaskan sebagai berikut:

1.1. Fungsi *map()*

Fungsi *map()* adalah fungsi bawaan yang tersedia di banyak bahasa pemrograman, termasuk Python, yang digunakan untuk menerapkan fungsi tertentu pada setiap item dari iterable (seperti daftar, tuple, atau himpunan), dan menghasilkan iterable baru sebagai hasil dari penerapan fungsi.

1.2. Fungsi *filter()*

Fungsi *filter()* adalah fungsi bawaan Python yang digunakan untuk menyaring elemen dari iterable (seperti list, tuple, atau string) berdasarkan fungsi tertentu yang mengembalikan nilai boolean. Fungsi ini berguna untuk memilih elemen yang memenuhi kondisi tertentu dan mengabaikan elemen yang tidak sesuai dengan kondisi.

1.3. Fungsi *reduce()*

Fungsi reduce() termasuk dalam modul *functools* Python yang digunakan untuk melakukan operasi penggabungan elemen-elemen dalam suatu iterable (seperti list atau tuple) menjadi satu nilai tunggal. Operasi ni dilakukan dengan menerapkan fungsi yang mengambil dua argumen dan mengembalikan satu nilai secara berulang pada elemen iterable.

1.4. Fungsi *lambda()*

Fungsi lambda memungkinkan pembuatan fungsi anonim (fungsi tanpa nama) secara cepat dan ringkas. Ketika fungsi sederhana diperlukan untuk penggunaan sementara, fungsi lambda biasanya digunakan sebagai argumen dalam fungsi tingkat tinggi seperti *map()*, *filter()*, atau *reduce()*.

3. Pembahasan

```
import pandas as pd

data1 = pd.read_csv("911_Calls_for_Service_(Last_30_Days).csv")
data1
```

Gambar 1

Pada Gambar 1 ini merupakan kode yang akan membaca data menggunakan *library pandas* dan menyimpannya ke dalam variabel data1 lalu memanggilnya.

```
import csv
from functools import reduce

# membuka file Laporan Polisi Detroit menggunakan modul csv
def baca_data(file_namer):
    with open(file_namer, 'r') as file:
        reader = csv.DictReader(file)
        return list(reader)

# mengubah data dari file tersebut ke dalam list dictionary
data = baca_data('911_Calls_for_Service_(Last_30_Days).csv')

# melakukan filter dengan fungsi lambda untuk menghilangkan baris yang memiliki data yang hilang di kolom zip_code atau kolom neighborhood
filtered_data = list(filter(lambda x: x['zip_code'] != '' and x['neighborhood'] != '' and x['totalresponsetime'] != '' and x['traveltime'] !

# menghitung total waktu respons rata-rata, waktu pengiriman rata-rata, dan total waktu rata-rata untuk kepolisian Detroit menggunakan fungs
total_totalresponsetime = reduce(lambda x, y: x + float(y['totalresponsetime']), filtered_data, 0)
total_traveltime = reduce(lambda x, y: x + float(y['traveltime']), filtered_data, 0)
total_totaltime = reduce(lambda x, y: x + float(y['totaltime']), filtered_data, 0)

rerata_totalresponsetime = total_totalresponsetime / len(filtered_data)
rerata_traveltime = total_traveltime / len(filtered_data)
rerata_totaltime = total_totaltime / len(filtered_data)
```

Gambar 2

Pada Gambar 2 ini kita akan membuat Model Populasi Polis Detroit. Pertama-tama kita akan meng*import* csv dan *reduce* dari *functools*. Kemudian membuka *file* laporan panggilan layanan kepolisian selama 30 hari terakhir di Detroit. Data dari *file* CSV dibaca dan di *filter* untuk menghapus baris yang memiliki data yang hilang di beberapa kolom kunci seperti *'zip_code'*, *'neighborhood'*, *'totalresponsetime'*, *'traveltime'*, dan *'totaltime'* lalu menyimpannya dalam variabel *filtered_data*. Setelah itu, dilakukan penghitungan rata-rata total waktu respon, waktu pengiriman, dan total waktu untuk kepolisian detroit menggunakan fungsi *lambda*() dan fungsi *reduce*(). Kemudian kode rerata_total ini merupakan proses perhitungan rata-rata waktu respons, waktu pengiriman, dan total waktu dengan membagi total waktu yang telah dihitung sebelumnya dengan panjang dari *'filtered_data'*.

```
print("Total Response Time:", total_totalresponsetime)
print("Total waktu respons rata-rata:", rerata_totalresponsetime)
print("\nTotal Travel Time:", total_traveltime)
print("Waktu pengiriman rata-rata:", rerata_traveltime)
print("\nTotal Total Time:", total_totaltime)
print("Total waktu rata-rata:", rerata_totaltime)
```

