TUGAS AKHIR - EF234801

# STUDI PERBANDINGAN ALGORITMA FUZZY C-MEANS, DBSCAN, DAN MEAN-SHIFT DALAM MENDETEKSI CACAT DALAM MODUL PERANGKAT LUNAK


**Frederick Hidayat**

NRP 5025211152

Dosen Pembimbing 1

**Ir. Siti Rochimah, MT., Ph.D.**

NIP 196810021994032001

Dosen Pembimbing 2

**Dini Adni Navastara, S.Kom, M.Sc.**

NIP 198510172015042001

**Program Studi S1 Teknik Informatika**

**Departemen Teknik Informatika**

**Fakultas Teknologi Elektro dan Informatika Cerdas**

**Institut Teknologi Sepuluh Nopember**

**Surabaya**


**2025**

**TUGAS AKHIR – EF234801**

# STUDI PERBANDINGAN ALGORITMA FUZZY C-MEANS, DBSCAN, DAN MEAN-SHIFT DALAM MENDETEKSI CACAT DALAM MODUL PERANGKAT LUNAK


**Frederick Hidayat**

NRP 5025211152

Dosen Pembimbing 1

**Ir. Siti Rochimah, MT., Ph.D.**

NIP 196810021994032001

Dosen Pembimbing 2

**Dini Adni Navastara, S.Kom., M.Sc.**

NIP 198510172015042001

**Program Studi S1 Teknik Informatika**

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya


2025

*[Halaman ini sengaja dikosongkan]*

**FINAL PROJECT – EF234801**

# A COMPARATIVE STUDY OF FUZZY C-MEANS, DBSCAN, AND MEAN-SHIFT IN IDENTIFYING DEFECT SOFTWARE MODULES


**Frederick Hidayat**

NRP 5025211152

Dosen Pembimbing 1

**Ir. Siti Rochimah, MT., Ph.D.**

NIP 196810021994032001

Dosen Pembimbing 2

**Dini Adni Navastara, S.Kom., M.Sc.**

NIP 198510172015042001

**Undergraduate Study Program of Informatics**

Department of Informatics

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

Surabaya


2025

`

*[Halaman ini sengaja dikosongkan]*

# LEMBAR PENGESAHAN

## STUDI PERBANDINGAN ALGORITMA FUZZY C-MEANS, DBSCAN, DAN MEAN-SHIFT DALAM MENDETEKSI CACAT DALAM MODUL PERANGKAT LUNAK

**PROPOSAL TUGAS AKHIR**

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Komputer pada
Program Studi S-1 Teknik Informatika
Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh: **Frederick Hidayat**
NRP. 5025211152

Disetujui oleh Tim Penguji Tugas Akhir:

1. Ir. Siti Rochimah, M.T., Ph.D. — Pembimbing
2. Dini Adni Navastara,S.Kom., M.Sc. — Ko-Pembimbing
3. Dr. Sarwosri, S.Kom., M.T. — Penguji
4. Ir. Adhatus Solichah Ahmadiyah, S.Kom., M.Sc. — Penguji

**SURABAYA**
**Juli, 2025**

*[Halaman ini sengaja dikosongkan]*

# APPROVAL SHEET

## A COMPARATIVE STUDY OF FUZZY C-MEANS, DBSCAN, AND MEAN-SHIFT IN IDENTIFYING DEFECT SOFTWARE MODULES

**FINAL PROJECT PROPOSAL**

Submitted to fulfill one of the requirements

for obtaining a bachelor degree at

Undergraduate Study Program of Informatics Engineering

Department of Informatics Engineering

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

By: **Frederick Hidayat**

NRP. 5025211152

Approved by Final Project Examiner Team:

1. Ir. Siti Rochimah, M.T., Ph.D. — Advisor
2. Dini Adni Navastara,S.Kom., M.Sc. — Co-Advisor
3. Dr. Sarwosri, S.Kom., M.T. — Examiner
4. Ir. Adhatus Solichah Ahmadiyah, S.Kom., M.Sc. — Examiner

**SURABAYA**

**July, 2025**

*[Halaman ini sengaja dikosongkan]*

## PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

| | | |
|---|---|---|
| Nama mahasiswa / NRP | : | Frederick Hidayat / 5025211152 |
| Program studi | : | Teknik Informatika |
| Dosen Pembimbing / NIP | : | Ir. Siti Rochimah, M.T., Ph.D. / 196810021994032001 |
| Dosen Ko-Pembimbing / NIP | : | Dini Adni Navastara, S.Kom, M.Sc. / 198510172015042001 |

dengan ini menyatakan bahwa Tugas Akhir dengan judul "Studi Perbandingan Algoritma Fuzzy C-Means, DBSCAN, Dan Mean-Shift Dalam Mendeteksi Cacat Dalam Modul Perangkat Lunak" adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 29 Juli 2025

Mengetahui

Dosen Pembimbing

Ir. Siti Rochimah, M.T., Ph.D.

NIP. 196810021994032001

Mahasiswa

Frederick Hidayat

NRP. 5025211152

Dosen Ko-Pembimbing

Dini Adni Navastara, S.Kom, M.Sc

NIP. 198510172015042001

*[Halaman ini sengaja dikosongkan]*

## STATEMENT OF ORIGINALITY

The undersigned below:

| | | |
|---|---|---|
| Name of Student / NRP | : | Frederick Hidayat / 5025211152 |
| Department | : | Informatics |
| Advisor / NIP | : | Ir. Siti Rochimah, M.T., Ph.D. / 196810021994032001 |
| Co-Advisor / NIP | : | Dini Adni Navastara, S.Kom, M.Sc / 198510172015042001 |

Hereby declare that Final Project with the title of "A Comparative Study Of Fuzzy C-Means, Dbscan, And Mean-Shift In Identifying Defect Software Modules" is the result of my own work, is original, and is written by following the rules of scientific writing.

If in the future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

Surabaya, 29 Juli 2025

Acknowledge
Advisor

Ir. Siti Rochimah, M.T., Ph.D.
NIP. 196810021994032001

Student

Frederick Hidayat
NRP. 5025211152

Co-Advisor

Dini Adni Navastara, S.Kom, M.Sc
NIP. 198510172015042001

*[Halaman ini sengaja dikosongkan]*

# PERNYATAAN KODE ETIK
# PENGGUNAAN AI GENERATIF

*Code of Conduct Statement: Generative AI or AI-Assisted Usage*

Saya yang bertanda tangan di bawah ini:
*I, the undersigned:*

| | | |
|---|---|---|
| Nama Mahasiswa / NRP<br>*Full Name / Student ID* | : | Frederick Hidayat / 5025211152 |
| Program Studi<br>*Study Program* | : | S-1 Teknik Informatika |
| Judul Tugas Akhir<br>*Final Project Title* | : | Studi Perbandingan Algoritma Fuzzy C-Means, DBSCAN, Dan Mean-Shift Dalam Mendeteksi Cacat Dalam Modul Perangkat Lunak |

dengan ini menyatakan bahwa pada Tugas Akhir dengan judul di atas tersebut:
*hereby declare that in the Final Project with the above title:*

| No. | Pernyataan<br>*Statement* | (☑) |
|---|---|---|
| 1 | Saya hanya menggunakan AI generatif sebagai alat bantu untuk memperbaiki tata bahasa. AI generatif tidak digunakan untuk membuat isi Tugas Akhir.<br>*I only used generative AI as a tool to improve the readability or language of the text in my Final Project. It was not used to generate a complete text of my work.* | ☑ |
| 2 | Saya telah memeriksa dan/atau memperbaiki seluruh bagian dari Tugas Akhir saya yang dibantu oleh AI generatif agar sesuai dengan baku mutu penulisan karya ilmiah.<br>*I have reviewed and refined all aspects of my work that generative AI assists with, ensuring it adheres to the standards of academic writing.* | ☑ |
| 3 | Saya tidak menggunakan AI generatif untuk pembuatan data primer, grafik dan/atau tabel pada Tugas Akhir saya.<br>*I did not use generative AI to generate primary data, figures, and/or tables in my work.* | ☑ |
| 4 | Saya telah memberikan atribusi/pengakuan terhadap alat AI yang digunakan, secara rinci pada suatu bagian pada lampiran.<br>*I have acknowledged the use of generative AI in any part of the work in the specific appendix page.* | ☑ |
| 5 | Saya memastikan tidak ada plagiarisme, termasuk hal yang berasal dari penggunaan AI generatif.<br>*I have ensured that there is no plagiarism issue in the work, including any parts generated by AI.* | ☑ |

Surabaya, 29 Juli 2025

Mahasiswa

Frederick Hidayat
NRP. 5025211152

*[Halaman ini sengaja dikosongkan]*

# ABSTRAK

## STUDI PERBANDINGAN ALGORITMA FUZZY C-MEANS, DBSCAN, DAN MEAN-SHIFT DALAM MENDETEKSI CACAT DALAM MODUL PERANGKAT LUNAK


**Nama Mahasiswa / NRP** : **Frederick Hidayat / 5025211152**

**Departemen** : **Teknik Informatika FTEIC-ITS**

**Dosen Pembimbing** : **Ir. Siti Rochimah, M.T., Ph.D.**

**Dini Adni Navastara,S.Kom., M.Sc.**



**Abstrak**

Deteksi cacat dalam perangkat lunak merupakan aspek penting dalam proses pengembangan untuk menjamin kualitas dan menekan biaya pemeliharaan di masa mendatang. Pendeteksian cacat dengan menggunakan data tak berlabel (*unsupervised defect prediction*) menjadi penting dan perlu untuk menunjang deteksi cacat pada projek yang tidak memiliki histori data berlabel. Penelitian mengenai deteksi cacat pada perangkat lunak dengan *dataset* NASA MDP Defect Dataset dijalankan dengan melakukan studi perbandingan terhadap tiga algoritma *clustering* yaitu Fuzzy C-Means (FCM), DBSCAN, dan Mean-Shift. *Dataset* ini mengandung metrik perangkat lunak yaitu kelompok fitur Halstead dan McCabe, kedua fitur akan dikombinasikan menjadi kelompok fitur dan digunakan sebagai kelompok fitur data yang diolah dalam penelitian. Evaluasi dilakukan menggunakan metrik *Silhouette Score* sebagai validasi internal dan metrik *Accuracy*, *Precision*, *Recall*, serta *F1-Score* sebagai validasi eksternal. Hasil eksperimen menunjukkan bahwa Fuzzy C-Means menghasilkan performa terbaik dibandingkan dua model lainnya dengan perbedaan nilai pada rentang 0,01 sampai 0,5, terutama setelah dilakukan seleksi fitur berbasis metrik Halstead yang selanjutnya dilakukan *hyperparameter tuning* dengan mengatur parameter *m*, *error*, dan *max_iter*. DBSCAN unggul dalam mendeteksi *outlier* dan *noise*, namun dalam mendeteksi cacat pada perangkat lunak dengan *dataset* terkait, model ini mengalami kesulitan karna data yang distribusinya tidak sejalan dengan prinsip pengelompokkan dari DBSCAN. Mean-Shift mampu menemukan jumlah klaster optimal tanpa inisialisasi awal, tetapi performanya terpengaruhi oleh *dataset* yang persebarannya tidak sejalan dengan prinsip model ini. Dengan demikian, Fuzzy C-Means dengan kelompok fitur Halstead menjadi model paling baik diantara ketiga model *unsupervised* dalam konteks deteksi cacat perangkat lunak pada *dataset* NASA dengan nilai *recall* tertinggi pada angka 0,8, namun performa tertinggi masih pada *supervised* model dengan *recall* pada nilai 1.



**Kata kunci:** Deteksi cacat perangkat lunak, DBSCAN, Fuzzy C-Means, Halstead, McCabe, Mean-Shift.

*[Halaman ini sengaja dikosongkan]*

# ABSTRACT

## A COMPARATIVE STUDY OF FUZZY C-MEANS, DBSCAN, AND MEAN-SHIFT IN IDENTIFYING DEFECT SOFTWARE MODULES


**Student Name / NRP** : **Frederick Hidayat / 5025211152**

**Department** : **Informatics ELECTICS-ITS**

**Advisor** : **Ir. Siti Rochimah, M.T., Ph.D.**

**Dini Adni Navastara,S.Kom., M.Sc.**



**Abstract**

Defect detection in software is a crucial aspect of the development process to ensure quality and reduce future maintenance costs. Detecting defects using unlabeled data (unsupervised defect prediction) is important and necessary to support defect detection in projects that lack labeled data history. This study investigates defect detection in software using the NASA MDP Defect Dataset by conducting a comparative study of three clustering algorithms: Fuzzy C-Means (FCM), DBSCAN, and Mean-Shift. The dataset contains software metrics, specifically Halstead and McCabe feature groups, which are combined and used as the feature set for processing in this study. Evaluation is carried out using the Silhouette Score as an internal validation metric, and Accuracy, Precision, Recall, and F1-Score as external validation metrics. Experimental results show that Fuzzy C-Means yields the best performance compared to the other two models, with differences ranging from 0.01 to 0.5, especially after feature selection based on Halstead metrics and subsequent hyperparameter tuning by adjusting the parameters *m*, *error*, and *max_iter*. DBSCAN excels at detecting outliers and noise but struggles to detect defects in this dataset due to distribution characteristics that do not align with DBSCAN's clustering principles. Mean-Shift can determine the optimal number of clusters without initial initialization, but its performance is affected by datasets whose distributions do not align well with the model's principles. Thus, Fuzzy C-Means with the Halstead feature group emerges as the best unsupervised model among the three for software defect detection on the NASA dataset, achieving the highest recall value of 0.7778. However, the highest overall performance is still achieved by supervised models, with a recall value of 1.



**Keywords:** DBSCAN, Fuzzy C-Means, Halstead, McCabe, Mean-Shift, Software defect detection

*[Halaman ini sengaja dikosongkan]*

# KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas rahmat dan karunia-Nya lah, penulis dapat menyelesaikan tugas akhir yang berjudul:

**Studi Perbandingan Algoritma Fuzzy C-Means, Dbscan, Dan Mean-Shift Dalam Mendeteksi Cacat Dalam Modul Perangkat Lunak**

Tugas akhir ini dibuat dalam rangka memenuhi syarat kelulusan dalam rangka mendapatkan gelar sarjana dan Teknik Informatika program Strata Satu (S-1) Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember.

Penulis menyadari bahwa tiada gading yang tak retak sehingga penyusunan buku Tugas Akhir ini pun tak luput dari halang rintang selama penelitian berlangsung. Seluruh rangkaian ini tidak mungkin dapat diselesaikan tanpa dukungan dan bantuan moril serta materiil dari berbagai pihak. Karena hal itu, penulis ingin menyampaikan rasa syukur, hormat dan terima kasih kepada berbagai pihak yang telah membantu selesainya proses penyelesaian tugas akhir ini:

1. Tuhan Yang Maha Esa, karena dengan berkat, rahmat, karunia, perlindungan serta kerahiman-Nya lah penulis dapat diterima di kampus perjuangan dan menyelesaikan rangkaian tugas akhir dengan lancar.
2. Ibu, ayah, dan kedua saudara pembimbing yang selalu memberikan doa dan dukungan emosional serta dukungan materiil selama menyelesaikan tugas akhir.
3. Ibu Ir. Siti Rochimah, M.T., Ph.D. selaku dosen wali dan dosen pembimbing serta Ibu Dini Adni Navastara, S.Kom., M.Sc. selaku dosen pembimbing yang selalu menyempatkan waktu dan tenaga untuk membimbing dan memberikan ilmu serta berbagai masukan yang membangun bagi penulis.
4. Teman-teman impulsif Malang, Vino, Keyisa, Nabila, dan Mbak Ryza serta para pencari hantu, Elfa dan Cathlea, yang selalu menolong, membantu, dan memberikan berbagai bantuan dari segi ilmu dan informasi serta memberikan suka duka yang akan selalu terkenang
5. Teman-teman PSM ITS, TPKB ITS, Admin RPL dan teman-teman jurusan yang membantu memberi semangat melalui canda tawa serta memberikan berbagai kebahagiaan yang tak dapat dilupakan selama menjalani perkuliahan
6. Berbagai pencipta musik dari Isyana Sarasvati, Bernadya, Taylor Swift, Avril Lavigne, Coldplay, Paramore, Clean Bandit, dan lain sebagainya yang menemani pengerjaan tugas akhir ini sehingga penuh harmoni.

Penulis telah melakukan usaha terbaik untuk menyusun dan menyelesaikan tugas akhir ini. Penulis menyadari dapat terjadi kesalahan maupun kekurangan dalam suatu penelitian yang tidak disengaja sehingga penulis memohon maaf jika terdapat kesalahan serta kekeliruan. Kritik dan saran dari pembaca akan sangat diapresiasi sebagai bahan evaluasi kedepannya. Semoga tugas akhir ini dapat memberikan manfaat bagi penulis sendiri serta pembaca guna menambah informasi dan wawasan

Surabaya, 22 Juni 2025

Frederick Hidayat

*[Halaman ini sengaja dikosongkan]*

`

# DAFTAR ISI

*[Halaman ini sengaja dikosongkan]*

# DAFTAR GAMBAR

# DAFTAR TABEL

*[Halaman ini sengaja dikosongkan]*

# DAFTAR KODE SEMU

*[Halaman ini sengaja dikosongkan]*

`

# BAB 1
# PENDAHULUAN

## 1.1. Latar Belakang

Keberhasilan sebuah sistem perangkat lunak dalam menjalankan fungsinya dipengaruhi oleh berbagai aspek. Salah satu aspek dasar yang menentukan keberhasilan sebuah perangkat lunak adalah kualitas perangkat lunak. Produk perangkat lunak dengan kualitas rendah berdampak besar dalam mempengaruhi kepuasan pengguna. Menurut penelitian yang dilakukan oleh Sambo pada tahun 2022, produk dengan mutu buruk menjadi faktor yang dapat mereduksi loyalitas pengguna sehingga hal tersebut menjadi sebuah metrik produk yang harus diperhatikan. Kualitas sebuah produk mempunyai pengaruh sebesar 65,1% dalam penentuan tingkat kepuasan pengguna sehingga peningkatan mutu berkontribusi secara linear dalam meningkatkan kepuasan pengguna (Sambo, 2022). Hal ini menunjukkan andil kualitas yang besar dalam memastikan kepuasan pengguna ketika menggunakan sebuah produk termasuk produk perangkat lunak. Perangkat lunak yang berkualitas dapat memberikan efisiensi biaya bagi pengembangnya. Dalam penemuan yang dilakukan oleh Krasner pada tahun 2020, kualitas perangkat lunak yang buruk dapat menyebabkan kerugian yang besar sampai sekitar 2,8 miliar untuk setiap kecacatan dalam sebuah perangkat lunak (Krasner, 2020). Dengan memperhatikan besarnya kerugian yang ditimbulkan oleh buruknya kualitas perangkat lunak, pengguna dapat menekan biaya yang datang di masa mendatang, salah satunya biaya perbaikan perangkat lunak yang cacat. Deteksi cacat perangkat lunak dini dapat menyiapkan pengguna dan pengembang untuk perbaikan lebih awal sehingga tidak terjadi peningkatan biaya secara kontinu. Melihat pentingnya deteksi cacat perangkat lunak dari sisi kualitas dan efisiensi biaya, metode deteksi cacat perangkat lunak dapat digunakan dan diuji demi meningkatkan efektivitas pengembangan perangkat lunak minim cacat. Dalam mendeteksi kecacatan perangkat lunak, algoritma *clustering* dapat digunakan sebagai dasar alat pendeteksi cacat terkait dengan kemampuan identifikasi pola melalui data metrik perangkat lunak yang ada. *Clustering* adalah salah satu bagian dalam *unsupervised machine learning* yang bekerja dengan mengelompokkan objek data serta menemukan pola data tersembunyi. Penggunaan kluster dalam algoritma *clustering* dapat digunakan untuk mereduksi dimensi dari sebuah data. Selain itu, *clustering* dapat mendeteksi anomali serta *outlier* sehingga sering digunakan dalam berbagai hal termasuk dalam pengenalan pola tertentu pada berbagai lini teknologi (Alahmari, 2021). Algoritma *clustering* yang termasuk dalam cakupan ini adalah Fuzzy C-Means, DBSCAN dan Mean-shift. Tiap algoritma memiliki berbagai keunggulan yang dapat diutilisasikan untuk mendeteksi cacat pada perangkat lunak melalui pengenalan pola dalam sebuah kumpulan data.

