

Smart Home

dan Analitik Konsumsi Energi

Metode
Multi-Clustering
untuk Optimasi
dan Evaluasi
Algoritma



Teti Desyani, M.Kom
Syaeful Machfud, M.Kom

SMART HOME *dan* **ANALITIK KONSUMSI ENERGI**

**Metode Multi-Clustering untuk Optimasi
dan Evaluasi Algoritma**

Teti Desyani, M.Kom
Syaeful Machfud, M.Kom



PENERBIT KBM INDONESIA

adalah penerbit dengan misi memudahkan proses penerbitan buku-buku penulis di tanah air Indonesia, serta menjadi media *sharing* proses penerbitan buku.

SMART HOME DAN ANALITIK KONSUMSI ENERGI
Metode Multi-Clustering untuk Optimasi dan
Evaluasi Algoritma

Copyright @ 2026 By Teti Desyani, M.Kom & Syaeful Machfud, M.Kom

All right reserved

Penulis

Teti Desyani, M.Kom
Syaeful Machfud, M.Kom

Desain Sampul

Eca Art

Tata Letak

Husnud Diniyah

Editor

Dr. Muhamad Husein Maruahey, Drs., M.Sc.
Background isi buku di ambil dari <https://www.freepik.com/>

Official

Depok, Sleman-Jogjakarta (Kantor)

Penerbit Karya Bakti Makmur (KBM) Indonesia

Anggota IKAPI/No. IKAPI 279/JTI/2021

081357517526 (Tlpn/WA)

Website

<https://penerbitkbm.com>
www.penerbitbukumurah.com

Penjualan Buku

Website: <https://penerbitkbm.com/toko-buku/>

Shopee: @penerbitKBM

Youtube

@PenerbitKBMOfficial

Tiktok

@penerbitkbm

Email

naskah@penerbitkbm.com

ISBN: 978-634-278-463-1

Cetakan ke-1, Juni 2026
15,5 x 23 cm, iv + 126 halaman

Isi buku diluar tanggungjawab penerbit
Hak cipta merek KBM Indonesia sudah terdaftar di DJKI-Kemenkumham
dan isi buku dilindungi undang-undang.

Dilarang keras menerjemahkan, memfotokopi, atau
memperbanyak sebagian atau seluruh isi buku ini
tanpa seizin penerbit karena beresiko sengketa hukum

Sanksi Pelanggaran Pasal 113

Undang-Undang No. 28 Tahun 2014 Tentang Hak Cipta

- i. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp 100. 000. 000 (seratus juta rupiah).
- ii. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp 500. 000. 000,00 (lima ratus juta rupiah).
- iii. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf a, huruf b, huruf e, dan/atau huruf g untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan/atau pidana denda paling banyak Rp 1. 000. 000. 000,00 (satu miliar rupiah).
- iv. Setiap Orang yang memenuhi unsur sebagaimana dimaksud pada ayat (3) yang dilakukan dalam bentuk pembajakan, dipidana dengan pidana penjara paling lama 10 (sepuluh) tahun dan/atau pidana denda paling banyak Rp 4. 000. 000. 000,00 (empat miliar rupiah).

KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya, sehingga buku yang berjudul “***Smart Home dan Analitik Konsumsi Energi: Metode Multi-Clustering untuk Optimasi dan Evaluasi Algoritma***” ini dapat diselesaikan dengan baik.

Perkembangan teknologi Internet of Things (IoT) dan sistem *smart home* telah menghasilkan data dalam jumlah besar yang berkaitan dengan konsumsi energi rumah tangga. Data tersebut tidak hanya mencerminkan penggunaan energi, tetapi juga memuat informasi penting mengenai pola aktivitas penghuni, efisiensi penggunaan perangkat, serta pengaruh faktor lingkungan seperti cuaca. Oleh karena itu, diperlukan pendekatan analisis data yang tepat untuk menggali informasi yang bernilai dari data tersebut.

Buku ini disusun dengan tujuan untuk memberikan pemahaman komprehensif mengenai penerapan teknik clustering dalam menganalisis pola konsumsi energi pada *smart home*. Secara khusus, buku ini mengkaji dan membandingkan beberapa algoritma clustering, yaitu K-Means, Fuzzy C-Means, DBSCAN, dan Hierarchical Clustering, serta dilengkapi dengan metode evaluasi kinerja seperti Silhouette Score, Davies-Bouldin Index, dan Calinski-Harabasz Index.

Melalui pendekatan multi-clustering, pembaca diharapkan dapat memahami kelebihan dan keterbatasan masing-masing algoritma dalam mengelompokkan data energi, serta memperoleh wawasan yang lebih mendalam mengenai pola konsumsi energi yang terjadi. Selain itu, buku ini juga menyajikan analisis berbasis data yang dapat digunakan sebagai dasar dalam pengambilan keputusan untuk meningkatkan efisiensi energi rumah tangga.

Penulis menyadari bahwa buku ini masih memiliki keterbatasan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi penyempurnaan di masa yang akan datang. Semoga buku ini dapat memberikan manfaat bagi akademisi, peneliti, mahasiswa, maupun praktisi yang tertarik pada bidang *Data Science*, machine learning, dan sistem *smart home*.

Akhir kata, penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan dalam penyusunan buku ini.

Penulis

DAFTAR ISI

KATA PENGANTAR.....	i
DAFTAR ISI	iii
BAB 1 PENDAHULUAN <i>SMART HOME</i> DAN ANALISIS ENERGI.....	1
BAB 2 DASAR TEORI <i>SMART HOME</i> DAN IOT	5
2.1 Konsep <i>Smart Home</i> dan <i>Internet of Things</i> (IoT)	5
2.2 Konsumsi Energi Rumah Tangga	6
2.3 Data Mining dan <i>Data Science</i>	7
2.4 Konsep <i>Clustering</i>	8
2.5 Algoritma <i>Clustering</i>	9
2.6 Metode Evaluasi <i>Clustering</i>	51
BAB 3 DATASET DAN KARAKTERISTIK DATA.....	55
3.1 Deskripsi Dataset <i>Smart Home</i>	55
3.2 Variabel yang Digunakan	56
3.3 Statistik Deskriptif Awal.....	59
BAB 4 DATA PREPROCESSING.....	63
4.1 Penanganan <i>Missing Value</i>	63
4.2 <i>Data Cleaning</i>	66
4.3 Transformasi Data	67
4.4 Normalisasi dan Standarisasi Data.....	67
4.5 Seleksi Fitur Awal	68
BAB 5 CLUSTERING MULTI-ALGORITMA	69
5.1 Skenario Eksperimen.....	69
5.2 Implementasi Algoritma K-Means	70

5.3 Implementasi <i>Fuzzy C-Means</i>	73
5.4 Implementasi DBSCAN.....	75
5.5 Implementasi <i>Hierarchical Clustering</i>	77
BAB 6 EVALUASI DAN PERBANDINGAN KINERJA <i>CLUSTERING</i>	83
6.1 Metode Evaluasi <i>Clustering</i>	83
6.2 Hasil Evaluasi <i>Clustering</i>	85
6.3 Analisis Perbandingan Metode.....	87
6.4 Penentuan Metode Terbaik	89
6.5 Kesimpulan	90
DAFTAR PUSTAKA	93
GLOSARIUM	97
LAMPIRAN	101
PROFIL PENULIS	123

BAB 1



PENDAHULUAN *SMART HOME* DAN ANALISIS ENERGI

Bab ini membahas latar belakang mengenai analisis konsumsi energi pada *smart home* menggunakan pendekatan clustering. Selain itu dijelaskan rumusan masalah, tujuan penelitian, manfaat penelitian, serta gambaran umum metode clustering yang digunakan dalam buku ini.