Gambar 3

Pada Gambar ketiga ini adalah kode yang akan mencetak hasil perhitungan yang telah dilakukan seperti rata-rata dan total waktu respon, waktu pengiriman, dan waktu pengiriman. Setiap hasil perhitungan akan dicetak dengan menggunakan fungsi *print*().

```
import csv
from functools import reduce

# fungsi untuk memisahkan data berdasarkan neighborhood dengan lambda dan filter
def pisahkan_berdasarkan_neighborhood(data):
    return {neighborhood: list(filter(lambda x: x['neighborhood'] == neighborhood, data)) for neighborhood in set(map(lambda x: x['neighborhoo

# fungsi untuk menghitung total waktu respons, waktu pengiriman, dan total waktu rata-rata untuk setiap neighborhood
def calculate_average_times(data):
    total_response = reduce(lambda x, y: x + (float(y['totalresponsetime']) if y['totalresponsetime'] != '' else 0), data, 0)
    total_pengiriman = reduce(lambda x, y: x + (float(y['traveltime']) if y['traveltime'] != '' else 0), data, 0)
    total_waktu = reduce(lambda x, y: x + (float(y['totaltime']) if y['totaltime'] != '' else 0), data, 0)
    count = len(data)
    rata_rata_response = total_response / count if count > 0 else 0
    rata_rata_pengiriman = total_pengiriman / count if count > 0 else 0
    rata_rata_waktu = total_waktu / count if count > 0 else 0

    return {
        'total_response': total_response,
        'rata_rata_response': rata_rata_response,
        'total_pengiriman': total_pengiriman,
        'rata_rata_pengiriman': rata_rata_pengiriman,
        'total_waktu': total_waktu,
        'rata_rata_waktu': rata_rata_waktu
    }

# memisahkan data berdasarkan neighborhood
data_berdasarkan_neighborhood = pisahkan_berdasarkan_neighborhood(filtered_data)

# menghitung total waktu respons, waktu pengiriman, dan total waktu rata-rata untuk setiap neighborhood
hasil_perhitungan = []
for neighborhood, entries in data_berdasarkan_neighborhood.items():
    hasil_perhitungan.append({'neighborhood': neighborhood, **calculate_average_times(entries)})

hasil_perhitungan.append({'neighborhood': 'Detroit', **calculate_average_times(filtered_data)})
```

Gambar 4

Gambar keempat adalah kode yang akan memodelkan *neighborhood samples*. Kode di atas akan mengimpor modul `csv` untuk membaca *file* CSV dan `*reduce*` dari `*functools*` untuk melakukan operasi reduksi pada data. Terdapat dua fungsi utama dalam gambar diatas yaitu:

1. *`pisahkan_berdasarkan_neighborhood`,*

   Fungsi ini akan memisahkan data berdasarkan lingkungan, menggunakan fungsi *filter* dan fungsi *lambda* untuk membentuk dictionary dengan kunci sebagai lingkungan dan nilai sebagai entri data terkait.

2. `*calculate_average_times*`

   Fungsi ini akan menghitung total dan rata-rata waktu respons, waktu pengiriman, dan total waktu untuk setiap lingkungan dengan mengurangi data dan menghitung statistik yang diperlukan.

