Penerapan Lambda, Map, Filter dan Reduce Dalam Modelkan Neighborhood Samples Dengan Model Populasi Polisi Detroit Pada 911 *Services*

Ahmad Sahidin Akbar[1], Nathanael Daniel Santoso[2], Safitri[3], Ferdy Kevin Naibaho[4], Elilya Octaviani[5]

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

Email: ahmad.122450044@student.itera.ac.id, nathanael.122450059@student.itera.ac.id, safitri.122450071@student.itera.ac.id, ferdy.122450107@student.itera.ac.id, elilya.122450009@student.itera.ac.id

ABSTRAK

Penerapan Lambda, Map, Filter dan Reduce Dalam Modelkan Neighborhood Samples dengan Model Populasi Polisi Detroit Pada 911 *Services* ini mengkaji penggunaan lambda, peta, filter, dan fungsi reduksi dalam pemodelan pengambilan sampel lingkungan menggunakan data populasi Departemen Kepolisian Detroit untuk layanan 911. Tujuan penelitiannya adalah untuk mengevaluasi potensi teknik pemrograman fungsional ini dalam menganalisis dan mengekstraksi wawasan dari alokasi sumber daya polisi yang kompleks dan waktu responnya. Studi ini menggunakan kumpulan data yang berisi informasi lingkungan, kantor polisi, dan panggilan darurat untuk menunjukkan bagaimana teknik pemrograman fungsional seperti fungsi lambda, peta, filter, dan pengurangan dapat digunakan untuk manipulasi data yang efisien, pengenalan pola, dan representasi ringkas ini sangat efektif untuk pemrosesan dan analisis. Informasi. Hasil penelitian ini dapat digunakan untuk meningkatkan alokasi sumber daya polisi, mengurangi waktu response, dan meningkatkan keselamatan publik secara keseluruhan.

Kata kunci : *call services 911***,** detroit, filter, fungsi lambda

PENDAHULUAN

Dalam kehidupan, banyak sekali masalah-masalah yang muncul salah satunya adalah banyaknya ancaman yang berasal dari luar. Hal ini menjadi keresahan di dalam kehidupan masyarakat. Solusi yang dilakukan atau ditawarkan salah satu negara di dunia ini dengan menyediakan layanan 911 yang memiliki peran penting seperti bantuan yang cepat ketika mengalami situasi darurat.

Dengan ini, banyak masyarakat yang memanfaatkan layanan 911 sehingga banyak data tentang informasi terkait permasalahan yang ada. Hal ini dapat dianalisis dan didapatkan informasi tentang pola dan jenis permasalahan yang sering terjadi dalam panggilan 911 tersebut karena data tersebut memuat banyak sekali informasi. Banyaknya informasi tersebut, dapat diketahui dengan analisa untuk mendapatkan informasi terkait faktor-faktor yang mempengaruhi ancaman terhadap panggilan 911 dengan menganalisis menggunakan pemrograman yang memudahkan dalam menganalisis dan memperoleh wawasan tentang bagaimana layanan ini merespon dalam permasalahan yang ada.

METODE

Dalam kasus ini kita menggunakan beberapa metode. Metode yang digunakan dalam kasus ini adalah:

1. Fungsi Lambda

   Ekspresi lambda, juga dikenal sebagai fungsi anonim, adalah sebuah fungsi tanpa nama. Ekspresi lambda digunakan untuk membuat fungsi kecil yang hanya terdiri dari satu baris kode. Salah satu karakteristik dari ekspresi lambda adalah kemampuannya untuk mengembalikan nilai. Untuk informasi lebih lanjut mengenai penggunaan fungsi di Python, silakan baca panduan cara menggunakan fungsi di Python [1].

2. Fungsi Map

   Fungsi map() dalam Python adalah fungsi bawaan yang menerapkan fungsi tertentu ke elemen apa pun yang dapat diubah (daftar, tupel, atau string) dan mengembalikan daftar hasil. Ini adalah cara untuk memproses setiap elemen dalam sebuah iterable tanpa menggunakan loop [2].

