Analisis Data dalam Pengelolaan Layanan Panggilan Darurat 911: Studi Kasus pada Populasi Detroit

Khaalishah Zuhrah Alyaa Vanefi[1], Vita Anggraini[2], Tobias David Manogari[3], Ratu Keisha Jasmine Deanova[4], Bastian Heskia Silaban[5].

Program Studi Sains Data, Fakultas Sains

Institut Teknologi Sumatera, , Jl. Terusan Ryacudu, Way Huwi, Kecamatan Jati Agung, Kabupaten Lampung Selatan, Lampung 35365, Indonesia.

1. Pendahuluan

Pada era digitalisasi yang terus mengalami perkembangan serta pembaruan, analisis data menjadi salah satu alat yang sangat penting guna melakukan pengolahan data. Analisis data merupakan proses sistematis untuk menguraikan, menginterpretasikan, dan mengolah data agar dapat diambil kesimpulan yang berharga. Data merupakan komponen utama dari sistem informasi perusahaan karena semua informasi untuk pengambilan keputusan berasal dari data. Oleh karena itu sudah sewajarnya jika Analisis data dipandang sebagai kebutuhan primer oleh perusahaan. Analisis data yang buruk dapat mengakibatkan tidak tersedianya data penting yang digunakan untuk menghasilkan informasi yang diperlukan dalam pengambilan keputusan. Untuk mengolah data menjadi bentuk yang lebih bermanfaat dibutuhkan analisis yang baik dan tajam. Analisis data merupakan metode yang digunakan untuk mengetahui bagaimana menggambarkan data, hubungan data, semantik data dan batasan data yang ada pada suatu sistem informasi. [1]

Dalam kehidupan tentu mempunyai banyak permasalahan yang muncul, salah satunya adalah banyaknya ancaman yang datang di luar. Hal ini menjadi permasalahan dalam kehidupan masyarakat. Salah satu negara di dunia menggunakan layanan 911 untuk memberikan solusi penting menangani permasalahan pada masyarakat, bentuk dari solusi tersebut seperti memberikan bantuan cepat dalam situasi darurat. Oleh karena itu, banyak orang yang menggunakan layanan 911, hal ini membuat 911 memiliki banyak data tentang masalah terkait. Karena datanya banyak mengandung informasi, maka dapat dianalisis dan diperoleh informasi tentang pola dan jenis masalah yang sering terjadi pada panggilan 911 dengan menggunakan pemrograman, sehingga memudahkan dalam menganalisis dan melakukan pengolahan data.

2. **Metode**

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

1. Fungsi filter() menyaring data berdasarkan nama kolom inisial. Fungsi ini memungkinkan pengguna untuk mengembalikan rentang data dengan lebar atau tinggi yang sama dengan larik sumber. Dalam python, fungsi filter() mengambil daftar dan fungsi sebagai argumen, memberikan metode yang luar biasa untuk menyaring elemen dari urutan "order" yang mengembalikan nilai benar. [2]
2. Fungsi Reduce
   Termasuk dalam modul functools dalam pemrograman python dan digunakan untuk menerapkan fungsi tertentu secara berulang ke sekumpulan elemen dengan mengumpulkan hasilnya. Fungsi ini memerlukan dua argumen: fungsi yang akan diterapkan dan urutan elemen yang akan diproses. [3]
3. Fungsi Lambda:
   Ekspresi lambda atau fungsi anonim adalah sebuah fungsi tanpa identitas. Salah satu fitur ekspresi lambda adalah kemampuan untuk mengembalikan nilai, ini digunakan untuk membuat fungsi kecil yang terdiri dari satu baris kode. Baca panduan cara menggunakan fungsi di python untuk informasi lebih lanjut tentang penggunaan fungsi di python. [4]
4. Fungsi Map
   Fungsi bawaan yang menerapkan fungsi tertentu ke elemen apapun yang dapat diubah (string, daftar, atau tupel) dan mengembalikan daftar hasil. Metode ini memungkinkan pemrosesan setiap elemen secara iterable tanpa menggunakan loop. [5]