Kemudian, proses pemisahan data juga akan dilakukan dengan memanggil fungsi `*pisahkan_berdasarkan_neighborhood*`. Hasil perhitungan tersebut kemudian disimpan dalam list `hasil_perhitungan`. Baris kode terakhir adalah hasil perhitungan yang akan dicetak ke layar menggunakan *loop* lalu akan menyajikan informasi tentang total dan rata-rata waktu respons, waktu pengiriman, dan total waktu untuk masing,

```
# menampilkan hasil perhitungan
for data in hasil_perhitungan:
    print(f"Neighborhood: {data['neighborhood']}")
    print(f"Total Response Time: {data['total_response']}")
    print(f"Total waktu respons rata-rata: {data['rata_rata_response']}")
    print(f"Total Travel Time: {data['total_pengiriman']}")
    print(f"Waktu pengiriman rata-rata: {data['rata_rata_pengiriman']}")
    print(f"Total Waktu: {data['total_waktu']}")
    print(f"Total waktu rata-rata: {data['rata_rata_waktu']}\n")
```

Gambar 5

Gambar kelima ini akan menampilkan sebuah iterasi *looping* melalui setiap elemen dalam list ‘hasil_perhitungan’. Setiap iterasi yang ada dalam list merupakan suatu *dictionary* yang berisi informasi statistik setiap *Neighborhood* yang telah dihitung sebelumnya. Pada setiap iterasi akan mencetak berbagai informasi terkait *neighborhood*, termasuk total waktu respons, total waktu pengiriman, rata-rata waktu pengiriman, total waktu, dan rata-rata total waktu.

```
import json

json_data = json.dumps(hasil_perhitungan, indent=4)

# menampilkan atau menyimpan hasil JSON
print(json_data)

# apabila ingin disimpan dalam bentuk file JSON, dapat menggunakan perintah berikut:
with open("hasil_perhitungan.json", "w") as json_file:
     json_file.write(json_data)
```

Gambar 6

Gambar keenam diatas diatas adalah pembuatan file format *list dictionary* dengan modul JSON. pertama-tama kita akan memanggil json dan membuat *variabel* json_data yang berisikan jsom.dumps dengan argumen hasil perhitungan dan indient=4. Kemudian mencetak hasil *JSON*

dengan print() lalu apabila ingin disimpan dalam bentuk file JSON, kita akan menggunakan with open (“hasil_perhitungan.json’,”w”) as json_file, lalu memanggilnya.

4. Kesimpulan

Berdasarkan penjelasan diatas, dapat disimpulkan bahwa dalam analisis keamanan perkotaan, model Populasi Polis Detroit dan penggunaan model Neighborhood Samples memainkan peran penting. Bagi Detroit, model populasi polis yang diterapkan akan membantu dalam menjaga keamanan masyarakat melalui pemahaman susunan, penyebaran, dan perilaku polisi. Di sisi lain, penggunaan model Neighborhood Samples memungkinkan peneliti untuk memahami faktor-faktor yang mempengaruhi tingkat kejahatan dan respons polisi, sehingga dapat mengembangkan strategi intervensi yang tepat dan efektif.

Pada laporan ini, penggunaan format data JSON sangat penting untuk mengumpulkan, menyimpan dan bertukar informasi antara peneliti dengan pihak terkait lainnya. JSON menyediakan kerangka kerja yang sederhana dan efisien untuk menyimpan informasi tentang populasi polisi, pola lingkungan, dan data keamanan lainnya. Fungsi seperti *map()*, *filter()*, *reduce()*, lambda, dan lainnya dapat digunakan untuk memproses data guna menghasilkan wawasan berharga mengenai model keamanan perkotaan. Misalnya, menilai efektivitas dan daya tanggap polisi dalam mengelola situasi keamanan dengan menghitung waktu respons dan pengiriman polisi, serta total waktu yang diperlukan untuk menanggapi panggilan layanan. Oleh karena itu, dengan memeriksa secara cermat dan menggunakan metode pemrosesan data yang tepat, kita dapat memperoleh pemahaman yang lebih mendalam tentang dinamika keamanan kota dan mengusulkan strategi peningkatan keamanan yang lebih efektif untuk menanggapi kebutuhan masyarakat.

5. Daftar Pustaka

Python Software Foundation. 2023. Python Documentation: Built-in Functions. Python Software Foundation.

Grinberg, Miguel. 2018. Flask Web Development: Developing Web Applications with Python. O'Reilly Media.

Hetland, Magnus Lie. 2017. Beginning Python: From Novice to Professional. 3rd edition, Apress.

Sedgewick, Robert, dan Kevin Wayne. 2015. Introduction to Programming in Python: An Interdisciplinary Approach. Addison-Wesley.