Penggunaan Fungsi Lambda Map, Reduce, dan Filter untuk Analisis Data 911 di Detroit dengan Python

Tessa Kania Sagala[1], Renisha Putri Giani[2], Rian Bintang Wijaya[3], Gymnastiar Al Khoarizmy[4], Virdio Samuel Saragih[5]

Jurusan Sains Data, Fakultas Sains, Institut Teknologi Sumatera, Lampung Selatan, Indonesia

Email: tessa.122450040@student.itera.ac.id, renisha.122450079@student.itera.ac.id, rian.122450094@student.itera.ac.id, gymnastiar.122450096@student.itera.ac.id, virdio.122450124@student.itera.ac.id

1. Pendahuluan

High Order Function atau fungsi tingkat tinggi adalah fungsi yang menggunakan satu atau lebih fungsi sebagai argumen, atau mengembalikan fungsi sebagai hasilnya. Ada beberapa jenis fungsi tingkat tinggi seperti map, reduce dan filter.

Dalam dunia komputasi, kita akan sering menemui kumpulan data yang besar dan kompleks. Memprosesnya secara manual untuk mendapatkan informasi akan sangat memakan waktu dan akan menjadi sulit. Maka dari itu, pada artikel ini kami akan membahas mengenai penggunaan fungsi lambda, map, reduce, dan filter untuk analisis data 911 di Detroit, yang dapat memberikan cara yang efisien untuk memproses dan menganalisis data yang besar sehingga memungkinkan peneliti membuat keputusan yang lebih baik berdasarkan data yang lebih akurat dan terorganisir. Data set yang diperoleh dari website City of Detroit Open Data Portal yang memungkinkan untuk mendapatkan informasi mengenai respons darurat polisi dan panggilan telepon yang dilakukan polisi. Analisis yang dilakukan pada artikel ini berupa penghitungan total waktu respons rata-rata, waktu pengiriman rata-rata, dan total waktu rata-rata, baik secara keseluruhan maupun berdasarkan lingkungan. Tujuan dari artikel ini untuk mendapatkan pemahaman yang mendalam mengenai penggunaan High Order Function untuk analisis data dengan menggunakan fungsi lambda, map, reduce, dan filter.

2. Metode

Dalam pengerjaan analisis Penggunaan Fungsi Lambda Map, Reduce, dan Filter untuk Analisis Data 911 di Detroit dengan Python ini, kami menggunakan beberapa fungsi yang esensial untuk mencapai tujuan kami:

2.1. Fungsi Map

Fungsi map adalah salah satu fungsi built-in di Python yang digunakan untuk menerapkan fungsi tertentu ke setiap item dari iterasi dan mengembalikan iterator dengan hasil yang diinginkan. 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).

Cara kerja dari fungsi Map sendiri adalah mengiterasi setiap elemen dari iterable yang diberikan, kemudian, fungsi yang diberikan diterapkan untuk menghasilkan nilai baru, setelah itu nilai nilai yang dihasilkan oleh fungsi dikumpulkan dalam iterasi baru yang berupa objek ‘Map’ selanjutnya hasil akhir dapat dikonversikan menjadi list, tuple atau jenis iterasi yang lain. Dengan pemahaman yang baik tentang cara kerjanya, ‘Map’ dapat digunakan untuk membuat kode lebih efisien dan lebih bersih.

2.2. Fungsi Reduce

Fungsi reduce adalah alat yang sangat berguna dalam pemrograman untuk menerapkan operasi secara kumulatif ke elemen-elemen dalam iterable (seperti list atau tuple) sehingga elemen-elemen tersebut dapat dikurangi menjadi satu nilai tunggal. Fungsi ini adalah bagian dari modul functools di Python, jadi perlu mengimpornya sebelum digunakan. Fungsi ini dapat membuat kode dengan lebih efisien dan ringkas.