Dalam usaha mengidentifikasi algoritma tepat guna dalam mendeteksi cacat perangkat lunak, dilakukan sebuah penelitian komparatif antar algoritma untuk menemukan metode yang dapat diterapkan secara tepat dalam konteks mencari cacat pada perangkat lunak. Selain itu, penelitian ini dilakukan untuk memvalidasi temuan penelitian terdahulu seperti pada penelitian Xu et al, 2021 menyangkut berbagai model *clustering* yang ada pada penelitian ini. Pemilihan algoritma yang tepat guna dapat mengoptimalisasi pendeteksian cacat pada perangkat lunak serta meningkatkan mutu perangkat lunak guna menekan biaya di masa mendatang demi menghasilkan efek positif untuk kepuasan pengguna. Penelitian ini membandingkan Fuzzy C-Means yang dipilih karena kemampuannya dalam menangani ambiguitas tinggi dalam data melalui sistem keanggotaan fuzzy yang fleksibel sedangkan setelah melakukan kajian pustaka terhadap penelitian terdahulu terhadap Mean-shift, algoritma ini belum digunakan untuk deteksi cacat pada perangkat lunak dengan data tabular meskipun karakteristik metode yang

menguntungkan karena bersifat nonparametrik dan handal dalam menemukan jumlah kluster tanpa inisiasi awal (berdasarkan kepadatan data), sehingga cocok untuk dataset dengan struktur yang tidak diketahui. Selain itu, DBSCAN dipilih juga karena kelebihannya dalam mengidentifikasi noise dalam data. Selain itu, penelitian dilakukan dengan harapan memberikan kontribusi positif dalam cakupan penggunaan algoritma *clustering* untuk pengembangan metode deteksi cacat pada perangkat lunak.

## 1.2. Rumusan Masalah

Pada tugas akhir ini, terdapat beberapa rumusan masalah yang dipaparkan sebagai berikut:

1. Bagaimana kinerja komparatif algoritma Fuzzy C-Means, DBSCAN dan Mean-shift dalam mendeteksi cacat dalam modul perangkat lunak?
2. Bagaimana proses seleksi fitur dapat mempengaruhi kinerja model algoritma dalam mendeteksi cacat pada modul perangkat lunak ?
3. Bagaimana pengaruh reduksi dimensi dengan PCA dengan menggunakan model terbaik ?
4. Bagaimana perbandingan performa metode *unsupervised* dengan metode *supervised* ?

## 1.3. Batasan Masalah

Permasalahan yang diangkat pada tugas akhir ini memiliki beberapa batasan sebagai berikut:

1. Implementasi dan uji coba menggunakan kode dengan bahasa pemrograman Python 3.10
2. *Dataset* yang digunakan adalah Metrics Data Program (MPD) dari NASA[1]

## 1.4. Tujuan

Tujuan dari pembuatan tugas akhir ini dipaparkan sebagai berikut:

1. Menganalisis akurasi dan presisi kinerja komparatif algoritma Fuzzy C-Means, DBSCAN, dan Mean-shift
2. Mengetahui fitur yang mempunyai pengaruh terhadap kinerja model algoritma dalam mendeteksi cacat pada modul perangkat lunak
3. Memberikan kontribusi dalam pengembangan metode deteksi cacat pada perangkat lunak

## 1.5. Manfaat

Manfaat dari pembuatan tugas akhir ini dipaparkan sebagai berikut:

1. Memberikan referensi tentang kinerja algoritma Fuzzy C-Means, DBSCAN, dan Mean-shift dalam mendeteksi cacat pada perangkat lunak
2. Menyediakan referensi bagi peneliti lain yang tertarik dalam pengembangan metode deteksi cacat pada perangkat lunak dengan algoritma *clustering machine learning*
3. Mendukung pengembangan perangkat lunak yang lebih dapat diandalkan dengan menggunakan hasil analisis komparatif algoritma *clustering*

[1] https://github.com/klainfo/NASADefectDataset

`

# BAB 2
# TINJAUAN PUSTAKA

## 2.1. Penelitian Terkait

Penelitian pada topik tugas akhir ini menagacu pada beberapa penelitian yang tercantum pada Tabel 2.1 yang berisi sepuluh penelitian dengan rentang tahun 2020-2025. Acuan ini digunakan untuk dasar pelaksanaan penelitian serta pencegahan terjadinya similaritas mutlak dengan penelitian preseden. Penelitian-penelitian tersebut membahas berbagai metode dalam mendeteksi cacat dalam modul perangkat lunak.

*Tabel 2.1 Penelitian Terkait*

| Penelitian | Dataset | Metode | Analisis |
|---|---|---|---|
| Ensemble machine learning model for software defect prediction (Dada et al, 2021) | NASA PROMISE Repository | *Novel Multi-model Ensemble machine-learning model* (88%) sebagai metode terbaik | Penelitian ini menunjukkan keakuratan multi-model *ensemble machine-learning* model dengan akurasi paling tinggi yaitu 88% diatas seluruh metode pembanding seperti KNN, LDA, dan GLMNet |
| Predicting software defects using machine learning techniques (Aquil et al, 2020) | NASA MDP Dataset | *Ensemble Learning Technique* STC (*Stacking Classifier*) dengan akurasi 88,63% dan *f-measure* 87,22% | Penelitian ini menunjukkan keandalan STC yang memiliki akurasi tertinggi |
| Optimizing Patient Stratifcation in Healthcare: A Comparative Analysis of Clustering Algorithms for EHR Data (Aljohani A., 2020) | Electronic Health Record | KNN sebagai algoritma terbaik dengan nilai 0,837 dan DBSCAN berada pada tempat ketiga dengan nilai 0,502 serta keandalannya dalam menangani berbagai skenario | Penelitian menunjukkan keandalan DBSCAN dengan performa yang terbaik setelah KNN dan *hierarchical clustering*. DBSCAN menunjukkan performa yang kompetitif ketika dihadapkan dengan skenario yang |

| Penelitian | Dataset | Metode | Analisis |
|---|---|---|---|
| | | | bersangkutan dengan *density-based clustering*. |
| A comprehensive comparative study of clustering-based unsupervised defect prediction models (Xu et al, 2021) | Ant, Camel, ivy, jedit, log4j, poi, Synapse, Velocity, xerces, Equinox framework, Eclipse JDT Core, Apache Lucene, My lyn, Eclipse PDE UI | Model dalam golongan IVSBC dengan EAF 0,65% dan PBC (Fuzzy C-Means) dengan nilai 0,58% memiliki performa yang menjanjikan | Penelitian ini menunjukkan performa model golongan PBC yang menjanjikan. |
| Modified Fuzzy C-Means Clustering for Anomaly Detection in Bio-medical Data (Sahoo et al, 2025) | Bio-medical data (Lymphograpgy, Wisconsin Breast Cancer) | *Optimalized* Fuzzy C-Means (berhasil mendeteksi pencilan sampai pada 83.33%) | Penelitian menunjukkan FCM termodifikasi dapat mendeteksi pencilan dengan baik |
| Impact of Software Metrics in Nasa Dataset Modules for Software Defect Prediction (Ramadhani et al, 2024) | NASA MDP Dataset | KNN model dengan k=11 dengan nilai AUC tertinggi 0,857 pada *dataset* PC4 | Nilai AUC paling baik dimiliki kombinasi LoC + McCabe + Misc, dengan McCabe yang keberadaannya menurunkan nilai AUC. |
| Comparative Study of Common Density-based Clusterin Algorithms (Alahami et al, 2021) | Iris, Wine | DBSCAN dengan keberhasilan pencarian jumlah kluster 100% dan *silhouette coefficient* 0,161 serta Meanshift dengan keberhasilan pencarian jumlah kluster 66,6% dan *silhouette coefficient* 0,686 | DBSCAN menunjukkan performa yang paling baik pada kebanyakan kasus dan Mean-shift menunjukkan performa yang buruk pada banyak kasus namun menunjukkan performa yang sangat baik pada beberapa kasus |
| Software Defect Prediction Based on Optimized Machine Learning Models: A | NASA MDP Dataset | KNN menunjukkan akurasi (98,97%), presisi (98,99%) dan *recall* tertinggi | KNN menunjukkan kecocokan paling tinggi untuk memprediksi cacat |

| Penelitian | Dataset | Metode | Analisis |
|---|---|---|---|
| Comparative Study (Siswantoro et al, 2023) | | (98,97%) pada *dataset* MC1. | dalam modul perangkat lunak pada beberapa *dataset* |
| Software Defect Prediction Method Based on Clustering Ensemble Learning (Tao et al, 2024) | Ant-1.7, ivy-1.4, jedit-4.0, log4j-1.0, Tomcat v6.0, xalan-2.6, xerces-1.3, lucene-2.4, camel-1.6, poi-3.0 | Clustering Overlap sebagai model terbaik dengan AUC 0,721 | Penelitian menunjukkan model *clustering ensemble learning* dengan *clustering overlap* lebih baik daripada metode seperti Newman, CNM, dan Fluid. |
| Metrics Based Feature Selection for Software Defect Prediction (Nugroho et al, 2020) | NASA MDP Dataset | Random Forest Classifier menunjukkan kinerja terbaik dengan rata-rata AUC 0,782 dibandingkan Naive Bayer, Decision Tree, KNN, dan SVM | Penelitian menunjukkan kinerja Random Forest Classifier memiliki performa AUC tertinggi dengan fitur terbaik berada pada LoC kemudian Misc dan diikuti dengan Halstead. |

Melihat keandalan dari model Fuzzy C-Means, DBSCAN, dan Mean-shift serta dibutuhkannya *Unsupervised Defect Prediction* (UDP) model yang memungkinkan pendeteksian cacat pada data tidak berlabel(Xu et al, 2021), diusung penelitian menggunakan tiga metode *clustering* yaitu Fuzzy C-Means, DBSCAN, dan Mean-shift untuk memungkinkan pendeteksian cacat untuk projek yang tidak memiliki histori akan data berlabel.

## 2.2. Dasar Teori

Teori-teori yang digunakan dalam penelitian dan pengerjaan tugas akhir ini dijelaskan sebagai berikut.

### 2.2.1. Deteksi Cacat dalam Modul Perangkat Lunak

Cacat pada perangkat lunak adalah eror, ketidaksesuaian, *bug*, kesalahan, dan kegagalan dalam sebuah perangkat lunak yang dapat menghasilkan ketidakakuratan atau hasil yang tidak sesuai dengan seharusnya. Selain itu, cacat pada perangkat lunak dapat juga menyebabkan perangkat lunak berperilaku diluar harapan program tersebut. Pada pengembangan sebuah perangkat lunak, dilakukan pengembangan dengan intensi untuk menghasilkan perangkat lunak dengan kecacatan yang minim atau bahkan tidak ada (Rawat et al, 2012). Deteksi cacat pada perangkat lunak menjadi hal esensial yang dilakukan untuk menjaga kualitas perangkat lunak memperhatikan dampak dari cacat perangkat lunak. Deteksi cacat dalam perangkat lunak

dilakukan untuk mengidentifikasi kecacatan pada modul perangkat lunak. Usaha deteksi cacat dapat dilakukan dengan mengandalkan model *machine learning* (Abbineni et al, 2018).

### 2.2.2. Python

Python merupakan bahasa pemrograman yang sering digunakan dalam konteks ilmu data dan pembelajaran mesin di masa sekarang. Python dibuat oleh Guido van Rossum pada tahun 1985-1990 dengan sintaks yang sangat sederhana. Python menawarkan bahasa pemrograman yang serbaguna dengan pustaka yang beragam untuk mengakomodasi kebutuhan pengolahan data serta pembelajaran mesin (Peta, 2022). Penggunaan pustaka seperti NumPy, Pandas, dan Scikit-learn pada Python memungkinkan pengolahan data secara efektif dan efisien dalam berbagai implementasi terutama pada bidang analisis data serta pembelajaran mesin. Simplisitas serta ramah bagi pemula menjadi poin utama Python populer dalam memenuhi kebutuhan lingkungan pendidikan dan profesional. Popularitas Python yang terus berkembang, berkontribusi aktif dalam pengembangan berbagai sumber daya informasi, panduan serta alat-alat yang memudahkan pengembangan kebutuhan pengolahan dan pembelajaran mesin (Pitroda et al, 2024). Pada penelitian ini, digunakan berbagai macam *library* untuk mengolah data dengan lebih efektif. Tiap pustaka memiliki peran dan fungsi masing-masing dalam mengefektifkan pengolahan data yang dilakukan pada penelitian ini. Terdapat 14 pustaka yang dirangkum beserta dengan fungsi serta kegunaannya. Rangkuman pustaka (*library*) dari Python yang digunakan dalam penelitian ini pada Tabel 2.2

*Tabel 2.2 Library Python dalam Penelitian*

| Library / Modul | Fungsi / Kegunaan |
|---|---|
| os | Untuk berinteraksi dengan sistem operasi. |
| pandas | Untuk manipulasi dan analisis data dalam format tabel (DataFrame). |
| numpy | Untuk komputasi numerik dan operasi array. |
| seaborn | Untuk visualisasi statistik data, berbasis matplotlib. |
| matplotlib.pyplot | Untuk membuat grafik dan visualisasi data. |
| sklearn.preprocessing.StandardScaler | Untuk standarisasi fitur sebelum pemodelan. |
| sklearn.model_selection.train_test_split | Untuk membagi data menjadi data latih dan data uji. |
| sklearn.metrics.silhouette_score | Untuk mengevaluasi kualitas klaster dengan *silhouette score*. |
| sklearn.metrics.precision_score | Untuk menghitung nilai *precision* dalam evaluasi model. |
| sklearn.metrics.recall_score | Untuk menghitung *recall* dalam evaluasi model. |
| sklearn.metrics.f1_score | Untuk menghitung *f1-score* dalam evaluasi model. |
| fcmeans.FCM | Untuk melakukan klastering menggunakan algoritma Fuzzy C-Means. |
| sklearn.cluster.DBSCAN | Untuk melakukan klastering menggunakan algoritma DBSCAN. |
| sklearn.cluster.MeanShift | Untuk melakukan klastering menggunakan algoritma Mean Shift. |

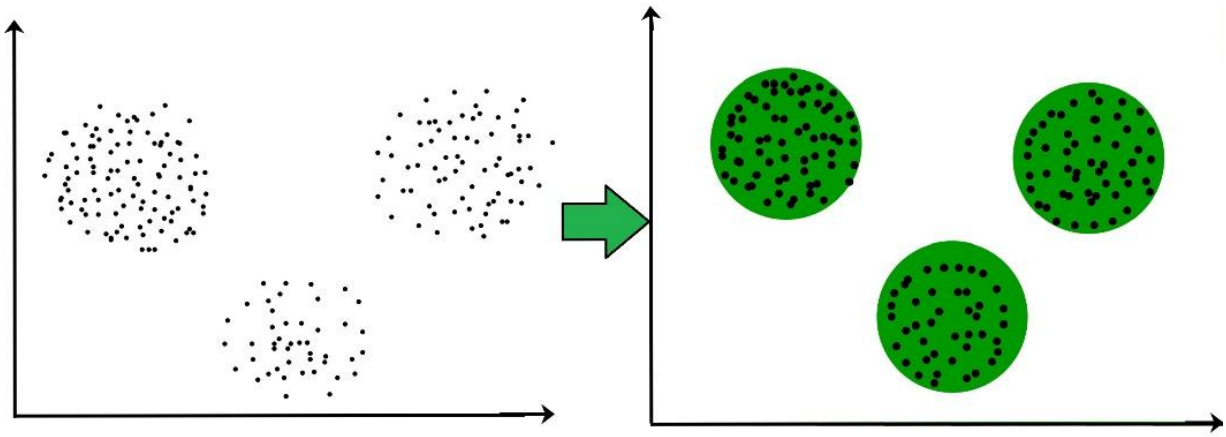
### 2.2.3. Pembelajaran Mesin

Pembelajaran mesin adalah bagian dari ilmu komputer yang melatih komputer untuk mempelajari data tanpa pemrograman eksplisit. Pembelajaran yang dikembangkan dalam algoritma pembelajaran mesin mampu melakukan identifikasi pola tertentu, membuat prediksi, serta melakukan pengambilan keputusan menggunakan data yang tersedia. Pembelejaran mesin terbagi menjadi tiga jenis yaitu *supervised learning, unsupervised learning,* dan *reinforcement learning*. *Supervised learning* menggunakan data latih berlabel dan mendefinisikan variabel yang ingin diperkirakan nilainya sedangkan *unsupervised learning* menggunakan data latih tak berlabel sehingga model mempelajari pola implisit serta pemahaman dari data yang diberikan. Disisi lain, *reinforcement learning* merupakan model pembelajaran berdasarkan umpan balik yang diterima oleh agen, dengan pemberian hadiah setiap kali aksi benar dilakukan dan denda bagi kesalahan yang dilakukan (Vyawahare, 2022).