Perkembangan teknologi digital dalam beberapa dekade terakhir telah mendorong lahirnya konsep *smart home*, yaitu sistem hunian cerdas yang memanfaatkan teknologi Internet of Things (IoT) untuk mengotomatisasi dan mengoptimalkan berbagai aktivitas rumah tangga. Dalam sistem ini, berbagai perangkat seperti peralatan dapur, sistem pemanas, pendingin, serta perangkat elektronik lainnya terhubung dan menghasilkan data secara kontinu. Data tersebut tidak

hanya mencerminkan aktivitas penggunaan energi, tetapi juga menyimpan informasi penting mengenai pola perilaku penghuni rumah.¹

Seiring dengan meningkatnya kebutuhan energi dan kesadaran terhadap efisiensi energi, analisis terhadap konsumsi energi rumah tangga menjadi isu yang semakin penting. Penggunaan energi yang tidak terkontrol dapat menyebabkan pemborosan biaya serta berdampak pada peningkatan emisi karbon. Oleh karena itu, diperlukan pendekatan berbasis data untuk memahami pola konsumsi energi secara lebih mendalam, sehingga dapat dihasilkan strategi yang efektif untuk meningkatkan efisiensi energi.²

Dataset *smart home* modern umumnya tidak hanya mencakup data konsumsi energi dari berbagai perangkat, tetapi juga dilengkapi dengan data lingkungan seperti suhu, kelembaban, tekanan udara, dan kondisi cuaca lainnya. Integrasi antara data energi dan data lingkungan membuka peluang untuk melakukan analisis yang lebih komprehensif, termasuk dalam mengidentifikasi faktor-faktor yang memengaruhi penggunaan energi.^{3,4}

Dalam konteks analisis data, teknik *clustering* merupakan salah satu metode yang banyak digunakan untuk mengelompokkan data berdasarkan kemiripan karakteristik. Metode ini sangat relevan untuk mengidentifikasi pola konsumsi energi tanpa memerlukan label data (*unsupervised learning*). Berbagai algoritma clustering seperti K-Means, Fuzzy C-Means, DBSCAN, dan *Hierarchical Clustering* memiliki karakteristik dan pendekatan yang berbeda dalam membentuk kelompok data.⁵

Namun demikian, tidak ada satu algoritma clustering yang secara universal paling unggul untuk semua jenis data. Oleh karena itu, diperlukan pendekatan *multi-clustering* yang mengombinasikan beberapa algoritma untuk memperoleh hasil yang lebih optimal dan komprehensif. Selain itu, evaluasi terhadap kualitas hasil clustering juga menjadi aspek penting yang harus diperhatikan. Penggunaan indeks

validasi seperti Silhouette Score, Davies-Bouldin Index, dan Calinski-Harabasz Index dapat memberikan gambaran objektif mengenai kinerja masing-masing algoritma.⁴

Berdasarkan latar belakang tersebut, penelitian ini bertujuan untuk menganalisis pola konsumsi energi pada *smart home* dengan menggunakan pendekatan multi-clustering serta melakukan evaluasi kinerja algoritma untuk menentukan metode yang paling sesuai dalam mengelompokkan data energi.

DAFTAR PUSTAKA

1. Eugene Prince, M. *et al.* Quantum-enhanced neuro-fusion framework for intelligent decision-making in *Smart Home* IoT surveillance. *Franklin Open* **15**, (2026).
2. Leonori, S., Rocca, R., Ricci, M. & Frattale Mascioli, F. M. *Smart Home* energy flow optimization, a practical case study. *Energy Conversion and Management: X* **28**, (2025).
3. Rodrigues, T., Morgado, J., Barros, M., Oliveira De Sá, A. & Cecílio, J. AI-driven IoT recommender system for enhancing energy efficient management in smart houses. *Expert Syst. Appl.* **296**, (2026).
4. Liu, L. & Zheng, F. An improved cohesive hierarchical clustering for indoor air quality monitoring in smart gymnasium with healthy sport areas. *Alexandria Engineering Journal* **105**, 204–217 (2024).
5. Aghdam, F. H., Rasti, M., Pongracz, E. & Anvari-Moghaddam, A. Digital twin assisted real-time energy management system for *Smart Homes*. *Energy Build.* **355**, 116978 (2026).
6. Alkanan, K., Zhong, Y., Luo, X. & Dong, R. Advanced multimodal biometric authentication for IoT-enabled *Smart Home* security: Integrating deep learning-based face recognition with behavioral patterns. *Kuwait Journal of Science* **53**, (2026).
7. Franco, P., Martinez, J. M., Kim, Y. C. & Ahmed, M. A. IoT Based Approach for Load Monitoring and Activity Recognition in *Smart Homes*. *IEEE Access* **9**, 45325–45339 (2021).

8. Ngo, M. H., Do, H. H., Pham, H. D., Do, Q. H. & Nguyen, H. S. Anomaly Detection Based on Frequent Pattern Mining in *Smart Home* Devices. *IEEE Access* **13**, 161436–161452 (2025).
9. Shahcheraghian, A., Ilinca, A. & Sommerfeldt, N. K-means and agglomerative clustering for source-load mapping in distributed district heating planning. *Energy Conversion and Management: X* **25**, 100860 (2025).
10. Liu, X., Vasantha, G., Goh, K., Lin, W. & Marzano, A. Data-driven smart product design, smart service design, and smart product-service system design – A comprehensive review. *Advanced Engineering Informatics* vol. 73 Preprint at <https://doi.org/10.1016/j.aei.2026.104534> (2026).
11. Fernandez-Basso, C., Díaz-Jimenez, D., López, J. L. & Espinilla, M. Fuzzy processing applied to improve multimodal sensor data fusion to discover frequent behavioral patterns for smart healthcare. *Information Fusion* **123**, (2025).
12. Rosyani, P., Suhendi, A., Apriyanti, D. H. & Waskita, A. A. Color Features Based Flower Image Segmentation Using K-Means and Fuzzy C-Means. *Building of Informatics, Technology and Science (BITS)* **3**, 253–259 (2021).
13. Sträng, H. & Dinh, T. Silhouette-upper-bound: An upper bound on the silhouette evaluation metric for clustering. *Software Impacts* **28**, 100835 (2026).
14. Firnando, R., Nurmaini, S. & Naufal Rachmatullah, M. Clustering Data Bibliografi menggunakan Algoritma DBSCAN dengan Author Matching Classifier Berbasis Deep Neural Network. *JSI: Jurnal Sistem Informasi (E-Journal)* **14**, 2022.
15. Bermejo, U., Almeida, A., Bilbao-Jayo, A. & Azkune, G. Embedding-based real-time change point detection with application to activity segmentation in *Smart Home* time series data. *Expert Syst. Appl.* **185**, (2021).

