

LAPORAN TUGAS BESAR 2

IF5150 Inteligensi Artifisial Lanjut

Implementasi Supervised Learning dan Unsupervised Learning Pada Dataset Konsumsi Listrik di Perumahan Kota Melbourne

Dipersiapkan oleh:

Kelompok 5

18221060 Auvarifqi Putra Diandra

18221071 Ahmad Rizki

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen	Halaman
		<i>IF5150-TB1-06</i>	<i><jml hlm></i>
<i>Revisi</i>	<i>1</i>	<i>30 Oktober 2024</i>	

Daftar Isi

Daftar Isi.....	2
1. Ringkasan.....	3
2. Penjelasan Kode.....	4
3. Business Understanding.....	6
4. Data Understanding.....	7
4.1 Deskripsi Data.....	7
4.1.1 Memahami Struktur Data.....	7
4.1.2 Menentukan Variabel Target.....	9
4.1.3 Memeriksa Nilai Ringkasan Statistik.....	9
4.2 Eksplorasi Data.....	9
4.2.1 Histogram pada fitur numerik.....	9
4.2.2 Keseimbangan fitur target.....	12
4.2.3 Korelasi antar fitur numerik.....	13
4.3 Verifikasi Kualitas Data.....	14
4.3.1 Identifikasi Missing Values.....	14
4.3.2 Identifikasi Duplikasi Data.....	14
4.3.3 Identifikasi Konsistensi Data.....	15
4.3.4 Identifikasi Format Data.....	15
4.3.5 Identifikasi Anomali Data (Outliers).....	16
5. Data Preparation.....	18
5.1 Pemilihan Data.....	18
5.1.1 Record Selection.....	18
5.1.2 Feature Selection.....	18
5.2 Perbaikan Data.....	18
5.2.1 Mengisi Missing Values.....	18
5.2.2 Penghapusan Duplikasi Data.....	18
5.2.3 Penanganan Outliers.....	18
5.2.4 Perbaikan Error Lainnya.....	18
5.3 Konstruksi Data (Training Set dan Testing Set).....	19
5.3.1 Train-Test Split.....	19
5.3.3 Rekayasa Fitur dengan Clustering.....	19
5.3.4 Transformasi Skala (Normalisasi).....	23
6. Implementasi Model.....	24

6.1 Supervised Learning.....	24
6.1.1 Logistic Regression.....	24
6.1.2 K Nearest Neighbours.....	26
6.2 Unsupervised Learning.....	28
6.2.1 K-Means.....	28
6.2.2 Density-Based Spatial Clustering of Applications with Noise (DBSCAN).....	30
7. Training, Testing, dan Analisis Hasil.....	32
7.1 Perbandingan Model Klasifikasi.....	32
7.2 Pengaruh Ensemble pada Model Klasifikasi.....	33
7.3 Perbandingan Hasil Clustering.....	34
7.3.1 Hasil Clustering dengan DBSCAN.....	34
7.3.2 Hasil Clustering dengan K-Means.....	36
8. Saran dan Perbaikan.....	39
9. Kontribusi Anggota.....	40

1. Ringkasan

Laporan ini mendokumentasikan implementasi supervised learning dan unsupervised learning untuk menganalisis konsumsi listrik di perumahan Kota Melbourne. Dalam proyek ini, dataset konsumsi listrik rumah tangga digunakan untuk memahami pola penggunaan energi, dengan fokus pada pengelompokan rumah berdasarkan pola konsumsi dan prediksi rumah yang memiliki kendaraan listrik (EV). Tren adopsi EV yang meningkat di Australia telah menciptakan tantangan baru bagi penyedia listrik, terutama dalam menangani lonjakan permintaan pada waktu tertentu. Oleh karena itu, laporan ini bertujuan memberikan solusi berbasis machine learning untuk mendeteksi pola konsumsi listrik yang kompleks.

Model supervised yang diterapkan mencakup Logistic Regression, K-Nearest Neighbors (KNN), dan Decision Tree. Ketiga model ini dievaluasi berdasarkan metrik seperti akurasi, F1-score, recall, dan precision untuk berbagai kombinasi teknik preprocessing dan clustering. Clustering digunakan untuk mendukung analisis data dan menghasilkan fitur tambahan. Dua metode clustering utama, yaitu K-Means dan DBSCAN, diterapkan untuk mengelompokkan rumah berdasarkan pola konsumsi listrik mereka. Selain itu, evaluasi dilakukan untuk membandingkan efektivitas pendekatan clustering ini dalam memberikan insight.

Hasil menunjukkan bahwa model Decision Tree dengan clustering K-Means menghasilkan performa terbaik dengan akurasi mencapai 84%, diikuti oleh KNN dengan DBSCAN sebesar 78%. Untuk clustering, K-Means mengidentifikasi enam cluster utama, sementara DBSCAN menambahkan kemampuan menangani noise. Laporan ini memberikan rekomendasi untuk meningkatkan analisis menggunakan metrik evaluasi internal dan eksternal serta optimalisasi parameter model, sehingga hasil clustering dan klasifikasi dapat lebih diandalkan untuk pengelolaan konsumsi energi listrik.

2. Penjelasan Kode

Dalam tugas besar ini, implementasi kode dibagi ke dalam beberapa folder sesuai dengan fungsinya, mencakup preprocessing, visualisasi data, model klasifikasi supervised learning, dan model clustering unsupervised learning. Berikut adalah penjelasan kode berdasarkan struktur folder:

STEI- ITB	<nomor dokumen>	Halaman 4 dari 40 halaman
Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB.		

```

.
├── README.md
├── data
│   └── EV_data.csv
├── main.ipynb
├── requirements.txt
└── utils
    ├── __pycache__
    │   ├── eda_statistics.cpython-310.pyc
    │   ├── eda_visualization.cpython-310.pyc
    │   └── preprocessing.cpython-310.pyc
    ├── classification_models
    │   ├── __pycache__
    │   │   ├── decision_tree.cpython-310.pyc
    │   │   ├── ensemble_voting.cpython-310.pyc
    │   │   ├── k_nearest_neighbor.cpython-310.pyc
    │   │   └── logistic_regression.cpython-310.pyc
    │   ├── decision_tree.py
    │   ├── ensemble_voting.py
    │   ├── k_nearest_neighbor.py
    │   └── logistic_regression.py
    ├── clustering_models
    │   ├── __pycache__
    │   │   ├── dbscan.cpython-310.pyc
    │   │   └── k_means.cpython-310.pyc
    │   ├── dbscan.py
    │   └── k_means.py
    └── eda_statistics.py
        └── eda_visualization.py
            └── preprocessing.py

```

- a. **Preprocessing:** tahapan ini dilakukan melalui file preprocessing.py yang berisi dua modul utama
 - **FeatureScaler:** Melakukan normalisasi data dengan Min-Max Scaling untuk memastikan fitur memiliki rentang nilai seragam.
 - **FeatureDimensionReducer:** Menggunakan PCA untuk mengurangi dimensi data tanpa kehilangan informasi signifikan, yang diperlukan terutama sebelum clustering.
- b. **Exploratory Data Analysis (EDA):** mencakup statistik deskriptif dan visualisasi. Fungsi utama berada di file eda_statistics.py dan eda_visualization.py
 - **Statistik:** Mengecek nilai kosong, nilai negatif, duplikasi data, dan outliers.
 - **Visualisasi:** Menyediakan histogram, boxplot, korelasi antar fitur, dan visualisasi cluster (2D dan 3D).
- c. **Supervised Learning:** Folder classification_models mengimplementasikan algoritma supervised learning
 - **Logistic Regression:** Menggunakan sigmoid dan gradient descent untuk memprediksi hasil biner.
 - **K-Nearest Neighbors (KNN):** Menggunakan jarak Euclidean untuk menentukan tetangga terdekat dan melakukan klasifikasi berdasarkan mayoritas.

- **Decision Tree:** Membagi data ke dalam subset berdasarkan atribut yang memaksimalkan informasi.
 - **Ensemble Voting:** Menggabungkan prediksi beberapa model untuk meningkatkan akurasi, dengan kemampuan menangani lebih dari dua model dan preferensi pada model tertentu jika terjadi voting seri.
- d. **Unsupervised Learning:** Clustering diimplementasikan dalam folder clustering_models, dimana clustering akan digunakan saat *feature engineering*
- **K-Means:** Membagi data menjadi K cluster dengan optimasi posisi centroid. Dilengkapi Elbow Method untuk menentukan jumlah cluster optimal.
 - **DBSCAN:** Mendeteksi cluster berdasarkan densitas data, tanpa memerlukan jumlah cluster sebagai parameter. Cocok untuk data kompleks dengan outliers.

3. Business Understanding

Pada Tugas Besar 2 mata kuliah AI Lanjut, kami akan menggunakan dataset tentang konsumsi listrik pada Perumahan di Melbourne. Dataset ini dilatarbelakangi oleh tren kendaraan listrik (*Electric Vehicle/EV*) yang telah melanda Australia, dengan 8,4% dari semua mobil baru yang terjual di Australia kini adalah EV, meningkat 120% dibandingkan tahun 2022. Meskipun perubahan ini sangat baik bagi lingkungan, ada tantangan baru yang muncul bagi penyedia listrik. EV membutuhkan energi listrik dalam jumlah besar untuk pengisian daya, yang sebenarnya tidak menjadi masalah jika hanya satu kendaraan. Namun, jika dalam satu blok atau wilayah terdapat banyak pengguna EV yang pulang kerja pada waktu yang sama dan mengisi daya kendaraan mereka secara bersamaan, ini dapat memberikan beban besar pada jaringan listrik di wilayah tersebut, bahkan berpotensi membuat sistem kewalahan. Oleh karena itu, kemampuan untuk mendeteksi rumah mana yang memiliki EV dan mengidentifikasi kelompok rumah seperti itu dalam suatu blok menjadi sangat penting. Inilah insight yang akan kami coba selesaikan menggunakan pembelajaran mesin atau *machine learning*.

4. Data Understanding

4.1 Deskripsi Data

4.1.1 Memahami Struktur Data

Berikut adalah overview utama dari dataset yang digunakan. Terdapat 50 fitur atau kolom berbeda yang akan kami gunakan. Adapun jumlah barisnya sebanyak 4048 baris. Pada fitur-fitur di dataset terdapat 1 tipe data object (read_date), 2 tipe int, dan sisanya (47) bertipe float. Berikut adalah kode beserta hasilnya.

STEI- ITB	<nomor dokumen>	Halaman 7 dari 40 halaman
Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB.		

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4048 entries, 0 to 4047
Data columns (total 51 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   read_date   4048 non-null    object  
 1   interval_1  4048 non-null    float64 
 2   interval_2  4048 non-null    float64 
 3   interval_3  4048 non-null    float64 
 4   interval_4  4048 non-null    float64 
 5   interval_5  4048 non-null    float64 
 6   interval_6  4048 non-null    float64 
 7   interval_7  4048 non-null    float64 
 8   interval_8  4048 non-null    float64 
 9   interval_9  4048 non-null    float64 
 10  interval_10 4048 non-null    float64 
 11  interval_11 4048 non-null    float64 
 12  interval_12 4048 non-null    float64 
 13  interval_13 4048 non-null    float64 
 14  interval_14 4048 non-null    float64 
 15  interval_15 4048 non-null    float64 
 16  interval_16 4048 non-null    float64 
 17  interval_17 4048 non-null    float64 
 18  interval_18 4048 non-null    float64 
 19  interval_19 4048 non-null    float64 
 20  interval_20 4048 non-null    float64 
 21  interval_21 4048 non-null    float64 
 22  interval_22 4048 non-null    float64 
 23  interval_23 4048 non-null    float64 
 24  interval_24 4048 non-null    float64 
 25  interval_25 4048 non-null    float64 
 26  interval_26 4048 non-null    float64 
 27  interval_27 4048 non-null    float64 
 28  interval_28 4048 non-null    float64 
 29  interval_29 4048 non-null    float64 
 30  interval_30 4048 non-null    float64 
 31  interval_31 4048 non-null    float64 
 32  interval_32 4048 non-null    float64 
 33  interval_33 4048 non-null    float64 
 34  interval_34 4048 non-null    float64 
 35  interval_35 4048 non-null    float64 
 36  interval_36 4048 non-null    float64 
 37  interval_37 4048 non-null    float64 
 38  interval_38 4048 non-null    float64 
 39  interval_39 4048 non-null    float64 
 40  interval_40 4048 non-null    float64 
 41  interval_41 4048 non-null    float64 
 42  interval_42 4048 non-null    float64 
 43  interval_43 4048 non-null    float64 
 44  interval_44 4048 non-null    float64 
 45  interval_45 4048 non-null    float64 
 46  interval_46 4048 non-null    float64 
 47  interval_47 4048 non-null    float64 
 48  interval_48 4048 non-null    float64 
 49  id          4048 non-null    int64  
 50  label        4048 non-null    int64  
dtypes: float64(48), int64(2), object(1)
```

4.1.2 Menentukan Variabel Target

Variabel target pada dataset yang kami gunakan adalah fitur “label”. Dimana pada fitur ini terdapat 2 values, yaitu 0 dan 1.