2.3 Fungsi Filter

Fungsi filter dalam Python merupakan alat yang digunakan untuk menyaring elemen-elemen dari sebuah iterasi berdasarkan sebuah kondisi atau kriteria tertentu. Fungsi Filter mengembalikan sebuah iterasi baru yang berisi elemen-elemen yang memenuhi kondisi tertentu. 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).

2.4 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). JSON adalah format teks yang sepenuhnya bahasa independen tetapi menggunakann bahasa yang lebih akbrab untuk programmer yaitu C++, Java, JavaScript, Python dan lain lain. Struktur data utama dari JSON adalah objek dan array. JSON mendukung beberapa tipe data yaitu string, number, objek, array, boolean dan juga nilai null.

JSON adalah format yang sangat penting dan banyak digunakan untuk pertukaran data dalam aplikasi web dan API. Dengan format yang sederhana dan fleksibel, JSON memudahkan pengembang untuk mengirim dan menerima data dengan cepat dan efisien. Dukungan luas di berbagai bahasa pemrograman membuat JSON menjadi pilihan utama untuk serialisasi dan deserialisasi data dalam banyak skenario pemrograman.

2.5 Library Pandas

Pandas merupakan library open-source untuk bahasa pemrograman Python yang menyediakan struktur data dan alat analisis data yang berkinerja tinggi dan mudah digunakan. Pandas digunakan untuk memanipulasi dan analisis data. Dengan berbagai fitur dan alat yang disediakan, Pandas memungkinkan Anda untuk bekerja dengan data secara efisien dan efektif, dari pembersihan dan transformasi data hingga analisis dan visualisasi data. Pandas menjadi alat yang sangat berguna bagi ilmuwan data, analis bisnis, dan banyak profesi lainnya yang bekerja dengan data.

3. Pembahasan

Pada artikel ini, kami menggunakan dataset yang terdiri dari data sepal length, sepal width, petal length, dan petal width dengan satuan cm. Untuk memvisualisasikan data pada dataset tersebut, kami menggabungkan beberapa fungsi, termasuk plot line dan plot scatter, serta memanfaatkan dua library utama, yaitu Pandas dan Scikit-learn. Library Pandas digunakan untuk membuat DataFrame kosong dan mengisi kolom-kolom DataFrame dengan nilai-nilai yang diimpor dari library Scikit-learn.

3.1. Menggunakan Pandas untuk Pengolahan Data

```
import pandas as pd
import json
from functools import reduce
import numpy as np

# import dataset
df = pd.read_csv('/content/911_Calls_for_Service_(Last_30_Days).csv')
# cleaning data
df = df.dropna(subset=['zip_code', 'neighborhood'])

# mengubah dataset ke dalam bentuk list dictionary
data = df.to_dict('records')
```

Langkah pertama dalam proses analisis ini adalah mengimpor dataset menggunakan pandas, sebuah library yang sangat kuat untuk manipulasi dan analisis data. Kami memulai dengan membaca file CSV yang berisi data panggilan 911. Selanjutnya, kami membersihkan dataset dengan menghapus baris yang memiliki nilai kosong di kolom zip_code dan neighborhood, memastikan bahwa analisis kami hanya menggunakan data yang lengkap dan valid. Data yang telah dibersihkan kemudian diubah menjadi list dictionary menggunakan metode to_dict('records'). Transformasi ini memungkinkan kita untuk dengan mudah memanipulasi dan mengakses data dalam bentuk yang lebih fleksibel untuk langkah-langkah analisis selanjutnya.

