Taktik Terdepan dalam Normalisasi Data: Menggali Potensi Min-Max Scaling untuk Kemajuan Sains Data

Natasya Ega Lina Marbun[1], Khusnun Nisa[2], Presilia[3], Irvan Alfaritzi[4], Syalaisha Andini Putriansyah[5]
Program Studi Sains Data, Fakultas Sains, Institut Teknologi Sumatera.

Email: natasya.122450024@student.itera.ac.id, khusnun.122450078@student.itera.ac.id, presilia.122450081@student.itera.ac.id, irvan.122450093@student.itera.ac.id, syalaisha.122450111@student.itera.ac.id

1. Pendahuluan

Dalam dunia Sains Data, normalisasi data merupakan langkah penting untuk mempersiapkan data agar siap dianalisis dan dimodelkan. Salah satu teknik normalisasi yang populer adalah Min-Max Scaling. Teknik ini memiliki keunggulan dalam kesederhanaan dan kemudahan implementasi, sehingga teknik sering digunakan.

Laporan ini bertujuan untuk menunjukan potensi Min-Max Scaling dalam memajukan Sains Data dengan membahas penerapannya dalam kasus ini, penelitian terdahulu juga telah menunjukkan efektivitas Min-Max Scaling dalam berbagai aplikasi Sains Data, bahwa Min-Max Scaling meningkatkan akurasi prediksi model pembelajaran mesin dalam analisis sentimen media sosial.

Min-Max Scaling mempercepat konvergensi algoritma pelatihan model regresi pada dataset keuangan. Min-Max Scaling mentransformasi nilai data ke dalam rentang tertentu, umumnya antara 0 dan 1. Transformasi ini dilakukan dengan memetakan nilai minimum data ke 0 dan nilai maksimum data ke 1. Proses ini memastikan bahwa semua variabel memiliki skala yang sama, sehingga meningkatkan stabilitas dan akurasi model.

2. Metode

2.1. Scope

Scope (lingkup atau cakupan) mengacu pada bagaimana variabel didefinisikan dan diakses dalam sebuah kode program [1]. Variabel yang didefinisikan di luar fungsi, secara default tidak bisa diakses dari dalam fungsi. Begitu juga sebaliknya, hal tersebut penting karena menentukan kapan dan di mana suatu variabel dapat digunakan. Berikut beberapa jenis scope dalam Python:

2.1.1. Global Scope

Cakupan global di dalam pemrograman Python mengacu pada blok kode tertentu dimana variabel didefinisikan di luar fungsi utama dan dapat diakses dari mana saja dalam program, termasuk di dalam fungsi [2]. Cakupan global ini perlu mendeklarasikan variabel tersebut sebagai global di dalam fungsi tersebut.

2.1.2. Local Scope

Cakupan lokal dalam pemrograman Python mengacu pada blok kode dimana variabel didefinisikan di dalam sebuah fungsi dan hanya dapat diakses dalam blok atau fungsi variabel tersebut dideklarasikan [3]. Jika mencoba mengakses variabel dalam cakupan lokal di luar fungsi maka akan terjadi error pada blok kode tersebut. Pemahaman mengenai cakupan lokal adalah kunci dalam pengembangan Python yang efisien dan terstruktur.

2.1.3. Enclosing Scope

Lingkup penutup dalam pemrograman Python mengacu pada lingkungan di mana sebuah fungsi bersarang (nested function) memiliki akses ke variabel dari lingkup luar fungsi tersebut [4]. Blok kode tersebut dapat berjalan ketika sebuah fungsi didefinisikan di dalam fungsi lain. Lingkup penutup penting dalam Python karena memungkinkan untuk mengatur dan memberikan informasi antara fungsi-fungsi yang bersarang.

2.2. Closure

Closure (penutup) adalah konsep yang berkaitan erat dengan konsep yang bersifat fungsional atau berorientasi objek. Closure dapat terjadi jika sebuah fungsi bersifat nested dan mengakses variabel non-local yang memungkinkan fungsi untuk menyimpan referensi ke variabel yang berada di luar cakupan fungsi tersebut. Hal tersebut menjadikan fungsi tersebut untuk memiliki memori tentang lingkungan di mana fungsi tersebut dibuat [5].