```
numerical_columns = initial_df.select_dtypes(include=[np.float64]).columns
categorical_columns = ['id']
date_columns = ['read_date']
label_column = ['label']
```

4.1.3 Memeriksa Nilai Ringkasan Statistik

Nilai ringkasan statistik dari dataset kami lakukan dengan menggunakan function describe dari pandas. Berikut adalah screenshot dari hasilnya.

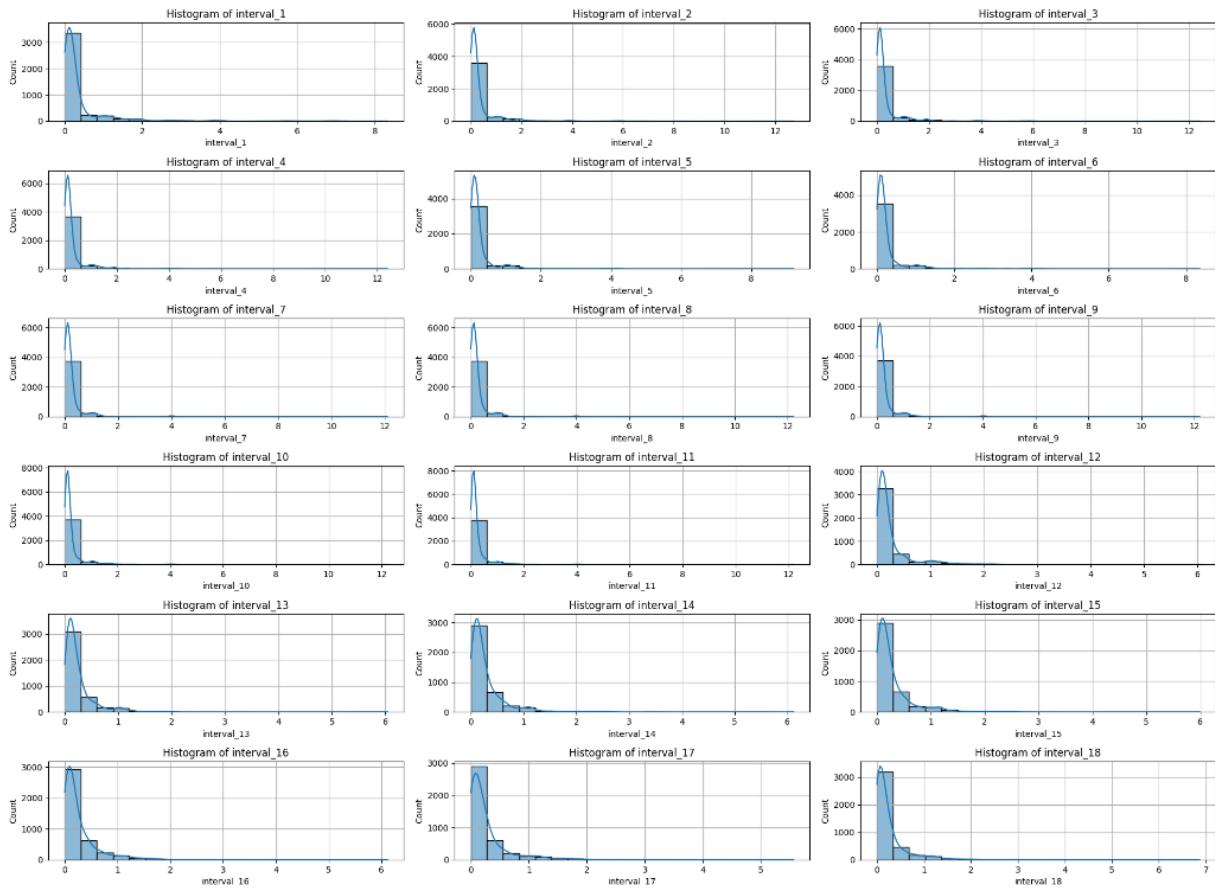
	interval_1	interval_2	interval_3	interval_4	interval_5	interval_6	interval_7	interval_8	interval_9
count	4048.000000	4048.000000	4048.000000	4048.000000	4048.000000	4048.000000	4048.000000	4048.000000	4048.000000
mean	0.344301	0.320546	0.297221	0.276360	0.255266	0.244258	0.250251	0.251163	0.255017
std	0.698534	0.655992	0.607303	0.554286	0.492384	0.461366	0.598976	0.606699	0.616976
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.073000	0.072000	0.070000	0.069000	0.066000	0.066000	0.064000	0.066000	0.064000
50%	0.137500	0.131200	0.125000	0.125000	0.125000	0.118700	0.118700	0.118700	0.118700
75%	0.250000	0.237500	0.225000	0.218700	0.212500	0.208250	0.200000	0.200000	0.206200
max	8.325000	12.712500	12.400000	12.387500	9.200000	8.350000	12.100000	12.212500	12.200000

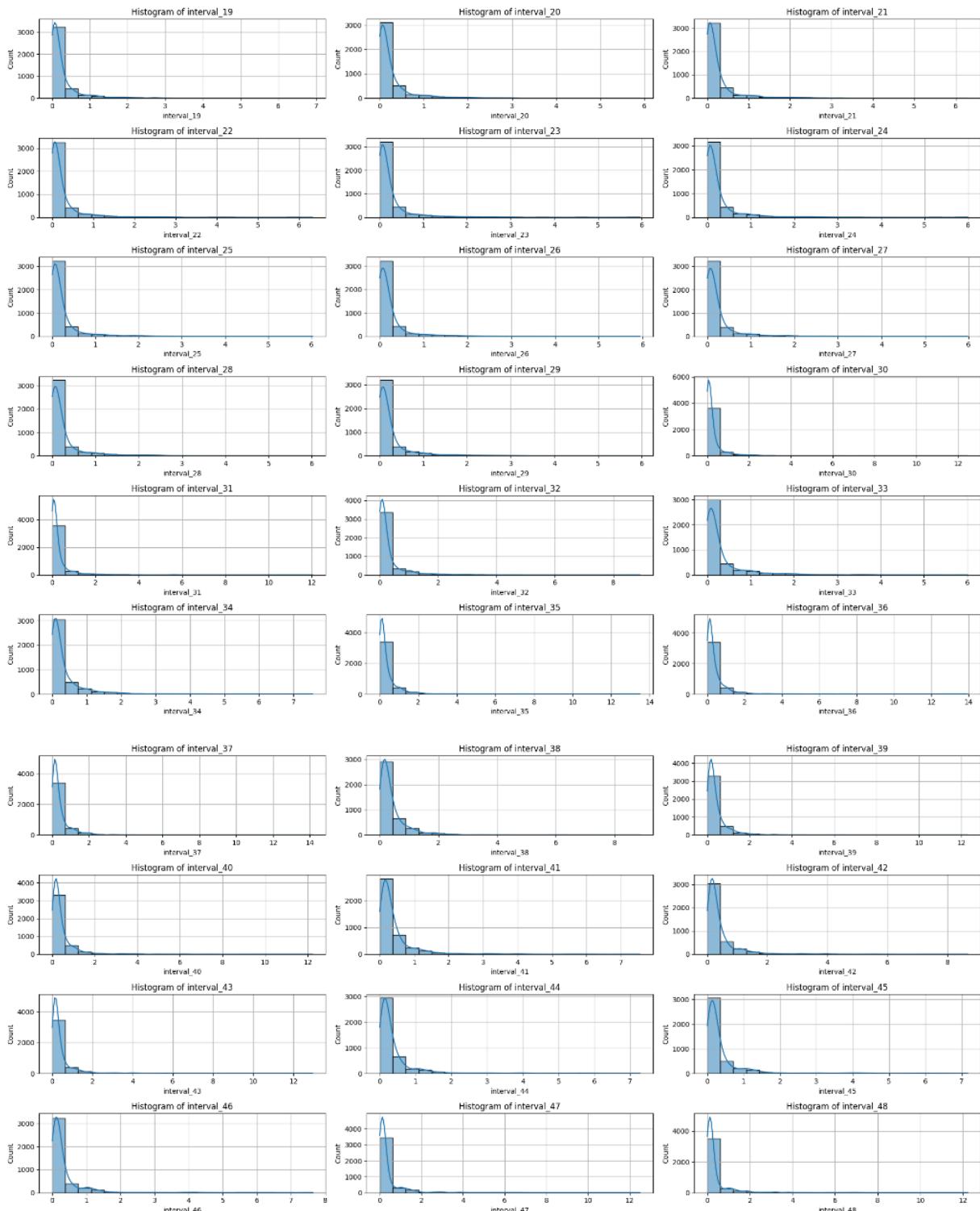
4.2 Eksplorasi Data

4.2.1 Histogram pada fitur numerik

Selanjutnya, kami membuat visualisasi histogram yang akan menggambarkan bagaimana nilai data tersebar di seluruh rentang nilai. Berikut adalah hasilnya.

Histogram untuk Fitur Numerik (Initial)





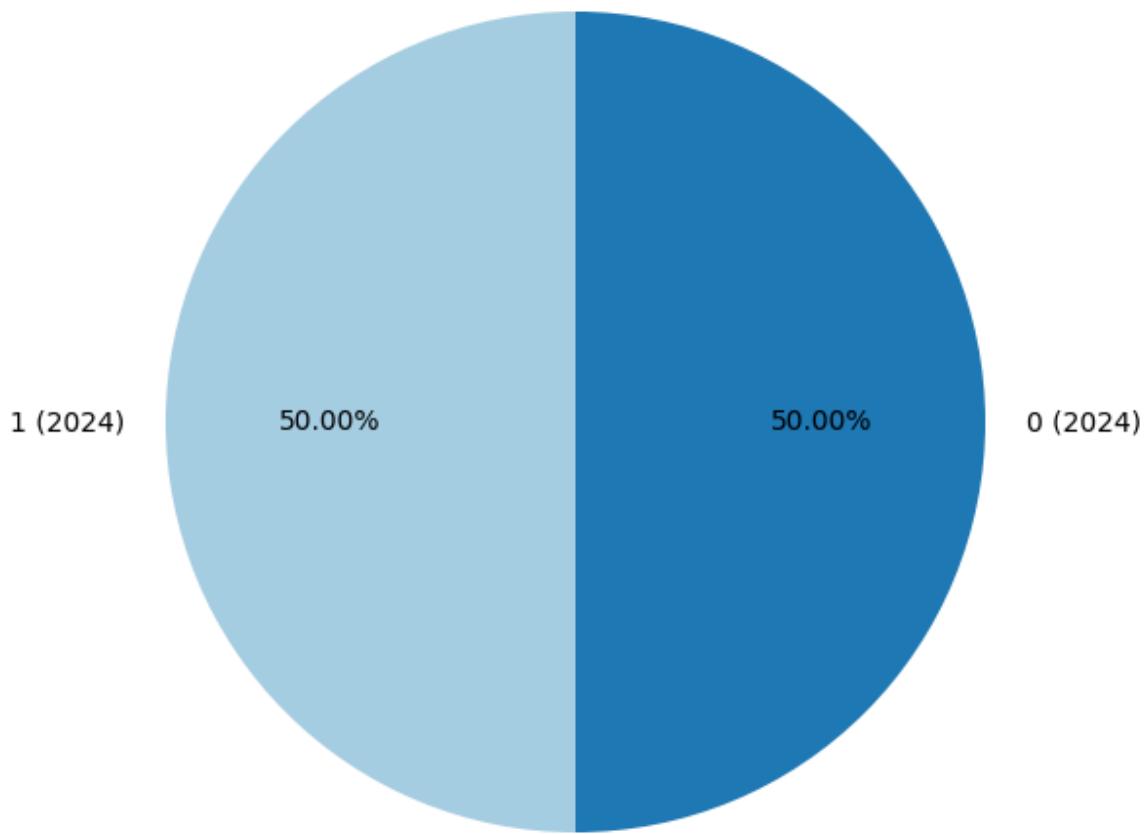
Berdasarkan hasil visualisasi histogram, kami mendapatkan hasil bahwa seluruh fitur cenderung memiliki pola *skewed right*. Sebagian besar nilai terkonsentrasi di dekat angka nol, dengan

frekuensi menurun secara tajam seiring meningkatnya nilai. Pola ini mengindikasikan bahwa mayoritas data berada dalam rentang nilai rendah, dengan beberapa nilai tinggi yang lebih jarang terjadi, yang berpotensi sebagai outlier. Selain itu, dengan adanya pola skewed, model yang akan digunakan harus selain model berbasis tree. Karena model tree cenderung memprioritaskan nilai mayoritas, sehingga mengabaikan minoritas dalam data yang tidak seimbang. Hal ini dapat menyebabkan pemisahan yang tidak optimal dan overfitting.

4.2.2 Keseimbangan fitur target

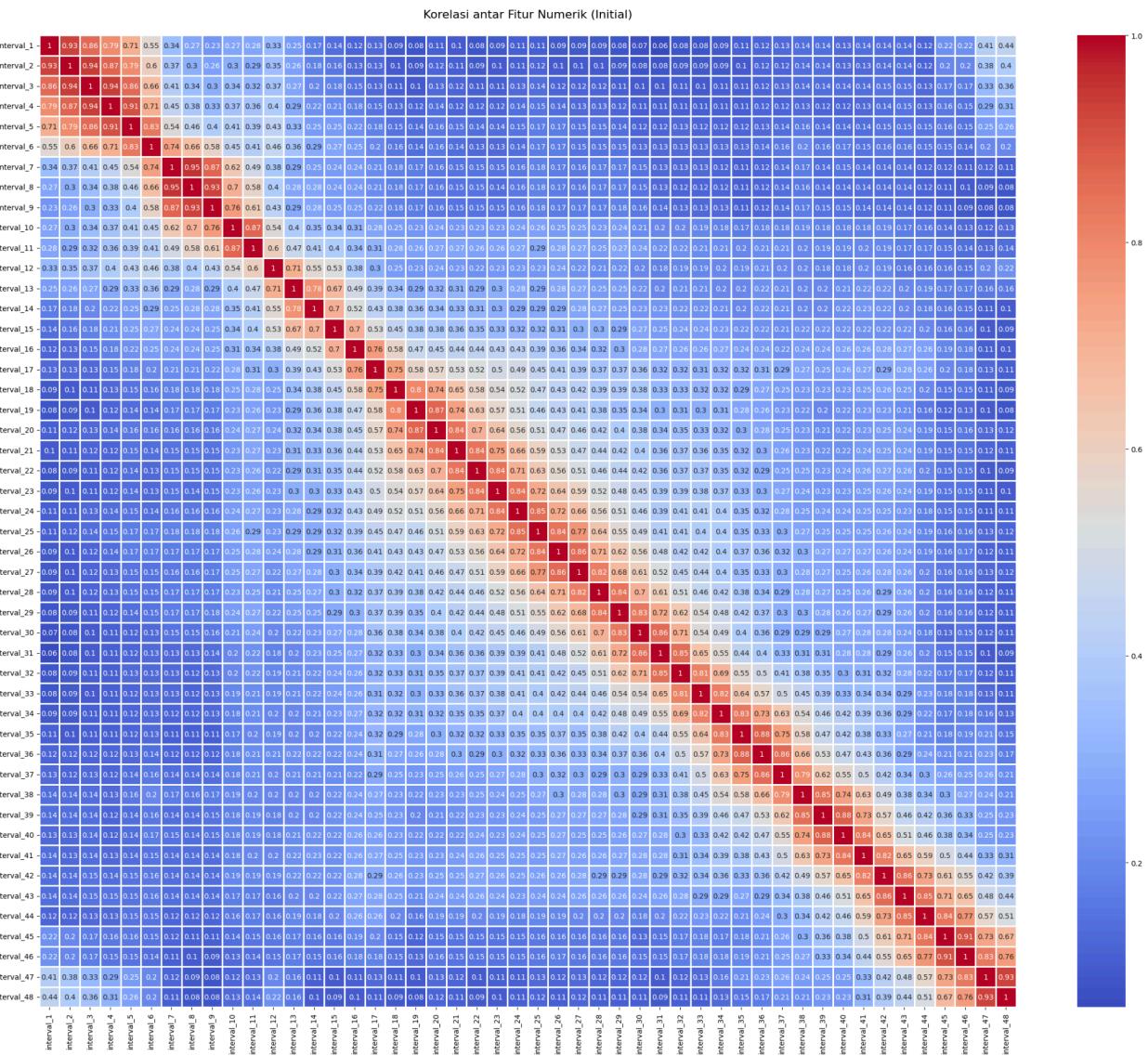
Persebaran pada fitur target, yaitu “label”, adalah *balanced*, dimana label 0 dan label 1 masing-masing berada di angka 50%. Hasil visualisasinya adalah sebagai berikut.

Class Distribution for 'label'



4.2.3 Korelasi antar fitur numerik

Kami melakukan analisis correlation matrix dengan pearson correlation matriks untuk melihat korelasi antar fitur pada dataset untuk fitur numerik. Berikut adalah visualisasinya.



Hasil dari analisis ini adalah mayoritas fitur numerik tidak memiliki korelasi yang kuat karena nilainya berkisar di 0.1-0.2. Akan tetapi, fitur-fitur memiliki kolerasi dengan fitur yang berdekatan waktu pengambilannya, misalnya interval_22 s.d. interval_25 masing-masingnya memiki kolerasi, hal ini ditunjukkan dengan nilai pearson diatas 0.5.

STEI- ITB	<nomor dokumen>	Halaman 13 dari 40 halaman
Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB.		

4.3 Verifikasi Kualitas Data

4.3.1 Identifikasi Missing Values

Selanjutnya kami mengecek jumlah null atau data yang kosong pada dataset. Didapatkan hasil sebagai berikut.

	Missing Count	Missing Percentage
read_date	0	0.0%
interval_1	0	0.0%
interval_2	0	0.0%
interval_3	0	0.0%
interval_4	0	0.0%
interval_5	0	0.0%
interval_6	0	0.0%
interval_7	0	0.0%
interval_8	0	0.0%
interval_9	0	0.0%
interval_10	0	0.0%
interval_11	0	0.0%
interval_12	0	0.0%
interval_13	0	0.0%
interval_14	0	0.0%
interval_15	0	0.0%
interval_16	0	0.0%
interval_17	0	0.0%
interval_18	0	0.0%
interval_19	0	0.0%
interval_20	0	0.0%
interval_21	0	0.0%
interval_22	0	0.0%
interval_23	0	0.0%
interval_24	0	0.0%
interval_25	0	0.0%
interval_26	0	0.0%
interval_27	0	0.0%
interval_28	0	0.0%
interval_29	0	0.0%
interval_30	0	0.0%
interval_31	0	0.0%
interval_32	0	0.0%
interval_33	0	0.0%
interval_34	0	0.0%
interval_35	0	0.0%
interval_36	0	0.0%
interval_37	0	0.0%
interval_38	0	0.0%
interval_39	0	0.0%
interval_40	0	0.0%
interval_41	0	0.0%
interval_42	0	0.0%
interval_43	0	0.0%
interval_44	0	0.0%
interval_45	0	0.0%
interval_46	0	0.0%
interval_47	0	0.0%
interval_48	0	0.0%
id	0	0.0%
label	0	0.0%

Berdasarkan hasil diatas, tidak terdapat values yang null. Oleh karena itu, kami tidak akan melakukan teknik imputasi kedepannya.

4.3.2 Identifikasi Duplikasi Data

Pada tahap ini, kami tidak menemukan adanya duplikasi data di dataset. Berikut adalah hasil identifikasi duplikasi data.

Duplicate Rows: 0
Duplicate Columns: 0

4.3.3 Identifikasi Konsistensi Data

Untuk melakukan pengecekan pada konsistensi data, kami melakukan pengecekan pada setiap fitur numerik untuk mengetahui apakah terdapat nilai negatif pada dataset. Hasil pada tahap ini menunjukkan bahwa tidak terdapat nilai yang negatif.

4.3.4 Identifikasi Format Data

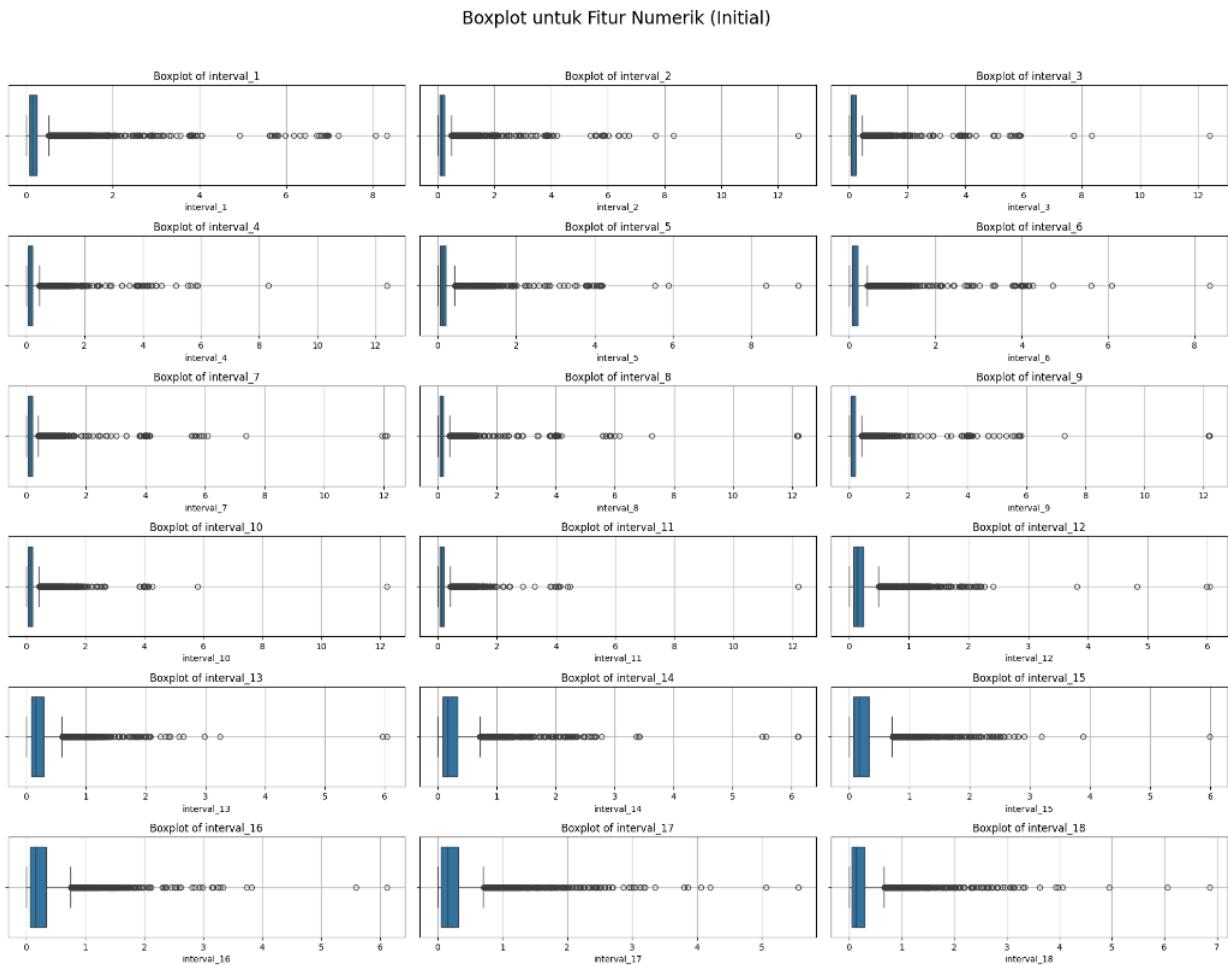
Pada tahap identifikasi format data, kami melakukan pengecekan format pada fitur “read_date” untuk melihat bagaimana penulisan format dari fitur ini. Hal ini bertujuan agar kami mengetahui perlu melakukan preprocessing yang tepat jika ada kesalahan format. Hasil dari tahap ini adalah semua kolom pada fitur “read_date” memiliki format yang serupa.

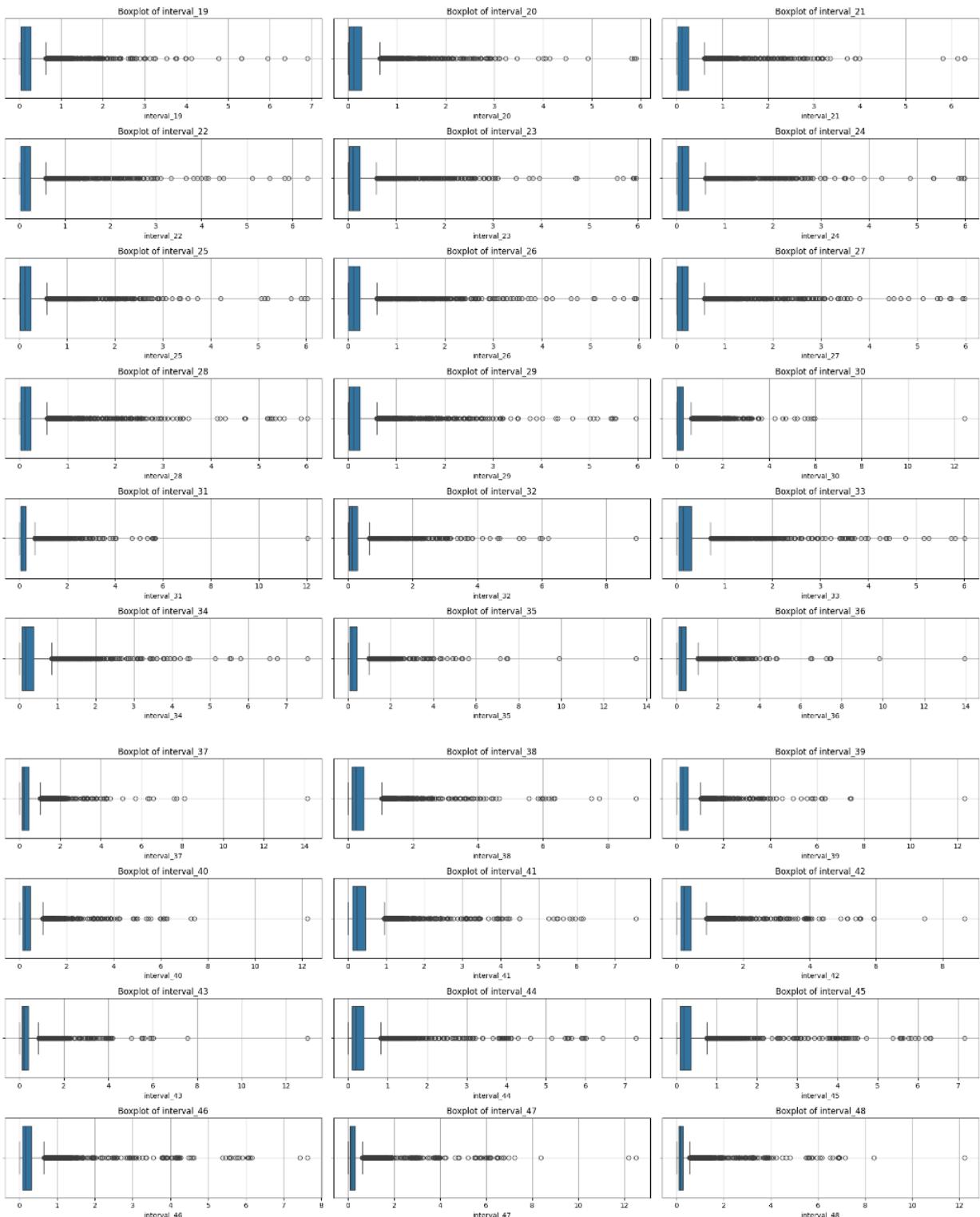
```
# Check Unique Values for read_date
unique_dates = initial_df['read_date'].unique()
unique_dates