3. Fungsi Filter

   Fungsi Filter() digunakan untuk menyaring data berdasarkan inisial nama kolom. Fungsi ini memungkinkan pengguna untuk mengembalikan rentang data yang memenuhi tinggi atau lebar yang sama seperti larik sumber. Fungsi filter() dalam Python mengambil fungsi dan daftar sebagai argumen, memberikan cara yang elegan untuk menyaring elemen dari urutan "order" yang mengembalikan nilai true [3].

4. Fungsi Reduce

   Dalam pemrograman Python, fungsi reduce() adalah bagian dari modul functools dan digunakan untuk menerapkan fungsi tertentu secara berulang ke sekumpulan elemen dengan mengumpulkan hasilnya. Fungsi ini membutuhkan dua argumen: fungsi yang akan diterapkan dan urutan elemen yang akan diproses [4].

HASIL DAN PEMBAHASAN

1. Mengimpor Modul

   Mengimpor dua modul `csv` untuk bekerja dengan file CSV, dan `reduce` dari `functools` untuk melakukan operasi akumulasi pada data.

```
import csv
from functools import reduce
```

2. Mengimpor Modul

   Membuka file CSV dan membaca isinya ke dalam list data. Setiap baris dalam file CSV akan disimpan sebagai dictionary (kamus) di dalam list tersebut.

```
data = []
with
open('/content/911_Calls_for_Servi
ce_(Last_30_Days).csv',  'r')  as
file:
    reader = csv.DictReader(file)
    for row in reader:
        data.append(row)
```

3. Mengimpor Modul

   Menyaring data untuk hanya menyertakan baris yang memiliki zip_code dan neighborhood yang tidak kosong.

```
filtered_data = list(filter(lambda
x:  x['zip_code']  !=  ''  and
x['neighborhood'] != '', data))
```

4. Mengubah Data ke Format Numerik

   Membuat list baru numeric_data dimana kita mengkonversi nilai waktu dari

string ke float (angka desimal). Jika ada nilai yang kosong, kita menggantinya dengan 0.

```
numeric_data = [{'total_response_time': float(row['totalresponsetime']) if row['totalresponsetime'] != '' else 0, 'dispatch_time': float(row['dispatchtime']) if row['dispatchtime'] != '' else 0, 'total_time': float(row['totaltime']) if row['totaltime'] != '' else 0} for row in filtered_data]
```

5. Menghitung Total Waktu

Menggunakan reduce untuk menjumlahkan total_response_time, dispatch_time, dan total_time dari semua baris yang telah disaring.

```
total_response_time = reduce(lambda x, y: x + y['total_response_time'], numeric_data, 0)
total_dispatch_time = reduce(lambda x, y: x + y['dispatch_time'], numeric_data, 0)
total_total_time = reduce(lambda x, y: x + y['total_time'], numeric_data, 0)
```

6. Menghitung Rata-Rata Waktu

Menghitung rata-rata dari masing-masing waktu dengan membagi total waktu dengan jumlah baris yang ada di numeric_data.

```
average_response_time = total_response_time / len(numeric_data)
average_dispatch_time = total_dispatch_time / len(numeric_data)
average_total_time = total_total_time / len(numeric_data)
```

7. Mencetak Hasil

Mencetak rata-rata waktu respons, waktu pengiriman, dan waktu keseluruhan.

```
print("Total rata-rata waktu respons:", average_response_time)
print("Total rata-rata waktu pengiriman:", average_dispatch_time)
print("Total rata-rata waktu keseluruhan:", average_total_time)
```

Output :

Total rata-rata waktu respons: 9.363020243313782
Total rata-rata waktu pengiriman: 5.8998965670098
Total rata-rata waktu keseluruhan: 30.55000246269012

8. Mengimpor Fungsi

Mengimpor fungsi reduce dari modul functools untuk membantu menghitung total waktu.

```
from functools import reduce
```