16. Lin, Y. W., Lin, Y. B., Wu, Y. F. & Shen, P. H. VoiceTalk: A No-Code Approach for Creating Voice-Controlled *Smart Home* Applications. *IEEE Open Journal of the Computer Society* **6**, 874–883 (2025).
17. Nimmy, K., Sankaran, S., Achuthan, K. & Calyam, P. Lightweight and Privacy-Preserving Remote User Authentication for *Smart Homes*. *IEEE Access* **10**, 176–190 (2022).
18. Sträng, H. & Dinh, T. An upper bound of the silhouette validation metric for clustering. <https://doi.org/10.1016/j.patcog.2026.113402> (2025) doi:10.1016/j.patcog.2026.113402.
19. Huang, Y., Jiang, J., Gao, Z., Sun, H. & Gao, Z. PIUL-DERs: Physics-informed pseudo-labeling unsupervised learning for *Smart Homes* with distributed energy resources. *Appl. Energy* **403**, (2026).
20. Devanathan, B. *et al.* AI-Based energy consumption forecasting in *Smart Homes* using multi-Algorithm analysis across temporal resolutions. *Results in Engineering* **29**, (2026).
21. Kumar, A. *et al.* Imputation of missing data for domain mean estimation using simple random sampling. *Kuwait Journal of Science* **52**, 1–10 (2025).
22. Guo, Y. & Holy, T. E. Recovering Missing Features in Nonnegative Matrix Factorization via Generalized Singular Value Decomposition. *iScience* 114708 (2026) doi:10.1016/j.isci.2026.114708.
23. de Sousa Guilhon Araujo, M. A., Cyrino Oliveira, F. L. & Souza, R. C. Missing data imputation using climate data reanalysis in solar energy generation time series. *Energy Conversion and Management: X* **30**, 101653 (2026).

24. Downing, N. J. *Missing Value* imputation in environmental, social, and governance data: an impact on emissions scores. *Financ. Res. Lett.* **85**, 1–10 (2025).
25. Yu, Y. *et al.* Dynamic Dual-Channel Asynchronous Graph Neural Network for Traffic Prediction with *Missing Values*. *Fundamental Research* <https://doi.org/10.1016/j.fmre.2026.01.019> (2026) doi:10.1016/j.fmre.2026.01.019.
26. Li, W., Subašić, G., Korolija, I., Guida, R. & Hong, S. M. Data imputation methods for missing U-values of building envelopes in building performance database. *Journal of Building Engineering* **118**, (2026).

GLOSARIUM

Istilah	Penjelasan
Agglomerative Clustering	Metode hierarchical clustering yang membentuk cluster dari bawah ke atas dengan menggabungkan cluster terdekat secara bertahap.
Artificial Intelligence (AI)	Teknologi kecerdasan buatan yang memungkinkan sistem komputer meniru kemampuan berpikir manusia.
Calinski-Harabasz Index	Metode evaluasi clustering yang mengukur rasio variansi antar cluster terhadap variansi dalam cluster.
Centroid	Titik pusat cluster yang diperoleh dari rata-rata anggota cluster.
Cluster	Kelompok data yang memiliki karakteristik atau pola yang mirip.
Clustering	Teknik pengelompokan data berdasarkan tingkat kemiripan antar data tanpa menggunakan label kelas.
CloudCover	Variabel cuaca yang menunjukkan tingkat tutupan awan pada lingkungan.
Confusion Matrix	Tabel evaluasi yang digunakan untuk melihat performa klasifikasi atau clustering terhadap data aktual.
Data Mining	Proses استخراج atau penemuan pola dan informasi penting dari kumpulan data dalam jumlah besar.

Istilah	Penjelasan
Data Preprocessing	Tahapan awal pengolahan data sebelum analisis dilakukan, seperti cleaning, normalisasi, dan transformasi data.
<i>Data Science</i>	Bidang ilmu yang menggabungkan statistik, machine learning, dan pemrograman untuk menganalisis data.
Davies-Bouldin Index	Indeks evaluasi clustering yang mengukur tingkat kemiripan antar cluster. Nilai lebih kecil menunjukkan kualitas cluster yang lebih baik.
DBSCAN	Algoritma clustering berbasis kepadatan (<i>Density-Based Spatial Clustering of Applications with Noise</i>) yang mampu mendeteksi noise dan outlier.
Dendrogram	Diagram berbentuk pohon yang digunakan untuk memvisualisasikan hasil hierarchical clustering.
Dew Point	Titik embun yang menunjukkan suhu ketika uap air mulai mengembun menjadi cairan.
Distance Metric	Metode pengukuran jarak antar data, seperti Euclidean Distance atau Manhattan Distance.
Euclidean Distance	Metode perhitungan jarak lurus antar dua titik data dalam ruang multidimensi.
Feature Selection	Proses pemilihan variabel yang relevan untuk digunakan dalam analisis data atau machine learning.
Fuzzy C-Means (FCM)	Algoritma clustering berbasis fuzzy logic di mana setiap data dapat menjadi anggota lebih dari satu cluster.
Hierarchical Clustering	Metode clustering yang membentuk struktur cluster secara bertingkat atau hierarki.

Istilah	Penjelasan
Humidity	Variabel cuaca yang menunjukkan tingkat kelembapan udara.
Internet of Things (IoT)	Konsep perangkat fisik yang saling terhubung melalui internet dan dapat bertukar data secara otomatis.
K-Means	Algoritma clustering yang membagi data ke dalam beberapa cluster berdasarkan centroid terdekat.
Machine Learning	Cabang kecerdasan buatan yang memungkinkan sistem belajar dari data tanpa diprogram secara eksplisit.
Mean Absolute Error (MAE)	Metrik evaluasi yang mengukur rata-rata kesalahan absolut antara nilai prediksi dan nilai aktual.
Membership Degree	Nilai derajat keanggotaan suatu data terhadap cluster pada metode fuzzy clustering.
<i>Missing Value</i>	Data yang hilang atau kosong pada suatu atribut dataset.
Noise	Data yang dianggap menyimpang atau tidak termasuk ke dalam cluster tertentu.
Normalisasi Data	Proses penyamaan skala data agar seluruh variabel memiliki rentang yang sebanding.
Outlier	Data pencilan yang memiliki nilai jauh berbeda dibandingkan mayoritas data lainnya.
PCA (Principal Component Analysis)	Teknik reduksi dimensi data untuk menyederhanakan visualisasi dan analisis data multidimensi.
Precision	Metrik evaluasi yang mengukur tingkat ketepatan prediksi positif.

