Analisis Kinerja Polisi Detroit dengan Paradigma Pemrograman Berbasis Fungsi

Cyntia Kristina Sidauruk (122450023)[1], Patricia Leondrea Diajeng Putri (122450050)[2], Berliana Enda Putri (122450065)[3] , Feryadi Yulius(122450087)[4], Naufal Fakhri (122450089)[5]

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

Email:[1]cyntia.122450023@student.itera.ac.id, [2]patricia.122450050@student.itera.ac.id, [3]berliana.122450065@student.itera.ac.id, [4]feryadi.122450087@student.itera.ac.id, [5]naufal.122450089@student.itera.ac.id

ABSTRAK

Penelitian ini akan menganalisis kinerja waktu respons, kedatangan, dan total waktu yang dihabiskan oleh kepolisian Detroit menggunakan dataset populasi polisi Detroit yang dapat diakses dari *website City of Detroit Open Data Portal*. Dengan tujuan mengevaluasi kecepatan respons polisi, metode Higher Order Functions (HOF) seperti Reduce, Filter, dan Map, serta paradigma pemrograman berbasis fungsi lainnya. Hasilnya akan menunjukkan bahwa penggunaan fungsi lambda dan reduce efektif dalam menghitung total waktu response rata-rata, waktu pengiriman rata-rata, dan total waktu rata-rata, yang kemudian disimpan dalam dictionary dan diekspor sebagai file JSON. Penelitian ini memberikan wawasan penting untuk meningkatkan efisiensi dan efektivitas layanan kepolisian di Detroit.

Kata kunci: *Dictionary, Higher Order Functions, Polisi Detroit*

PENDAHULUAN

Di era modern yang ditandai dengan kemajuan teknologi informasi, data menjadi salah satu faktor penting dalam menilai kinerja lembaga publik termasuk kepolisian. Laporan ini memanfaatkan data set yang mencatat layanan panggilan 911 pada Kota Detroit sejak tahun 2016. Dataset ini mencakup berbagai panggilan darurat dan inisiatif polisi, termasuk data waktu response, waktu pengiriman, serta total waktu yang diperlukan untuk setiap insiden untuk ditangani oleh polisi Detroit. Dataset yang digunakan dapat diakses dari website *City of Detroit Open Data Portal* yang memberikan informasi rinci mengenai respons darurat polisi dan panggilan telepon yang dilakukan polisi.

Pada tanggal 20 September 2016, dataset ini telah mengumpulkan lebih dari 6 juta catatan, yang artinya menyediakan sumber data yang kaya untuk dianalisis. Dengan menggunakan model pemrograman berbasis fungsi, laporan ini bertujuan untuk memproses data ini secara efisien guna mendapatkan wawasan yang lebih luas mengenai kinerja Departemen Kepolisian Detroit. Analisis ini akan mencakup penghitungan total waktu respons rata-rata, waktu pengiriman rata-rata, dan total waktu rata-rata, baik secara keseluruhan maupun berdasarkan lingkungan. Pendekatan pemrograman berbasis fungsi akan memungkinkan pemrosesan data yang lebih modular, mudah dipahami, dan dapat diulang, memastikan bahwa hasil analisis dapat diandalkan dan berguna untuk pengambilan keputusan strategis.

METODE DASAR

Map

Fungsi map( ) merupakan fungsi bawaan dari Python yang biasanya dikenal sebagai teknik pemetaan. Penggunaan fungsi ini adalah ketika ingin memproses ataupun mengubah semua item dalam iterable tanpa menggunakan fungsi for loop eksplisit dengan item yang dikirim ke fungsi digunakan sebagai parameter (Ramos, 2020).

Fungsi map( ) biasanya digunakan ketika perlu menerapkan fungsi transformasi (fungsi bawaan, kelas, metode, fungsi lambda, dan fungsi yang ditentukan pengguna) ke setiap item dalam sebuah iterable apapun seperti list, tuple, set dan lainnya kemudian menerapkan fungsi yang diberikan ke iterable tersebut dan membuatnya menjadi iterable baru. Setelah menerapkan fungsi ke semua elemen, maka fungsi map( ) akan mengembalikan objek yang digunakan.

Karena fungsi map( ) ditulis dalam C sehingga fungsi ini sangat dioptimalkan karena perulangan tersirat yang dimilikinya lebih efisien daripada for loop Python yang biasa. Selain itu, penggunaan map( ) pengguna mendapatkan item sesuai dengan permintaan dan hanya satu item tersebut yang terdapat di memori sistem sehingga tidak menghabiskan terlalu banyak memori sistem pada waktu tertentu.

Filter

Fungsi filter( ) dapat menghasilkan iterator baru ketika diterapkan pada iterable dan fungsi ini merupakan fungsi bawaan dari Python. Iterator yang dihasilkan akan secara efektif menyaring elemen tertentu dengan menentukan kriteria (Vadapallim, 2024)

Fungsi filter ( ) ini beroperasi pada objek yang dapat diubah seperti daftar, tuple,
dan lain sebagainya. Secara efektif, dengan menerapkan kondisi tertentu atau mengekstrasi elemen tertentu seperti fungsi dan iterable, maka akan menghasilkan iterator secara eksklusif mencakup elemen yang memenuhi kriteria khusus tersebut.