array(['3/1/2021', '3/2/2021', '3/3/2021', '3/4/2021', '3/5/2021',
       '3/6/2021', '3/7/2021', '3/8/2021', '3/9/2021', '3/10/2021',
       '3/11/2021', '3/12/2021', '3/13/2021', '3/14/2021', '3/15/2021',
       '3/16/2021', '3/17/2021', '3/18/2021', '3/19/2021', '3/20/2021',
       '3/21/2021', '3/22/2021', '3/23/2021', '3/24/2021', '3/25/2021',
       '3/26/2021', '3/27/2021', '3/28/2021', '3/29/2021', '3/30/2021',
       '3/31/2021', '4/1/2021', '4/2/2021', '4/3/2021', '4/4/2021',
       '4/5/2021', '4/6/2021', '4/7/2021', '4/8/2021', '4/9/2021',
       '4/10/2021', '4/11/2021', '4/12/2021', '4/13/2021', '4/14/2021',
       '4/15/2021', '4/18/2021 0:00', '2/19/2021 0:00', '2/20/2021 0:00',
       '2/21/2021 0:00', '2/22/2021 0:00', '2/23/2021 0:00',
       '2/24/2021 0:00', '2/25/2021 0:00', '2/26/2021 0:00',
       '2/27/2021 0:00', '2/28/2021 0:00', '3/1/2021 0:00',
       '3/2/2021 0:00', '3/3/2021 0:00', '3/4/2021 0:00', '3/5/2021 0:00',
       '3/6/2021 0:00', '3/7/2021 0:00', '3/8/2021 0:00', '3/9/2021 0:00',
       '3/10/2021 0:00', '3/11/2021 0:00', '3/12/2021 0:00',
       '3/13/2021 0:00', '3/14/2021 0:00', '3/15/2021 0:00',
       '3/16/2021 0:00', '3/17/2021 0:00', '3/18/2021 0:00',
       '3/19/2021 0:00', '3/20/2021 0:00', '3/21/2021 0:00',
       '3/22/2021 0:00', '3/23/2021 0:00', '3/24/2021 0:00',
       '3/25/2021 0:00', '3/26/2021 0:00', '3/27/2021 0:00',
       '3/28/2021 0:00', '3/29/2021 0:00', '3/30/2021 0:00',
       '3/31/2021 0:00', '4/1/2021 0:00', '4/2/2021 0:00',
       '4/3/2021 0:00', '4/4/2021 0:00'], dtype=object)
```

4.3.5 Identifikasi Anomali Data (Outliers)

Boxplot memberikan ringkasan lima angka (minimum, kuartil pertama/Q1, median/Q2, kuartil ketiga/Q3, dan maksimum). Selain itu, boxplot juga memvisualisasikan outlier sebagai titik-titik yang berada di luar whiskers. Berikut adalah hasil visualisasi boxplot pada fitur numerik.





Berdasarkan boxplot diatas, setiap interval memiliki distribusi data yang sangat terpusat di sekitar nilai rendah (dekat nol), dengan sejumlah besar outlier yang muncul sebagai titik-titik di

luar whiskers. Rentang interkuartil (IQR) terlihat sempit, menunjukkan bahwa sebagian besar data terkonsentrasi dalam rentang nilai yang kecil, sementara nilai ekstrim yang lebih besar menyebar jauh dari median. Pola ini konsisten dengan distribusi yang *skewed right*, seperti yang terlihat pada histogram sebelumnya. Banyaknya *outlier* menunjukkan bahwa dataset ini mungkin memiliki nilai-nilai ekstrem yang perlu dianalisis lebih lanjut untuk memahami apakah ini adalah data valid atau kesalahan (error).

5. Data Preparation

5.1 Pemilihan Data

5.1.1 Record Selection

Pada tugas besar ini kami akan menggunakan keseluruhan baris atau *record* dari dataset, yaitu sebanyak 4048 baris.

5.1.2 Feature Selection

Beberapa fitur tidak akan digunakan untuk melatih model, seperti fitur “read_date” dan “id”. Hal ini dilakukan agar tidak merusak representasi dari fitur-fitur lainnya ketika proses modelling.

5.2 Perbaikan Data

5.2.1 Mengisi Missing Values

Tidak dilakukan sebab tidak ada missing values.

5.2.2 Penghapusan Duplikasi Data

Tidak dilakukan sebab tidak ada duplikasi data.

5.2.3 Penanganan Outliers

Tidak dilakukan sebab outlier dapat memberikan informasi yang berharga.

5.2.4 Perbaikan Error Lainnya

STEI- ITB	<nomor dokumen>	Halaman 18 dari 40 halaman
Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB.		

5.3 Konstruksi Data (Training Set dan Testing Set)

5.3.1 Train-Test Split

Pada tahap ini, Kami memisahkan data menjadi data training dan data testing, untuk mencegah leakage. Split yang kami gunakan pada dataset ini adalah sebesar 10% untuk test set dan 90% untuk train set. Sehingga menghasilkan jumlah data training sebanyak 3643 baris dan jumlah data testing sebanyak 405 baris.

5.3.2 Encoding Categorical Data

Tidak dilakukan sebab data sudah dalam bentuk numerik.

5.3.3 Rekayasa Fitur dengan Clustering

Selanjutnya, kami menggunakan teknik feature engineering dengan melakukan clustering menggunakan 2 metode yaitu KMeans dan AGNES. Dengan beberapa langkah berikut untuk mengoptimalkan proses clustering.

a. Grouping berdasarkan id (rumah)

Kami melakukan pengelompokan atau grouping pada fitur “id”. Hal ini bertujuan untuk mendapatkan pengelompokan untuk setiap rumah dan mendapatkan nilai mean dari fitur lainnya (fitur interval). Berikut adalah implementasi dan hasil yang didapatkan.

b. Reduksi Dimensi

Selanjutnya kami melakukan dimensionality reduction menggunakan teknik Principal Component Analysis (PCA). PCA bekerja dengan mengekstraksi fitur yang paling relevan dari data asli berdasarkan variansi, sehingga data dengan dimensi tinggi dapat direpresentasikan dengan lebih ringkas tanpa kehilangan informasi penting. Pada langkah awal, fitur numerik dipilih untuk direduksi (dengan mengabaikan kolom seperti id). Kami menentukan threshold sebesar 95%, yang berarti PCA akan memilih jumlah komponen utama yang cukup untuk menjelaskan setidaknya 95% dari total variansi data. Berikut adalah implemetasi PCA yang dilakukan.