Istilah	Penjelasan
Recall	Metrik evaluasi yang mengukur kemampuan model dalam menemukan seluruh data positif.
Silhouette Score	Metode evaluasi clustering yang mengukur kualitas pemisahan dan kepadatan cluster.
<i>Smart Home</i>	Sistem rumah pintar yang memanfaatkan teknologi IoT untuk mengotomatisasi perangkat rumah tangga.
Standardisasi	Proses mengubah data agar memiliki rata-rata 0 dan standar deviasi 1.
Time Series	Data yang tersusun berdasarkan urutan waktu tertentu.
Unsupervised Learning	Metode pembelajaran mesin tanpa menggunakan label data sebagai acuan pelatihan.
WindSpeed	Variabel cuaca yang menunjukkan kecepatan angin.
Z-Score Normalization	Teknik standardisasi data menggunakan rata-rata dan standar deviasi.

```
import time
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler,
LabelEncoder
from sklearn.cluster import KMeans, DBSCAN,
AgglomerativeClustering
from sklearn.metrics import silhouette_score,
davies_bouldin_score, calinski_harabasz_score
from sklearn.metrics import confusion_matrix,
accuracy_score, precision_score, recall_score,
f1_score
from collections import Counter
import warnings
from scipy.cluster.hierarchy import dendrogram,
linkage
from sklearn.decomposition import PCA
import seaborn as sns

# Set style
plt.style.use('ggplot')
sns.set_palette("husl")
warnings.simplefilter("ignore")

# -----
# Data Loading and Preprocessing
# -----

print("="*70)
print("CLUSTERING COMPARISON: K-Means vs Fuzzy C-
Means vs DBSCAN vs Hierarchical")
```

```

print("="*70)

# Load dataset
file_path = '/kaggle/input/smart-home-dataset-with-
weather-information/HomeC.csv'
home_data = pd.read_csv(file_path)

# Selecting features
selected_features = ['use [kW]', 'temperature',
'humidity', 'pressure', 'windSpeed']
home_data =
home_data.dropna(subset=selected_features +
['summary'])

# Encoding summary
label_encoder = LabelEncoder()
home_data['summary_encoded'] =
label_encoder.fit_transform(home_data['summary'])

# Take sample for faster computation (5000 samples)
numeric_data_sample = home_data[selected_features +
['summary_encoded', 'summary']].sample(n=5000,
random_state=42)

# Normalize features
scaler = StandardScaler()
X_normalized =
scaler.fit_transform(numeric_data_sample[selected_f
eatures])

print(f"\nDataset shape: {X_normalized.shape}")
print(f"Number of samples: {len(X_normalized)}")
print(f"Number of features:
{X_normalized.shape[1]}")

# -----
# Define clustering methods
# -----

```

```

n_clusters = 3 # Number of clusters for K-Means,
FCM, Hierarchical
eps = 0.5      # DBSCAN parameter
min_samples = 10 # DBSCAN parameter

print(f"\nClustering parameters:")
print(f" - Number of clusters: {n_clusters}")
print(f" - DBSCAN eps: {eps}, min_samples:
{min_samples}")

# 1. K-Means Clustering
print("\n" + "="*70)
print("1. TRAINING K-MEANS CLUSTERING")
print("="*70)
start_time = time.time()
kmeans = KMeans(n_clusters=n_clusters,
random_state=42, n_init=10)
kmeans_labels = kmeans.fit_predict(X_normalized)
kmeans_time = time.time() - start_time
print(f" Time: {kmeans_time:.2f} seconds")
print(f" Cluster distribution:
{np.bincount(kmeans_labels)}")

# 2. Fuzzy C-Means Clustering
print("\n" + "="*70)
print("2. TRAINING FUZZY C-MEANS CLUSTERING")
print("="*70)

class FuzzyCMeans:
    def __init__(self, n_clusters=3, m=2,
max_iter=100, tol=1e-4, random_state=42):
        self.n_clusters = n_clusters
        self.m = m
        self.max_iter = max_iter
        self.tol = tol
        self.random_state = random_state

    def fit(self, X):
        np.random.seed(self.random_state)

```

```

        n_samples, n_features = X.shape

        # Initialize membership matrix
        self.U = np.random.rand(n_samples,
self.n_clusters)
        self.U = self.U / np.sum(self.U, axis=1,
keepdims=True)

        for iteration in range(self.max_iter):
            # Calculate centroids
            centers = []
            for j in range(self.n_clusters):
                um = self.U[:, j] ** self.m
                center = np.dot(um, X) / np.sum(um)
                centers.append(center)
            self.centers_ = np.array(centers)

            # Update membership matrix
            new_U = np.zeros_like(self.U)
            for i in range(n_samples):
                for j in range(self.n_clusters):
                    dist = np.linalg.norm(X[i] -
self.centers_[j])

                    if dist == 0:
                        new_U[i, j] = 1
                    else:
                        denominator = sum((dist /
np.linalg.norm(X[i] - self.centers_[k])) **
(2/(self.m-1))
                                for k in
range(self.n_clusters))
                        new_U[i, j] = 1 /
denominator

            # Check convergence
            if np.max(np.abs(new_U - self.U)) <
self.tol:
                print(f"        Converged after
{iteration+1} iterations")

```

```

        break

        self.U = new_U

        # Get hard labels
        self.labels_ = np.argmax(self.U, axis=1)
        return self

    def predict(self, X):
        n_samples = X.shape[0]
        U_pred = np.zeros((n_samples,
self.n_clusters))
        for i in range(n_samples):
            for j in range(self.n_clusters):
                dist = np.linalg.norm(X[i] -
self.centers_[j])
                if dist == 0:
                    U_pred[i, j] = 1
                else:
                    denominator = sum((dist /
np.linalg.norm(X[i] - self.centers_[k])) **
(2/(self.m-1))
                                for k in
range(self.n_clusters))
                    U_pred[i, j] = 1 / denominator
        return np.argmax(U_pred, axis=1)

start_time = time.time()
fcm = FuzzyCMeans(n_clusters=n_clusters, m=2,
max_iter=100)
fcm_labels = fcm.fit(X_normalized).labels_
fcm_time = time.time() - start_time
print(f" Time: {fcm_time:.2f} seconds")
print(f" Cluster distribution:
{np.bincount(fcm_labels)}")

# 3. DBSCAN Clustering
print("\n" + "="*70)
print("3. TRAINING DBSCAN CLUSTERING")