Pembelajaran mesin telah berkembang dalam memenuhi kebutuhan berbagai bidang. Kebutuhan ini kian meningkat tiap harinya dikarenakan teknologi ini dapat melakukan tugas yang terlalu kompleks untuk dilakukan secara langsung oleh manusia. Limitasi manusia dalam berbagai kemungkinan, seperti pengaksesan data yang besar dalam satu waktu secara manual, membuat teknologi ini menjadi pilihan dalam memudahkan pekerjaan manusia. Disisi lain, penggunaan pembelajaran mesin dapat mengefisiensi fungsi biaya dan waktu. Contoh konkret penggunaan pembelajaran mesin dapat dilihat pada mobil pengemudi otomatis, deteksi penipuan siber, dan pengenal wajah (Vyawahare, 2022).

### 2.2.4. Clustering Algortihm

*Clustering* merupakan sebuah algoritma dalam *unsupervised learning* pada pembelajaran mesin. *Unsupervised learning* mencari pola dan kesamaan dalam mengklasifikasikan titik-titik data tidak berlabel. Penggunaan data tidak berlabel menjadi salah satu kelebihan *unsupervised learning* dalam menangani *dataset* yang kompleks dibandingkan dengan *supervised learning*. Proses *clustering* (pengelompokan) dalam *unsupervised learning* berjalan dengan membagi data menjadi beberapa bentuk sehingga setiap kumpulan data hanya menjadi anggota dari satu kelompok (Naeem et al, 2023). Algoritma yang termasuk dalam golongan *clustering* adalah Fuzzy C-Means, DBSCAN, dan Mean-shift. Implementasi algoritma pengelompokan tingkat lanjut terlihat pada deteksi objek dan deteksi anomali (Zhou et al, 2022).



*Gambar 2.1 Ilustrasi Clustering*

*Sumber: (GeeksforGeeks, 2025)*

Pada Gambar 2.1, tergambar poin-poin data yang dikelompokkan sesuai kelompoknya kemudian digolongkan seperti terlihat pada bagian kanan dari gambar. Tiap lingkaran hijau mengindikasikan sebuah *cluster*, kelompok yang dihasilkan dari proses pengelompokkan tiap point data.

### 2.2.5. Fuzzy C-Means

Fuzzy C-Means (FCM) adalah teknik pengelompokan non hirarki dengan basis model *clustering* yaitu *fuzzy.* Fuzzy diperkenalkan sebagai *fuzzy set* oleh Lothfi Zadeh pertama kali pada tahun 1965. *Penggunaan* model *fuzzy* memungkinkan pembentukan cluster berdasarkan pada tingkat keanggotaan dari 0 sampai 1 (Auliya et al, 2024). Matriks tingkat keanggotaan serta pusat kluster pada FCM diperbarui secara berulang setiap observasi (Saatchi et al, 2024). Dengan berlandaskan *fuzzy*, tingkat keanggotaan didalamnya dapat menangani data dengan ambiguitas tinggi dengan fleksibilitasnya (Ghaffari, 2024). Proses algoritma FCM menurut Bezdek (1984) meliputi,

1. Inisialisasi beberapa dasar yaitu
   a. Jumlah kluster ($2 \leq c <$ *jumlah data*)
   b. Eksponen pembobot ($1 \leq m < \infty$)
   c. Matriks positif-definit ($A$) berukuran $n \times n$ dengan $n$ adalah jumlah data
   d. Tentukan norma yang digunakan ($\|k\|_A$) untuk mengukur jarak dalam ruang $\mathbb{R}^n$
   e. Inisialisasi matriks keanggotaan *fuzzy* awal ($U^{(o)} \in M_{fc}$ ) secara acak
   f. Jumlah maksismum iterasi ($i_{MAX}$)
   g. Nilai iterasi awal ($k = 1$)

2. Hitung pusat kluster dengan menggunakan nilai keanggotaan *fuzzy* menggunakan Persamaan 2.1

$$\hat{v}_i = \frac{\sum_{k=1}^{N} (\hat{u}_{ik})^m \, y_k}{\sum_{k=1}^{N} (\hat{u}_{ik})^m} \; ; \; 1 \leq i \leq c \; ; \tag{2.1}$$

   dengan

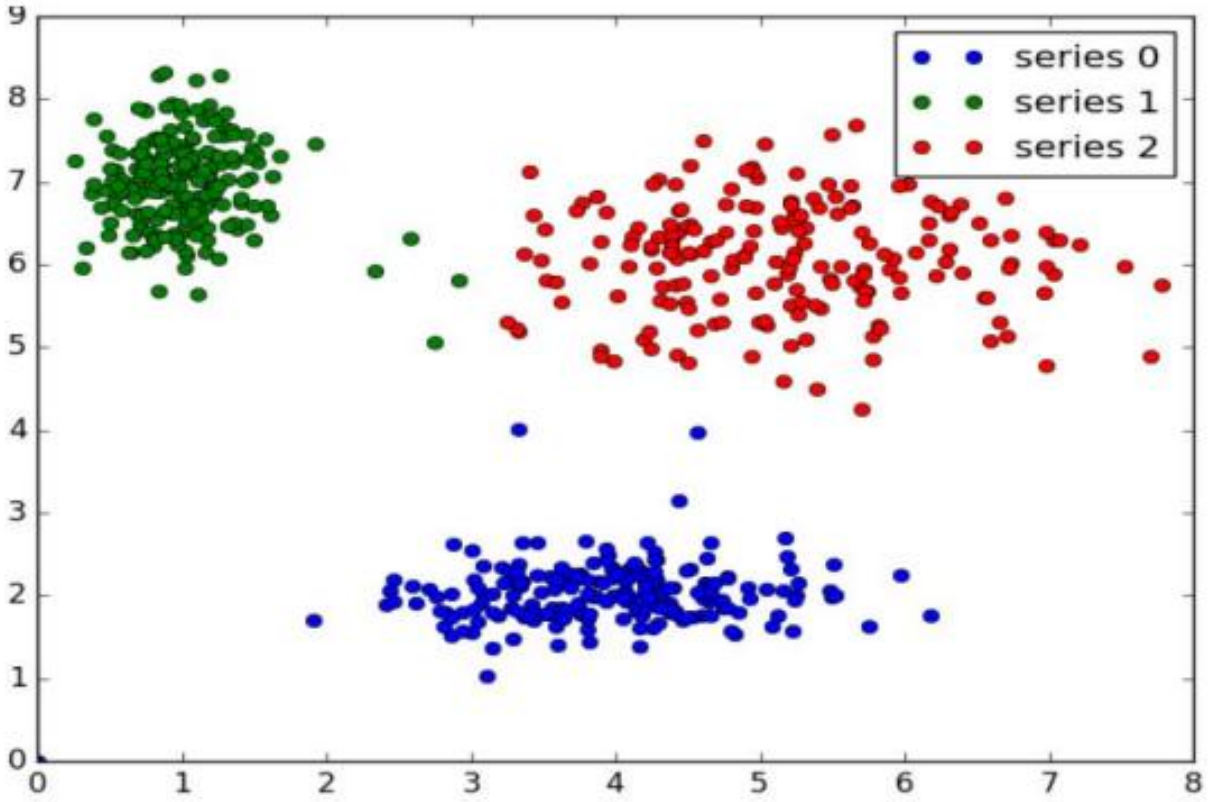
   $y_k$ adalah titik data dari himpunan data $Y$ dengan $Y = \{y_1, y_2, \dots, y_N\} \subset R^n$
   $\hat{u}_{ik}$ adalah matriks keanggotaan *fuzzy* tiap titik data.

3. Perbarui nilai matriks keanggotaan $\hat{U}^{(k+1)} = \hat{u}_{ik}^{(k+1)}$ dengan menggunakan Persamaan 2.2,

$$\hat{u}_{ik} = \left(\sum_{j=1}^{c} \frac{\hat{d}_{ik}}{\hat{d}_{jk}}^{\frac{2}{(m-1)}}\right)^{-1} \; ; \; 1 \leq k \leq N \; ; 1 \leq i \leq c \; ; \tag{2.2}$$

4. Bandingkan matriks keanggotaan baru $\hat{U}^{(k+1)}$ dengan $\hat{U}^k$ menggunakan norma matriks. Jika $\left\|\hat{U}^{(k+1)} - \hat{U}^k\right\| < \epsilon$, maka proses akan berhenti dan jika sebaliknya, tetapkan nilai $\hat{U}^k = \hat{U}^{(k+1)}$ dan kembali ke langkah kedua.

Berikut ilustrasi pengelompokan Fuzzy C-Means yang dapat terlihat pada Gambar 2.2



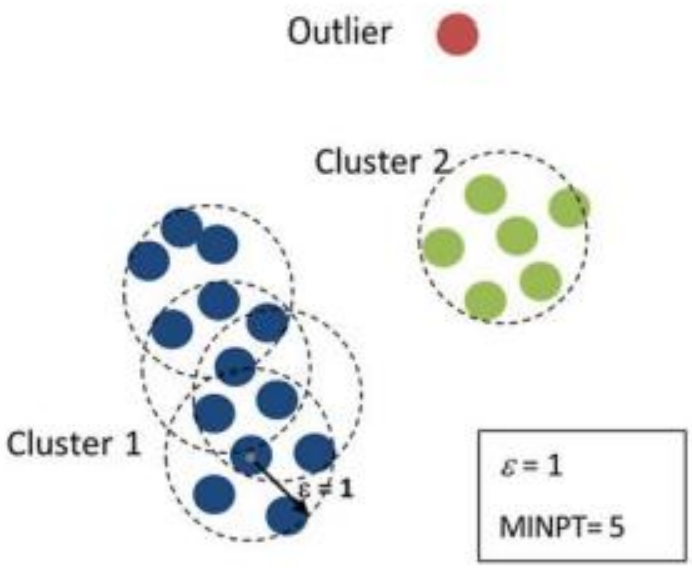

*Gambar 2.2 Ilustrasi Fuzzy C-Means (FCM)*

*Sumber: (Thakur et al, 2020)*

Pada Gambar 2.2, terlihat tiap poin data yang telah dikelompokkan menjadi kluster-kluster yang telah selesai dikelompokkan sesuai dengan karakteristik Fuzzy C-Means. Dihasilkan tiga kluster dengan warna yang berbeda mengindikasikan tiap poin data telah menjadi anggota dari suatu kluster. Tiap poin data yang ada pada suatu kumpulan data dapat terlihat tidak terkumpul dengan suatu kumpulan poin data, namun poin data tersebut tetaplah menjadi bagian dari kluster atau kelompok maupun golongan menurut aturan metode algoritma Fuzzy C-Means.

### 2.2.6. DBSCAN

*Density-Based Spatial Clustering of Application with Noise* (DBSCAN) adalah teknik pengelompokan mengacu pada karakteristik data yaitu kepadatan data atau *minimum points* (*MinPts*) dan radius atau cakupan dari pusat suatu kepadatan data yang disebut *Eps*(ε) dari setiap data. Kepadatan data merupakan banyaknya titik data atau poin data maupun datum dalam area inti yang cakupannya dipengaruhi oleh radius *Eps*. Metode DBSCAN mengelompokkan dengan membuat kluster menurut kepadatan yang cukup tinggi sehingga data yang tidak masuk ke dalam kluster akan dianggap sebagai pencilan atau *noise* (Risman et al., 2019). Penggunaan DBSCAN menghasilkan tiga status yang meliputi *noise*, *core*, dan *border*. *Core* merupakan area inti dari kluster didasarkan pada densitas [jumlah titik minimum pada radius *Eps*(ε)] sedangkan *noise* adalah sebuah titik posisi yang tidak terjangkau oleh *core*. Titik yang terletak pada batasan antara area inti dengan area *noise* akan memiliki status *border* (Dewi, 2021).



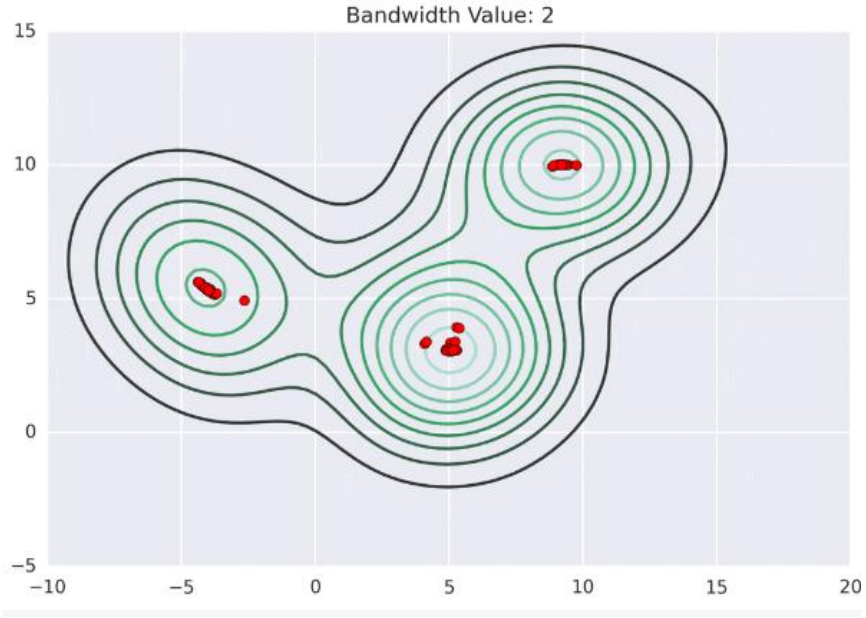

*Gambar 2.3 Ilustrasi DBSCAN*

*Sumber: (Pusadan, 2019)*

Pengelompokan pada DBSCAN menurut Ester dkk. (1996) dimulai dengan melakukan inisialisasi untuk dua nilai yaitu kepadatan data minimal (*MinPts*) dan radius *Eps*(ε). Algoritma pada DBSCAN akan memilih titik acak dan melakukan pengecekan tetangganya dalam radius *Eps*. Pengecekan jarak antar titik yang dilakukan oleh Ester menggunakan prinsip *euclidean*. Jika jumlah titik dalam radius tersebut mencapai atau melebihi jumlah minimum (*MinPts*), titik tersebut diklasifikasikan sebagai titik inti (*core*) dan sebuah kluster akan terbentuk. Jika titik acak yang dipilih merupakan titik tepi (*border*), berarti jumlah titik dalam radius *Eps* kurang dari nilai kepadatan data minimal dan algoritma akan melanjutkan pengecekan ke titik data selanjutnya.

### 2.2.7. Mean-Shift

Mean-shift dikenal sebagai teknik pengelompokan tanpa parameter (*nonparametric*) dengan mengidentifikasi mode atau *local 9maxima* menggunakan fungsi *Kernel Density Estimates* (KDE) lalu menggeser titik data secara iteratif menuju area dengan kepadatan terdekat sesuai prinsip *gradient ascent* (Qiao et al, 2022). Teknik ini dikenalkan oleh Fukunaga dan Hostetler pada tahun 1975 dan digeneralisir atau diperluas oleh Cheng pada tahun 1995. Berbeda dengan metode *clustering* K-Means yang membutuhkan jumlah kluster sebagai parameter awal, mean-shift bekerja dengan melakukan estimasti terhadap distribusi kepadatan data kemudian secara otomatis menemukan jumlah kluster berdasarkan pola yang ada dalam data tersebut. Karakteristik ini membuat mean-shift berguna dalam situasi ketika dihadapkan struktur data yang tidak diketahui sebelumnya atau pola distribusinya kompleks. Karakteristik mean-shift membuat algoritma ini unggul dengan sifat adaptif dan fleksibel karena tidak membutuhkan pengetahuan terdahulu mengenai jumlah kluster serta tidak membatasi bentuk kluster sehingga membantu dalam menangani pengelompokan data yang seringkali tidak terdistribusi dalam pola tertentu (Bepery et al, 2021).




*Gambar 2.4 Ilustrasi Mean-shift*

*Sumber: (Analytics India Magazine, 2025)*

Menurut Demirovic pada tahun 2019, Mean-shift berjalan dengan langkah sebagai berikut,

1. Melakukan konstruksi untuk probabilitas kepadatan menggunakan *Kernel Density Estimation* (KDE) melalui Persamaan 2.3

$$\hat{f}(x) = \frac{1}{Nh^D} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right) \tag{2.3}$$

dengan,

$N$ adalah jumlah titik data,
$h$ adalah parameter,
$D$ adalah dimensi data,
$K$ adalah fungsi kernel,
$x$ adalah data sampel,
$\frac{x-x_i}{h}$ adalah jarak ternormalisasi

2. Melakukan konstruksi *Density Gradient Estimation* untuk menentukan maksimum peningkatan kepadatan dengan Persamaan 2.4

$$\hat{\nabla} f_{h,K}(x) \equiv \nabla \hat{f}_{h,K}(x) = \frac{2c_{k,D}}{Nh^{D+2}} \sum_{i=1}^{N} (x - x_i)\, k'\left(\left\|\frac{x - x_i}{h}\right\|^2\right) \tag{2.4}$$

dengan,

$C$ adalah konstanta normalisasi,
$k'$ adalah turunan fungsi kernel

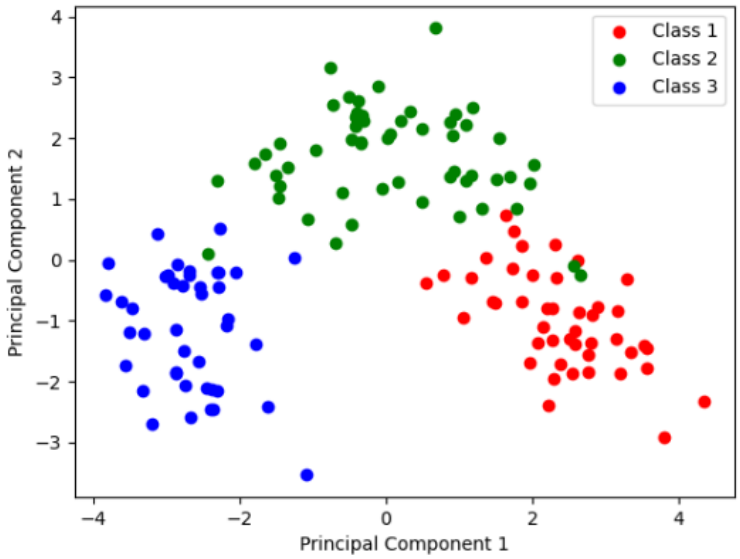
3. Menghitung vektor Mean-shift untuk mengidentifikasi pusat distribusi data

$$m_{h,K}(x) = \frac{\sum_{i=1}^{n} x_i k\left(\left\|\frac{x - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} k\left(\left\|\frac{x - x_i}{h}\right\|^2\right)} - x \tag{2.5}$$

4. Setiap poin akan berpindah secara iteratif berdasarkan pada vektor Mean-shift menuju pusat kepadatan data
5. Proses akan berhenti ketika tidak ada titik data yang bergerak secara signifikan terhadap vektor mean-shift atau konvergen

### 2.2.8. Principal Component Analysis (PCA)

*Principal Component Analysis* (PCA) adalah teknik reduksi dimensi non-parametrik yang bekerja dengan cara melakukan transformasi terhadap fitur atau variabel asli yang berkorelatif menjadi sebuah kumpulan variabel baru yang representatif. Sekumpulan variabel baru ini disebut *principal components* atau PC dimana komponen ini dihasilkan melalui kombinasi linear ortogonal dari variabel asli sebelumnya kemudian diurutkan berdasarkan varian yang ditentukan. Komponen pertama, biasanya disebut PC1, menangkap varians terbesar dalam data, dan komponen kedua, yang biasanya disebut PC2, menangkap varians terbesar berikutnya yang ortogonal terhadap PC1, dan hal ini berlaku seterusnya. Proses ini dilakukan terus menerus dengan menghitung *eigen value* serta *eigen vector* dari matriks kovarians, biasa disebut matriks korelasi, dari sebuah data. Secara matematis, *principal component* diperoleh dari dekomposisi nilai *singular* (*Singular Value Decomposition* atau SVD) atau diagonalisasi matriks kovarians, dimana vektor *eigen* melakukan representasi arah dari *principal component* tersebu serta nilai *eigen* akan mengukur besarnya varians (Jolliffe et al, 2016).