```

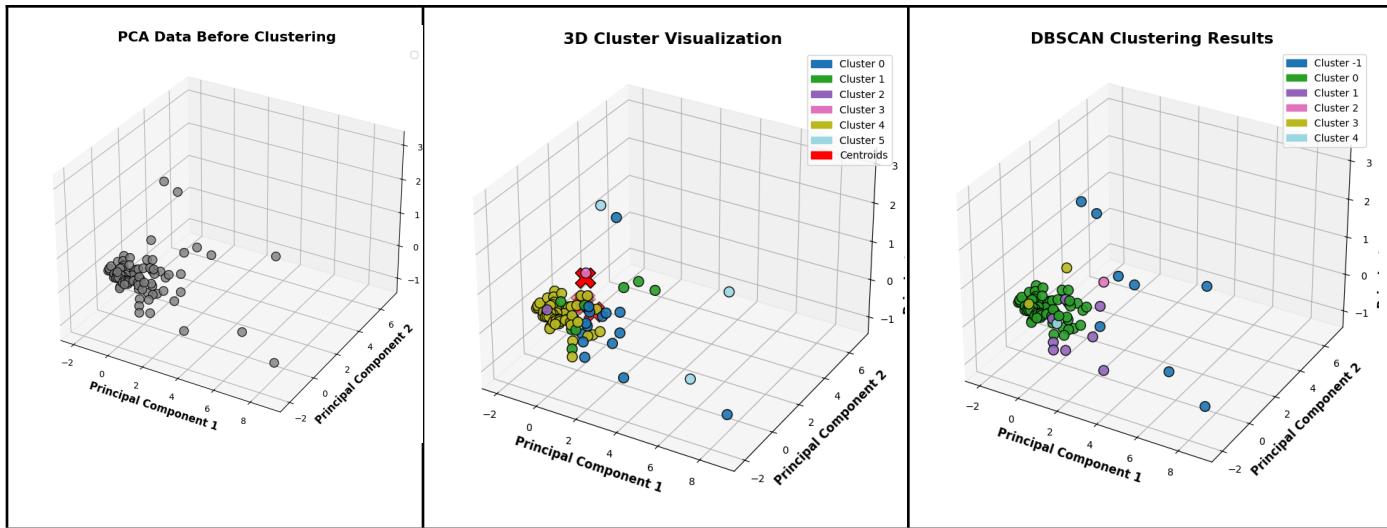
Cumulative Variance Ratio: [ 0.5979996  0.82737428  0.88355434  0.92601407  0.94504691  0.95922071
 0.96656434  0.97336566  0.97897427  0.98334416  0.9863264   0.98846404
 0.99024563  0.99169487  0.99289407  0.99378068  0.99449001  0.99515163
 0.99578085  0.99628163  0.9967409   0.99715901  0.99751225  0.9978194
 0.99809929  0.9983606   0.99861145  0.99880637  0.99897402  0.99911519
 0.99923892  0.9993462   0.99944393  0.99952959  0.99960678  0.99967127
 0.99973096  0.99978229  0.99982203  0.99985576  0.99988883  0.99991754
 0.99993617  0.99995303  0.9999685   0.99998102  0.99999203  1.          ]
Selected number of components (threshold=95.0%): 6