3.2. Model Populasi Polisi Detroit

```
list_dict_output = [x for x in data if x["zip_code"] and x["neighborhood"] and
x["totalresponsetime"] and x["time_on_scene"] and x["totaltime"]]

total_response_time = sum(x.get("totalresponsetime", 0) for x in list_dict_output) /
len(list_dict_output)
delivery_time = sum(int(x.get("time_on_scene", 0)) if not
np.isnan(x.get("time_on_scene", 0)) else 0 for x in list_dict_output) /
len(list_dict_output)
```

```
total_time = sum(int(x.get("totaltime", 0)) if not np.isnan(x.get("totaltime", 0)) else
0 for x in list_dict_output) / len(list_dict_output)

print("Total Response Time Average: ", total_response_time)
print("Delivery Time Average: ", delivery_time)
print("Total Time Average: ", total_time)
```

Setelah data dibersihkan dan diubah menjadi list dictionary, langkah berikutnya adalah menghitung rata-rata waktu respons, waktu pengiriman, dan total waktu untuk kepolisian Detroit. Pertama, kami menggunakan filter dengan fungsi lambda untuk menyaring data yang valid, memastikan hanya baris yang memiliki nilai pada kolom totalresponsetime, time_on_scene, dan totaltime yang diproses. Dengan menggunakan kombinasi fungsi lambda, map, dan reduce, kami menghitung rata-rata dari masing-masing waktu tersebut. map digunakan untuk mengekstrak nilai dari list dictionary, dan reduce untuk menjumlahkan nilai-nilai tersebut sebelum dibagi dengan jumlah entri yang valid untuk mendapatkan rata-rata. Hasil perhitungan ini memberikan gambaran umum tentang kinerja kepolisian Detroit dalam merespons panggilan darurat, termasuk total waktu respons rata-rata, waktu pengiriman rata-rata, dan total waktu rata-rata. Hasil dari kode output tersebut sebagai berikut:

```
Total Response Time Average:  8.306335351073573
Delivery Time Average:  31.406902350747632
Total Time Average:  54.51698168391036
```

3.3. Modelkan Neighborhood Samples

```
neighborhoods = list(set(map(lambda x: x["neighborhood"], data)))
dictNeighbor = {}
for neighborhood in neighborhoods:
        dictNeighbor[neighborhood] = list(filter(lambda x: x["neighborhood"] ==
neighborhood, data))

for neighborhood in dictNeighbor:
    dictNeighbor[neighborhood] = {
                "total_response_time": reduce(lambda x, y: x + y,map(lambda x:
x.get("totalresponsetime", 0),dictNeighbor[neighborhood]))
        / len(dictNeighbor[neighborhood]),
                    "delivery_time": reduce(lambda x, y: x + y,map(lambda x:
int(x.get("time_on_scene", 0)) if not np.isnan(x.get("time_on_scene", 0)) else
0,dictNeighbor[neighborhood]))
        / len(dictNeighbor[neighborhood]),
          "total_time": reduce(lambda x, y: x + y,map(lambda x: int(x.get("totaltime",
0)) if not np.isnan(x.get("totaltime", 0)) else 0,dictNeighbor[neighborhood]))
        / len(dictNeighbor[neighborhood]),
    }

dictNeighbor["All Detroit"] = {
    "total_response_time": total_response_time,
    "delivery_time": delivery_time,
    "total_time": total_time,
}

dictNeighbor["All Detroit"]
```

Setelah menghitung waktu rata-rata untuk keseluruhan Detroit, analisis dilanjutkan dengan pemisahan data berdasarkan neighborhood. Langkah pertama adalah mengidentifikasi semua neighborhood yang unik dalam dataset dengan menggunakan set dan map. Setelah itu, data dikelompokkan berdasarkan neighborhood dengan menggunakan fungsi filter untuk setiap kelompok. Untuk setiap neighborhood, kami menghitung rata-rata total waktu respons, waktu pengiriman, dan total waktu menggunakan kombinasi fungsi lambda, map, dan reduce. Proses ini mirip dengan perhitungan sebelumnya untuk keseluruhan Detroit, tetapi dilakukan untuk setiap kelompok neighborhood secara terpisah. Hasilnya adalah dictionary yang berisi waktu rata-rata untuk setiap neighborhood. Kami juga menambahkan entri khusus untuk keseluruhan Detroit, yang mencakup total waktu respons rata-rata, waktu pengiriman rata-rata, dan total waktu rata-rata, memberikan gambaran yang lengkap tentang kinerja kepolisian di berbagai neighborhood di Detroit.