*Gambar 2.5 Visualisasi Principal Component pada PCA*

*Sumber: (GeeksforGeeks, 2025)*

Tujuan utama PCA adalah mengkompres data dengan cara melakukan pengurangan terhadap dimensi fitur-fitur sambil mempertahankan sebanyak mungkin variabilitas asli. Hal ini dicapai dengan melakukan proyeksi data ke ruang dimensi lebih rendah yang direntang oleh beberapa *principal component* utama. Dalam praktiknya sendiri, teknik PCA sendiri sering digunakan untuk menghilangkan multikolinearitas, visualisasi data multidimensi, serta ekstraksi fitur dalam *machine learning*. Kriteria pemilihan jumlah *principal component* salah satunya adalah akumalsi varians dengan nilai ≥80-95%. Kelebihan PCA terletak pada kemampuannya mengidentifikasi pola dominan secara objektif, meskipun interpretasi *principal component* bergantung pada pemahaman domain (Abdi et al, 2010)

### 2.2.9. NASA Metrics Data Program Defect Dataset

NASA atau *National Aeronautics and Space Administration* adalah sebuah organisasi dari Amerika yang bergerak dalam bidang eksplorasi antariksa dan riset aeronautika. NASA Software Independent Verification and Validation Facility (IV&V) merupakan bagian dari NASA yang bergerak dengan banyak perangkat lunak. IV&V mengeluarkan data yang berisi metrik data produk perangkat lunak yang pernah diulas. Badan ini tidak akan mengulas modul perangkat lunak jika tidak ditemukan kemungkinan cacat melalui alat bantu seperti perangkat McCabe (Jiang, 2007). Data dari *dataset* merupakan data tabular yang awalnya disebarluaskan oleh NASA pada situs web resmi IV&V, namun situs tersebut sudah tidak lagi beroperasi. Sebuah repositori daring dari Github milik klainfo kemudian mengunggah *backup* pada tahun 2016 yang dapat diakses melalui tautan berikut https://github.com/klainfo/NASADefectDataset. Dari modul-modul produk perangkat lunak yang ada, terdapat beberapa modul dengan detail sebagai yang terangkum pada Tabel 2.3 dan Tabel 2.4. Pada Gambar 2.6, diberikan contoh potongan data dengan format data .arff yang didapatkan langsung dari sumber Github milik Klainfo yang telah disebutkan diatas. Selain itu, terdapat fitur-fitur yang digunakan pada penelitian ini. Fitur tersebut dirangkum pada Tabel 2.5 dengan nilai minimum dan nilai maksimum dari tiap fitur.

*Tabel 2.3 Deskripsi Kumpulan Data NASA MDP Defect Dataset*

| Nama Kumpulan Data | Deskripsi |
|---|---|
| CM1 | Instrumen sistem kapal luar angkasa NASA dalam bahasa C |
| JM1 | Prediksi r*eal-time ground system* dalam bahasa C |
| KC1 | Sistem manajemen untuk menerima dan memproses *ground* data dalam bahasa C++ |
| KC3 | Sistem manajemen untuk menerima dan memproses *ground* data dalam bahasa Java |
| MC1 | Projek dalam bahasa C++ |
| MC2 | Sistem video panduan dalam bahasa C |
| MW1 | Projek dalam bahasa C |
| PC1 | Perangkat lunak penerbangan untuk satelit pengorbit bumi dalam bahasa C |
| PC3 | - |
| PC4 | - |
| PC5 | Projek perangkat lunak dalam bahasa C |

*Tabel 2.4 Rincian NASA MDP Dataset*

| Nama Kumpulan Data | Jumlah Atribut | Jumlah Modul | Jumlah Modul Cacat (Y) | Jumlah Modul Tidak Cacat (N) | Jumlah Modul Cacat (%) |
|---|---|---|---|---|---|
| CM1 | 38 | 327 | 42 | 285 | 12,8 |
| JM1 | 22 | 7720 | 1612 | 6108 | 20,8 |
| KC1 | 22 | 1162 | 294 | 868 | 25,3 |
| KC3 | 40 | 194 | 36 | 158 | 18,5 |
| MC1 | 39 | 1952 | 36 | 1916 | 1,8 |
| MC2 | 40 | 124 | 44 | 80 | 35,4 |
| MW1 | 38 | 250 | 25 | 225 | 10,0 |
| PC1 | 38 | 679 | 55 | 624 | 8,1 |
| PC3 | 38 | 1053 | 130 | 923 | 12,3 |
| PC4 | 38 | 1270 | 176 | 1094 | 13,9 |
| PC5 | 39 | 1694 | 458 | 1236 | 27,0 |

```
1. @relation JM1
2.
3. @attribute LOC_BLANK numeric
4. @attribute BRANCH_COUNT numeric
5. @attribute LOC_CODE_AND_COMMENT numeric
6. @attribute LOC_COMMENTS numeric
7. @attribute CYCLOMATIC_COMPLEXITY numeric
8. @attribute DESIGN_COMPLEXITY numeric
9. @attribute ESSENTIAL_COMPLEXITY numeric
10. @attribute LOC_EXECUTABLE numeric
11. @attribute HALSTEAD_CONTENT numeric
12. @attribute HALSTEAD_DIFFICULTY numeric
13. @attribute HALSTEAD_EFFORT numeric
14. @attribute HALSTEAD_ERROR_EST numeric
15. @attribute HALSTEAD_LENGTH numeric
16. @attribute HALSTEAD_LEVEL numeric
17. @attribute HALSTEAD_PROG_TIME numeric
18. @attribute HALSTEAD_VOLUME numeric
19. @attribute NUM_OPERANDS numeric
20. @attribute NUM_OPERATORS numeric
21. @attribute NUM_UNIQUE_OPERANDS numeric
22. @attribute NUM_UNIQUE_OPERATORS numeric
23. @attribute LOC_TOTAL numeric
24. @attribute label {Y,N}
25.
26. @data
27. 3,1,0,0,1,1,1,12,57.51,4.26,1046,0.08,55,0.23,58.11,245.27,29,26,17,5,17,N
28. .
```

*Gambar 2.6 Dataset JM1*

*Tabel 2.5 Fitur yang dimiliki Seluruh Dataset*

| Fitur | Deskripsi | Nilai Minimum | Nilai Maksimum |
|---|---|---|---|
| CYCLOMATIC_COMPLEXITY atau v(g) | Kompleksitas *cyclomatic* | 1 | 470 |
| ESSENTIAL_COMPLEXITY atau ev(g) | Kompleksitas *essential* | 1 | 402 |

| Fitur | Deskripsi | Nilai Minimum | Nilai Maksimum |
|---|---|---|---|
| DESIGN_COMPLEXITY atau iv(g) | Kompleksitas desain | 1 | 290 |
| HALSTEAD_VOLUME (V) | Volume | 0.0 | 174650.29 |
| HALSTEAD_LENGTH (N) | *Program length* | 0 | 15682 |
| HALSTEAD_LEVEL (L) | *Program level* | 0.0 | 2.0 |
| HALSTEAD_DIFFICULTY (D) | *Difficulty* | 0.0 | 833.78 |
| HALSTEAD_CONTENT (I) | *Intelligence Content* | 0.0 | 14763.91 |
| HALSTEAD_EFFORT (E) | *Effort* | 0.0 | 31159817.68 |
| HALSTEAD_ERROR_EST (B) | *Number of error estimation* | 0.0 | 58.22 |
| HALSTEAD_PROG_TIME (T) | *Time estimator* | 0.0 | 1731100.99 |
| NUM_UNIQUE_OPERATORS | Jumlah operator unik | 0 | 411 |
| NUM_UNIQUE_OPERANDS | Jumlah operan unik | 0 | 2241 |
| NUM_OPERATORS | Total operator | 0 | 10862 |
| NUM_OPERANDS | Total operan | 0 | 5169 |
| LOC_TOTAL | Jumlah baris | 0 | 704 |
| LOC_COMMENTS | Jumlah baris komentar | 0 | 901 |
| LOC_BLANK | Jumlah baris kosong (*whitespace*) | 0 | 704 |
| LOC_EXECUTABLE | Jumlah baris berisi kode dan *whitespace* | 0 | 2824 |
| LOC_CODE_AND_COMMENT | Jumlah kode dan komentar | 0 | 180 |
| BRANCH_COUNT | *Flow graph branch's count* | 1 | 826 |
| Label (*defective state*) | Label apakah terdapat cacat atau tidak pada modul perangkat lunak dengan nilai N atau Y | Fitur ini hanya memiliki nilai N dan Y | |

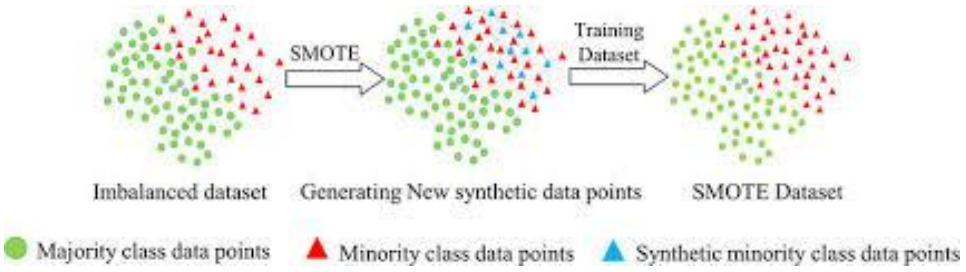
Pada *dataset* NASA MDP JM1, terdapat label yang menyatakan ditemukannya kecacatan atau tidak dengan label N (*No* atau tidak ada kecacatan) atau Y (*Yes* atau ada kecacatan). Label tersebut didapatkan dari NASA melalui laporan yang NASA terima mengenai proyek tersebut. Setiap baris data berisi runtutan angka seperti pada baris ke-27 dari Gambar 2.6. Setiap angka merepresentasikan nilai dari suatu metrik, misal pada angka pertama yang berarti baris data tersebut memiliki nilai LOC_BLANK 3. Dari Tabel 2.4, terlihat persentase jumlah modul cacat dan tidak cacat yang tidak seimbang melalui kolom persentase jumlah modul cacat sehingga diperlukan penanganan lebih lanjut melalui praproses.

### 2.2.10. Synthetic Minority Over-sampling Technique (SMOTE)

SMOTE diperkenalkan oleh Chawla dkk, pada tahun 2002. SMOTE sendiri adalah teknik *oversampling* yang digunakan untuk menangani ketidakseimbangan sebuah kelas dalam *dataset*. Hal ini biasa dilakukan ketika jumlah data pada kelas minoritas jauh lebih sedikit dibandingkan dengan kelas mayoritas. SMOTE menciptakan data sintetis untuk kelas minoritas, bukan hanya

menduplikasi data yang ada seperti pada *random oversampling* yang menduplikasi poin data pada kelas minoritas. SMOTE bekerja dengan mengikuti cara sebagai berikut,

a. Untuk setiap poin data pada kelas minoritas, akan dicari $k$ terdekat dari kelas minoritas,
b. Pilih satu poin data yang merupakan tetangga secara acak
c. Buat poin data baru dengan menggunakan interpolasi linear antara data asli dan tetangganya.




*Gambar 2.7 Ilustrasi SMOTE*

*Sumber: (RPubs, 2025)*

### 2.2.11. Metrik Kompleksitas Program dari Halstead dan McCabe

Dalam membahas kompleksitas sebuat program, terdapat beberapa metrik yang dapat digunakan. Halstead (Halstead, 1977) pada konteks ini menyediakan metrik sebagai berikut,

**a. Vocabulary (η)**
*Vocabulary* menunjukkan jumlah keunikan semesta yang terdiri dari jumlah jenis operator dan operan yang ada dari sebuah program.

$$\eta = \eta_1 + \eta_2 \tag{2.6}$$

$\eta$ : Jumlah keunikan semesta
$\eta_1$: Jumlah jenis operator unik yang muncul, seperti: $+, -, *, :, <, >$
$\eta_2$: Jumlah jenis operan (variabel) yang muncul

**b. Program Length (N)**
*Program Length* menunjukkan kemunculan semesta dari semua operator dan operan dari sebuah program

$$N = N_1 + N_2 \tag{2.7}$$

$N$: Jumlah kemunculan semesta
$N_1$: Jumlah kemunculan operator
$N_2$: Jumlah kemunculan operan

**c. Volume (V)**
*Volume* (V) menunjukkan ukuran volume sebuah program dengan mempertimbangkan *program length* dan *vocabulary*. Volume dari sebuah program dapat berubah ketika program dikonversi menjadi bahasa pemrograman lain

$$V = N \times \log_2 \eta \quad (2.8)$$

**d. Program Level (L)**

*Program Level* (L) menunjukkan seberapa mudah sebuah program dapat dipahami. Semakin tinggi nilainya, semakin sulit sebuah program untuk dipahami

$$L = \frac{2}{\eta_1} \times \frac{\eta_2}{N_2} \quad (2.9)$$

**e. Difficulty (D)**

*Difficulty* (D) menunjukkan seberapa mudah sebuah kode untuk dibaca dan dimodifikasi. Semakin tinggi nilainya, semakin sulit kode dibaca dan dimodifikasi

$$D = \frac{1}{L} = \frac{\eta_1}{2} \times \frac{N_2}{\eta_2} \quad (2.10)$$

**f. Intelligent Content (I)**

*Intelligent Content* (I) menunjukkan kompleksitas suatu algoritma dalam sebuah program, hal ini tidak bergantung pada bahasa yang digunakan untuk mengekspresikan algoritma tersebut. Semakin tinggi nilai I, semakin kompleks algoritma yang ada.

$$I = LV \quad (2.11)$$

**g. Effort (E)**

*Effort* (E) menunjukkan seberapa tinggi usaha yang dibutuhkan untuk mengembangkan suatu program. Semakin tinggi maka semakin banyak usaha yang dibutuhkan sebuah program untuk dikembangkan.

$$E = \frac{V}{L} = D \times V \quad (2.12)$$

**h. Implementation Time / Programming Time (T)**

*Implementation Time* atau *Programming Time* (T) menujukkan berapa perkiraan waktu implementasi atau waktu pengembangan sebuah program. Nilai perkiraan waktu implementasi didapatkan dengan membagi nilai *effort* dengan *Strout's number* (S). *Strout's number* yang biasanya digunakan adalah 18.

$$T = \frac{E}{S} = \frac{E}{18} \quad (2.13)$$

**i. Number of Delivered Errors / Error Estimation (B)**

Persamaan 2.13 menampilkan *Error Estimation* (B) yang menunjukkan estimasi jumlah error yang terdeteksi pada sebuah modul perangkat lunak dengan batas nilai kritis 3000.

$$B = \frac{V}{3000} \quad (2.14)$$

Selain itu, terdapat metrik perangkat lunak milik McCabe (McCabe, 1976) sebagai berikut,

**a. Cyclomatic Complexity [v(g)]**

*Cyclomatic complexity* merupakan sebuah turunan dari *flowgraph* dan secara matematis dihitung dengan teori graf. *Cyclomatic complexity* menunjukkan jumlah jalur dari sebuah program.

$$v(g) = E - N + 2P \tag{2.15}$$

dengan

v(g) adalah *Cyclomatic complexity*,
E adalah jumlah *edge* (garis penghubung dua titik pada graf),
N adalah jumlah *vertices* atau *node* (titik pada graf),
P adalah jumlah komponen penghubung.

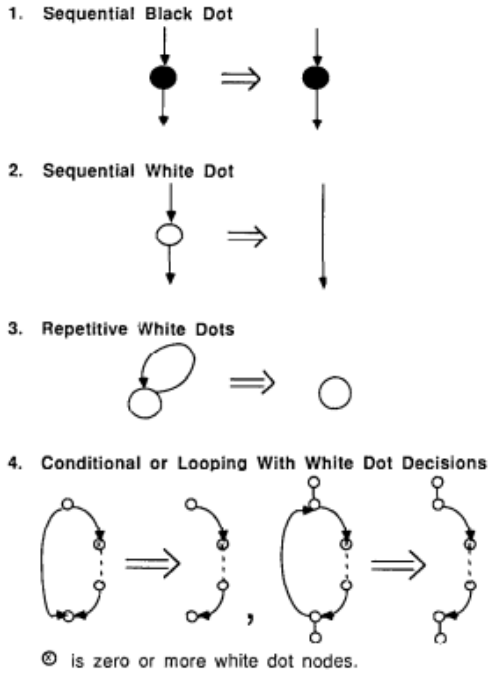
**b. Essential Complexity [ev(g)]**

*Essential complexity* atau ev(g) menunjukkan kompleksitas yang esensial dari sebuah struktur program. *Essential complexity* didapatkan dengan mereduksi graf yang bukan merupakan esensi utama dari sebuah program.

$$ev(g) = v(g) - m \tag{2.16}$$

**c. Design Complexity [iv(g)]**

iv(g) merupakan kompleksitas desain modul yang berasal dari *cyclomatic complexity* dari graf yang sudah direduksi. Peraturan reduksi grafnya adalah sebagai berikut.