      id component_1 component_2 component_3 component_4 component_5 component_6
 0    1     -0.803264     0.435725    -0.071188     0.066271     0.331311   -0.278895
 1    2     -0.070587     1.480572    -0.328994    -0.196996    -0.065874     0.026944
 2    3      1.214502     0.163576     0.112634     0.121882     0.168127     0.170285
 3    4      0.616512    -0.075902    -0.154446     0.330629     0.432833   -0.203359
 4    5     -1.178961     0.087659     0.131939     0.274622    -0.111103   -0.010401
 ...
 ...
 83   84     -0.545998    -0.392862    -0.121194     0.084257     0.029917   -0.002204
 84   85     -0.895901    -0.599072    -0.292079    -0.371008    -0.302637   -0.139829
 85   86      1.611186    -0.117840     0.353985    -0.290498    -0.014312     0.002938
 86   87      0.087980    -0.243163    -0.078609    -0.007348     0.136554   -0.045367
 87   88     -0.608316    -0.595110    -0.129317    -0.578196     0.106496     0.155377

```

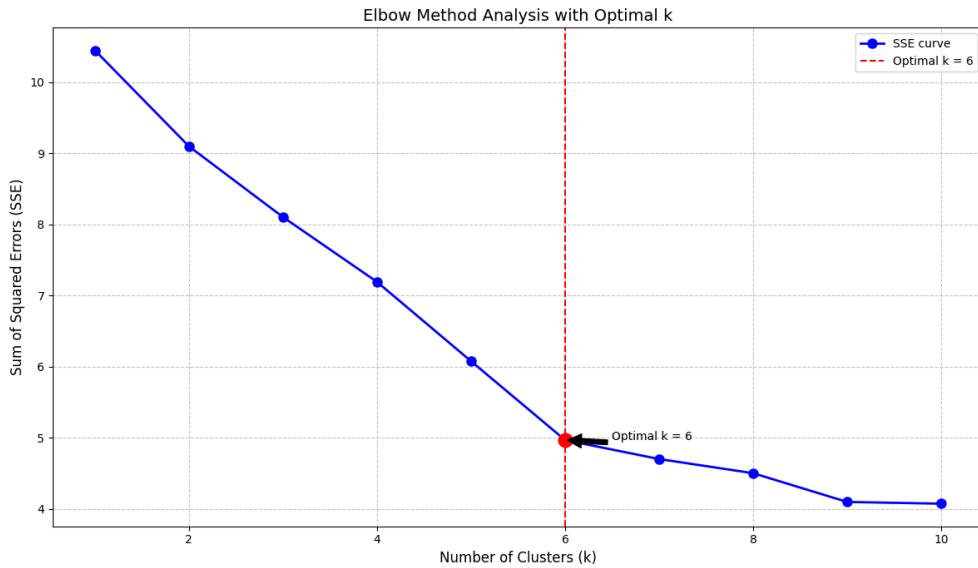
Hasil dari PCA menunjukkan rasio variansi yang dijelaskan oleh masing-masing komponen utama, serta kumulatifnya. Dari output, terlihat bahwa enam komponen utama dipilih karena bersama-sama mereka menjelaskan lebih dari 95% dari total variansi dalam data. Dengan demikian, dimensi data yang awalnya lebih tinggi dapat direduksi menjadi enam fitur utama ini tanpa kehilangan terlalu banyak informasi. Berikut adalah visualisasi dari sebelum dan sesudah dilakukan PCA.

Sebelum	Sesudah (K-Means)	Sesudah (DBSCAN)
---------	-------------------	------------------



c. Clustering

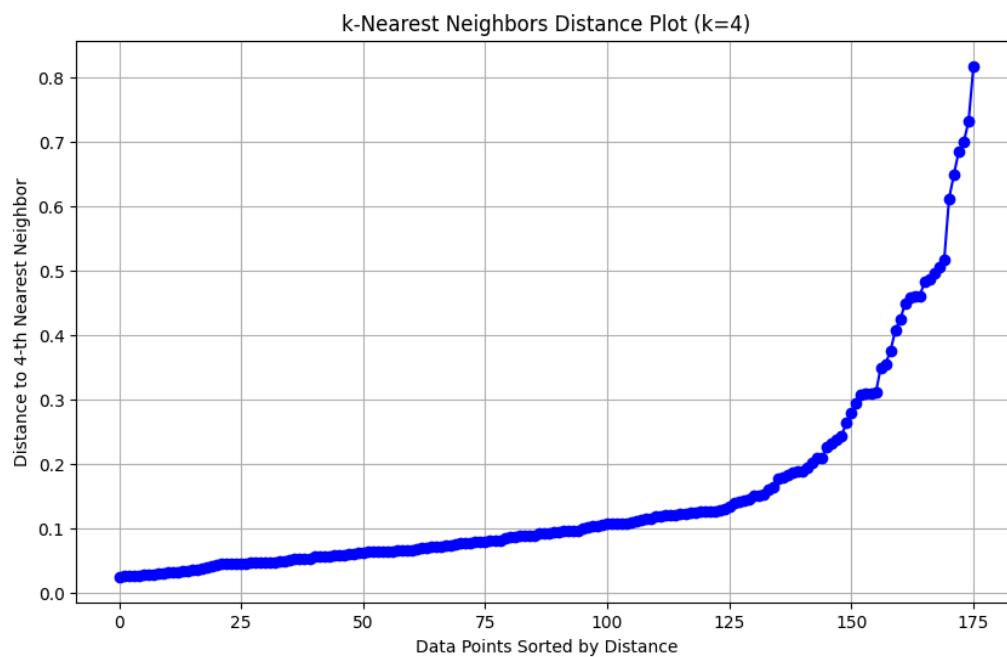
Sebelum melakukan clustering, kami menggunakan Elbow Method untuk menentukan jumlah cluster yang optimal dalam algoritma clustering seperti K-Means. Elbow Method bekerja dengan menghitung Sum of Squared Errors (SSE) untuk berbagai nilai jumlah cluster k , di mana SSE mengukur total kuadrat jarak antara setiap data dan centroid cluster-nya. Nilai SSE yang lebih kecil menunjukkan bahwa data lebih dekat ke centroidnya, yang berarti cluster lebih kompak. Berikut adalah hasil elbow method.



Pada grafik, sumbu x merepresentasikan jumlah cluster k , dan sumbu y menunjukkan nilai SSE. Dengan bertambahnya k , SSE cenderung menurun karena data dibagi ke dalam lebih banyak cluster, sehingga jarak dalam setiap cluster berkurang. Namun, penurunan SSE menjadi tidak signifikan setelah titik tertentu. Titik ini, yang tampak seperti "siku" pada grafik, adalah jumlah cluster yang optimal karena menyeimbangkan kompaksi cluster dan kompleksitas model. Dari grafik, terlihat bahwa titik "elbow" berada pada $k=6$. Pada $k=6$, penurunan SSE setelahnya mulai berkurang drastis, menunjukkan bahwa menambah jumlah cluster lebih lanjut tidak memberikan pengurangan SSE yang signifikan. Oleh karena itu, $k=6$ dipilih sebagai jumlah cluster yang optimal untuk menghasilkan pembagian data yang efisien dengan kompaksi cluster yang baik tanpa menambah kompleksitas yang tidak diperlukan.

Akan tetapi perhitungan elbow method untuk DBSCAN agak berbeda, yaitu elbow method bertujuan untuk menentukan nilai optimal parameter eps dalam algoritma DBSCAN. Dalam konteks DBSCAN, eps merupakan jarak maksimum yang menentukan apakah sebuah titik data termasuk dalam cluster berdasarkan kedekatan dengan tetangganya. Grafik ini dibuat dengan menghitung jarak ke tetangga k -terdekat (dalam hal ini, $k=4$) untuk setiap titik data, lalu mengurutkan

jarak tersebut secara bertahap. Pada grafik, terlihat bahwa jarak ke tetangga ke-4 meningkat perlahan pada awalnya, yang mencerminkan area dengan densitas tinggi, tetapi setelah titik tertentu terjadi lonjakan tajam. Lonjakan ini menandai perbedaan antara titik dalam cluster yang padat dan titik yang dianggap sebagai noise atau outlier. Titik elbow, di mana grafik mulai meningkat tajam, digunakan untuk menentukan nilai eps yang optimal. Nilai eps di titik ini memastikan bahwa DBSCAN dapat mendeteksi cluster secara efektif dengan menjaga koneksi antar titik dalam cluster, sekaligus mengisolasi noise. Metode ini membantu memilih parameter eps secara data-driven untuk mencerminkan struktur sebenarnya dari dataset.



5.3.4 Transformasi Skala (Normalisasi)

Kami melakukan transformasi pada fitur-fitur interval menggunakan metode minmax. Hal ini sesuai dengan hasil analisis data, dimana fitur-fitur interval memiliki pola *skewed* dan tidak terdistribusi normal. Berikut adalah implementasi dan hasil dari transformasi skala yang dilakukan.

X_train_scaled													
	interval_1	interval_2	interval_3	interval_4	interval_5	interval_6	interval_7	interval_8	interval_9	interval_10	...	interval_40	...
0	0.124625	0.081613	0.064008	0.005045	0.006109	0.007485	0.004645	0.004616	0.005123	0.004597	...	0.084352	...
1	0.003748	0.007866	0.003024	0.005546	0.006793	0.007485	0.004645	0.005133	0.011270	0.008687	...	0.015845	...
2	0.011892	0.005742	0.010484	0.012513	0.014022	0.012455	0.009008	0.010021	0.008197	0.007853	...	0.018896	...
3	0.032072	0.009204	0.006774	0.007750	0.011848	0.008623	0.008017	0.007228	0.006803	0.008098	...	0.123926	...
4	0.023267	0.013270	0.013605	0.009082	0.012228	0.014216	0.008777	0.009240	0.008197	0.007665	...	0.071575	...
...
3638	0.015976	0.011170	0.009839	0.010817	0.015326	0.015329	0.010744	0.011663	0.010984	0.009816	...	0.018896	...
3639	0.083604	0.013845	0.008306	0.008315	0.009565	0.010898	0.006529	0.007474	0.018934	0.009898	...	0.033947	...
3640	0.007928	0.004641	0.004355	0.002664	0.003478	0.003832	0.002645	0.002793	0.002295	0.010225	...	0.018732	...
3641	0.015015	0.008850	0.008565	0.010091	0.011543	0.014216	0.009298	0.009240	0.009730	0.008687	...	0.049080	...
3642	0.111111	0.030482	0.021169	0.020182	0.025815	0.026946	0.019628	0.020534	0.017418	0.021472	...	0.027607	...

3643 rows x 49 columns

6. Implementasi Model

6.1 Supervised Learning

6.1.1 Logistic Regression

Dataset yang digunakan merupakan studi kasus *binary classification*. Oleh karena itu, Logistic Regression dipilih untuk model supervised learning yang pertama dengan tujuan memprediksi hasil biner pada target fitur dataset, dimana kami akan mendapat insight tentang apakah suatu rumah memiliki kendaraan listrik (EV) atau tidak. Fungsi sigmoid dalam algoritma ini mampu memetakan prediksi ke dalam bentuk probabilitas, sehingga cocok untuk tugas klasifikasi. Logistic Regression juga efisien secara komputasi dan mudah diinterpretasikan, karena bobot (*weights*) dalam model menunjukkan pentingnya masing-masing fitur. Algoritma ini sangat sesuai untuk data yang bersifat linier atau ketika kesederhanaan dan interpretabilitas lebih diutamakan dibandingkan hubungan non-linier yang kompleks. Implementasi yang digunakan menggunakan metode gradient descent untuk optimasi, sehingga dapat beradaptasi dengan dataset yang berukuran besar maupun kecil. Meskipun model yang lebih kompleks, seperti neural network, dapat menangani data non-linier dengan lebih baik, Logistic Regression dipilih karena biaya komputasinya lebih rendah dan memiliki batas keputusan (*decision boundary*) yang sederhana. Berikut adalah kode algoritma yang digunakan.