Dengan digunakannya sebagai parameter masukan, maka fungsi ini akan bertanggung jawab untuk memfilter elemen individual dalam iterable yang akan menghasilkan iterator.

Reduce

Fungsi reduce( ) pada python digunakan ketika akan mengulangi setiap item dalam daftar ataupun tipe data yang dapat diubah lainnya dengan mengembalikan satu nilai. Fungsi ini merupakan metode kelas functools bawaan Python. Fungsi ini beroperasi pada semua iterable (Nurvinda, 2022).

Dengan mengimplementasikan teknik matematik yang disebut dengan pelipatan atau reduksi, maka ketika pengguna menerapkan fungsi ke suatu iterable kemudian menguranginya menjadi nilai kumulatif tunggal. Fungsi ini digunakan ketika mengambil fungsi yang sudah ada, lalu menerapkannya secara kumulatif ke dalam semua item di iterable yang akhirnya akan menghasilkan satu nilai akhir.

Fungsi ini dapat memproses iterable tanpa for loop eksplisit karena ditulis dalam C sehingga loop internal yang dimilikinya lebih cepat dibanding for loop eksplit tersebut.

Library CSV

Library ini digunakan untuk membaca atau melakukan proses pada data yang memiliki format CSV (Comma Separated Values). Format ini merupakan salah satu format file umum yang dapat diakses untuk menyimpan dan bertukar data tabular menggunakan *spreadsheet* ataupun database (Muhardian, 2019).

Dengan menggunakan library ini, penyimpanan data tabular seperti angka dan teks diubah menjadi teks biasa yang pada setiap baris file nya adalah catatan data. Setiap catatan data tersebut akan terdiri dari satu atau lebih bidang yang dipisahkan dengan tanda baca koma. Pemisahan ini digunakan untuk memisahkan field menjadi sumber nama format file.

JSON

JSON adalah suatu singkatan dari Java Script Objek Notation. Fungsi ini merupakan format file berbasis teks untuk menyimpan dan mengangkut data dan biasanya digunakan ketika data tersebut dikirim dari server ke halaman web (Faradilla, 2022). File dari JSON memiliki format .json serta menggunakan teks yang dapat dibaca oleh pengguna dan dipahami oleh komputer. Dengan format yang dimiliki, JSON dapat menyimpan informasi secara terstruktur serta banyak digunakan sebagai alternatif yang lebih ringan serta lebih sederhana.

JSON banyak digunakan karena penggunaannya yang mudah dipahami, ringan, ringkas, serta menunjukkan bahwa data yang digunakan terstruktur berdasarkan syntax JavaScript. Program JavaScript sendiri mengubah data JSON menjadi objek native tanpa harus melakukan parsing ataupun serializing. Dengan begitu, JSON menjadi populer dengan kelebihan kompatibelnya yang memiliki banyak bahasa pemrograman, environment, serta library.

HASIL DAN PEMBAHASAN

1. **Import Library**

```
Python
import csvsv
import json
from functools import reduce
```

Kode ini mengimport csv,json, reduce untuk mendefinisikan dan membaca file csv dan json kemudian kita bisa membuka file csv dan menulisnya ke json sehingga kita bisa menghitung total nilai kolom menggunakan import reduce

2. **Import dataset**

```
Python
file_path = '/content/911_Calls_for_Service_(Last_30_Days).csv'

with open(file_path, mode='r', encoding='utf-8-sig') as file:
    csv_reader = csv.DictReader(file)
    data = [row for row in csv_reader]
```

file_path Kode ini digunakan untuk membaca file csv yang kita ingin gunakan. selanjutnya dengan menggunakan csv .DictReader(file) kita melakukan konversi dari setiap baris menjadi sebuah dictionary dengan nama kolom sebagai key atau kunci.kemudian kita menggunakan [row for row in csv_reader] untuk mengumpulkan dari setiap barisnya menjadi satu kedalam data

3. **Bagian 1**

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

def convert_time_to_float(time_str):
    try:
        return float(time_str)
    except ValueError:
        return 0.0

total_response_time = reduce(lambda acc, x: acc + convert_time_to_float(x['totalresponsetime']),
filtered_data, 0.0)
```

```
total_dispatch_time = reduce(lambda acc, x: acc + convert_time_to_float(x['dispatchtime']), filtered_data,
0.0)
total_time = reduce(lambda acc, x: acc + convert_time_to_float(x['totaltime']), filtered_data, 0.0)

avg_response_time = total_response_time / len(filtered_data)
avg_dispatch_time = total_dispatch_time / len(filtered_data)
avg_total_time = total_time / len(filtered_data)
```

Kode di atas kita menggunakan fungsi filter dan lambda untuk melakukan penyaringan data kedalam kode zip_code dan neighborhood agar kita bisa membuang baris data kosong.

Selanjutnya melakukan konversi data ke dalam total_response_time dan total_dispatch_time kemudian kita ubah menjadi tipe float convert_time_to_float untuk mengatasi kasus konvensi gagal kita menggunakan fungsi total_response_time, total_dispatch_time, dan total_time kita hitung dengan menjumlahkan nilai konversi setiap entri. Setelah itu kita menghitung rata-rata dari total waktu (avg_total_time) dengan cara membagi total waktu(total_time) dengan banyaknya data(len(filtered_data)

4. Bagian 2