```

```

print("="*70)
start_time = time.time()
dbscan = DBSCAN(eps=eps, min_samples=min_samples,
metric='euclidean')
dbscan_labels = dbscan.fit_predict(X_normalized)
dbscan_time = time.time() - start_time

n_dbscan_clusters = len(set(dbscan_labels)) - (1 if
-1 in dbscan_labels else 0)
n_noise = list(dbscan_labels).count(-1)
print(f"   Time: {dbscan_time:.2f} seconds")
print(f"   Number of clusters: {n_dbscan_clusters}")
print(f"   Noise points: {n_noise}
({n_noise/len(dbscan_labels)*100:.1f}%)")
if n_dbscan_clusters > 0:
    print(f"   Cluster distribution:
{np.bincount(dbscan_labels[dbscan_labels != -1])}")

# 4. Hierarchical Clustering
print("\n" + "="*70)
print("4. TRAINING HIERARCHICAL CLUSTERING")
print("="*70)
start_time = time.time()
hierarchical =
AgglomerativeClustering(n_clusters=n_clusters,
linkage='ward')
hierarchical_labels =
hierarchical.fit_predict(X_normalized)
hierarchical_time = time.time() - start_time
print(f"   Time: {hierarchical_time:.2f} seconds")
print(f"   Cluster distribution:
{np.bincount(hierarchical_labels)}")

# -----
# Evaluation Metrics
# -----

print("\n" + "="*70)
print("CLUSTERING EVALUATION METRICS")

```

```

print("="*70)

def evaluate_clustering(labels, X, method_name):
    """Calculate clustering evaluation metrics"""
    # Filter out noise points for DBSCAN
    if method_name == "DBSCAN" and -1 in labels:
        mask = labels != -1
        labels_filtered = labels[mask]
        X_filtered = X[mask]
        if len(labels_filtered) < 2:
            return {'Silhouette': np.nan, 'Davies-
Bouldin': np.nan,
                    'Calinski-Harabasz': np.nan,
                    'Valid_samples': len(labels_filtered)}
        else:
            labels_filtered = labels
            X_filtered = X
            mask = slice(None)

    results = {}
    results['Valid_samples'] = len(labels_filtered)

    # Calculate metrics only if we have at least 2
clusters and 2 samples
    unique_labels = len(set(labels_filtered))
    if unique_labels >= 2 and len(labels_filtered)
>= 2:
        try:
            results['Silhouette'] =
silhouette_score(X_filtered, labels_filtered)
        except:
            results['Silhouette'] = np.nan

        try:
            results['Davies-Bouldin'] =
davies_bouldin_score(X_filtered, labels_filtered)
        except:
            results['Davies-Bouldin'] = np.nan

```

```

        try:
            results['Calinski-Harabasz'] =
calinski_harabasz_score(X_filtered,
labels_filtered)
        except:
            results['Calinski-Harabasz'] = np.nan
    else:
        results['Silhouette'] = np.nan
        results['Davies-Bouldin'] = np.nan
        results['Calinski-Harabasz'] = np.nan

    return results

# Evaluate all methods
methods = {
    'K-Means': kmeans_labels,
    'Fuzzy C-Means': fcm_labels,
    'DBSCAN': dbscan_labels,
    'Hierarchical': hierarchical_labels
}

results_df = pd.DataFrame()
for method_name, labels in methods.items():
    eval_results = evaluate_clustering(labels,
X_normalized, method_name)
    eval_results['Method'] = method_name
    results_df = pd.concat([results_df,
pd.DataFrame([eval_results])], ignore_index=True)

# Display results
print("\n" + "="*70)
print("EVALUATION RESULTS")
print("="*70)
print(results_df.to_string(index=False))
print("\nMetric Interpretations:")
print("    • Silhouette Score: [-1, 1] → Higher is
better (good separation)")
print("    • Davies-Bouldin Index: [0, ∞) → Lower is
better (compact clusters)")

```

```
print(" • Calinski-Harabasz Index: [0, ∞) → Higher  
is better (variance ratio)")
```

```
# -----  
# Visualization 1: Bar Chart Comparison  
# -----
```

```
fig, axes = plt.subplots(2, 2, figsize=(15, 12),  
facecolor='white')
```

```
# 1a. Silhouette Score Comparison  
ax1 = axes[0, 0]  
sil_scores = results_df[['Method',  
'Silhouette']].dropna()  
colors_bar = ['#FF6B6B', '#4ECDC4', '#45B7D1',  
'#96CEB4']  
bars = ax1.bar(sil_scores['Method'],  
sil_scores['Silhouette'], color=colors_bar,  
edgecolor='black', linewidth=1.5)  
ax1.set_ylabel('Silhouette Score', fontsize=12,  
fontweight='bold')  
ax1.set_title('Silhouette Score Comparison\n(Higher  
is Better)', fontsize=14, fontweight='bold')  
ax1.set_ylim([-0.5, 1])  
ax1.axhline(y=0, color='black', linestyle='-',  
linewidth=0.5)  
ax1.grid(True, alpha=0.3, axis='y')
```

```
# Add value labels  
for bar, score in zip(bars,  
sil_scores['Silhouette']):  
    if not np.isnan(score):  
        height = bar.get_height()  
        ax1.text(bar.get_x() + bar.get_width()/2.,  
height + 0.02,  
f'{score:.3f}', ha='center',  
va='bottom', fontweight='bold')
```

```
# 1b. Davies-Bouldin Index Comparison
```

```

ax2 = axes[0, 1]
db_scores = results_df[['Method', 'Davies-
Bouldin']].dropna()
bars = ax2.bar(db_scores['Method'],
db_scores['Davies-Bouldin'], color=colors_bar,
edgecolor='black', linewidth=1.5)
ax2.set_ylabel('Davies-Bouldin Index', fontsize=12,
fontweight='bold')
ax2.set_title('Davies-Bouldin Index
Comparison\n(Lower is Better)', fontsize=14,
fontweight='bold')
ax2.grid(True, alpha=0.3, axis='y')

for bar, score in zip(bars, db_scores['Davies-
Bouldin']):
    if not np.isnan(score):
        height = bar.get_height()
        ax2.text(bar.get_x() + bar.get_width()/2.,
height + 0.02,
                f'{score:.3f}', ha='center',
va='bottom', fontweight='bold')

# 1c. Calinski-Harabasz Index Comparison
ax3 = axes[1, 0]
ch_scores = results_df[['Method', 'Calinski-
Harabasz']].dropna()
bars = ax3.bar(ch_scores['Method'],
ch_scores['Calinski-Harabasz'], color=colors_bar,
edgecolor='black', linewidth=1.5)
ax3.set_ylabel('Calinski-Harabasz Index',
fontsize=12, fontweight='bold')
ax3.set_title('Calinski-Harabasz Index
Comparison\n(Higher is Better)', fontsize=14,
fontweight='bold')
ax3.grid(True, alpha=0.3, axis='y')

for bar, score in zip(bars, ch_scores['Calinski-
Harabasz']):
    if not np.isnan(score):