```

import numpy as np

# Define Kelas untuk LogReg
class LogisticRegression():
    # Init konstruktor dengan params learning rate dan jumlah iterasi
    def __init__(self, lr=0.001, n_iters=200):
        self.lr = lr
        self.n_iters = n_iters
        self.weights = None
        self.bias = None

    # Define fungsi sigmoid
    def sigmoid(self, x):
        # Merubah input menjadi probabilitas di range (0,1)
        return 1/(1+np.exp(-x))

    # fit dengan data train
    def fit(self, X, y):
        n_samples, n_features = X.shape
        self.weights = np.zeros(n_features)
        self.bias = 0

        # Loop untuk update parameter sebanyak n_iters
        for _ in range(self.n_iters):
            # hitung linear prediction
            linear_pred = np.dot(X, self.weights) + self.bias
            # masukkan ke sigmoid
            predictions = self.sigmoid(linear_pred)

            # cari gradien untuk weight dan bias (optimization)
            dw = (1/n_samples) * np.dot(X.T, (predictions - y))
            db = (1/n_samples) * np.sum(predictions-y)

            # update weight dan bias
            self.weights = self.weights - self.lr*dw
            self.bias = self.bias - self.lr*db

        return self

    # prediksi data test
    def predict(self, X):
        linear_pred = np.dot(X, self.weights) + self.bias
        y_pred = self.sigmoid(linear_pred)
        # jika probabilitas > 0.5, maka kelas 1, jika tidak kelas 0
        class_pred = [0 if y<0.5 else 1 for y in y_pred]
        return class_pred

    def fit_predict(self, X, y, X_test):
        self.fit(X, y).predict(X_test)

```

Logistic Regression bekerja dengan memodelkan hubungan antara input fitur dan output target sebagai fungsi linier yang diubah menjadi probabilitas menggunakan fungsi sigmoid. Pada awalnya, parameter model berupa bobot ('weights') dan bias ('bias') diinisialisasi dengan nilai nol. Proses pelatihan dilakukan dalam metode 'fit', di mana model menerima data input 'X' dan target 'y'. Selama sejumlah iterasi yang telah ditentukan, model secara bertahap memperbarui bobot dan bias menggunakan algoritma Gradient Descent. Gradien loss dihitung berdasarkan selisih antara prediksi dan nilai aktual untuk mengoptimalkan parameter model. Prediksi dilakukan menggunakan metode 'predict', di mana model menghitung nilai linier, menerapkan fungsi sigmoid untuk mendapatkan probabilitas, dan menentukan kelas akhir berdasarkan threshold 0.5. Fungsi tambahan 'fit_predict' menggabungkan proses pelatihan dan prediksi dalam satu langkah. Secara keseluruhan, kode ini dirancang untuk meminimalkan kesalahan prediksi dengan mengoptimalkan parameter sehingga dapat memetakan data input ke kelas target dengan akurasi tinggi.

6.1.2 K Nearest Neighbours

KNN adalah algoritma pembelajaran mesin supervised yang bekerja dengan prinsip mencari tetangga terdekat dari suatu data untuk melakukan klasifikasi. Dalam konteks deteksi rumah yang memiliki kendaraan listrik (EV), KNN dapat memanfaatkan pola konsumsi listrik yang sudah diketahui dari rumah-rumah yang memiliki EV sebagai data training. Algoritma ini cocok untuk kasus ini karena beberapa alasan. Pertama, KNN dapat menangkap pola konsumsi listrik yang kompleks dan bervariasi sepanjang waktu, yang merupakan karakteristik umum dari penggunaan listrik rumah tangga dengan EV. Kedua, KNN memiliki kemampuan untuk mempertimbangkan berbagai fitur sekaligus, seperti pola konsumsi di waktu yang berbeda (malam hari saat pengisian EV) dan total konsumsi harian, yang dapat menjadi indikator kuat kepemilikan EV. Ketiga, sifat data konsumsi listrik yang kontinu dan memiliki pola tertentu sangat sesuai dengan cara kerja KNN yang mengandalkan kedekatan jarak antar data. Berikut adalah implementasi yang akan kami gunakan.

```
class KNearestNeighbors:
    def __init__(self, k=3):
        """
        Initialize the KNN classifier.

        Parameters:
        - k (int): Number of neighbors to consider.
        """
        self.k = k

    def fit(self, X, y):
        """
        Fit the model to the training data.

        Parameters:
        - X (np.ndarray): Feature matrix for training data.
        - y (np.ndarray): Labels for training data.
        """
        self.X_train = X
        self.y_train = y

    def predict(self, X):
        """
        Predict the class labels for the provided data.

        Parameters:
        - X (np.ndarray): Feature matrix for test data.

        Returns:
        - np.ndarray: Predicted class labels.
        """
        predictions = [self._predict_single(x) for x in X]
        return np.array(predictions)

    def _predict_single(self, x):
        """
        Predict the class label for a single data point.

        Parameters:
        - x (np.ndarray): A single test data point.

        Returns:
        - int: Predicted class label.
        """
        # Compute distances between x and all training samples
        distances = [self._euclidean_distance(x, x_train) for x_train in self.X_train]
        # Get indices of k nearest neighbors
        k_indices = np.argsort(distances)[:self.k]

        # Extract the labels of the k nearest neighbors
        k_nearest_labels = [self.y_train[i] for i in k_indices]

        # Determine the most common label among the k neighbors
        most_common = Counter(k_nearest_labels).most_common(1)
        return most_common[0][0]

    def _euclidean_distance(self, x1, x2):
        """
        Compute the Euclidean distance between two points.

        Parameters:
        - x1 (np.ndarray): First data point.
        - x2 (np.ndarray): Second data point.

        Returns:
        - float: Euclidean distance.
        """
        return np.sqrt(np.sum((x1 - x2) ** 2))
```

Intuisi KNN adalah bahwa data baru akan diklasifikasikan berdasarkan kelas mayoritas dari tetangga terdekatnya dalam ruang fitur. Algoritma ini dimulai dengan inisialisasi parameter k , yang menunjukkan jumlah tetangga terdekat yang akan dipertimbangkan. Fungsi `fit` menyimpan data pelatihan (fitur dan label) untuk referensi, karena KNN tidak melakukan pelatihan eksplisit seperti model berbasis parameter; ini adalah metode lazy learning di mana perhitungan dilakukan saat prediksi. Saat melakukan prediksi, algoritma menghitung jarak Euclidean antara data yang akan diklasifikasikan dengan semua data pelatihan menggunakan fungsi euclidian distance. Jarak-jarak ini kemudian diurutkan, dan k tetangga terdekat dipilih berdasarkan nilai jarak terendah. Label dari k tetangga ini diambil, dan label yang paling sering muncul di antara mereka dipilih sebagai prediksi untuk data tersebut. Proses ini dilakukan secara iteratif untuk setiap data

dalam dataset uji melalui fungsi predict. Dengan pendekatan ini, KNN memanfaatkan hubungan lokal dalam data untuk membuat prediksi. Nilai k memengaruhi performa algoritma: nilai k yang kecil membuat model lebih sensitif terhadap noise, sementara nilai k yang terlalu besar dapat mengurangi akurasi dengan mempertimbangkan terlalu banyak tetangga yang tidak relevan. Kode ini mendemonstrasikan cara kerja KNN secara sederhana, dengan fokus pada perhitungan jarak, pemilihan tetangga, dan pengambilan keputusan berbasis mayoritas.

6.2 Unsupervised Learning

6.2.1 K-Means

Selanjutnya, K-Means dipilih untuk model pertama untuk unsupervised learning. K-Means clustering sangat relevan untuk mengidentifikasi kelompok-kelompok rumah dengan pola konsumsi listrik serupa. Implementasi yang diberikan menggunakan jarak Euclidean untuk mengukur kesamaan antara titik data, dan secara iteratif memperbaiki posisi centroid hingga konvergen. Algoritma ini cocok untuk kasus ini karena beberapa alasan. Pertama, dapat mengelompokkan rumah-rumah berdasarkan pola konsumsi listrik tanpa memerlukan label, yang berguna ketika data histori tentang kepemilikan EV tidak tersedia. Kedua, dapat mengidentifikasi "hotspot" atau kelompok rumah dengan karakteristik konsumsi listrik tinggi yang mungkin mengindikasikan konsentrasi pengguna EV dalam suatu area. Ketiga, kompleksitas komputasi yang relatif rendah ($O(n)$) memungkinkan analisis skala besar. Namun, kelemahannya adalah sensitifitas terhadap pemilihan jumlah cluster (K) dan initial centroid, serta asumsi bahwa cluster berbentuk spherical yang mungkin tidak selalu mencerminkan pola sebenarnya.

```

class KMeans:
    def __init__(self, K=3, max_iters=100, random_state=42):
        self.K = K
        self.max_iters = max_iters
        self.random_state = random_state

        # List untuk menyimpan indeks sampel pada tiap cluster
        self.clusters = [[] for _ in range(self.K)]

        # List untuk menyimpan centroid (mean) dari tiap cluster
        self.centroids = []

    def fit(self, X):
        """
        Fit the KMeans model to the data.
        Parameters:
        - X: np.ndarray
            The input data to cluster.
        """
        self.X = X
        self.n_samples, self.n_features = X.shape

        if self.random_state is not None:
            np.random.seed(self.random_state)

        # init centroid secara acak dari sampel dalam dataset
        random_sample_idxs = np.random.choice(self.n_samples, self.K, replace=False)
        self.centroids = [self.X[idx] for idx in random_sample_idxs]

        # optimize clusters
        for _ in range(self.max_iters):
            # assign samples ke centroid terdekat
            self.clusters = self._create_clusters(self.centroids)

            # hitung centroid baru berdasarkan cluster
            centroids_old = self.centroids
            self.centroids = self._get_centroids(self.clusters)

            # Periksa konvergensi (apakah centroid tidak berubah)
            if self._is_converged(centroids_old, self.centroids):
                break

    def predict(self, X):
        """
        Predict the closest cluster each sample in X belongs to.
        Parameters:
        - X: np.ndarray
            The input data to predict.
        Returns:
        - labels: np.ndarray
            Cluster labels for each sample in X.
        """
        labels = np.empty(X.shape[0])

        for idx, sample in enumerate(X):
            distances = [self.euclidean_distance(sample, centroid) for centroid in self.centroids]
            labels[idx] = np.argmin(distances)

        return labels.astype(int)

    def fit_predict(self, X):
        """
        Fit the KMeans model and return cluster labels.
        Parameters:
        - X: np.ndarray
            The input data to cluster.
        Returns:
        - labels: np.ndarray
            Cluster labels for each sample in X.
        """
        self.fit(X)
        return self.predict(X)

    # Fungsi untuk menghitung jarak euclidean
    def euclidean_distance(self, x1, x2):
        return np.sqrt(np.sum((x1 - x2) ** 2))

    # Mendapatkan label cluster untuk tiap sampel berdasarkan assignment cluster.
    def _get_cluster_labels(self, clusters):
        labels = np.empty(self.n_samples)
        for cluster_idx, cluster in enumerate(clusters):
            for sample_idx in cluster:
                labels[sample_idx] = cluster_idx
        return labels

    # Membuat cluster dengan mengassign sampel ke centroid terdekat.
    def _create_clusters(self, centroids):
        clusters = [[] for _ in range(self.K)]
        for idx, sample in enumerate(self.X):
            centroid_idx = self._closest_centroid(sample, centroids)
            clusters[centroid_idx].append(idx)
        return clusters

    # Menentukan centroid terdekat untuk sebuah sampel.
    def _closest_centroid(self, sample, centroids):
        distances = [self.euclidean_distance(sample, point) for point in centroids]
        closest_idx = np.argmin(distances)
        return closest_idx

    # Mendapatkan centroid baru berdasarkan rata-rata nilai sampel dalam cluster.
    def _get_centroids(self, clusters):
        centroids = np.zeros((self.K, self.n_features))
        for cluster_idx, cluster in enumerate(clusters):
            cluster_mean = np.mean(self.X[cluster], axis=0)
            centroids[cluster_idx] = cluster_mean
        return centroids

    # Periksa konvergenSI dengan membandingkan centroid baru dan centroid lama.
    def _is_converged(self, centroids_old, centroids):
        distances = [self.euclidean_distance(centroids_old[i], centroids[i]) for i in range(self.K)]
        return sum(distances) == 0

```

Kode di atas merupakan implementasi algoritma K-Means Clustering, sebuah metode unsupervised learning untuk mengelompokkan data ke dalam sejumlah cluster berdasarkan kesamaan fitur. Model diinisialisasi dengan jumlah cluster yang diinginkan (K), jumlah iterasi maksimum (max_iters), dan seed randomisasi (random_state) untuk memastikan hasil yang konsisten. Selama pelatihan, model memilih centroid awal secara acak dari dataset. Kemudian, untuk setiap iterasi, setiap data dihitung jaraknya ke centroid menggunakan jarak Euclidean dan diassign ke cluster dengan centroid terdekat. Setelah semua data diassign, centroid baru dihitung sebagai rata-rata posisi data dalam setiap cluster. Proses ini terus berulang hingga posisi centroid tidak lagi berubah (konvergen) atau jumlah iterasi maksimum tercapai. Setelah model dilatih, fungsi prediksi (predict) dapat digunakan untuk menentukan cluster dari data baru dengan cara menghitung jarak ke centroid terdekat. Algoritma ini memastikan bahwa jarak antara data dalam cluster yang sama diminimalkan, sementara jarak antar cluster dimaksimalkan. Dengan menggunakan metode tambahan seperti _create_clusters untuk membentuk ulang cluster dan _is_converged untuk memeriksa konvergensi, kode ini dirancang untuk mengelompokkan data dengan efisien. Algoritma ini bekerja secara iteratif, dan hasil akhirnya sangat tergantung pada inisialisasi centroid serta jumlah cluster yang dipilih.

6.2.2 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Terakhir, DBSCAN dipilih untuk model di kasus ini karena keunggulannya dalam mengelompokkan data berbasis densitas tanpa perlu menentukan jumlah cluster sebelumnya. Dalam konteks dataset yang digunakan, algoritma ini cocok untuk mengidentifikasi pola atau kelompok dalam data yang tidak memiliki distribusi yang jelas, seperti data geografis atau data dengan kepadatan yang bervariasi. DBSCAN bekerja dengan menemukan area dengan densitas tinggi, menjadikannya ideal untuk mendeteksi kelompok data seperti "hotspot" perilaku tertentu atau cluster pengguna dengan karakteristik unik. DBSCAN juga unggul dalam menangani outlier, dengan secara otomatis melabelinya sebagai noise. Ini penting dalam dataset yang mungkin mengandung data ekstrem yang tidak relevan dengan pola utama. Selain itu, algoritma ini mampu menangani cluster dengan bentuk arbitrer, tidak seperti K-Means yang mengasumsikan cluster berbentuk bulat. Hal ini membuat DBSCAN sangat relevan untuk dataset dengan distribusi yang kompleks atau non-linear. Berikut adalah implementasi dari DBSCAN pada tugas kami.

STEI- ITB	<nomor dokumen>	Halaman 30 dari 40 halaman
Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB.		