3.4. Buat File Output JSON

```
import json

# Menyimpan hasil ke file JSON
with open("DataPopulasiDetroit.json", "w") as outfile:
    json.dump(list(neighborhood_dict.values()), outfile)
```

Langkah terakhir dalam analisis ini adalah menyimpan hasil perhitungan ke dalam file JSON. Hal ini dilakukan untuk memastikan data hasil analisis dapat diakses dan digunakan untuk analisis lebih lanjut atau pelaporan. Menggunakan modul json, kami mengubah dictionary yang berisi hasil perhitungan menjadi format JSON. File "detroit_population_data.json" yang dihasilkan berisi list dictionary yang mencakup waktu rata-rata untuk setiap neighborhood serta keseluruhan Detroit. Penyimpanan dalam format JSON memungkinkan data ini untuk mudah dibagikan dan diintegrasikan dengan sistem lain, serta digunakan untuk visualisasi atau analisis lanjutan. Format JSON yang serbaguna dan mudah dibaca manusia membuatnya ideal untuk menyimpan dan mentransfer data hasil analisis ini.

4. Kesimpulan

Dalam tugas praktikum ini, kami berhasil menerapkan pemrograman berbasis fungsi untuk menganalisis data panggilan darurat 911 di Detroit. Dengan menggunakan package `pandas`, kami memulai dengan mengimpor dan membersihkan dataset, menghapus entri yang tidak lengkap untuk memastikan keakuratan analisis. Data yang telah dibersihkan kemudian diubah menjadi list dictionary untuk memudahkan pemrosesan lebih lanjut. Penggunaan kombinasi fungsi `lambda`, `map`, dan `reduce` memungkinkan kami menghitung rata-rata total waktu respons, waktu pengiriman, dan total waktu untuk kepolisian Detroit, memberikan wawasan penting tentang kinerja respon kepolisian secara keseluruhan.

Selanjutnya, kami memisahkan data berdasarkan `neighborhood` dan menghitung waktu rata-rata yang sama untuk setiap kelompok. Ini memberikan gambaran lebih rinci tentang kinerja kepolisian di berbagai wilayah di Detroit, memungkinkan identifikasi area-area yang mungkin memerlukan perhatian lebih. Dengan mengelompokkan dan menganalisis data berdasarkan `neighborhood`, kami dapat melihat variasi dalam kinerja respon yang mungkin terkait dengan karakteristik lokal atau distribusi sumber daya. Hasil analisis ini menunjukkan perbedaan signifikan antara berbagai wilayah, yang dapat digunakan untuk perencanaan strategis oleh pihak berwenang.

Akhirnya, hasil analisis disimpan dalam file JSON menggunakan modul `json`, memastikan data dapat diakses untuk analisis lebih lanjut atau pelaporan. File ini mencakup data rata-rata waktu untuk setiap `neighborhood` serta keseluruhan Detroit, yang dapat digunakan oleh pihak berwenang untuk meningkatkan efisiensi respons darurat. Secara keseluruhan, tugas ini menunjukkan bagaimana pemrograman berbasis fungsi dapat diterapkan secara efektif untuk mengolah dan menganalisis data dalam skala besar. Teknik yang digunakan tidak hanya memudahkan proses analisis tetapi juga memastikan hasil yang akurat dan dapat diandalkan, memberikan wawasan yang bermanfaat untuk meningkatkan layanan darurat di Detroit.

5. Referensi

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

Santoso, D. J. T. (2020). *Analisis Big Data*. Yayasan Prima Agus Teknik.

Syaferi, F. (2023, November 2). *Memahami Konsep Functional Programming di Python*. Asia Career Blog. Retrieved May 16, 2024, from https://blog.unmaha.ac.id/memahami-konsep-functional-programming-di-python