3. **Pembahasan**

3.1 Import Data Set

```
import pandas as pd

path_dataset = "/content/911_Calls_for_Service_(Last_30_Days) (1).csv"
df = pd.read_csv(path_dataset)
df
```

Mengimpor data dengan file csv dengan menggunakan library pandas dan membaca data set dengan kode df = pd.read_csv()

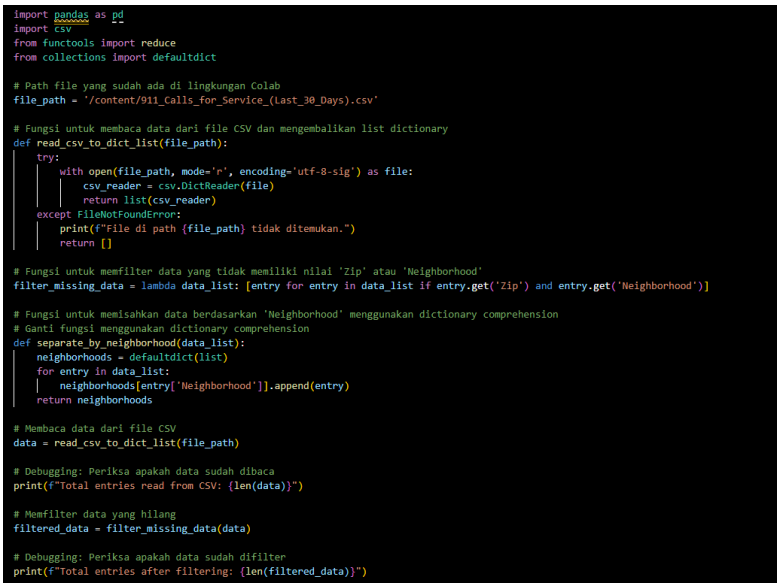
3.2 Memodelkan Populasi Detroit

```python
import csv
from collections import defaultdict

# Path file yang sudah ada di lingkungan Colab
file_path = '/content/911_Calls_for_Service_(Last_30_Days).csv'

# Fungsi untuk membaca data dari file CSV dan mengembalikan list dictionary
def read_csv_to_dict_list(file_path):
    try:
        with open(file_path, mode='r', encoding='utf-8-sig') as file:
            csv_reader = csv.DictReader(file)
            return list(csv_reader)
    except FileNotFoundError:
        print(f"File di path {file_path} tidak ditemukan.")
        return []

# Fungsi untuk memfilter data yang tidak memiliki nilai 'Zip' atau 'Neighborhood'
filter_missing_data = lambda data_list: [entry for entry in data_list if entry.get('Zip') and entry.get('Neighborhood')]

# Fungsi untuk memisahkan data berdasarkan 'Neighborhood' menggunakan dictionary comprehension
def separate_by_neighborhood(data_list):
    neighborhoods = defaultdict(list)
    for entry in data_list:
        neighborhoods[entry['Neighborhood']].append(entry)
    return neighborhoods

# Membaca data dari file CSV
data = read_csv_to_dict_list(file_path)
print(f"Total entri yang dibaca dari CSV: {len(data)}")

# Memfilter data yang hilang
filtered_data = filter_missing_data(data)

# Debugging: Periksa apakah data sudah difilter
print(f"Total entri setelah difilter: {len(filtered_data)}")

# Memisahkan data berdasarkan 'Neighborhood'
neighborhood_data = separate_by_neighborhood(filtered_data)
print(f"Total neighborhood unik: {len(neighborhood_data)}")

# Tampilkan data yang sudah dipisahkan berdasarkan 'Neighborhood'
for neighborhood, data_list in neighborhood_data.items():
    print(f"Neighborhood: {neighborhood}")
    for entry in data_list:
        print(entry)
```