*Gambar 2.8 Prinsip Reduksi Kompleksitas Desain Modul*

*Sumber: (McCabe, 1989)*

1. Titik hitam: pemanggilan ke modul bawahan (*subordinate module*) tidak dapat disederhanakan (reduksi)

2. Titik putih: node (*vertices*) yang berurutan dapat disederhanakan (reduksi) menjadi satu garis
3. Titik putih berulang: pengulangan logika tanpa titik hitam dapat disederhanakan (reduksi) menjadi satu node
4. Titik putih bersyarat: Keputusan logis dengan dua jalur tanpa titik hitam dapat disederhanakan (reduksi) menjadi satu garis

### 2.2.12. Metrik Evaluasi

Dalam proses evaluasi, metrik evaluasi diperlukan sebagai alat analisis performa dengan cara menilai kinerja suatu model algoritma berdasarkan metrik yang sesuai dengan tujuan penelitian. Ketika menganalisis hasil pengelompokan pada *unsupervised learning*, dibutuhkan validasi eksternal dan internal untuk dapat melakukan validasi pada hasil algoritma (Niño, 2019). Validasi internal dan eksternal yang dapat dipakai untuk melakukan validasi yaitu:

**a. Validasi Internal**

Dalam melakukan validasi internal, dilakukan evaluasi kualitas pengelompokan tanpa campur tangan serta pengaruh dari data eksternal. Validasi dengan *Silhouette Score* dipakai dalam mengukur koherensi klaster dengan pembandingan jarak rata-rata antara data satu dengan data lainnya dalam satu kluster dengan jarak ke klaster terdekat. Metrik ini berkaitan erat dengan jarak antar data internal dalam sebuah kluster serta separasi antar kluster. Pentingnya jarak antar titik data internal yang rendah serta tingkat separasi yang tinggi antar kluster menjadikan *silhouette score* metrik yang tepat dalam mengukur kualitas kluster. Kisaran nilai yang dihasilkan oleh metrik ini adalah -1 sampai 1 dengan nilai yang lebih baik jika lebih tinggi (Rousseeuw, 1987) (Niño, 2019). Persamaan dari *Silhouette Score* menurut Rousseeuw, 1987 adalah,

$$s(i) = \frac{b(i) - a(i)}{max\,\{a(i),\ \ b(i)\}} \tag{2.17}$$

Dengan,

$s(i)$: Nilai *Silhouette* poin data ke $i$

$b(i)$: Rata-rata jarak terendah antara poin data $i$ dan semua poin data dari kluster terdekat lainnya

$a(i)$: Rata-rata jarak antara poin data $i$ dan semua poin data lain dalam kluster yang sama

**b. Validasi Eksternal**

Validasi eksternal bertujuan untuk membandingkan hasil pengelompokan model *unsupervised learning* dengan *ground truth* atau data berlabel yang telah diketahui untuk mengetahui kesesuaian pengelompokan. Proses ini penting untuk mengevaluasi sejauh mana model dapat melakukan replikasi pola label yang diharapkan dalam sebuah data. Akurasi, Presisi, *Recall*, dan *F1-Score* menjadi metrik yang efektif untuk melakukan validasi eksternal. Pada metrik-metrik validasi eksternal, terdapat nilai-nilai yang menjelaskan kesesuaian pengelompokan. Hal yang dimaksud adalah *confussion*

*matrix* yang diperkenalkan oleh Karl Pearson. *Confussion matrix* menunjukkan performa pengklasifikasi, apakah dia dapat mengklasifikasikan dengan benar dan seberapa banyak dan apa saja kesalahan yang dilakukan pengklasifikasi tersebut (Sathyanarayanan et al, 2024). Contoh *confussion matrix* pada umumnya terlihat pada Tabel 2.6.

*Tabel 2.6 Contoh Confussion Matrix*

| | | *Actual Values* | |
|---|---|---|---|
| | | *Positive* | *Negative* |
| *Predicted Values* | *Positive* | *True Positive* | *False Positive (Type I Error)* |
| | *Negative* | *False Negative (Type II Error)* | *True Negative* |

Terdapat dua tipe prediksi yaitu benar dan tidak benar. Selain daripada hal tersebut, tipe hasil pengklasifikasian yang ditunjukkan pada struktur *confussion matrix* pada tabel 2.3 adalah

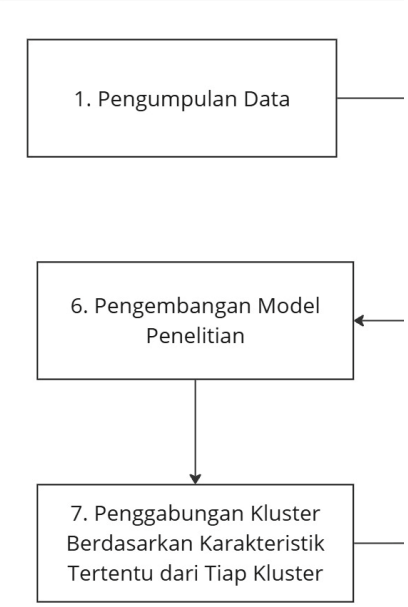
a. *True Positive* (TP), hal ini terjadi ketika kedua nilai yaitu nilai aktual dan nilai prediksi sama-sama memiliki nilai positif.
b. *False Positive* (FP), hal ini terjadi ketika prediksi dari pengklasifikasi menunjukkan nilai positif namun pada kenyataannya, nilai aktualnya adalah negatif atau bisa juga disebut sebagai *Type I Error*.
c. *False Negative* (FN), hal ini terjadi ketika prediksi dari pengklasifikasi menunjukkan nilai negatif namun pada kenyataannya, nilai aktualnya adalah positif atau bisa disebut sebagai *Type II Error*.
d. *True Negative* (TN), hal ini terjadi ketika nilai aktual dan nilai prediksi sama-sama memiliki nilai negatif

Pada bagian pertama dari TP, TN, FP, dan FN, kata *true* atau *false* mengindikasikan bahwa nilai prediksi benar atau tidak. Di sisi lain, nilai prediksi dari sebuah model pengklasifikasi berada pada bagian selanjutnya dari TP, TN, FP, dan FN yaitu berisi kata *positive* atau *negative* (Sathyanarayanan et al, 2024).

Pada pengklasifikasi biner terdapat dua hasil klasifikasi. Sebuah *confussion matrix* pada kasus pengklasifikasian biner, memberikan beberapa macam metriks yang dapat digunakan untuk menunjukkan berbagai hal seperti kesesuaian dan akurasi sebuah model. Pada penelitian ini, digunakan beberapa metrik yaitu akurasi, presisi, *recall*, dan *f1 score*.

Akurasi merupakan metrik jumlah ketepatan prediksi benar yang diberikan oleh model. Metrik ini digunakan untuk menunjukkan proporsi klasifikasi benar, positif maupun negatif. Formula akurasi adalah

$Accuracy$




$$= \frac{True\ Positives\ (TP) + True\ Negatives\ (TN}{Total\ Data} \quad (2.18)$$

Presisi mengukur jumlah positif yang benar dibagi dengan seluruh prediksi positif yang diberikan oleh model. Metrik ini digunakan untuk menunjukkan ketepatan model dalam mendeteksi data positif. Formula presisi adalah

$$Precision = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Positives\ (FP)} \quad (2.19)$$

*Recall* mengukur jumlah positif yang berhasil dideteksi dibandingkan dengan total data positif yang sebenarnya ada. Hal ini akan menunjukan sensitivitas model dalam mendeteksi data positif yang seharusnya terdeteksi. Formula *recall* adalah

$$Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Negatives\ (FN)} \quad (2.20)$$

*F1-Score* ini mengukur nilai rata-rata harmonik presisi dan *recall* sehingga dapat menangani data yang tidak seimbang (Powers et al, 2011) (Niño, 2019). *F1-Score* memiliki nilai antara 0 hingga 1. Nilai yang lebih tinggi menunjukkan performa pengelompokan yang lebih baik dalam konteks replikasi label yang benar. Formula *F1-Score* adalah

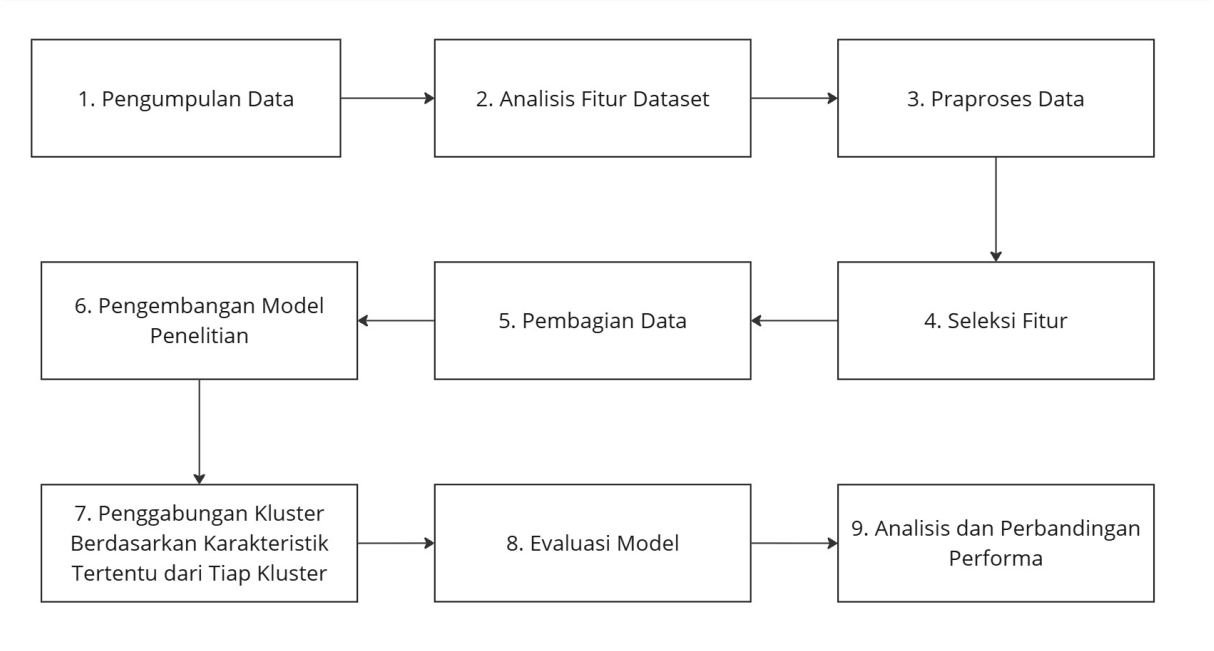
$$F1 = 2 \times \frac{Precision\ \times Recall}{Precision + Recall} \quad (2.21)$$

*[Halaman ini sengaja dikosongkan]*

`

# BAB 3
# METODOLOGI

Dalam memenuhi tujuan penelitian, dirancang sebuah metode untuk melakukan studi perbandingan performa algoritma Fuzzy C-Means, DBSCAN, dan Mean-shift dalam mendeteksi cacat dalam perangkat lunak dengan menggunakan kumpulan data yang sudah tersedia sebelumnya. Penelitian ini dijalankan dengan alur yang sistematis dan terukur. Alur penelitian ini terlihat pada diagram alir melalui Gambar 3.1




*Gambar 3.1 Diagram Alir Penelitian*

Rancangan penelitian dimulai dengan melakukan koleksi atau pengumpulan data yang berasal dari sumber data yaitu NASA MDP Software Defect. Fitur data akan dianalisis untuk memahami dasar dari tiap fitur data. Kumpulan data akan melalui praproses data, standarisasi dan seleksi fitur. Praproses data akan melalui berbagai tahap seperti transformasi data, pengecekan duplikasi data serta penanganan nilai data yang hilang. Data yang sudah melewati praproses, standarisasi, dan seleksi fitur data akan dibagi menjadi data latih dan data uji. Data tersebut menjadi dasar pengembangan model algoritma terkait yaitu Fuzzy C-Mean, DBSCAN, dan Mean-shift kemudian dijalankan untuk melakukan pengelompokan atau *clustering* data sesuai dengan karakteristik dan pola dari masing-masing algoritma sehingga menghasilkan kluster yang akan dievaluasi. Hasil pengelompokan akan dievaluasi kualitas internalnya dengan menggunakan *silhouette score* untuk mengetahui algoritma mana yang menghasilkan kluster terbaik. Setelah mengetahui kualitas hasil kluster masing-masing algoritma, dilakukan evaluasi eksternal dengan menggunakan metrik *accuracy*, *precision*, *recall*, dan *f1-score* menggunakan label data yang telah didapatkan sebelumnya (*groundtruth* dari *dataset* asli) untuk mengetahui keandalan algoritma untuk memprediksi label cacat. Dari validasi dan evaluasi yang ada, dilakukan perbandingan dan analisis terkait dengan performa masing-masing algoritma. Hasil analisis serta perbandingan akan hal yang mendasari penarikan kesimpulan algoritma dengan performa terbaik untuk mendeteksi cacat pada perangkat lunak berdasarkan metrik yang disebutkan sebelumnya.

## 3.1. Pengumpulan Data

Data yang digunakan dalam penelitian ini dikumpulkan dari NASA MDP Software Defect Dataset yang sebelumnya terletak di http://nasa-softwaredefectdatasets.wikispaces.com/space/content namun laman tersebut sudah tidak berfungsi dan disediakan kembali pada https://github.com/klainfo/NASADefectDataset. Data yang digunakan merupakan *dataset* publik berisi data yang dikumpulkan dari modul perangkat lunak NASA guna penelitian deteksi cacat pada perangkat lunak. Data mencakup metrik perangkat lunak seperti jumlah baris kode, kompleksitas design kode, dan jumlah cacat dilaporkan. Pemilihan *dataset* ini dilakukan karena karakteristik yang disajikan oleh NASA MDP Software Defect Dataset telah dispesifikasikan untuk penelitian deteksi cacat serta ketersediaan label dalam *dataset* ini dapat dimanfaatkan untuk melakukan evaluasi eksternal guna mengetahui kinerja tiap model.

Pengumpulan data dilakukan dengan melakukan *cloning* terhadap repositori kode Github pada tautan yang telah disebutkan sebelumnya. Dari sumber data yang didapatkan, terdapat 13 *dataset* yang dapat diolah. Tiap subset berisikan data dengan jumlah yang beragam seperti disebutkan pada Tabel 2.4. Data yang telah diperoleh berupa data dalam bentuk .arff. Contoh data pada *dataset* JM1 terlihat pada Gambar 2.6. Tiap baris data memiliki label N dan Y yang menunjukkan kondisi modul perangkat lunak yang tidak cacat (N) atau cacat (Y). Perlakuan ini memudahkan peninjauan *dataset* serta pengolahan data lebih lanjut pada tahap praproses data.

## 3.2. Analisis Fitur pada Dataset

*Dataset* yang dipilih merupakan data yang berasal dari National Aeronautics and Space Administration (NASA) Metrics Data Program (MDP) Defect Dataset seperti pada Gambar 2.6. Kumpulan data ini berisi permasalahan, produk serta metrik data terkait yang dikumpulkan dari perangkat lunak milik NASA. Data Dalam *dataset* ini terdapat fitur seperti CYCLOMATIC_COMPLEXITY, ESSENTIAL_COMPLEXITY, dan DESIGN_COMPLEXITY. Di dalam artikel milik Ramadhani (2024), fitur dalam *dataset* NASA dapat dideskripsikan sebagai berikut,

*Tabel 3.1 Tabel Fitur pada Dataset*

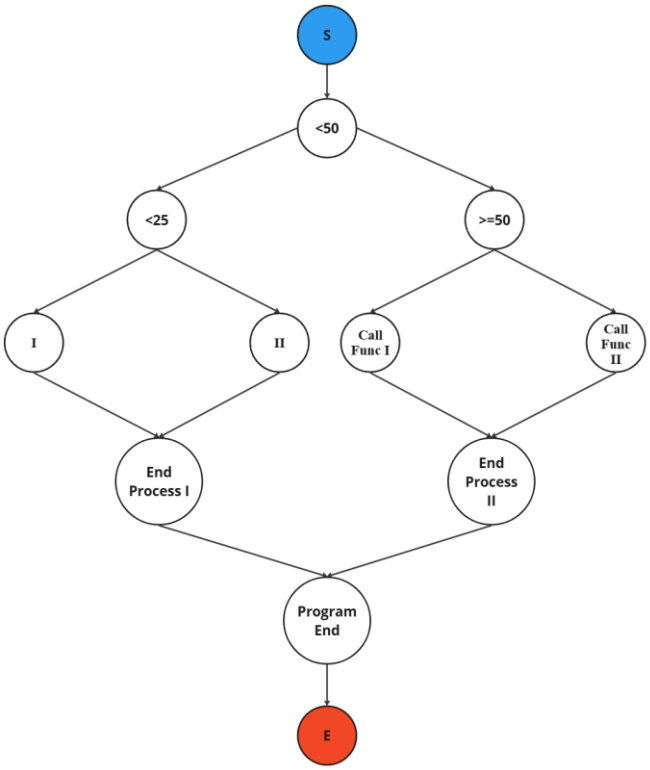
| Variabel | Deskripsi | Tipe Metrik |
|---|---|---|
| CYCLOMATIC_COMPLEXITY | Kompleksitas *cyclomatic* | McCabe |
| ESSENTIAL_COMPLEXITY | Kompleksitas *essential* | McCabe |
| DESIGN_COMPLEXITY | Kompleksitas desain | McCabe |
| HALSTEAD_VOLUME | Volume | Halstead |
| HALSTEAD_LENGTH | *Program length* | Halstead |
| HALSTEAD_LEVEL | *Program level* | Halstead |
| HALSTEAD_DIFFICULTY | *Difficulty* | Halstead |
| HALSTEAD_CONTENT | *Intelligence Content* | Halstead |
| HALSTEAD_EFFORT | *Effort* | Halstead |
| HALSTEAD_ERROR_EST | *Number of error estimation* | Halstead |
| HALSTEAD_PROG_TIME | *Time estimator* | Halstead |
| NUM_UNIQUE_OPERATORS | Jumlah operator unik | Halstead |
| NUM_UNIQUE_OPERANDS | Jumlah operan unik | Halstead |
| NUM_OPERATORS | Total operator | Halstead |

| Variabel | Deskripsi | Tipe Metrik |
|---|---|---|
| NUM_OPERANDS | Total operan | Halstead |
| LOC_TOTAL | Jumlah baris | - |
| LOC_COMMENTS | Jumlah baris komentar | - |
| LOC_BLANK | Jumlah baris kosong (*whitespace*) | - |
| LOC_EXECUTABLE | Jumlah baris berisi kode dan *whitespace* | - |
| LOC_CODE_AND_COMMENT | Jumlah kode dan komentar | - |
| BRANCH_COUNT | *Flow graph branch's count* | - |
| Defective | Terdeteksi cacat atau tidak (label diberikan oleh NASA) | - |