```

```

        height = bar.get_height()
        ax3.text(bar.get_x() + bar.get_width()/2.,
height + height*0.02,
                f'{score:.0f}', ha='center',
va='bottom', fontweight='bold')

# 1d. Computation Time Comparison
ax4 = axes[1, 1]
times = [kmeans_time, fcm_time, dbscan_time,
hierarchical_time]
methods_names = ['K-Means', 'Fuzzy C-Means',
'DBSCAN', 'Hierarchical']
bars = ax4.bar(methods_names, times,
color=colors_bar, edgecolor='black', linewidth=1.5)
ax4.set_ylabel('Time (seconds)', fontsize=12,
fontweight='bold')
ax4.set_title('Computation Time Comparison',
fontsize=14, fontweight='bold')
ax4.set_yscale('log')
ax4.grid(True, alpha=0.3, axis='y')

for bar, time_val in zip(bars, times):
    height = bar.get_height()
    ax4.text(bar.get_x() + bar.get_width()/2.,
height + height*0.02,
            f'{time_val:.2f}s', ha='center',
va='bottom', fontweight='bold')

plt.suptitle('Clustering Methods Performance
Comparison', fontsize=16, fontweight='bold',
y=1.02)
plt.tight_layout()
plt.show()

# -----
# Visualization 2: PCA Visualization of Clusters
# -----

# Reduce dimensionality to 2D for visualization

```

```

pca = PCA(n_components=2, random_state=42)
X_pca = pca.fit_transform(X_normalized)

fig, axes = plt.subplots(2, 2, figsize=(16, 14),
facecolor='white')

# Colors for clusters
cluster_colors = ['#FF6B6B', '#4ECDC4', '#45B7D1',
'#96CEB4', '#FFEAA7', '#DDA0DD']

# 2a. K-Means Clusters
ax = axes[0, 0]
for i in range(n_clusters):
    mask = kmeans_labels == i
    ax.scatter(X_pca[mask, 0], X_pca[mask, 1],
               c=cluster_colors[i], label=f'Cluster
{i}', alpha=0.6, s=30)
ax.scatter(kmeans.cluster_centers_[0],
kmeans.cluster_centers_[1],
           c='black', marker='X', s=200,
           linewidths=3, edgecolors='white',
           label='Centroids')
ax.set_title(f'K-Means Clusters (PCA
Projection)\nTime: {kmeans_time:.2f}s',
            fontsize=12, fontweight='bold')
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.legend()
ax.grid(True, alpha=0.3)

# 2b. Fuzzy C-Means Clusters
ax = axes[0, 1]
for i in range(n_clusters):
    mask = fcm_labels == i
    ax.scatter(X_pca[mask, 0], X_pca[mask, 1],
               c=cluster_colors[i], label=f'Cluster
{i}', alpha=0.6, s=30)

```

```

ax.set_title(f'Fuzzy C-Means Clusters (PCA
Projection)\nTime: {fcm_time:.2f}s', fontsize=12,
fontweight='bold')
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.legend()
ax.grid(True, alpha=0.3)

# 2c. DBSCAN Clusters
ax = axes[1, 0]
unique_labels = set(dbscan_labels)
for i, label in enumerate(unique_labels):
    mask = dbscan_labels == label
    if label == -1:
        ax.scatter(X_pca[mask, 0], X_pca[mask, 1],
                    c='black', label='Noise',
alpha=0.4, s=20, marker='x')
    else:
        ax.scatter(X_pca[mask, 0], X_pca[mask, 1],
                    c=cluster_colors[i %
len(cluster_colors)],
                    label=f'Cluster {label}',
alpha=0.6, s=30)
ax.set_title(f'DBSCAN Clusters (PCA
Projection)\nClusters: {n_dbscan_clusters}, Noise:
{n_noise}\nTime: {dbscan_time:.2f}s',
            fontsize=12, fontweight='bold')
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.legend()
ax.grid(True, alpha=0.3)

# 2d. Hierarchical Clusters
ax = axes[1, 1]
for i in range(n_clusters):
    mask = hierarchical_labels == i
    ax.scatter(X_pca[mask, 0], X_pca[mask, 1],
                c=cluster_colors[i], label=f'Cluster
{i}', alpha=0.6, s=30)

```

```

ax.set_title(f'Hierarchical Clusters (PCA
Projection)\nTime: {hierarchical_time:.2f}s',
fontsize=12, fontweight='bold')
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.legend()
ax.grid(True, alpha=0.3)

plt.suptitle('Cluster Visualization using PCA (2D
Projection)', fontsize=16, fontweight='bold',
y=1.02)
plt.tight_layout()
plt.show()

# -----
# Visualization 3: Radar Chart for Metrics
Comparison
# -----

from math import pi

# Prepare data for radar chart
metrics_to_plot = ['Silhouette', 'Davies-Bouldin',
'Calinski-Harabasz']
methods_radar = ['K-Means', 'Fuzzy C-Means',
'Hierarchical'] # Exclude DBSCAN for radar

# Normalize metrics
normalized_data = {}
for metric in metrics_to_plot:
    values = []
    for method in methods_radar:
        val = results_df[results_df['Method'] ==
method][metric].values[0]
        if np.isnan(val):
            val = 0
        values.append(val)

```

```

        if metric == 'Davies-Bouldin': # Lower is
better, so invert
            max_val = max(values) if max(values) > 0
else 1
            values = [1 - (v/max_val) if max_val > 0
else 0 for v in values]
        else: # Higher is better
            max_val = max(values) if max(values) > 0
else 1
            values = [v/max_val if max_val > 0 else 0
for v in values]
        normalized_data[metric] = values

# Create radar chart
fig, ax = plt.subplots(figsize=(10, 8),
subplot_kw=dict(projection='polar'),
facecolor='white')
angles = [n / float(len(metrics_to_plot)) * 2 * pi
for n in range(len(metrics_to_plot))]
angles += angles[:1]

for idx, method in enumerate(methods_radar):
    values = [normalized_data[metric][idx] for
metric in metrics_to_plot]
    values += values[:1]
    ax.plot(angles, values, 'o-', linewidth=2,
label=method)
    ax.fill(angles, values, alpha=0.1)

ax.set_xticks(angles[:-1])
ax.set_xticklabels(metrics_to_plot, fontsize=11,
fontweight='bold')
ax.set_ylim(0, 1)
ax.set_title('Clustering Methods Performance Radar
Chart\n(Normalized Scores, Higher is Better)',
            fontsize=14, fontweight='bold',
pad=20)
ax.legend(loc='upper right', bbox_to_anchor=(1.3,
1.0))