3. Pembahasan

3.1. Kode Program

3.1.1. Impor Modul

```
import pandas as pd
```

Gambar 1. Impor Modul

Langkah pertama dalam membuat program yaitu mengimpor modul. Dalam program normalize data, modul yang digunakan adalah modul *pandas*.

3.1.2. Fungsi normalize_data

```
def normalize_data(data):
    min_val = data.min()
    max_val = data.max()
    return (data - min_val) / (max_val - min_val)
```

Gambar 2. Fungsi normalize data

Pada gambar 2 menggunakan fungsi *normalize_data* yang bertujuan untuk melakukan normalisasi data. Normalisasi data adalah proses mengubah nilai-nilai dalam dataset sehingga mereka memiliki skala yang seragam. Hal ini umumnya dilakukan untuk memudahkan perbandingan antara data dengan rentang nilai yang berbeda-beda. Adapun fungsi dari pemrograman diatas yaitu :

1) Menghitung Nilai Minimum dan Maksimum: Pada baris pertama dan kedua, fungsi mencari nilai minimum dan maksimum dari data menggunakan method ‘.min()’ dan ‘.max()’ dari objek ‘data’. Ini akan memberikan batas bawah dan batas atas dari rentang data.
2) Normalisasi Data: Setelah mendapatkan nilai minimum dan maksimum, langkah selanjutnya adalah menghitung normalisasi setiap nilai dalam data.
3) Mengembalikan Data yang dinormalisasi: Setelah proses normalisasi selesai, hasilnya dikembalikan sebagai output dari fungsi.

3.1.3. Fungsi normalize_column

```
def normalize_column(df, column_name):
    df[column_name] = normalize_data(df[column_name])
    return df[column_name]  # Mengembalikan data kolom yang telah dinormalisasi
```

Gambar 3. Fungsi normalize_column

Gambar 3 digunakan untuk melakukan normalisasi pada satu kolom tertentu dalam DataFrame (‘df’) dengan menggunakan fungsi ‘normalize_data’ yang telah didefinisikan sebelumnya, dan kemudian mengembalikan data dari kolom yang telah dinormalisasi.

Langkah-langkah fungsi ini adalah sebagai berikut:

1) Input Fungsi:
   - ‘df’: DataFrame yang berisi data yang ingin dinormalisasi.
   - ‘column_name’: Nama kolom dalam DataFrame yang ingin dinormalisasi.
2) Proses Normalisasi:

- Fungsi ‘normalize_column’ menggunakan fungsi ‘normalize_data’ yang telah didefinisikan sebelumnya untuk melakukan normalisasi pada kolom yang ditentukan dari DataFrame.
- Normalisasi dilakukan dengan mengakses kolom tertentu dalam DataFrame menggunakan ‘df[column_name]’, kemudian memanggil fungsi ‘normalize_data’ untuk melakukan normalisasi pada kolom tersebut.
- Hasil normalisasi ditetapkan kembali ke kolom yang sama dalam DataFrame.

3.1.4. Fungsi normalize_file

```
def normalize_file(file_path, column_name):
    try:
        if file_path.endswith('.csv'):
            df = pd.read_csv(file_path, encoding='utf-8')
        elif file_path.endswith('.xlsx'):
            df = pd.read_excel(file_path)
        else:
            raise ValueError("Format file tidak didukung. Gunakan file CSV atau XLSX.")

        if column_name not in df.columns:
            raise ValueError("Kolom yang dimaksud tidak ditemukan dalam file.")

        normalized_column = normalize_column(df, column_name)
        return normalized_column
    except Exception as e:
        raise ValueError("Terjadi kesalahan saat membaca file:", e)
```

Gambar 4. Fungsi normalize_file

Pada gambar 4, mendefinisikan fungsi normalize_file yang digunakan untuk membaca file dan melakukan normalisasi pada salah satu kolom data. Fungsi ini menerima dua argumen, yaitu file_path yang merupakan path dari file yang akan dibaca dan column_name yang merupakan nama kolom yang akan dinormalisasi. Fungsi ini juga menggunakan exception handling (try-except), dimana digunakan untuk menangani kesalahan yang mungkin terjadi selama proses pembacaan dan normalisasi file.