Tiap fitur merupakan hasil transformasi yang dihitung dengan berpadanan pada Halstead tahun 1977 dan McCabe tahun 1976. Sebagai contoh, terdapat kode dalam bahasa C pada Gambar 3.2. Pada kode di bawah, terdapat sebuah fungsi bernama *decision_tree* yang mempunyai parameter *value* bertipe data integer. Fungsi ini memiliki logika percabangan di dalamnya dengan dua cabang yang menggambarkan dua keputusan yang dapat terjadi. Tiap cabang memiliki dua cabang dengan keputusan yang berbeda-beda tiap cabangnya. Tiap cabang akan mengeluarkan keputusan atau luaran yang berbeda seperti ”Decision I\n”, “Decision II\n”, pemanggilan *function_call_one( )*, dan pemanggilan *function_call_two( )*. Setiap percabangan utama diakhiri oleh sebuah perintah luaran *printf* yang menggambarkan selesainya sebuah akhir dari bagian sebuah cabang. Kode ini dapat dikuantifikasikan metriks kompleksitasnya dengan memperhatikan prinsip-prinsip pada metrik kompleksitas McCabe dan Halstead. Demonstrasi kuantifikasi metrik McCabe dan Halstead melalui kode ini dapat memberikan gambaran tentang bagaimana NASA mendapatkan metrik-metrik terkait menggunakan *software* McCabeIQ 7.1.

```
 1. void decision_tree(int value) {
 2.     if (value < 50) {
 3.         if (value < 25) {
 4.             printf("Decision I\n");
 5.         } else {
 6.             printf("Decision II\n");
 7.         }
 8.         printf("End of Process I\n");
 9.     } else {
10.         if (value < 75) {
11.             function_call_one();
12.         } else {
13.             function_call_two();
14.         }
15.         printf("End of Process II\n");
16.     }
17. }
18.
```

*Gambar 3.2 Representasi Kode Contoh dalam Bahasa C*

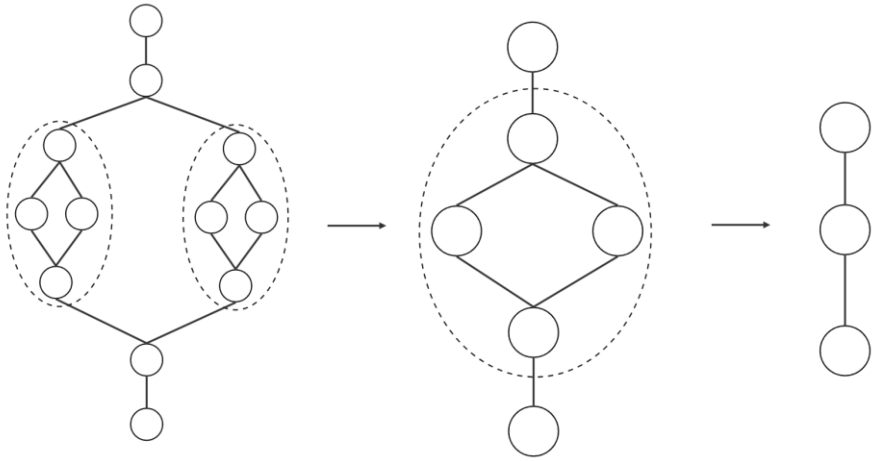



*Gambar 3.3 Control Flow Graph (CFG) Kode Contoh*

Dari Persamaan yang terdapat pada bagian 2.2.112.2.11, didapatkan perhitungan metrik kompleksitas.

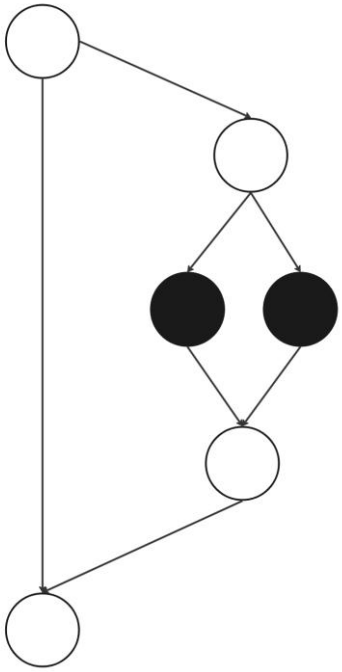
Pada v(g) atau *Cyclomatic Complexity*, didapatkan 4 dengan detail yang didapatkan dari Control From Graph diatas yaitu

*Edge* = 14, *Node* = 12, dan P = 1 sehingga E-N+2P adalah 4



*Gambar 3.4 Control Flow Graph tereduksi sesuai Prinsip Essential Complexity*

Pada ev(g) atau *Essential Complexity*, didapatkan nilai 1 didapatkan dari pengurangan 3 subgraf yang terstruktur dengan baik



*Gambar 3.5 Control Flow Graph (CFG) tereduksi sesuai Prinsip Design Complexity*

Pada iv(g) atau *Design Complexity*, didapatkan 3 setelah direduksi bagian kiri dari graf

Penghitungan apa yang dapat disebut satu operator pada bagian selanjutnya, mengikuti teori Halstead serta mengikuti perubahan yang dibuat oleh NASA pada *http://mdp.ivv.nasa.gov/count_metrics.html* yang terangkum pada tabel di bawah ini

*Tabel 3.2 Satu Operator pada Dataset NASA*

| | |
|---|---|
| if (...) ... | if (...) ... else |
| switch (.....) ... | default: |
| case:... | goto |
| do ... while (...) ... | while (...) ...do |
| for (... ; ...; ...) ... | this-> |
| [ ] ... | { } |
| <function name>( ) ... | ( ) in any other cases not covered |
| ... '?' ... ' ': ... | |

Untuk metrik Halstead, didapatkan data sebagai berikut.

Jumlah operator unik = 11 (void, decision_tree, int, ( ), { }, if, <, printf, ;, function_call_one, function_call_two)

Jumlah operan unik = 9 (value, 50, 25, 75, “Decision I\n”, “Decision II\n”, “End of Process I\n”, “End of Process II\n”)

Jumlah operator = 29 (void:1, decision_tree:1, int:1, ( ):1, { }:7, if(...) ... else:3, <:3, printf:4, ;:6, function_call_one:1, function_call_two:1).

Jumlah operan = 11 (value:4, 50:1, 25:1, 75:1, “Decision I\n”:1, “Decision II\n”:1, “End of Process I\n”:1, “End of Process II\n”:1)

Pada LOC_TOTAL, didapatkan 17 sesuai dengan jumlah baris kode.

Sehingga, dengan bantuan Persamaan pada bab 2.2.11 didapatkan,

Total operator dan operan ($N$) = 29 + 11 = 40

$$V = N \times \log_2 \eta = 40 \times \log_2 21 \approx 175{,}68$$

$$L = \frac{2}{\eta_1} \times \frac{\eta_2}{N_2} = \frac{2 \times 9}{12 \times 11} = \frac{18}{132} \approx 0{,}136$$

$$D = \frac{1}{L} = \frac{132}{18} \approx 7{,}333$$

$$I = L \times V = 0{,}136 \times 175{,}68 \approx 23{,}892$$

$$E = D \times V = 7{,}333 \times 175{,}68 \approx 1288{,}261$$

$$T = \frac{E}{S} = \frac{E}{18} = \frac{1288{,}261}{18} \approx 71{,}570$$

$$B = \frac{V}{3000} = \frac{175{,}68}{3000} \approx 0{,}059$$

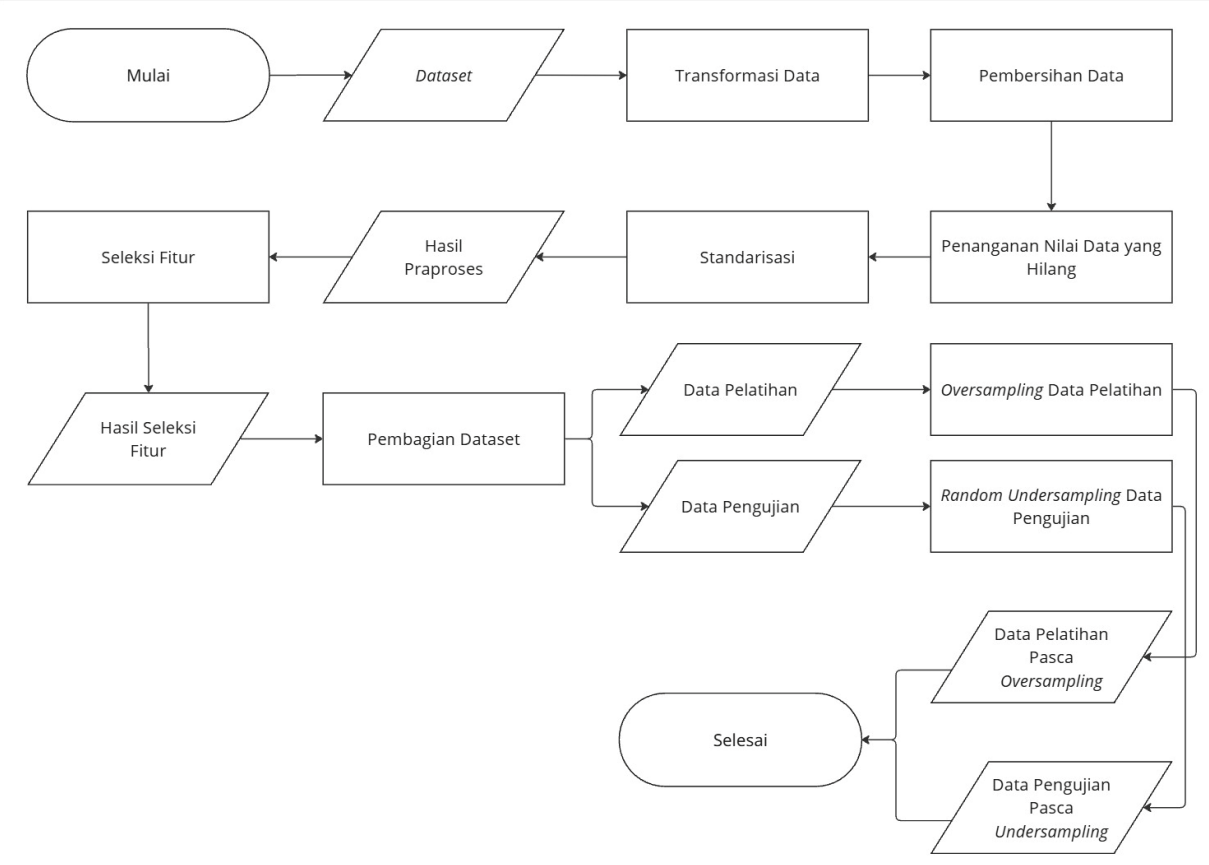
Selain itu, didapatkan branchCount sebesar 6 sesuai dengan jumlah cabang yang keluar dari *decision node* pada *flow graph* sebelum direduksi

*Tabel 3.3 Perhitungan Jumlah Metrik Kode Contoh*

| Fitur | Deskripsi | Tipe Metrik | Jumlah |
|---|---|---|---|
| CYCLOMATIC_COMPLEXITY atau v(g) | Kompleksitas *cyclomatic* | McCabe | 4 |
| ESSENTIAL_COMPLEXITY atau ev(g) | Kompleksitas *essential* | McCabe | 1 |
| DESIGN_COMPLEXITY atau iv(g) | Kompleksitas desain | McCabe | 3 |
| HALSTEAD_VOLUME (V) | Volume | Halstead | 175,68 |
| HALSTEAD_LENGTH (N) | *Program length* | Halstead | 40 |
| HALSTEAD_LEVEL (L) | *Program level* | Halstead | 0,136 |
| HALSTEAD_DIFFICULTY (D) | *Difficulty* | Halstead | 7,333 |
| HALSTEAD_CONTENT (I) | *Intelligence Content* | Halstead | 23,892 |
| HALSTEAD_EFFORT (E) | *Effort* | Halstead | 1288,261 |
| HALSTEAD_ERROR_EST (B) | *Number of error estimation* | Halstead | 0,059 |
| HALSTEAD_PROG_TIME (T) | *Time estimator* | Halstead | 71,570 |
| NUM_UNIQUE_OPERATORS | Jumlah operator unik | Halstead | 11 |
| NUM_UNIQUE_OPERANDS | Jumlah operan unik | Halstead | 9 |

| Fitur | Deskripsi | Tipe Metrik | Jumlah |
|---|---|---|---|
| NUM_OPERATORS | Total operator | Halstead | 29 |
| NUM_OPERANDS | Total operan | Halstead | 11 |
| LOC_TOTAL | Jumlah baris | - | 17 |
| LOC_COMMENTS | Jumlah baris komentar | - | 0 |
| LOC_BLANK | Jumlah baris kosong (*whitespace*) | - | 0 |
| LOC_EXECUTABLE | Jumlah baris berisi kode dan *whitespace* | - | 17 |
| LOC_CODE_AND_COMMENT | Jumlah kode dan komentar | - | 17 |
| BRANCH_COUNT | *Flow graph branch's count* | - | 6 |

## 3.3. Praproses Data, Seleksi Fitur, dan Pembagian Dataset




*Gambar 3.6 Diagram Alir Praproses, Seleksi Fitur dan Pembagian Dataset*

Data yang telah diambil dari sumber, selanjutnya dapat dipraproses untuk meningkatkan kualitas data sebelum data digunakan dalam pengembangan model. Praproses data meliputi transformasi data, pembersihan data, dan penanganan pada nilai data yang hilang. Transformasi

data dilakukan untuk mendapatkan format yang memudahkan pengolahan data. Format data yang didapatkan pada awalnya adalah .arff yang kemudian akan dirubah menjadi .csv. Setelah itu, dilakukan pembersihan data yang berfokus pada penghilangan fitur yang tidak relevan dengan penelitian dimana terdapat fitur yang tidak selalu dipunyai oleh semua *dataset* sehingga fitur yang dipertahankan adalah fitur yang dimiliki oleh semua *dataset*. Fitur data yang digunakan adalah fitur yang dimiliki oleh semua *dataset* agar dapat diseleksi sesuai kelompok fitur data pada tahap seleksi fiur selanjutnya. Selain itu, dilakukan pengecekan duplikasi data untuk mengurangi redudansi data namun tidak ditemukan data terkait sehingga dapat dilanjutkan pada proses selanjutnya. Setelah itu, dilakukan penanganan nilai data yang hilang guna mendayagunakan kembali nilai data yang tidak lengkap. Standarisasi fitur dijalankan untuk menyepadankan skala metrik modul perangkat lunak dari *dataset* yang tidak berdistribusi normal dengan membuat standar deviasinya menjadi 1 dan membuat rata-rata nya menjadi 0 sehingga performa model dapat mengalami peningkatan ketika mendeteksi cacat pada modul perangkat lunak. Setelah dilakukan praproses data, dilakukan seleksi fitur sesuai *domain knowledge* dari teori McCabe dan Halstead terhadap *dataset* terkait. Hal ini bertujuan meningkatkan performa prediksi model pembelajaran mesin serta mengurangi kompleksitas dimensi data dalam pelatihan model.

Setelah data dipraproses, data tersebut akan dibagi menjadi beberapa bagian. Data akan dipisah dengan perbandingan 8:2 dengan komposisi 80% data pelatihan dan 20% data pengujian. Data latih akan berperan dalam melatih model yang akan dikembangkan. Model akan mempelajari pola implisit dari data latih dan data uji akan menguji performa model dalam melakukan prediksi. Sebelum data digunakan, data latih akan di-*oversampling* untuk menyeimbangkan data yang akan dipelajari oleh model dan data uji akan di-*undersampling* untuk menyeimbangkan data uji agar metrik evaluasi lebih seimbang. Dari pola ini, model akan memiliki pemahaman akan data sehingga model dapat digunakan untuk menguji data-data lain dengan karakteristik yang sesuai.