```

```

ax.grid(True)

plt.tight_layout()
plt.show()

# -----
# Visualization 4: Heatmap of Cluster Distributions
# -----

fig, axes = plt.subplots(2, 2, figsize=(14, 12),
facecolor='white')

# Create heatmaps for each method
methods_heatmap = ['K-Means', 'Fuzzy C-Means',
'DBSCAN', 'Hierarchical']
for idx, method in enumerate(methods_heatmap):
    ax = axes[idx // 2, idx % 2]
    labels = methods[method]

    if method == 'DBSCAN':
        # For DBSCAN, count clusters including
noise
        unique, counts = np.unique(labels,
return_counts=True)
        cluster_names = [f'C{int(u)}' if u != -1
else 'Noise' for u in unique]
    else:
        unique, counts = np.unique(labels,
return_counts=True)
        cluster_names = [f'Cluster {u}' for u in
unique]

    # Create heatmap data
    heatmap_data = counts.reshape(1, -1)

    im = ax.imshow(heatmap_data, cmap='YlOrRd',
aspect='auto')
    ax.set_xticks(range(len(cluster_names)))

```

```

    ax.set_xticklabels(cluster_names, rotation=45,
ha='right')
    ax.set_yticks([])
    ax.set_title(f'{method} - Cluster
Distribution', fontsize=12, fontweight='bold')

    # Add value labels
    for i, count in enumerate(counts):
        ax.text(i, 0, str(count), ha='center',
va='center',
                color='white' if count >
max(counts)/2 else 'black', fontweight='bold')

    plt.colorbar(im, ax=ax, label='Number of
Samples')

plt.suptitle('Cluster Size Distribution by Method',
fontsize=16, fontweight='bold', y=1.02)
plt.tight_layout()
plt.show()

# -----
# Visualization 5: Summary Table with Colors
# -----

fig, ax = plt.subplots(figsize=(12, 4),
facecolor='white')
ax.axis('tight')
ax.axis('off')

# Prepare summary data WITHOUT special characters
summary_data = []
for method in methods_radar + ['DBSCAN']:
    row = [method]
    for metric in ['Silhouette', 'Davies-Bouldin',
'Calinski-Harabasz']:
        val = results_df[results_df['Method'] ==
method][metric].values[0]
        if np.isnan(val):

```

```

        row.append('N/A')
    else:
        row.append(f'{val:.3f}')

row.append(f"{times[methods_names.index(method)]:.2f}s")
summary_data.append(row)

# Create table
table = ax.table(cellText=summary_data,
                 colLabels=['Method', 'Silhouette',
                             'Davies-Bouldin', 'Calinski-Harabasz', 'Time'],
                 cellLoc='center',
                 loc='center',
                 colWidths=[0.2, 0.2, 0.2, 0.2,
0.15])

# Style the table
table.auto_set_font_size(False)
table.set_fontsize(11)
table.scale(1.2, 1.5)

# Color coding function
def get_color_for_metric(metric_name, value,
is_numeric):
    if not is_numeric or value == 'N/A':
        return '#FFFFFF'

    try:
        val = float(value)
    except:
        return '#FFFFFF'

    if metric_name == 'Silhouette': # Higher is
better
        if val > 0.3:
            return '#90EE90' # Green
        elif val > 0.1:
            return '#FFD700' # Yellow

```

```

        else:
            return '#FFB6C1' # Red
        elif metric_name == 'Davies-Bouldin': # Lower
is better
            if val < 0.8:
                return '#90EE90'
            elif val < 1.2:
                return '#FFD700'
            else:
                return '#FFB6C1'
        elif metric_name == 'Calinski-Harabasz': #
Higher is better
            if val > 2000:
                return '#90EE90'
            elif val > 1000:
                return '#FFD700'
            else:
                return '#FFB6C1'
        return '#FFFFFF'

# Apply color coding
for i in range(len(summary_data)):
    for j, metric in enumerate(['Silhouette',
'Davies-Bouldin', 'Calinski-Harabasz'], start=1):
        cell = table[(i+1, j)]
        value = summary_data[i][j]
        if value != 'N/A':
            color = get_color_for_metric(metric,
value, True)
            cell.set_facecolor(color)

# Header styling
for j in range(5):
    cell = table[(0, j)]
    cell.set_facecolor('#40466e')
    cell.set_text_props(weight='bold',
color='white')

# Add note about interpretation

```

```

fig.text(0.5, 0.02, 'Note: Green=Good,
Yellow=Medium, Red=Poor',
        ha='center', fontsize=10, style='italic')

plt.suptitle('Clustering Methods Performance
Summary', fontsize=14, fontweight='bold', y=0.95)

plt.tight_layout()
plt.show()

# -----
# Final Recommendations
# -----

print("\n" + "="*70)
print("FINAL RECOMMENDATIONS")
print("="*70)

# Find best method for each metric
best_silhouette =
results_df.loc[results_df['Silhouette'].idxmax()]
if not results_df['Silhouette'].isna().all() else
None
best_db = results_df.loc[results_df['Davies-
Bouldin'].idxmin()] if not results_df['Davies-
Bouldin'].isna().all() else None
best_ch = results_df.loc[results_df['Calinski-
Harabasz'].idxmax()] if not results_df['Calinski-
Harabasz'].isna().all() else None

print("\n Best Method by Metric:")
if best_silhouette is not None and not
np.isnan(best_silhouette['Silhouette']):
    print(f"    • Silhouette Score:
{best_silhouette['Method']}
({best_silhouette['Silhouette']:.3f})")
if best_db is not None and not
np.isnan(best_db['Davies-Bouldin']):

```

```

    print(f"    • Davies-Bouldin Index:
    {best_db['Method']} ({best_db['Davies-
    Bouldin']:.3f})")
    if best_ch is not None and not
    np.isnan(best_ch['Calinski-Harabasz']):
        print(f"    • Calinski-Harabasz Index:
    {best_ch['Method']} ({best_ch['Calinski-
    Harabasz']:.0f})")

print("\n💡 Recommendations:")
print("    • Use K-Means if you need fast, scalable
    clustering with good performance")
print("    • Use Fuzzy C-Means for overlapping
    clusters and probabilistic assignments")
print("    • Use DBSCAN if you have irregular cluster
    shapes and want to detect outliers")
print("    • Use Hierarchical clustering for small
    datasets and when you need a dendrogram")

print("\n✅ Clustering comparison completed
    successfully!")