Proses normalisasi fungsi ini adalah sebagai berikut:

1. Melakukan pengecekan terhadap file yang diinputkan memiliki format CSV. Jika file memiliki format selain CSV maka, akan menampilkan pesan ValueError.
2. Membaca file dengan menggunakan kode ‘pd.read_csv’.
3. Memeriksa kolom yang yang dimaksud ada dalam dataframe. Jika tidak ditemukan, akan mengembalikkan ValueError. Dan jika ditemukan, fungsi ‘normalize_column’ akan dipanggil untuk melakukan normalisasi yang hasilnya dikembalikkan.

3.1.5. variabel file_path

```
file_path = input("Masukkan path file (CSV atau XLSX): ")
```

Gambar 5. Fungsi file_path

Kode pada gambar 5. Digunakan untuk pengguna memasukkan alamat file bertipe CSV atau XLSX yang akan dinormalisasi.

3.1.6. Variabel column_name

```
column_name = input("Masukkan nama kolom yang akan dinormalisasi: ")
```

Gambar 6. Variabel column_name

Kode pada gambar 6. Digunakan untuk pengguna memasukkan nama kolom yang ingin dinormalisasikan.

3.1.7. Mencetak Hasil Normalisasi

```
try:
    normalized_data = normalize_file(file_path, column_name)
    print("Data kolom setelah dinormalisasi:")
    for value in normalized_data:
        print(value)
except ValueError as ve:
    print(ve)
```

Gambar 7. Fungsi pembaca file

Pada gambar 7. Fungsi mencoba untuk menjalankan proses normalisasi dengan memanggil fungsi normalize_file dan mencetak pesan kesalahan jika terdapat ValueError. Fungsi ini menggunakan *loop* untuk mencetak setiap nilai yang telah dinormalisasikan.

4. Kesimpulan

Melalui program yang telah dibuat, dapat disimpulkan bahwa Min-Max Scaling memiliki potensi besar dalam memajukan sains data dengan mempersiapkan data untuk analisis dan pemodelan. Hal ini menggambarkan pentingnya normalisasi data dan efektivitas Min-Max Scaling dalam memudahkan proses ini. Referensi penelitian terdahulu juga menggarisbawahi manfaat teknik ini dalam meningkatkan akurasi model dan mempercepat konvergensi algoritma pelatihan.

Dari program tersebut, dapat pula diambil adalah bahwa Min-Max Scaling dapat diterapkan dengan mudah melalui kode program untuk normalisasi data dalam konteks sains data. Program ini memungkinkan pengguna untuk dengan cepat dan efisien melakukan normalisasi data dari file CSV atau XLSX berdasarkan kolom yang dipilih. Dengan memasukkan path file dan nama kolom, pengguna dapat dengan mudah melihat hasil normalisasi data dalam bentuk baris, sehingga memudahkan untuk mempersiapkan data untuk analisis lebih lanjut.

Pada akhirnya program tersebut program tersebut menegaskan bahwa Min-Max Scaling merupakan taktik terdepan dalam normalisasi data dalam konteks sains data. Dengan mempergunakan Min-Max Scaling, pengguna dapat menggali potensi teknik ini untuk mempersiapkan data secara efisien untuk analisis dan pemodelan lebih lanjut. Dengan memperhatikan kesederhanaan dan efektivitas Min-Max Scaling, program yang telah di buat ini secara langsung menggarisbawahi pentingnya teknik ini dalam meningkatkan kemajuan sains data.

5. Daftar Pustaka

[1] McKinney, W. (2017). Python for Data Analysis (2nd ed.). O'Reilly Media.

[2] Python Software Foundation. (2020). The Python Language Reference. Python 3.9.2 documentation.

[3] J. Zelle (2003). "Python Programming: An Introduction to Computer Science." 1st ed. Franklin, Beedle & Associates Inc.,

[4] Ramalho, L. (2015). Fluent Python: Clear, Concise, and Effective Programming (1st ed.). O'Reilly Media.

[5] Beazley, D. (2019). Python Essential Reference (4th ed.). Addison-Wesley Professional.