```

import numpy as np
from scipy.spatial.distance import euclidean

class DBSCAN:
    def __init__(self, eps=0.5, min_samples=5):
        """
        Initialize DBSCAN clustering.

        Parameters:
        - eps (float): Maximum distance between two points to be considered neighbors.
        - min_samples (int): Minimum number of points required to form a dense region.
        """
        self.eps = eps
        self.min_samples = min_samples
        self.labels_ = None

    def fit(self, X):
        """
        Fit the DBSCAN model to the data.

        Parameters:
        - X (np.ndarray): Input data.
        """
        n_samples = X.shape[0]
        self.labels_ = np.full(n_samples, -1) # Initialize all labels as noise (-1)
        cluster_id = 0

        for i in range(n_samples):
            if self.labels_[i] != -1: # Skip already visited points
                continue

            # Find neighbors of point i
            neighbors = self._region_query(X, i)

            if len(neighbors) < self.min_samples:
                self.labels_[i] = -1 # Mark as noise
            else:
                # Expand the cluster
                self._expand_cluster(X, i, neighbors, cluster_id)
                cluster_id += 1

        def _region_query(self, X, point_idx):
            """
            Find neighbors within eps distance.
            """
            neighbors = []
            for i, point in enumerate(X):
                if euclidean(X[point_idx], point) <= self.eps:
                    neighbors.append(i)
            return neighbors

        def _expand_cluster(self, X, point_idx, neighbors, cluster_id):
            """
            Expand the cluster using neighbors.
            """
            self.labels_[point_idx] = cluster_id

            i = 0
            while i < len(neighbors):
                neighbor_idx = neighbors[i]

                if self.labels_[neighbor_idx] == -1: # If it is noise, mark as border point
                    self.labels_[neighbor_idx] = cluster_id
                elif self.labels_[neighbor_idx] == -1: # If unvisited
                    self.labels_[neighbor_idx] = cluster_id
                    new_neighbors = self._region_query(X, neighbor_idx)
                    if len(new_neighbors) >= self.min_samples:
                        neighbors.extend(new_neighbors) # Add to cluster