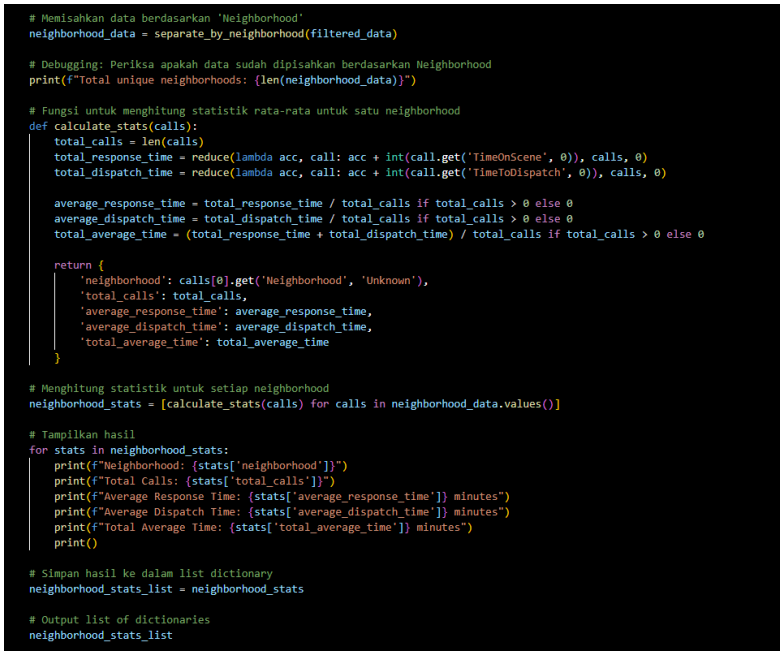
Sebelum memodelkan populasi detroit hal yang pertama dilakukan yaitu mengimpor file csv yang sudah di dilakukan pada langkah sebelumnya dengan menggunakan import csv, lalu membaca file dari lokasi data dan mengembalikan ke dalam daftar dictionary dengan menggunakan kode csv.DictReader supaya dapat membaca data set perbarisnya. Setelah membaca dataset lalu membuat fungsi dimana fungsi tersebut akan menyaring baris pada dataset yang tidak memiliki nilai kolom dengan menggunakan filter dan fungsi filter akan memeriksa setiap bari dan hanya memasukkan baris yang memiliki nilai untuk kedua kolom, kolom yang dimaksud yaitu zip dan neighborhood. lalu pada lokasi_dataset berfungsi untuk mendefinisikan path ke file dataset csv. Setelah itu menggunakan data_filter untuk mensaring data yang tidak lengkap dan disimpan dalam data_terlfilter dan menampilkan data yang sudah tersaring. Selanjutnya melakukan pemodelan data, pertama melakukan import reduce agar dapat menghitung statistik. Fungsi def hitung_statistik_polisi_detroit yaitu sebagai menerima satu parameter. Lalu menggunakan filerter dengan lambda x dimana x merupakan nilai agecny yang mana nilai agency merupakan detroit police sehingga nilai tersebut akan masuk kedalam panggilan detroit. Setelah itu menginisiasi total panggilan,waktu respon dan total waktu pengiriman dengan nilai awal adalah 0. Lalu menggunakan pengulangan dengan menggunakan loop untuk setiap panggilan_detroit , ini digunakan untuk mengumpulkan statistik, dimana total panggilan+= 1 artinya menambah jumlah panggilan dengan 1 untuk setiap iterasi, total_waktu_respon += int() untuk menambah total waktu respon

dengan nilai dari kolom timeonscene dan total_waktu_pengiriman += int() untuk menambah waktu pengiriman dengan nilai kolom time todispatch