9. Mengelompokkan Data Berdasarkan Lingkungan:

Membuat dictionary bernama neighborhood_data. Untuk setiap baris data

yang sudah difilter (filtered_data), kita memeriksa apakah lingkungan (neighborhood) sudah ada di dictionary. Jika belum, kita tambahkan entri baru. Kemudian, kita tambahkan data baris tersebut ke list di dalam dictionary sesuai dengan lingkungannya.

```
neighborhood_data = {}
for row in filtered_data:
    if row['neighborhood'] not in neighborhood_data:

neighborhood_data[row['neighborhood']] = []

neighborhood_data[row['neighborhood']].append(row)
```

10. Menghitung Statistik untuk Setiap Lingkungan:

    Membuat list `neighborhood_stats` untuk menyimpan hasil rata-rata waktu untuk setiap lingkungan. Untuk setiap lingkungan, nilai waktu dikonversi dari string ke float dan nilai kosong diabaikan. Selanjutnya, total waktu respons, total waktu pengiriman, dan total waktu keseluruhan dihitung menggunakan `reduce`. Dari total ini, rata-rata waktu dihitung dengan membagi total waktu dengan jumlah insiden.

```
neighborhood_stats = []
for neighborhood, incidents in neighborhood_data.items():
                numeric_data = [{'total_response_time': float(row['totalresponsetime']) if row['totalresponsetime'] != '' else 0,

'dispatch_time': float(row['dispatchtime']) if row['dispatchtime'] != '' else 0,
                    'total_time': float(row['totaltime']) if row['totaltime'] != '' else 0} for row in incidents]

        total_response_time = reduce(lambda x, y: x + y['total_response_time'], numeric_data, 0)
        total_dispatch_time = reduce(lambda x, y: x + y['dispatch_time'], numeric_data, 0)
            total_total_time = reduce(lambda x, y: x + y['total_time'], numeric_data, 0)

        average_response_time = total_response_time / len(numeric_data)
        average_dispatch_time = total_dispatch_time / len(numeric_data)
            average_total_time = total_total_time / len(numeric_data)

    neighborhood_stats.append({
                    'neighborhood': neighborhood,
        'average_response_time': average_response_time,
        'average_dispatch_time': average_dispatch_time,
            'average_total_time': average_total_time
    })
```

11. Menghitung Statistik untuk Seluruh Data:

Melakukan langkah yang sama seperti di atas untuk seluruh data tanpa mengelompokkannya berdasarkan lingkungan, lalu menambahkan hasilnya ke neighborhood_stats.

```
total_numeric_data = [{'total_response_time': float(row['totalresponsetime']) if row['totalresponsetime'] != '' else 0,

'dispatch_time': float(row['dispatchtime']) if row['dispatchtime'] != '' else 0,

'total_time': float(row['totaltime']) if row['totaltime'] != '' else 0} for row in filtered_data]

total_response_time_detroit = reduce(lambda x, y: x + y['total_response_time'], total_numeric_data, 0)
total_dispatch_time_detroit = reduce(lambda x, y: x + y['dispatch_time'], total_numeric_data, 0)
total_total_time_detroit = reduce(lambda x, y: x + y['total_time'], total_numeric_data, 0)

average_response_time_detroit = total_response_time_detroit / len(total_numeric_data)
average_dispatch_time_detroit = total_dispatch_time_detroit / len(total_numeric_data)
average_total_time_detroit = total_total_time_detroit / len(total_numeric_data)

for stats in neighborhood_stats:
    print("Lingkungan:", stats['neighborhood'])
    print("Rata-rata waktu respon :", stats['average_response_time'])
    print("Rata-rata waktu pengiriman:", stats['average_dispatch_time'])
    print("Total waktu dalam rata-rata:", stats['average_total_time'])
```

12. Mencetak Hasil:

Mencetak nama lingkungan bersama dengan rata-rata waktu respons, waktu pengiriman, dan waktu keseluruhan.

```
for stats in neighborhood_stats:
    print("Neighborhood:", stats['neighborhood'])
    print("Average Response Time:", stats['average_response_time'])
    print("Average Dispatch Time:", stats['average_dispatch_time'])
    print("Average Total Time:", stats['average_total_time'])
```

Output :