                i += 1

    def predict(self):
        """
        Return the cluster labels.
        """
        return self.labels_

```

Kami mengimplementasikan algoritma DBSCAN, sebuah metode clustering berbasis densitas yang menentukan cluster berdasarkan kedekatan dan jumlah titik dalam suatu area. DBSCAN bekerja dengan dua parameter utama, yaitu `eps`, jarak maksimum untuk menentukan tetangga suatu titik, dan `min_samples`, jumlah minimum titik untuk membentuk area berdensitas tinggi (cluster). Pada awalnya, semua titik dilabeli sebagai noise -1, lalu algoritma mencari tetangga setiap titik dalam radius `eps`. Jika suatu titik memiliki tetangga yang cukup ($\geq \text{min_samples}$),

algoritma memulai pembentukan cluster baru, menjadikan titik tersebut sebagai inti, dan memperluas cluster dengan memeriksa tetangga dari setiap titik baru yang ditambahkan. Titik dengan tetangga yang tidak mencukupi tetap dilabeli sebagai noise. DBSCAN dapat menemukan cluster dengan bentuk arbitrer dan secara otomatis menangani outlier, tetapi kinerjanya sensitif terhadap pemilihan parameter `eps` dan `min_samples`. Implementasi ini mencakup setiap langkah penting DBSCAN, termasuk pencarian tetangga menggunakan jarak Euclidean, pembentukan cluster, dan perluasan iteratifnya untuk mengelompokkan data dengan efisien.

7. Training, Testing, dan Analisis Hasil

7.1 Perbandingan Model Klasifikasi

Setelah melakukan modelling terhadap keseluruhan model beserta teknik rekayasa yang berbeda, kami mendapatkan hasil seperti tabel berikut. (Disini kami menambahkan model Decision Tree agar bisa melihat dampak saat pengujian ensemble kedepannya).

No	Jenis Model	Param	Metrics			
			Accuracy	F1 Score	Recall	Precision
1	Logistic Regression	+ clustering (k-means)	59%	57%	59%	61%
2	KNN	+ clustering (k-means)	83%	83%	83%	83%
3	Decision Tree	+ clustering (k-means)	84%	83%	83%	83%
4	Logistic Regression	+ clustering (DBSCAN)	62%	61%	62%	65%
5	KNN	+ clustering (DBSCAN)	78%	78%	78%	78%
6	Decision Tree	+ clustering (DBSCAN)	80%	79%	78%	79%

Tabel diatas menunjukkan hasil evaluasi performa tiga model klasifikasi, yaitu Logistic Regression, KNN, dan Decision Tree, yang digabungkan dengan dua metode clustering berbeda: K-Means dan DBSCAN. Dari hasil, terlihat bahwa kombinasi Decision Tree dengan K-Means

menghasilkan performa terbaik secara keseluruhan, dengan Accuracy, F1 Score, Recall, dan Precision masing-masing mencapai 84%. Kombinasi ini sedikit lebih unggul dibandingkan dengan KNN dan K-Means, yang memiliki performa 83% di semua metrik. Ketika menggunakan metode clustering DBSCAN, Decision Tree juga menunjukkan hasil yang paling baik, dengan Accuracy 80% dan F1 Score 79%, yang mengungguli kombinasi lainnya. KNN dengan DBSCAN berada di urutan kedua untuk DBSCAN, dengan performa yang relatif baik di semua metrik (78%). Sebaliknya, Logistic Regression menunjukkan performa terendah baik dengan K-Means maupun DBSCAN, dengan akurasi masing-masing hanya 59% dan 62%. Hasil ini menunjukkan bahwa Decision Tree unggul dalam kedua pendekatan clustering, terutama dalam mengelompokkan data berdasarkan pola yang lebih kompleks. Sementara itu, KNN memberikan performa yang stabil tetapi sedikit di bawah Decision Tree, sedangkan Logistic Regression kurang efektif untuk skenario ini. Ini menunjukkan bahwa kompleksitas model (seperti Decision Tree) dan kemampuan clustering (terutama K-Means) memberikan kontribusi besar terhadap keberhasilan model dalam mendeteksi pola dalam dataset.

7.2 Pengaruh Ensemble pada Model Klasifikasi

Pada tahap ini kami mengimplementasikan metode ensemble voting dengan tiga model, Logistic Regression, K-Nearest Neighbors (KNN), dan Decision Tree. Kami menambahkan satu model lagi agar bisa menggunakan pendekatan hard voting. Prediksi akhir ditentukan berdasarkan suara mayoritas dari semua model dalam ensemble. Dalam kasus suara yang sama (tie), model tambahan (tie_break_model) dapat digunakan untuk memutuskan hasil akhir, memberikan fleksibilitas tambahan dalam situasi seperti itu. Untuk memastikan kemampuan melakukan hard voting, sebuah model tambahan ditambahkan ke dalam ensemble sehingga totalnya menjadi tiga model, memungkinkan sistem untuk secara konsisten menghasilkan keputusan berbasis mayoritas tanpa risiko tie yang tidak terselesaikan. Dengan demikian, ensemble ini menggabungkan kelebihan dari model-model yang digunakan, memanfaatkan keragaman mereka untuk menghasilkan prediksi yang lebih andal dan robust, sekaligus memitigasi kelemahan dari model individu. Berikut adalah detail dari hasil uji ensemble pada model klasifikasi.

No	Jenis Model	Param	Metrics
----	-------------	-------	---------

			Accuracy	F1 Score	Recall	Precision
1	Ensemble Hard Voting	+ clustering (k-means)	81%	80%	80%	80%
2	Ensemble Hard Voting	+ clustering (DBSCAN)	80%	79%	78%	79%

7.3 Perbandingan Hasil Clustering

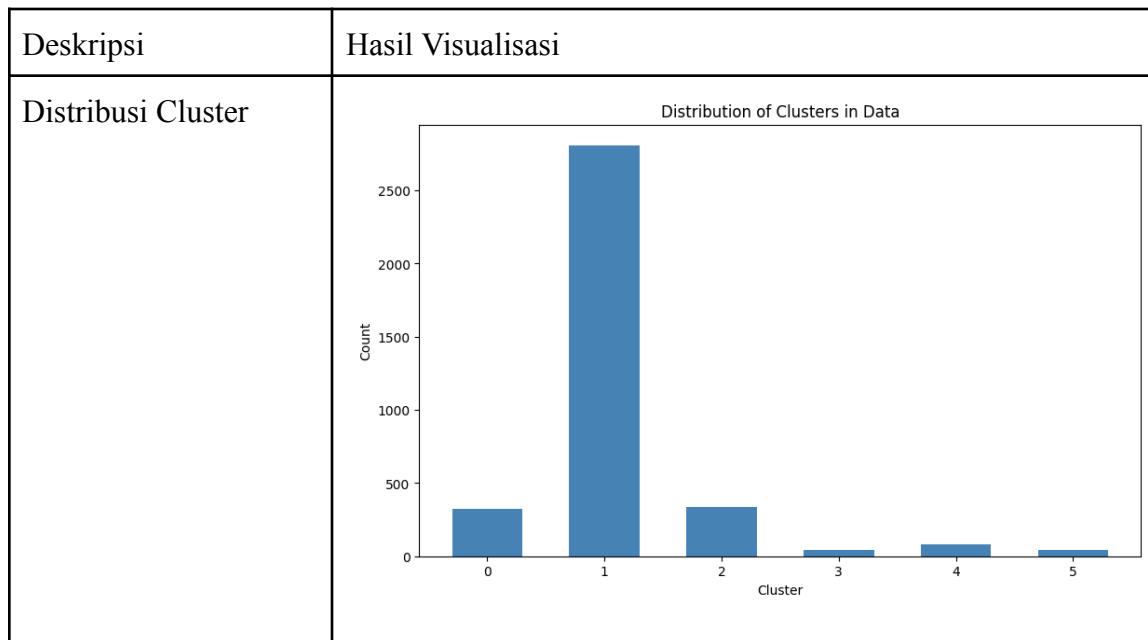
7.3.1 Hasil Clustering dengan DBSCAN

Berdasarkan hasil analisis clustering dengan DBSCAN, teridentifikasi enam cluster penggunaan listrik yang berbeda:

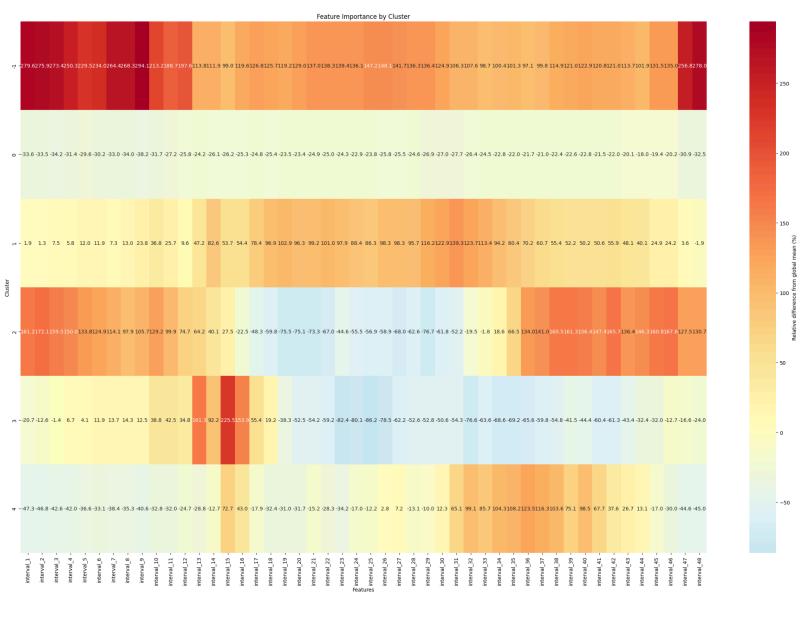
1. Cluster 0 (2809 rumah) merupakan kelompok terbesar dengan pola konsumsi listrik yang konsisten di bawah rata-rata. Konsumsi terendah terjadi di pagi hari (interval 3, 8, 9) dengan penggunaan 34-38% di bawah rata-rata, sementara penggunaan tertinggi di malam hari (interval 43-45) masih 18-20% di bawah rata-rata. Kelompok ini kemungkinan terdiri dari rumah-rumah tanpa EV atau dengan pola konsumsi listrik yang efisien.
2. Cluster 1 (339 rumah) menunjukkan lonjakan konsumsi signifikan pada sore hari, khususnya interval 30-32 dengan penggunaan 122-139% di atas rata-rata. Konsumsi pada awal dan akhir hari relatif normal (1.3-1.9% di atas rata-rata). Pola ini mungkin mengindikasikan rumah-rumah yang mengisi EV setelah jam kerja.
3. Cluster 2 (42 rumah) memiliki karakteristik konsumsi sangat tinggi di awal hari dan malam hari (interval 2, 42, 46) mencapai 165-172% di atas rata-rata, namun sangat rendah di siang hari (interval 19, 20, 29) hingga 75-77% di bawah rata-rata. Ini mungkin merepresentasikan pola penggunaan industri atau komersial spesifik.
4. Cluster 3 (84 rumah) menunjukkan pola unik dengan konsumsi ekstrem tinggi di siang hari (interval 13, 15, 16) mencapai 153-225% di atas rata-rata, namun sangat rendah pada interval 23-25 (80-86% di bawah rata-rata). Kelompok ini mungkin terdiri dari bangunan komersial atau kantor.

5. Cluster 4 (44 rumah) memperlihatkan konsumsi tinggi di sore menjelang malam (interval 35-37) dengan 108-123% di atas rata-rata, namun rendah di awal hari (interval 1-2) hingga 46-47% di bawah rata-rata. Ini mungkin menunjukkan rumah-rumah dengan pola pengisian EV di sore hari.
6. Cluster -1 (325 rumah) menunjukkan konsumsi sangat tinggi di awal dan akhir hari (interval 1, 9, 48) mencapai 278-294% di atas rata-rata, dengan konsumsi terendah masih cukup tinggi (97-99% di atas rata-rata). Kelompok ini kemungkinan merupakan pengguna listrik besar seperti industri atau fasilitas komersial besar.

Distribusi ini memiliki implikasi penting bagi manajemen jaringan listrik. Dengan mayoritas rumah berada di Cluster 0 yang memiliki konsumsi rendah, penyedia listrik perlu fokus pada manajemen beban dari cluster-cluster lain yang memiliki pola konsumsi tinggi pada waktu-waktu tertentu. Berikut adalah detail hasil dari implementasi cluster DBSCAN.



Matriks feature importance



7.3.2 Hasil Clustering dengan K-Means

Berdasarkan hasil analisis clustering dengan K-means, teridentifikasi enam cluster penggunaan listrik yang berbeda:

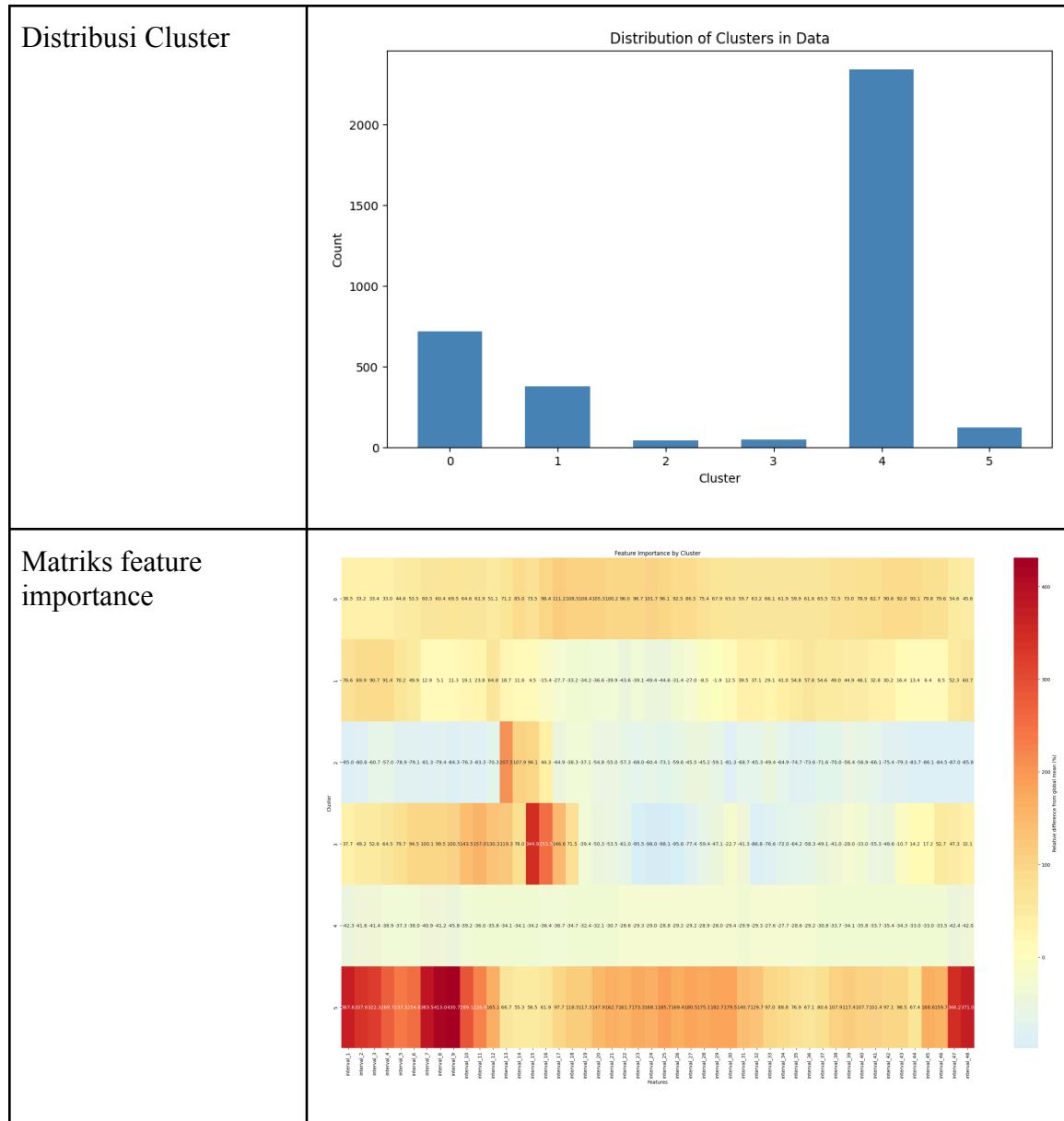
1. Cluster 0 (718 rumah) menunjukkan pola konsumsi listrik yang tinggi pada siang hari, khususnya pada interval 17-19 dengan penggunaan sekitar 108-111% di atas rata-rata. Namun, konsumsi pada pagi hari (interval 2-4) relatif rendah, hanya sekitar 33% di atas rata-rata. Pola ini mungkin mengindikasikan rumah-rumah yang mengisi EV mereka selama jam kerja atau siang hari.
2. Cluster 1 (376 rumah) memiliki karakteristik konsumsi tinggi di pagi hari, dengan penggunaan sekitar 90% di atas rata-rata pada interval 2-4. Sebaliknya, konsumsi menurun signifikan pada siang hari (interval 22-25) hingga 43-49% di bawah rata-rata. Ini mungkin merepresentasikan rumah-rumah yang mengisi EV sebelum berangkat kerja.
3. Cluster 2 (40 rumah) menunjukkan pola unik dengan konsumsi sangat tinggi pada interval 13-15 (mencapai 207% di atas rata-rata), namun sangat rendah di malam hari (interval 45-48) hingga 85-87% di bawah rata-rata. Kelompok kecil ini mungkin memiliki pola penggunaan listrik yang sangat spesifik.

STEI- ITB	<nomor dokumen>	Halaman 36 dari 40 halaman
Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB.		

4. Cluster 3 (44 rumah) memperlihatkan lonjakan konsumsi ekstrem pada interval 15-16 (hingga 344% di atas rata-rata) dan interval 11 (157% di atas rata-rata), namun sangat rendah pada interval 24-26 (sekitar 95-98% di bawah rata-rata). Ini mungkin mengindikasikan penggunaan industri atau komersial khusus.
5. Cluster 4 (2344 rumah) merupakan kelompok terbesar dengan pola konsumsi yang relatif stabil dan di bawah rata-rata, khususnya pada interval 29-34 (sekitar 27-28% di bawah rata-rata) dan lebih rendah lagi di awal hari. Kelompok ini kemungkinan terdiri dari rumah-rumah tanpa EV atau dengan pola konsumsi listrik yang efisien.
6. Cluster 5 (121 rumah) menunjukkan konsumsi ekstrem tinggi di pagi hari, terutama pada interval 7-9 (383-430% di atas rata-rata), namun relatif normal pada interval 14-16 (55-61% di atas rata-rata). Pola ini mungkin menunjukkan penggunaan industri atau komersial yang intensif di pagi hari.

Distribusi ini memiliki implikasi penting bagi manajemen jaringan listrik. Dengan adanya variasi yang signifikan dalam pola konsumsi dan ukuran cluster, penyedia listrik perlu mengembangkan strategi yang lebih kompleks untuk manajemen beban. Rekomendasi meliputi penerapan tarif dinamis berdasarkan waktu penggunaan, penguatan infrastruktur yang ditargetkan di area dengan konsentrasi tinggi cluster berkebutuhan tinggi, dan pengembangan program manajemen permintaan yang disesuaikan dengan karakteristik masing-masing cluster. Berikut adalah detail hasil dari implementasi cluster k-means.

Deskripsi	Hasil Visualisasi
-----------	-------------------



8. Saran dan Perbaikan

Berikut beberapa saran dan perbaikan yang dapat dilakukan:

a. Evaluasi Clustering dengan Metrik Internal dan Eksternal

Sebagai bagian dari analisis lebih lanjut, disarankan untuk mengevaluasi hasil clustering menggunakan metrik evaluasi internal seperti Silhouette Coefficient, Dunn Index, dan Cohesion and Separation. Metrik ini dapat membantu memastikan bahwa cluster yang dihasilkan memiliki kekompakan internal yang tinggi dan pemisahan antar cluster yang jelas. Selain itu, metrik eksternal seperti Purity, Entropy, dan Rand Index juga dapat diterapkan jika tersedia data label ground truth untuk membandingkan hasil clustering dengan klasifikasi aktual. Evaluasi yang komprehensif akan memberikan pandangan yang lebih jelas mengenai keefektifan metode clustering yang digunakan dan memastikan hasil yang sesuai dengan kebutuhan domain.

b. Optimalisasi Parameter Model

Proses clustering seperti K-Means dan DBSCAN memerlukan pemilihan parameter yang optimal, seperti jumlah cluster k untuk K-Means dan parameter eps serta minPts untuk DBSCAN. Penggunaan pendekatan berbasis data seperti Elbow Method untuk K-Means dan metode k-distance graph untuk DBSCAN sangat disarankan. Penyesuaian parameter yang cermat akan meningkatkan akurasi dan kegunaan hasil clustering dalam mendeteksi pola konsumsi listrik.

c. Studi Lanjutan tentang Distribusi Data

Hasil analisis menunjukkan adanya outliers yang signifikan dalam data. Penelitian lebih lanjut diperlukan untuk memahami apakah outliers tersebut adalah bagian dari pola konsumsi yang valid atau hanya merupakan anomali data. Metode berbasis density seperti DBSCAN sudah efektif dalam menangani noise, namun penggunaan teknik clustering lainnya seperti Gaussian Mixture Model (GMM) atau Agglomerative Clustering dapat dipertimbangkan untuk mengevaluasi potensi struktur data lainnya yang mungkin tidak terlihat dengan metode saat ini.

9. Kontribusi Anggota

Berikut adalah tabel dari kontribusi kelompok

No	Nama	Kontribusi
1	Auvarifqi Putra Diandra	<ul style="list-style-type: none">- Mengerjakan Logistic Regression- Mengerjakan K-Means- Mengerjakan EDA- Mengerjakan Ensemble Voting
2	Ahmad Rizki	<ul style="list-style-type: none">- Mengerjakan K Nearest Neighbour- Mengerjakan DBSCAN- Mengerjakan Data Preprocessing- Mengerjakan Decision Tree