Lalu nilai akan dihitung dengan menggunakan rata_rata_waktu_respon = total_waktu_respon dibagi total_panggilan, jika total panggilan > 0 maka hasilnya total rata-rata waktu, dan jika sebaliknya, maka hasilnya 0



```python
import pandas as pd
import csv
from functools import reduce
from collections import defaultdict

# Path file yang sudah ada di lingkungan Colab
file_path = '/content/911_Calls_for_Service_(Last_30_Days).csv'

# Fungsi untuk membaca data dari file CSV dan mengembalikan list dictionary
def read_csv_to_dict_list(file_path):
    try:
        with open(file_path, mode='r', encoding='utf-8-sig') as file:
            csv_reader = csv.DictReader(file)
            return list(csv_reader)
    except FileNotFoundError:
        print(f"File di path {file_path} tidak ditemukan.")
        return []

# Fungsi untuk memfilter data yang tidak memiliki nilai 'Zip' atau 'Neighborhood'
filter_missing_data = lambda data_list: [entry for entry in data_list if entry.get('Zip') and entry.get('Neighborhood')]

# Fungsi untuk memisahkan data berdasarkan 'Neighborhood' menggunakan dictionary comprehension
# Ganti fungsi menggunakan dictionary comprehension
def separate_by_neighborhood(data_list):
    neighborhoods = defaultdict(list)
    for entry in data_list:
        neighborhoods[entry['Neighborhood']].append(entry)
    return neighborhoods

# Membaca data dari file CSV
data = read_csv_to_dict_list(file_path)

# Debugging: Periksa apakah data sudah dibaca
print(f"Total entries read from CSV: {len(data)}")

# Memfilter data yang hilang
filtered_data = filter_missing_data(data)

# Debugging: Periksa apakah data sudah difilter
print(f"Total entries after filtering: {len(filtered_data)}")
```



```python
# Memisahkan data berdasarkan 'Neighborhood'
neighborhood_data = separate_by_neighborhood(filtered_data)

# Debugging: Periksa apakah data sudah dipisahkan berdasarkan Neighborhood
print(f"Total unique neighborhoods: {len(neighborhood_data)}")

# Fungsi untuk menghitung statistik rata-rata untuk satu neighborhood
def calculate_stats(calls):
    total_calls = len(calls)
    total_response_time = reduce(lambda acc, call: acc + int(call.get('TimeOnScene', 0)), calls, 0)
    total_dispatch_time = reduce(lambda acc, call: acc + int(call.get('TimeToDispatch', 0)), calls, 0)

    average_response_time = total_response_time / total_calls if total_calls > 0 else 0
    average_dispatch_time = total_dispatch_time / total_calls if total_calls > 0 else 0
    total_average_time = (total_response_time + total_dispatch_time) / total_calls if total_calls > 0 else 0

    return {
        'neighborhood': calls[0].get('Neighborhood', 'Unknown'),
        'total_calls': total_calls,
        'average_response_time': average_response_time,
        'average_dispatch_time': average_dispatch_time,
        'total_average_time': total_average_time
    }

# Menghitung statistik untuk setiap neighborhood
neighborhood_stats = [calculate_stats(calls) for calls in neighborhood_data.values()]

# Tampilkan hasil
for stats in neighborhood_stats:
    print(f"Neighborhood: {stats['neighborhood']}")
    print(f"Total Calls: {stats['total_calls']}")
    print(f"Average Response Time: {stats['average_response_time']} minutes")
    print(f"Average Dispatch Time: {stats['average_dispatch_time']} minutes")
    print(f"Total Average Time: {stats['total_average_time']} minutes")
    print()

# Simpan hasil ke dalam list dictionary
neighborhood_stats_list = neighborhood_stats

# Output list of dictionaries
neighborhood_stats_list
```

Program ini membaca data panggilan darurat dari file CSV, memfilter data yang tidak lengkap, memisahkan data berdasarkan 'Neighborhood', dan menghitung statistik rata-rata panggilan untuk setiap 'Neighborhood'. Dengan menggunakan fungsi seperate_by_neigborhood untuk memisahkan data berdasarkan neigborhood

menggunakan defaultdict supaya setiap entry di data_list dimasukkan kedalam daftar yang sesuai dengan neigborhoodnya lalu menghitung statistik rata- rata untuk satu neighborhood dan menghitung statistik untuk setiap neighborhood dengan menggunakan calculate_stats. Hasilnya adalah daftar dictionary yang berisi statistik .