Lingkungan: Eden Gardens
Rata-rata waktu respon :
11.254545454545452
Rata-rata waktu pengiriman:
6.736363636363637

Total waktu dalam rata-rata:
31.356363636363646
Lingkungan: Buffalo Charles
Rata-rata waktu respon :
17.135593220338983
Rata-rata waktu pengiriman:
12.427118644067797
Total waktu dalam rata-rata:
44.515254237288154
Lingkungan: New Center
Rata-rata waktu respon :
0.9734597156398103
Rata-rata waktu pengiriman:
0.22369668246445498
Total waktu dalam rata-rata:
15.604739336492898
Lingkungan: O'Hair Park
Rata-rata waktu respon :
18.500000000000004
Rata-rata waktu pengiriman:
12.912359550561796
Total waktu dalam rata-rata:
52.36516853932585
Lingkungan: Miller Grove
Rata-rata waktu respon :
10.01186440677966
Rata-rata waktu pengiriman:
6.672881355932204
Total waktu dalam rata-rata:
37.33728813559322
Lingkungan: Elmwood Park
Rata-rata waktu respon :
13.889130434782611
Rata-rata waktu pengiriman:
9.4054347826087
Total waktu dalam rata-rata:
49.891304347826086
Lingkungan: Downtown
Rata-rata waktu respon :
3.107888249794576
Rata-rata waktu pengiriman:
1.7829087921117508
Total waktu dalam rata-rata:
18.664667214461804
Lingkungan: Warrendale
Rata-rata waktu respon :
12.74895287958115
Rata-rata waktu pengiriman:
8.182722513089002
Total waktu dalam rata-rata:
37.204712041884804
Lingkungan: Palmer Park
Rata-rata waktu respon : 14.978125
Rata-rata waktu pengiriman: 8.7921875
Total waktu dalam rata-rata: 35.309375
Lingkungan: Brightmoor
Rata-rata waktu respon :
14.378832116788313
Rata-rata waktu pengiriman:
9.95547445255474
Total waktu dalam rata-rata:
39.54124087591238
Lingkungan: McDougall-Hunt
Rata-rata waktu respon :
7.815447154471545
Rata-rata waktu pengiriman:
3.7447154471544715
Total waktu dalam rata-rata:
26.758536585365874
Lingkungan: Detroit Golf
Rata-rata waktu respon : 22.25
Rata-rata waktu pengiriman:
13.530000000000001
Total waktu dalam rata-rata: 52.3
Lingkungan: Central Southwest
Rata-rata waktu respon :
6.269834710743804
Rata-rata waktu pengiriman:
2.985950413223141

Total waktu dalam rata-rata:
23.757851239669417
Lingkungan: Eastern Market
Rata-rata waktu respon :
1.448048048048048
Rata-rata waktu pengiriman:
0.9900900900900897
Total waktu dalam rata-rata:
16.716516516516513
Lingkungan: Davison
Rata-rata waktu respon :
2.3440677966101693
Rata-rata waktu pengiriman:
0.8546610169491528
Total waktu dalam rata-rata:
14.03177966101695
Lingkungan: Schulze
Rata-rata waktu respon :
15.81396648044693
Rata-rata waktu pengiriman:
12.640782122905025
Total waktu dalam rata-rata:
39.29832402234635
Lingkungan: Grand River-I96
Rata-rata waktu respon :
5.617187499999999
Rata-rata waktu pengiriman:
3.7433593749999994
Total waktu dalam rata-rata:
23.33164062500001
Lingkungan: Oakman Blvd Community
Rata-rata waktu respon :
13.979245283018864
Rata-rata waktu pengiriman:
8.751698113207544
Total waktu dalam rata-rata:
39.44981132075475
Lingkungan: Boynton
Rata-rata waktu respon : 8.83533834586466
Rata-rata waktu pengiriman:
4.640601503759399
Total waktu dalam rata-rata:
33.26165413533835
Lingkungan: Farwell
Rata-rata waktu respon :
3.265432098765432
Rata-rata waktu pengiriman:
1.7462962962962962
Total waktu dalam rata-rata:
20.158641975308637
Lingkungan: Cornerstone Village
Rata-rata waktu respon :
16.01569767441861
Rata-rata waktu pengiriman:
10.6953488372093
Total waktu dalam rata-rata:
46.24709302325582
Lingkungan: Bethune Community
Rata-rata waktu respon :
17.669965870307173
Rata-rata waktu pengiriman:
11.650170648464162
Total waktu dalam rata-rata:
42.51092150170651
Lingkungan: Russell Woods
Rata-rata waktu respon :
10.474074074074073
Rata-rata waktu pengiriman:
6.88148148148148
Total waktu dalam rata-rata:
22.361111111111114
Lingkungan: Gratiot-Findlay
Rata-rata waktu respon :
1.3758957654723127
Rata-rata waktu pengiriman:
0.6003257328990228
Total waktu dalam rata-rata:
11.95472312703583
Lingkungan: Gratiot Town/Ketterring