```


PROFIL PENULIS



Teti Desyani, S.Kom., M.Kom. merupakan akademisi dan peneliti di bidang Teknik Informatika yang aktif mengajar di Universitas Pamulang, Tangerang Selatan, Banten, Indonesia. Fokus kajian dan penelitiannya meliputi *Artificial Intelligence, Machine Learning, Data Mining, Computer Vision*, Sistem Pendukung Keputusan, serta pengembangan aplikasi berbasis web dan teknologi digital.

Penulis aktif menghasilkan berbagai karya ilmiah yang dipublikasikan pada jurnal nasional maupun internasional, khususnya pada bidang kecerdasan buatan dan penerapan teknologi informasi dalam pendidikan, kesehatan, serta klasifikasi data. Beberapa penelitian yang telah dilakukan di antaranya mengenai klasifikasi penyakit tanaman menggunakan *Convolutional Neural Network (CNN)*, penerapan algoritma K-Means, metode SAW dan TOPSIS untuk sistem pendukung keputusan, hingga implementasi model deep learning untuk klasifikasi citra.

Selain aktif dalam penelitian, penulis juga terlibat dalam kegiatan pengabdian kepada masyarakat, khususnya pada pengembangan kompetensi digital dan media pembelajaran berbasis teknologi. Melalui karya tulis dan penelitian yang dilakukan, penulis berharap dapat memberikan kontribusi nyata terhadap perkembangan ilmu pengetahuan dan pemanfaatan teknologi informasi di Indonesia.

Google Scholar menunjukkan bahwa penulis telah memiliki ratusan sitasi ilmiah dengan indeks akademik yang terus berkembang sebagai bentuk kontribusi dalam dunia penelitian dan pendidikan tinggi.



Syaeful Machfud, S.Kom, M.Kom. Kelahiran Bogor Jawa Barat, menamatkan Pendidikan SDN Tahun 1995, dan SLTPN Tahun 1998, serta SLTA Tahun 2001, Lulus S1 Program Studi Teknik Informatika Tahun 2014, Lulus S2 Tahun 2020. Saat ini adalah Dosen Tetap Universitas Pamulang, Prodi Teknik Informatika S-1 dan Mengampu beberapa mata kuliah Jaringan Komputer, Komunikasi Data, Interaksi Manusia Dengan Komputer, Analisa dan Perancangan Sistem, Pemrograman Web 1, Sistem Informasi Manajemen, Jaringan Nirkabel dll.

Menulis beberapa artikel jurnal ilmiah.

Penelitian terikait tentang :

1. Sistem penunjang keputusan penentuan penerima dana bantuan rumah tidak layak huni dengan metode smarter dan topsis pada desa rawakalong
2. Analisis Prediksi Penyakit Demensia Alzheimer Menggunakan Metode Decision Tree C4.5 dan Naive Bayes
3. PENDEKATAN HYBRID SVM DAN KNN UNTUK KLASIFIKASI PENYAKIT TIROID
4. ANALISIS DAMPAK PENERAPAN TEKNOLOGI ARTIFICIAL INTELLIGENCE TERHADAP EFEKTIFITAS PEMBELAJARAN BAGI SISWA SMK
5. Klasifikasi Penyakit Kanker Paru-Paru Menggunakan Metode C4.5
6. Klasifikasi Penyakit Mata Pada Data OCT Menggunakan Convolutional Neural
7. KLASIFIKASI PENYAKIT JANTUNG KORONER MENGGUNAKAN EXTREME LEARNING MACHINE
8. ANALYSIS OF SECURITY VULNERABILITY ASSESSMENT IN WEB APPLICATION ENVIRONMENT USING COMMON VULNERABILITY SCORING SYSTEM METHOD

9. Rekayasa Teknologi dalam Implementasi Rumah Pintar Berbasis Internet untuk Segala (IoT)
10. Mapping of Darunnajah 6 Muko-Muko Bengkulu Oil Palm Plantation Using GIS
11. Perancangan Sistem Informasi Penduduk Berbasis Desktop

Dan PkM terkait :

1. Pelatihan Komputer Microsoft Excel Di SMPN 2 Gunung Sindur
2. Sosialisasi Penggenalan Pentingnya Cyber Security Guna Menjaga Keamanan Data Di Era Digital Pada Siswa/i SMK Bakti Idhata Jakarta
3. PEMANFAATAN DIGITAL MARKETING UNTUK MENINGKATKAN BRAND AWARENESS TAMAN BACA MASYARAKAT (TBM) KOLONG CIPUTAT
4. Laporan Akhir Pengabdian Kepada Masyarakat : Otomatisasi Penyusunan Presentasi Menggunakan AI Gamma Untuk Meningkatkan Kreativitas Dan Efisiensi Di Bidang Pendidikan
5. Perancangan Sistem Informasi Perpustakaan Berbasis Web SMPN 2 Solear

Buku "Smart Home dan Analitik Konsumsi Energi: Metode Multi-Clustering untuk Optimasi dan Evaluasi Algoritma" membahas penerapan teknik data mining dan machine learning dalam menganalisis pola konsumsi energi pada sistem smart home berbasis Internet of Things (IoT). Seiring berkembangnya teknologi rumah pintar, berbagai perangkat elektronik mampu menghasilkan data konsumsi energi secara real-time yang dapat dimanfaatkan untuk memahami perilaku penggunaan energi rumah tangga secara lebih mendalam.

Buku ini mengintegrasikan konsep smart home, data science, clustering, serta evaluasi performa algoritma untuk menghasilkan analisis pola konsumsi energi yang lebih komprehensif. Berbagai algoritma clustering seperti K-Means, Fuzzy C-Means, DBSCAN, dan Hierarchical Clustering dibahas secara teoritis maupun implementatif menggunakan dataset smart home yang terintegrasi dengan data cuaca. Selain itu, buku ini juga menjelaskan tahapan preprocessing data, penanganan missing value, normalisasi data, serta teknik evaluasi clustering menggunakan Silhouette Score, Davies-Bouldin Index, dan Calinski-Harabasz Index.

Melalui pendekatan multi-clustering, pembaca diajak memahami kelebihan, kekurangan, serta karakteristik masing-masing algoritma dalam mengelompokkan pola konsumsi energi rumah tangga. Buku ini juga dilengkapi dengan contoh perhitungan, visualisasi cluster, kode implementasi Python, interpretasi hasil, serta analisis perbandingan performa algoritma clustering.

Buku ini sangat cocok digunakan oleh mahasiswa, dosen, peneliti, maupun praktisi yang tertarik pada bidang:

- Data Science
- Machine Learning
- Clustering
- Smart Home dan IoT
- Analisis Data Energi
- Artificial Intelligence (AI)

Dengan pendekatan teoritis dan praktis yang seimbang, buku ini diharapkan dapat menjadi referensi dalam pengembangan penelitian maupun implementasi sistem analisis energi cerdas berbasis data.