```
Python
neighborhoods = set(row['neighborhood'] for row in filtered_data)

neighborhood_samples = []

for neighborhood in neighborhoods:
    neighborhood_data = list(filter(lambda x: x['neighborhood'] == neighborhood, filtered_data))

    total_response_time_neigh = reduce(lambda acc, x: acc + convert_time_to_float(x['totalresponsetime']),
neighborhood_data, 0.0)
    total_dispatch_time_neigh = reduce(lambda acc, x: acc + convert_time_to_float(x['dispatchtime']),
neighborhood_data, 0.0)
    total_time_neigh = reduce(lambda acc, x: acc + convert_time_to_float(x['totaltime']),
neighborhood_data, 0.0)

    avg_response_time_neigh = total_response_time_neigh / len(neighborhood_data)
    avg_dispatch_time_neigh = total_dispatch_time_neigh / len(neighborhood_data)
    avg_total_time_neigh = total_time_neigh / len(neighborhood_data)

    neighborhood_samples.append({
        'neighborhood': neighborhood,
        'AvgResponseTime': avg_response_time_neigh,
```

```
        'AvgDispatchTime': avg_dispatch_time_neigh,
        'AvgTotalTime': avg_total_time_neigh
    })

detroit_data = {
    'neighborhood': 'All Detroit',
    'AvgResponseTime': avg_response_time,
    'AvgDispatchTime': avg_dispatch_time,
    'AvgTotalTime': avg_total_time
}

neighborhood_samples.append(detroit_data)
```

Kode tersebut pertama-tama kita melakukan identifikasi semua data neighborhood selanjutnya kita lakukan filter dengan menggunakan set(row['neighborhood'] for row in filtered_data). Kemudian kita lakukan juga menyaring data untuk mendapatkan baris list yang sesuai dengan kriteria di neighborhood menggunakan (filter(lambda x: x['neighborhood'] == neighborhood, filtered_data)).

Setelah kita mendapatkan data yang sesuai dengan neighborhood, kita menghitung totalresponsetime,dispatchtime, totaltime dengan menggunakan fungsi lambda dan kita ubah menggunakan fungsi reduce menjadi tipe float (convert_time_to_float).

Selanjutnya kita menghitung rata-rata untuk masing-masing waktu total_response_time_neigh,total_dispatch_time_neigh,total_time_neigh dengan membagi total dari waktu dengan banyaknya data yang terdapat pada neighborhood len(neighborhood_data) kemudian disimpan ke dalam dictionary baru bernama neigborhood_samples.

5. Bagian 3

```
Python
output_file = '/content/detroit_police_report.json'

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

print(f'Output written to {output_file}')
```

Setelah kita mendapatkan dictionary baru berupa neighborhood sampel rata-rata 'total response time' dan 'dispatch time' dan total_time setiap neighborhood kita melakukan penyimpanan data menjadi sebuah file json.

KESIMPULAN

Berdasarkan hasil dan pembahasan dalam jurnal praktik ini, kami menyimpulkan bahwa penggunaan teknik HOF (high order function) seperti reduksi, filtering, dan pemetaan dalam menilai kecepatan respon polisi sangat efektif dan efisien. HOF memungkinkan Anda melakukan analisis dengan lebih cepat dan akurat, menghitung rata-rata respons keseluruhan dan waktu rata-rata yang dikelompokkan berdasarkan wilayah, sehingga Anda dapat melihat perbedaan kinerja polisi antar wilayah. Menerapkan perpustakaan seperti CSV dan JSON memudahkan pemrosesan dan penyimpanan data dalam format terstruktur dan dapat diakses. Secara khusus, penggunaan format JSON menawarkan fleksibilitas dalam penyimpanan dan pertukaran data serta mendukung interoperabilitas dengan berbagai platform dan bahasa pemrograman. Secara keseluruhan, penggunaan HOF dalam analisis kinerja kepolisian tidak hanya meningkatkan efisiensi dan efektivitas, namun juga membuka peluang untuk lebih mengembangkan analisis data di sektor publik.

REFERENSI

Faradilla. 2022. *Apa Itu JSON?* Hostinger.

Muhardian, A. 2019. *File CSV Python*. Petani Kode.

Nurvinda, K. 2022. *Mengenal Fungsi Python untuk Memperpendek Coding*. DQLab

Ramos, L.P. 2020. *Peta Python() : Memproses Iterable Tanpa Perulangan*. Real Python.

Vadapallim P. 2024. *Fungsi Filter( ) dengan Python*. upGrad.