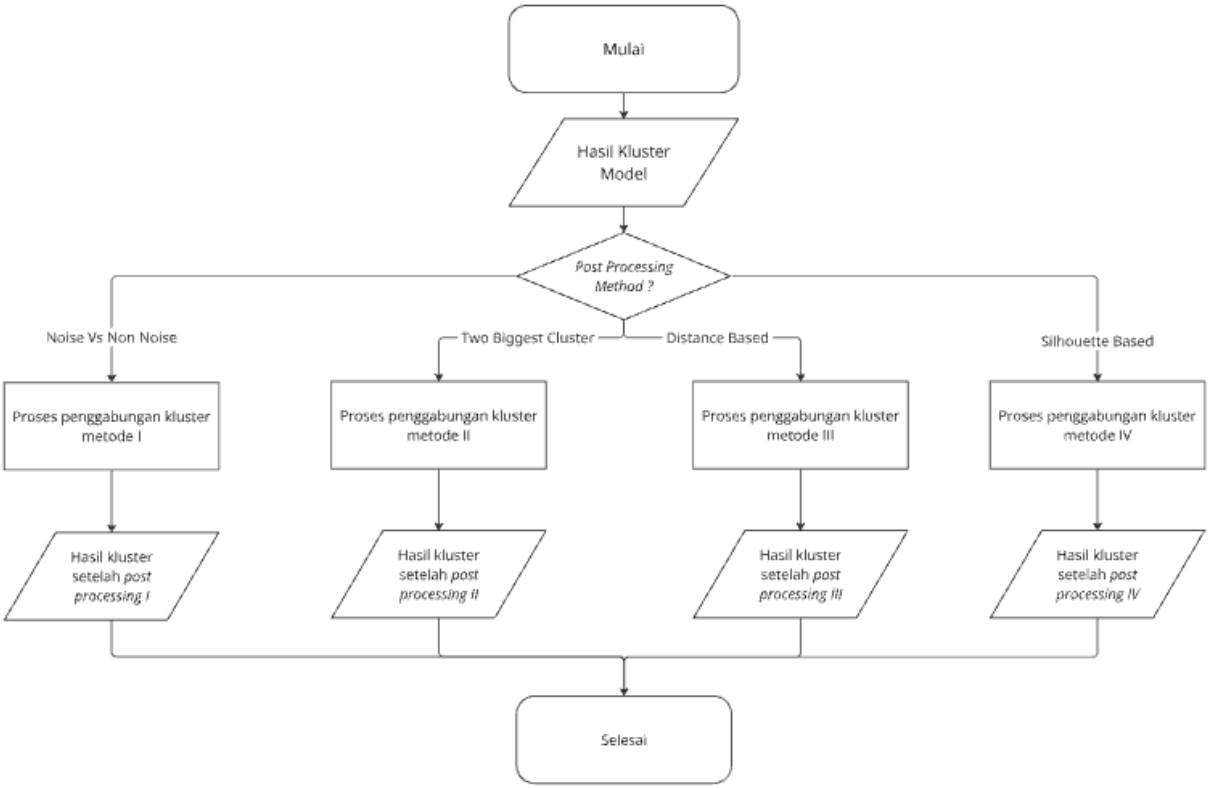
## 3.4. Pengembangan Model Penelitian

Tahap pengembangan model meliputi pengembangan tiga model yaitu Fuzzy C-Means, DBSCAN, dan Mean-shift. Tiap metode menggunakan pendekatan unik dalam mengelompokkan modul perangkat lunak dari *dataset* yang sudah dipraporses sebelumnya. Model-model ini dikembangkan untuk menguji keandalan tiap model dalam mendeteksi metrik yang sudah dipilih pada tahap praproses. Pengembangan dilakukan dengan melakukan optimalisasi parameter untuk mendapatkan hasil yang optimal dari tiap model. *Dataset* NASA yang sebelumnya telah dipraproses dan dibagi menjadi data pelatihan dan data pengujian untuk mengevaluasi kinerja model kemudian dipakai untuk melatih model yang telah dikembangkan. Ketika model telah dikembangkan, pengujian kemudian dilakukan dengan data pengujian yang telah disediakan sebelumnya.

## 3.5. Penggabungan Kluster berdasarkan Karakteristik Tertentu dari Tiap Kluster

Pada tahap ini, dilakukan pemrosesan pasca pengujian dengan hasil kluster dari model *density-based* terkait. Pemrosesan lanjut pasca pengujian model dilakukan dengan menggabungkan kluster berdasarkan karakteristik tertentu dari tiap kluster untuk hasil model *density based* (DBSCAN dan Mean-shift). Proses ini hanya dilakukan kepada hasil kluster model *density based* karena hasil kluster dari model *density based* lebih banyak daripada jumlah label yang diprediksi pada penelitian ini yaitu dua prediksi atau (*binary classification*) sehingga diperlukan penggabungan kluster agar didapatkan dua kluster yang didapatkan dengan

menggunakan metode penggabungan tertentu sehingga dapat dievaluasi pada tahap selanjutnya. Metode penggabungan dapat dilihat pada Gambar 3.7.




*Gambar 3.7 Diagram Alir Penggabungan Kluster berdasarkan Karakteristik Tertentu dari Tiap Kluster (Post Processing)*

Dari Gambar 3.7, tergambar empat metode dalam proses penggabungan kluster untuk model *density based.*

1. *Noise vs Non Noise,* merubah semua poin data dalam kluster menjadi 1 dan merubah *outlier* menjadi 0.
2. *Two Biggest Cluster*, mengekstrak semua label kluster dan mencari kluster terpadat jika label lebih dari satu (tidak termasuk *noise*) dan menjadikan semua poin data dalam dua kluster terbesar menjadi 1 dan merubah poin data sisanya menjadi 0.
3. *Distance Based,* melakukan pengecekan jumlah kluster selain *noise* apakah berjumlah lebih dari satu kemudian menghitung rata-rata *centroid* dari tiap kluster yang sebelumnya terbuat dan mengukur jarak tiap data ke *centroid* semua kluster dilanjutkan dengan memasukkan poin data ke *centroid* terdekat. Langkah terakhir mengubah label menjadi kelompok kluster ganjil dan genap dengan modulo.
4. *Silhouette Based*„ melakukan pengecekan jumlah kluster selain *noise* apakah berjumlah lebih dari satu kemudian menghitung rata-rata *silhouette score* tiap kluster kemudian memilah kluster berdasarkan nilai rata-rata *silhouette score* positif dan nilai rata-rata *silhouette score* negatif.

Setelah dilakukan *post processing*, dilakukan evaluasi model dilakukan dengan menggunakan metrik evaluasi yang telah disebutkan sebelumnya.

## 3.6. Evaluasi Model

Evaluasi internal dilakukan guna mengukur efektivitas model dalam melakukan klasifikasi cacat berdasarkan data pelatihan yang telah diberikan serta data pengujian yang diuji. Dalam penelitian ini, data pengujian berasal dari NASA MDP Software Defect Dataset yang telah melalui tahap praproses untuk meningkatkan kualitasnya. Evaluasi ini bertujuan untuk menilai sejauh mana model dapat mengenali pola dalam data serta membentuk klaster yang representatif terhadap cacat perangkat lunak. Dengan demikian, model diharapkan mampu menghasilkan klaster yang memiliki performa yang tinggi dalam mengelompokkan data sesuai dengan karakteristiknya. Selain itu, hasil evaluasi internal juga dapat memberikan wawasan mengenai kekuatan dan kelemahan model dalam menangani variasi data yang ada.

Dalam evaluasi internal, efektivitas klasifikasi diukur dengan memperhatikan dua aspek utama, yaitu kohesi dalam klaster dan pemisahan antar klaster. Kohesi pada sebuah klaster mengacu pada seberapa jauh data dalam satu klaster memiliki kesamaan yang tinggi, sehingga menunjukkan bahwa model dapat mengelompokkan data berkarakteristik sama dengan baik. Disisi lain, pemisahan atau *separation* antar klaster menggambarkan jarak antara klaster satu dengan klaster lainnya sehingga mampu menunjukkan kemampuan model dalam membedakan karakteristik perangkat lunak dengan baik. Semakin tinggi kohesi dalam klaster dan semakin besar pemisahan atau *separation* antar klaster, semakin baik performa model dalam klasifikasi. Oleh karena itu, metrik seperti *Silhouette Score* digunakan untuk mengukur efektivitas model dalam melakukan klasifikasi cacat perangkat lunak.

Evaluasi eksternal dilakukan guna mengukur kinerja model dalam menganalisis cacat yang terdapat dalam sebuah modul perangkat lunak. Proses ini bertujuan untuk memastikan bahwa model tidak hanya bekerja dengan baik pada data pelatihan, tetapi juga mampu menggeneralisasi terhadap data yang belum pernah dilihat sebelumnya. Dalam evaluasi ini, data uji yang berbeda dari data latih digunakan untuk mengidentifikasi sejauh mana model mampu mendeteksi cacat dengan tingkat akurasi yang tinggi. Selain itu, evaluasi ini juga membantu dalam menentukan apakah model mengalami overfitting atau underfitting dengan membandingkan performa pada berbagai skenario pengujian. Dengan demikian, hasil evaluasi eksternal memberikan gambaran yang lebih objektif mengenai efektivitas model dalam konteks yang lebih luas. Dalam melakukan evaluasi eksternal, dipilih F1-Score sebagai tolak ukur kemampuan model dalam meklasifikasikan cacat dengan benar. Metrik ini memungkinkan penilaian terhadap keseimbangan antara presisi dan *recall*. Hasil dari evaluasi ini akan digunakan untuk membandingkan performa tiap algoritma yang telah digunakan. Dengan adanya hasil evaluasi eksternal, dapat dianalisis dan ditemukan model yang memiliki kinerja optimal dalam mendeteksi cacat pada modul perangkat lunak.

## 3.7. Analisis dan Perbandingan Performa

Pada tahap akhir, dibuatlah perbandingan dari tiga model untuk menentukan model yang paling efektif dalam mendeteksi cacat pada modul perangkat lunak berdasarkan NASA MDP Software Defect Dataset. Perbandingan dilakukan bersamaan dengan analisis metrik performa yang diperoleh dari evaluasi internal maupun eksternal serta mempertimbangkan aspek kecepatan eksekusi dari model untuk mengetahui model dengan komputasi paling sederhana.

Analisis juga akan membahas faktor-faktor yang menyebabkan suatu model algoritma dapat menghasilkan hasil yang lebih optimal daripada model lain. Hasil dari analisis ini dapat memberikan wawasan lebih dalam mengenai efektivitas suatu model algoritma dalam mendeteksi cacat pada perangkat lunak berdasarkan *dataset* terkait.

Analisis dan perbandingan performa model akan mempertimbangan hasil yang didapatkan dari skenario-skenario yang direncanakan. Skenario pengujian yang dilakukan adalah,

1. Pengujian model pada kondisi dasar
2. Pengujian pengaruh pemilihan fitur terhadap performa model
3. Pengujian pengaruh *hyperparameter tuning* terhadap performa model terbaik
4. Pengujian PCA sebagai strategi reduksi dimensi untuk menyederhanakan kompleksitas fitur
5. Perbandingan dengan metode *supervised* dengan memanfaatkan PCA

## 3.8. Peralatan Pendukung

Untuk mendukung penelitian ini, digunakan peralatan dengan spesifikasi sebagai berikut

1. Perangkat keras
    a. *Personal Computer* (PC), sebagai piranti utama pengembang kode dan evaluasi kode. Detail spesifikasi PC yang digunakan adalah:
        1. Prosesor AMD Ryzen 5 7640HS
        2. Memori 16 GB RAM
        3. Windows 11 Home Single Language
        4. GPU NVIDIA GeForce RTX 4060, 8 GB Graphics
2. Perangkat lunak
    a. *Integrated Development Environment* (IDE), sebagai editor kode serta *compiler* kode,
    b. Python 3.10, sebagai bahasa pemrograman yang digunakan untuk menguji dan mengembangkan algoritma

## 3.9. Rencana Implementasi dan Uji Coba

Pada penelitian ini, dirumuskan langkah-langkah implementasi metode serta uji coba.

### 3.9.1. Implementasi Metode Penelitian

Pada implementasi metode penelitian, digunakan piranti serta pustaka digital dalam bahasa pemrograman Python. Pelibatan pustaka digital pada penelitian ini memudahkan praproses serta memudahkan pengembangan model. Proses uji coba akan membandingkan nilai output hasil masing-masing algoritma yang digunakan. Pada tahap awal akan dilakukan koleksi data diikuti dengan transformasi data. Pada pengkoleksian data, dibutuhkan transformasi data yang didapatkan dari sumber. Data yang memiliki format .arff atau *attribute-relation file format* akan diubah menjadi dokumen dengan format .csv atau *comma separated values*. Transformasi data yang dilakukan pada bagian ini memiliki tujuan yang dapat berguna bagi kelancaran penelitian kedepannya. Pengolahan data dengan menggunakan dokumen yang memiliki format .arff dapat meningkatkan kompleksitas pelaksanaan penelitian dan meningkatkan kemungkinan *human error* pada beberapa kasus tertentu sehingga transformasi yang dilakukan pada tahap ini dibutuhkan untuk memudahkan pengolahan selanjutnya. Pada bagian Kode Semu 3.1, terdapat sebuah kode semu yang menunjukkan sebuah fungsi untuk merubah masukan berupa dokumen yang memiliki format .arff dan kemudian fungsi tersebut akan mengolah dan membaca dokumen tersebut sedemikian rupa (seperti pembacaan bagian *attribute* dan *data*) sehingga

dapat disimpan tiap datanya pada sebuah dokumen yang memiliki format .csv seperti yang telah dijelaskan sebelumnya.

```
INPUT -> arff_file
OUTPUT -> csv_file
--------------------------------------------------------------------------------
BEGIN
    FUNCTION transform_arff_to_csv(arff_file, csv_file)
        OPEN arff_file FOR READING
        OPEN csv_file FOR WRITING

        SET in_data_section TO FALSE
        SET attributes TO EMPTY LIST

        FOR EACH line IN arff_file
            REMOVE leading and trailing whitespace FROM line

            IF line STARTS WITH "@attribute"
                PARSE attribute name FROM line
                APPEND attribute name TO attributes

            ELSE IF line STARTS WITH "@data"
                SET in_data_section TO TRUE
                WRITE attributes AS CSV HEADER TO csv_file

            ELSE IF in_data_section IS TRUE AND line IS NOT EMPTY
                WRITE line AS CSV ROW TO csv_file

        CLOSE arff_file
        CLOSE csv_file
    END FUNCTION
END
```

*Kode Semu 3.1 Kode Semu Transformasi Data*

```
INPUT -> data from file .csv
OUTPUT -> clean data and processed data
-----------------------------------------
BEGIN
    FUNCTION clean_data(data)
        FOR EACH row IN data
            FOR EACH column IN row
                CONVERT inconsistent data types IF NECESSARY
                REMOVE special characters IF NECESSARY
        RETURN cleaned_data
    END FUNCTION

    FUNCTION handle_missing_values(data, method)
        FOR EACH column IN data
            IF column CONTAINS missing values
                /*
                    Using desirable method to fill or remove missing value
                */
        RETURN processed_data
    END FUNCTION

    /*
        Checking data if it needs normalization or standardization from its distribution
    */

    FUNCTION handle_data_scale
        /*
            Normalize or standardize if needed
        */
END
```

*Kode Semu 3.2 Kode Semu Praproses Data*

Praproses data menangani inkosistensi data seperti fungsi pada baris ke-5 dan menangani nilai yang hilang seperti fungsi pada baris ke-13 serta penanganan skalasi data seperti fungsi pada baris ke-26. Setelah data dipraproses, data akan melalui seleksi fitur sesuai dengan hasil pencarian korelasi antar fitur data.

```
 1. INPUT -> Clean and processed data from previous step
 2. OUTPUT -> Group of dataset with corresponding selected feature
 3. ---------------------------------------------------------------
 4. BEGIN
 5.
 6. LOAD dataset into memory
 7.
 8. DEFINE feature groups:
 9.     mccabe = {
10.         "CYCLOMATIC_COMPLEXITY",
11.         "ESSENTIAL_COMPLEXITY",
12.         "DESIGN_COMPLEXITY",
13.         "BRANCH_COUNT"
14.     }
15.
16.     halstead = {
17.         "NUM_OPERATORS",
18.         "NUM_OPERANDS",
19.         "NUM_UNIQUE_OPERATORS",
20.         "NUM_UNIQUE_OPERANDS",
21.         "HALSTEAD_LENGTH",
22.         "HALSTEAD_VOLUME",
23.         "HALSTEAD_LEVEL",
24.         "HALSTEAD_DIFFICULTY",
25.         "HALSTEAD_EFFORT",
26.         "HALSTEAD_PROG_TIME",
27.         "HALSTEAD_CONTENT",
28.         "HALSTEAD_ERROR_EST"
29.     }
30.
31.     misc = {
32.         "LOC_CODE_AND_COMMENT",
33.         "LOC_EXECUTABLE",
34.         "LOC_COMMENTS",
35.         "LOC_BLANK",
36.         "LOC_TOTAL",
37.     }
38.
39.     feature_groups = {
40.         "mccabe": mccabe,
41.         "halstead": halstead,
42.         "misc": misc,
43.         "mccabe_halstead": mccabe ∪ halstead,
44.         "halstead_misc": halstead ∪ misc,
45.         "mccabe_misc": mccabe ∪ misc,
46.         "all": mccabe ∪ halstead ∪ misc
47.     }
48.
49. FOR each group_name, group_features IN feature_groups:
50.     SET selected_features = EMPTY_LIST
51.     FOR each column IN dataset.columns:
52.         IF column IN group_features:
53.             ADD column TO selected_features
54.
55.     dataset_group[group_name] = SELECT selected_features FROM dataset
56.
57. RETURN dataset_group
58.
59. END
60.
```

*Kode Semu 3.3 Kode Semu Seleksi Fitur*

Pada Kode Semu 3.3, dilakukan seleksi fitur untuk memaksimalkan kinerja model. Data yang telah melalui seleksi fitur akan dibagi menjadi beberapa bagian yaitu menjadi data latih dan data uji.

```
INPUT -> datasets from default case and feature-selected case
OUTPUT -> datasets from default case and feature-selected case that has been split
----------------------------------
BEGIN

INITIALIZE train_test_splits = EMPTY_MAP
INITIALIZE train_test_splits_original = EMPTY_MAP

FOR each dataset_name, balanced_dataframe IN dataframes:
    SET X = all columns of balanced_dataframe EXCEPT 'Defective'
    SET y = 'Defective' column of balanced_dataframe

    SPLIT X, y into:
        X_train, X_test, y_train, y_test
        USING 80% train, 20% test, stratified on y, with fixed random seed

    CONCATENATE X_train and y_train → train_df
    CONCATENATE X_test and y_test → test_df

    SAVE train_df and test_df INTO train_test_splits[dataset_name]

    CREATE directory: "../splits_feature_selection_case/dataset_name"
    SAVE train_df TO "train.csv" in the directory
    SAVE test_df TO "test.csv" in the directory

    PRINT class distribution of y_train and y_test

FOR each dataset_name, balanced_dataframe IN dataframes:
    REPEAT the same SPLIT process as above

    CONCATENATE X_train and y_train → train_ori_df
    CONCATENATE X_test and y_test → test_ori_df

    SAVE train_ori_df and test_ori_df INTO train_test_splits_original[dataset_name]

    CREATE directory: "../splits_original/dataset_name"
    SAVE train_ori_df TO "train.csv" in the directory
    SAVE test_ori_df TO "test.csv" in the directory

    PRINT class distribution of y_train and y_test

PRINT "All datasets split and saved!"

END