Rata-rata waktu respon :
16.392537313432836
Rata-rata waktu pengiriman:
11.035820895522388
Total waktu dalam rata-rata:
54.48358208955223
Lingkungan: Grandmont #1
Rata-rata waktu respon :
15.163157894736841
Rata-rata waktu pengiriman:
8.410526315789474
Total waktu dalam rata-rata:
31.194736842105264
Lingkungan: Greenfield
Rata-rata waktu respon :
14.114864864864865
Rata-rata waktu pengiriman:
8.052027027027027
Total waktu dalam rata-rata:
35.87027027027027

13. Menyimpan Data Statistik Lingkungan ke dalam File JSON:

    Mengimpor modul json untuk bekerja dengan data JSON dan menentukan nama file output sebagai neighborhood_stats.json. Selanjutnya, kode membuka file JSON dalam mode penulisan dan menggunakan json.dump untuk menulis data neighborhood_stats ke file tersebut dengan indentasi 4 spasi agar formatnya lebih mudah dibaca. Setelah file berhasil dibuat, kode mencetak pesan keberhasilan yang menginformasikan bahwa file JSON telah berhasil dibuat.

```
import json

output_file = 'neighborhood_stats.json'
with open(output_file, 'w') as json_file:
    json.dump(neighborhood_stats, json_file, indent=4)

print(f"File JSON '{output_file}' telah berhasil dibuat.")
```

Output :
File JSON 'neighborhood_stats.json' telah berhasil dibuat.

KESIMPULAN

Analisis ini dilakukan untuk mengevaluasi kinerja waktu respons layanan darurat berdasarkan data panggilan darurat selama 30 hari terakhir di Detroit. Data diambil dari file CSV yang mencakup berbagai metrik waktu dan kemudian dianalisis berdasarkan lingkungan (neighborhood).

Ada variasi signifikan dalam efisiensi layanan darurat berdasarkan lingkungan di Detroit. Beberapa lingkungan menunjukkan kinerja yang sangat baik, sementara yang lain menghadapi tantangan signifikan. Data ini bisa digunakan oleh otoritas terkait untuk mengidentifikasi area yang memerlukan perbaikan dan alokasi sumber daya yang lebih baik untuk meningkatkan efisiensi layanan darurat.

DAFTAR PUSTAKA

[1] Hidayat, A. (2007). Studi Bahasa Pemrograman Haskell sebagai Bahasa Pemrograman Fungsional.

[2] Nugroho, P. A., Fenriana, I., & Arijanto, R. (2020). Implementasi deep learning menggunakan convolutional neural network (CNN) pada ekspresi manusia. *Algor*, *2*(1), 12-20.

[3] Sarosa, M., Nailul Muna, S. S. T., Mila Kusumawardani, S. T., Suyono, A., Azis, I. Y. M., & MPd, S. (2022). *Pemrograman Python Dalam Contoh dan Penerapan*. Media Nusa Creative (MNC Publishing).

[4] Fahrudin, T. M., & S ST, M. T. (2023). *Algoritma dan Pemrograman Dasar dalam Bahasa Pemrograman Python*. Tholabul Ilmi Publishing & Education.