3. 3 Membuat File Output JSON

```python
import pandas as pd
import json

# Fungsi untuk menyaring data dengan format zip atau neighborhood tidak kosong
def filter_data(df):
    try:
        filtered_df = df[(df['Zip'].notnull()) & (df['Neighborhood'].notnull())]
        return filtered_df
    except KeyError as e:
        print(f"Error: {e}. Pastikan kolom 'Zip' dan 'Neighborhood' ada dalam DataFrame.")
        return pd.DataFrame()  # Mengembalikan DataFrame kosong jika terjadi kesalahan

# Path atau lokasi dataset
file_path = '/content/911_Calls_for_Service_(Last_30_Days) (1).csv'

# Membaca data dari file CSV
try:
    df = pd.read_csv(file_path)
except FileNotFoundError:
    print(f"Error: File di path {file_path} tidak ditemukan.")
    df = pd.DataFrame()  # Mengembalikan DataFrame kosong jika file tidak ditemukan

# Menyaring data dengan format zip atau neighborhood tidak kosong
filtered_df = filter_data(df)

# Mengonversi DataFrame ke list of dictionaries
data_list = filtered_df.to_dict(orient='records')

# Mengonversi list of dictionaries ke format JSON
data_json = json.dumps(data_list, indent=4)

# Menampilkan hasil dalam format JSON
print(data_json)

# Menyimpan hasil dalam file JSON
output_file_path = '/content/911_Calls_for_Service_filtered.json'
with open(output_file_path, 'w') as f:
    f.write(data_json)

print(f'Data telah disimpan dalam file: {output_file_path}')
```

Program ini membaca data panggilan darurat dari file CSV, menyaring data yang tidak lengkap (dengan 'Zip' atau 'Neighborhood' yang kosong), mengonversi data yang telah disaring menjadi format JSON, dan menyimpan hasilnya ke dalam file JSON. Hasil akhir dapat digunakan untuk analisis lebih lanjut atau integrasi dengan aplikasi lain yang memerlukan format data JSON

4. Kesimpulan

Dalam era digitalisasi yang terus berkembang, analisis data menjadi kunci penting dalam pengolahan informasi perusahaan. Data merupakan elemen utama dalam sistem informasi perusahaan, dan analisis data yang baik diperlukan untuk menghasilkan informasi berharga yang mendukung pengambilan keputusan.

Dalam konteks layanan panggilan darurat 911, banyaknya data yang terkumpul memberikan peluang untuk menganalisis pola dan jenis masalah yang sering terjadi. Dengan menggunakan pemrograman dan berbagai metode seperti filter(), reduce, lambda, dan map, data panggilan darurat dapat diolah untuk menghasilkan informasi yang berguna.

Metode yang digunakan mencakup penyaringan data, perhitungan statistik, dan pemisahan data berdasarkan kategori tertentu seperti daerah (neighborhood). Hasil analisis dapat digunakan untuk pemodelan populasi, perhitungan statistik rata-rata panggilan darurat untuk setiap daerah, dan konversi data ke format JSON untuk keperluan analisis lebih lanjut atau integrasi dengan aplikasi lain.

Dengan demikian, analisis data tidak hanya menjadi kebutuhan primer bagi perusahaan, tetapi juga alat yang dapat memberikan wawasan berharga dalam penanganan masalah di masyarakat seperti layanan panggilan darurat. Dengan memanfaatkan teknik analisis data yang tepat, informasi yang diperoleh dapat digunakan secara efektif untuk meningkatkan layanan dan pengambilan keputusan.

5. Daftar Pustaka

[1] S. B. Doro Edi, "Analisis Data dengan Menggunakan ERD dan Model Konseptual Data Warehouse," *Jurnal Informatika,* p. 71, 2009.

[2] A. Hidayat, "Studi Bahasa Pemograman Haskell sebagai Bahasa Pemograman Fungsional," 2007.

[3] P. A. F. I. &. A. R. Nugroho, " Implementasi deep learning menggunakan convolutional neural network (CNN) pada ekspresi manusia.," pp. 12-20, 2020.

[4] M. N. M. S. S. T. M. K. S. T. S. A. A. I. Y. M. &. M. S. Sarosa, " Pemrograman Python Dalam Contoh dan Penerapan. Media Nusa Creative (MNC Publishing).," 2022.

[5] T. M. &. S. S. M. T. Fahrudin, "Algoritma dan Pemrograman Dasar dalam Bahasa Pemrograman Python," *Tholabul Ilmi Publishing & Education.,* 2023.