```

*Kode Semu 3.4 Kode Semu Data Splitting*

Pada pembagian data yang disebutkan pada Kode Semu 3.4, dilakukan pembagian data latih dan data uji dengan bobot data latih sebesar 80 persen dan data uji sebesar 20 persen. Pembagian data ini dilakukan pada seluruh data yang termasuk data awal (original atau *default* sebelum seleksi fitur) dan data yang telah melewati seleksi fitur. Data yang dibagi pada tahap ini digunakan untuk kebutuhan model Fuzzy C-Means, sedangkan pada kasus model yang termasuk dalam keluarga *density based clustering* seperti DBSCAN dan Meanshift akan menggunakan seluruh data untuk membuat kluster dengan menggunakan pola data berdasarkan seluruh data yang tersedia. Langkah selanjutnya adalah *oversampling* pada data latih dan *undersampling* pada data uji. Kode semu representasi bagaimana logika *oversampling* berjalan terlihat pada Kode Semu 3.5 dan Kode Semu 3.6.

```
INPUT -> Train data
OUTPUT -> Balanced train data
--------------------------------------------------
BEGIN
DEFINE function balance_with_smote(dataset, target_column = "Defective"):

    COUNT instances of each class in the target_column

    IF there is only one class:
        PRINT warning and RETURN a copy of the original dataset

    IDENTIFY:
        - Majority class and its count
        - Minority class and its count

    INITIALIZE SMOTE with default strategy (equalize all classes)
    APPLY SMOTE on:
        - Features = dataset excluding the target_column
        - Labels = target_column

    COMBINE the oversampled features and labels INTO a new balanced dataset

    RETURN the balanced dataset

END
```

*Kode Semu 3.5 Kode Semu Oversampling dengan SMOTE*

```
INPUT -> Test data
OUTPUT -> Balanced test data
----------------------------------
BEGIN

INITIALIZE undersampled_test_splits as empty dictionary

FOR each dataset_name and splits in train_test_splits:

    COPY the test set from splits

    IDENTIFY:
        - Majority class in test set
        - Minority class in test set

    SEPARATE test set INTO:
        - df_majority = rows with majority class
        - df_minority = rows with minority class

    DOWNSAMPLE df_majority to match size of df_minority with random undersampling

    CONCATENATE downsampled majority and minority rows INTO test_df_undersampled

    STORE Undersampled test set
      INTO undersampled_test_splits under dataset_name

    CREATE output directory for this dataset IF it doesn't exist

    SAVE undersampled test set to CSV in:
        ".../undersampled_splits/[dataset_name]/test.csv"
END
```

*Kode Semu 3.6 Kode Semu Undersampling dengan Random Undersampling*

Pada langkah ini, didapatkan data yang telah seimbang sehingga dapat digunakan untuk membangun model pada langkah selanjutnya. Langkah selanjutnya adalah melakukan pengembangan kode model. Model yang pertama dibangun adalah FCM seperti tertera pada Kode Semu 3.7.

```
INPUT -> datasets for corresponding scenario
OUTPUT -> evaluation metrics and confussion matrix
----------------------------------------------------------------
// Fuzzy C-Means Code

BEGIN

FOR each dataset_name IN data_case:

    SET X_train = training features from data_case[dataset_name] (excluding 'Defective')
    SET X_test = test features from data_case[dataset_name] (excluding 'Defective')
    SET y_test = ground-truth labels from fuzzy_def_case_groundtruth[dataset_name]

    INITIALIZE FCM model with 2 clusters and fixed random seed
    FIT FCM model on X_train

    PREDICT cluster labels for X_test → fcm_labels

    IF accuracy between y_test and fcm_labels < 0.5:
        INVERT fcm_labels (flip 0 ↔ 1)

    COMPUTE confusion matrix between y_test and fcm_labels → cm

    IF cm is a 2x2 matrix:
        UNPACK cm into tn, fp, fn, tp
    ELSE:
        HANDLE corner cases where one class is missing in prediction:
            - If label is 0 → tn = cm[0,0]; others = 0
            - If label is 1 → tp = cm[0,0]; others = 0

    CREATE prediction_data:
        - 'True_Label' = y_test
        - 'FCM_Raw_Prediction' = fcm_labels

    CONVERT prediction_data TO DataFrame → prediction_df

    GET test features (excluding 'Defective') → feat_df

    CONCATENATE feat_df with prediction_df → full_df

    COMPUTE evaluation metrics:
        - Accuracy
        - Precision (handle divide-by-zero)
        - Recall (handle divide-by-zero)
        - F1-score (handle divide-by-zero)
        - Silhouette score (based on X_test and fcm_labels)

    STORE all metrics along with:
        - TP, TN, FP, FN
        - Model path

END

```

*Kode Semu 3.7 Kode Semu Pengembangan Kode Model Fuzzy C-Means*

Pada bagian ini, akan dikembangkan model yang digunakan untuk mengelompokan berbagai data yang nantinya digunakan pada tiap skenario berbeda. Pada Kode Semu 3.7, dibuatlah sebuah model Fuzzy C-Means untuk menjalankan pengelompokan data dengan berbagai skenario uji. Data yang dimasukkan kedalam model berupa data latih dan data uji dengan keadaan dasar yaitu parameter n_cluster dipastikan bernilai 2 pada awalnya sehingga dapat membuat dua kluster yang merepresentasikan kasus terkait.

```
1. INPUT -> datasets for corresponding scenarios
2. OUTPUT -> evaluation metrics and confusion matrix
3. ----------------------------------
4. // DBSCAN Code
5. BEGIN
6. FOR each dataset_name and dataset_features in data_case:
7.
8.     SET dataset = filtered_dataframes[dataset_name]
9.     SET y_true = ground-truth labels for the dataset
10.
11.    SET X = feature values from dataset (excluding 'Defective' if present)
12.
13.    INITIALIZE DBSCAN with default EPS and MIN_SAMPLES
14.    FIT DBSCAN on X and GET predicted cluster labels → y_pred
15.
16.    IF y_pred has more than 1 unique label:
17.        CALCULATE silhouette score
18.    ELSE:
19.        SET silhouette score = NaN
20.
21.    INITIALIZE method_predictions as empty dictionary
22.
23.    FOR each method in methods:
24.
25.        CALL post_process_results() with:
26.            - dataset_name
27.            - method_name = method
28.            - label data and groundtruth
29.            - cluster result
30.
31.        GET:
32.            - method_results (metrics evaluation)
33.            - y_pred_bin (binary prediction for the method)
34.
35.        STORE y_pred_bin in method_predictions under current method
36.
37.        COMPUTE evaluation metrics for each method:
38.            - Accuracy
39.            - Precision (handle divide-by-zero)
40.            - Recall (handle divide-by-zero)
41.            - F1-score (handle divide-by-zero)
42.            - Silhouette score
43.
44.        STORE all metrics along with for each method:
45.            - TP, TN, FP, FN
46.            - Model path
47.
48. END
49.
```

*Kode Semu 3.8 Kode Semu Pengembangan Kode Model DBSCAN*

Pada bagian Kode Semu 3.8, diberikan kode semu pengembangan model DBSCAN. Pada bagian ini, dilakukan juga *post processing* seperti terlihat pada baris ke-23. *Post processing* pada bagian yang disebutkan merupakan penggabungan kluster menurut karakteristik tertentu dari tiap kluster untuk mendapatkan jumlah kluster yang dibutuhkan. Pada penelitian ini, dibutuhkan dua kluster yang nantinya dapat merepresentasikan kluster tidak cacat dan kluster cacat. Hasil dari tiap pemrosesan pasca pengujian model kemudian akan disimpan bersamaan dengan segala metrik evaluasi yang dibutuhkan serta *confussion matrix*. Selanjutnya, dikembangkan model untuk Mean-shift untuk melihat performa dari model terkait dalam mengelompokan dan mendeteksi cacat melalui metrik-metrik evaluasi yang dihitung melalui

hasil pengelompokan dan *groundtruth* yang berasal dari *dataset* asli NASA MDP Dataset. Pengembangan model Mean-shift terlihat pada Kode Semu 3.9.

```
INPUT -> datasets for corresponding scenarios
OUTPUT -> evaluation metrics and confusion matrix
----------------------------------
// Meanshift Code
BEGIN
FOR each dataset_name and dataset_features in data_case:

    SET dataset = filtered_dataframes[dataset_name]
    SET y_true = ground-truth labels for the dataset

    SET X = feature values from dataset (excluding 'Defective' if present)

    INITIALIZE MEANSHIFT
    FIT MEANSHIFT on X and GET predicted cluster labels → y_pred

    IF y_pred has more than 1 unique label:
        CALCULATE silhouette score
    ELSE:
        SET silhouette score = NaN

    INITIALIZE method_predictions as empty dictionary

    FOR each method in methods:

        CALL post_process_results() with:
            - dataset_name
            - method_name = method
            - label data and groundtruth
            - cluster result

        GET:
            - method_results (metrics evaluation)
            - y_pred_bin (binary prediction for the method)

        STORE y_pred_bin in method_predictions under current method

        COMPUTE evaluation metrics for each method:
            - Accuracy
            - Precision (handle divide-by-zero)
            - Recall (handle divide-by-zero)
            - F1-score (handle divide-by-zero)
            - Silhouette score

        STORE all metrics along with for each method:
            - TP, TN, FP, FN
            - Model path

END

```

*Kode Semu 3.9 Kode Semu Pengembangan Kode Model Mean-shift*

Pada kode diatas, dilakukan pengembangan model menggunakan parameter masing-masing. Parameter yang digunakan pada tiap kode pada skenario pertama adalah parameter dasar atau *default*. Setelah model dikembangkan, hasil prediksi model akan dilakukan *post processing* untuk model *density based* sebelum dievaluasi dengan metrik-metrik yang telah disebutkan sebelumnya. Berbagai *post processing* yang telah dijelaskan sebelumnya digunakan untuk memungkinkan *density based* model seperti DBSCAN dan Mean-shift mempunyai jumlah kluster yang sesuai dengan kebutuhan deteksi cacat pada penelitian ini. Kode semu dari *post processing* dipanggil pada tiap model *density based* terlihat pada Kode Semu 3.10.

```
INPUT -> prediction result from density based model
OUTPUT -> processed cluster after post processing
----------------------------------
BEGIN
LET y_pred = clustering_output and X_data = original_feature_data
IF method_name == "noise_vs_nonnoise" THEN
        IF label == -1 THEN
            SET label = 0 // Noise
        ELSE
            SET label = 1 // Non-noise
ELSE IF method_name == "two_largest_clusters" THEN
    EXTRACT all labels except -1 into valid_labels
    IF COUNT(unique(valid_labels)) >= 2 THEN
        COMPUTE frequency of each label
        SELECT two labels with highest counts as cluster_A and cluster_B
        FOR each label in y_pred DO
            IF label == cluster_A THEN
                SET label = 0
            ELSE IF label == cluster_B THEN
                SET label = 1
            ELSE
                SET label = -1
            ENDIF
        ENDFOR
        FOR each label in y_pred DO
            IF label == -1 THEN
                SET label = 0
            ELSE
                SET label = 1
            ENDIF
        ENDFOR
    ENDIF
ELSE IF method_name == "distance_based" AND X_data is not NULL THEN
    EXTRACT all labels except -1 into valid_labels
    IF COUNT(unique(valid_labels)) >= 2 THEN
        FOR each cluster_id in valid_labels DO
            EXTRACT points assigned to cluster_id
            COMPUTE mean point as cluster centroid
            ADD centroid to cluster_centers
        ENDFOR
        FOR each point in X_data DO
            COMPUTE euclidean distance to all cluster_centers
            ASSIGN point to nearest centroid
        ENDFOR
        FOR each assigned cluster index DO
            SET label = assigned_cluster_index MOD 2
        ENDFOR
    ENDIF
ELSE IF method_name == "silhouette_based" AND X_data is not NULL THEN
    EXTRACT all labels except -1 into valid_labels
    IF COUNT(unique(valid_labels)) >= 2 THEN
        COMPUTE silhouette score for each point
        FOR each cluster_id in valid_labels DO
            CALCULATE mean silhouette score for the cluster
            IF mean_score > 0 THEN
                SET label of cluster = 1
            ELSE
                SET label of cluster = 0
            ENDIF
        ENDFOR
    ENDIF
ENDIF
IF binary_classification_accuracy < 0,5 THEN INVERT label
ENDIF
RETURN y_pred as binary label
END
```

*Kode Semu 3.10 Kode Semu Post Processing Hasil Model Density Based*

### 3.9.2. Evaluasi Model dan Analisis Hasil Uji Coba

Model yang dihasilkan untuk mendeteksi cacat pada modul perangkat lunak akan dievaluasi menggunakan metrik *silhouette score* untuk mengetahui kualitas kluster dan F1-*Score*, *precision*, dan *recall*. Setelah melalui evaluasi secara kuantitatif, akan dilakukan evaluasi dan analisis kelompok fitur data yang memiliki pengaruh terbesar dalam mendeteksi cacat pada modul perangkat lunak. Analisis akan dilakukan untuk mengetahui model terbaik.

`

# BAB 4
# HASIL DAN PEMBAHASAN

Penelitian dilakukan dalam lingkungan terkontrol yaitu Python 3.10 dengan peralatan pendukung yang telah dijelaskan pada bagian 3.2. Penelitian mengikuti skenario yang hasilnya dilaporkan dan dibahas pada bab ini.

## 4.1. Hasil Penelitian

Hasil penelitian meliputi beberapa skenario uji coba untuk memperoleh berbagai hasil dari masing-masing skenario uji coba. Seluruh uji coba dilakukan untuk mendapatkan hasil metrik evaluasi masing-masing skenario guna mendapatkan bahan komparasi performa dari setiap model. Skenario uji coba yang dilakukan meliputi,

1. Skenario 1 : Skenario ini merupakan skenario untuk mengetahui kondisi dasar model dengan fitur-fitur yang dipunyai oleh semua *dataset* kemudian dilakukan dan standarisasi serta *oversampling* pada data *train* dan *undersampling* pada data *test* untuk model Fuzzy C-Means dan *post processing distance based* untuk model DBSCAN dan Meanshift menggunakan *dataset* tanpa *oversampling*.
2. Skenario 2 : Data pada skenario ini mengikuti data skenario satu kemudian dilakukan seleksi fitur menggunakan kelompok fitur terdefinisi berdasarkan teori metrik kompleksitas perangkat lunak dari Halstead dan McCabe.
3. Skenario 3 : Skenario ini dilakukan dengan menjalankan *hyperparameter tuning* pada model terbaik dengan fitur terbaik dari skenario dua dengan parameter *m*, *error*, *max_iter* untuk mengoptimalkan performa model.
4. Skenario 4 : Skenario ini menggunakan *dataset* asli tanpa mengurangi fitur dari *dataset* dan membiarkan tiap *dataset* dengan fiturnya sendiri-sendiri, kemudian dimensi fitur akan direduksi menggunakan Principal Component Analysis (PCA) kemudian data di standarisasi dan serta *oversampling* pada data *train* dan *undersampling* pada data *test* untuk model Fuzzy C-Means untuk membandingkan pengaruh reduksi dimensi dengan seleksi fitur dan PCA terhadap performa.
5. Skenario 5 : Skenario perbandingan dengan metode *supervised* dengan memanfaatkan PCA sebagai usaha reduksi dimensi fitur

### 4.1.1. Kondisi Dasar Model

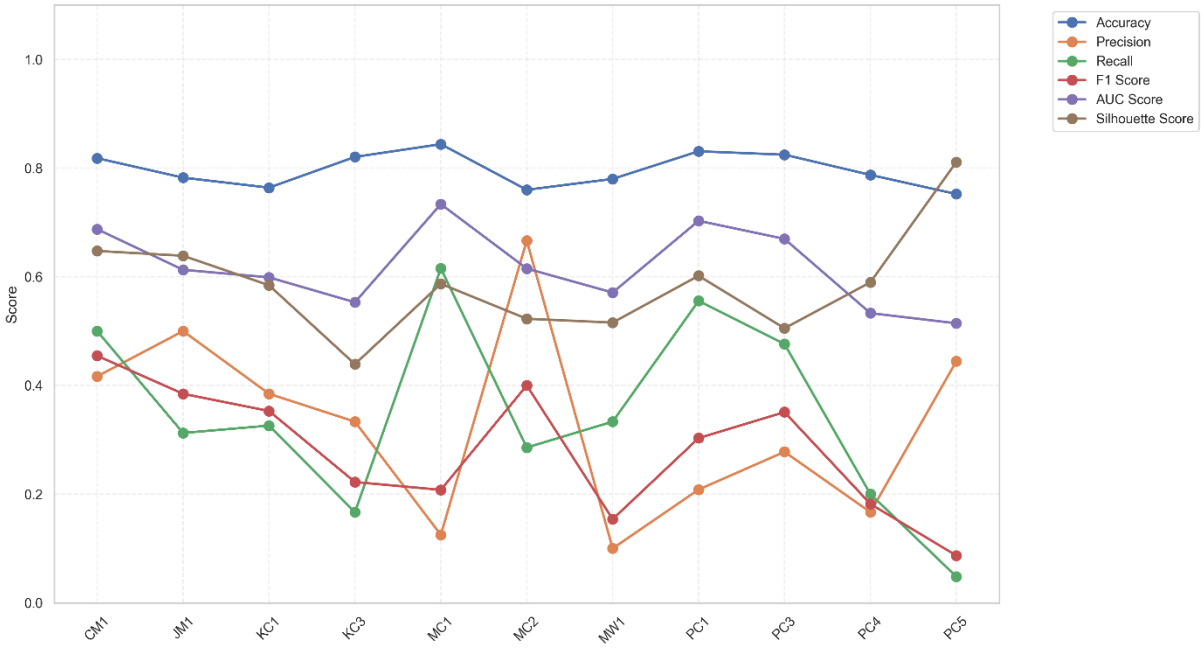
Skenario ini dilakukan dengan menjalankan model Fuzzy C-Means (FCM), DBSCAN, dan Mean-Shift dalam kondisi dasarnya tanpa melalui *hyperparameter tuning*. Pada skenario ini, dilakukan koleksi data diikuti dengan pengaksesan *dataset*. *Dataset* kemudian melalui praproses hingga siap digunakan. Data selanjutnya distandarisasi untuk mendapatkan konsistensi dan keseragaman sehingga meningkatkan kualitas dari model yang digunakan. Setelah itu, model Fuzzy C-Means, DBSCAN, dan Mean-Shift dibangun agar dapat digunakan. Pada Fuzzy C-Means, digunakan parameter n_cluster = 2 untuk membuat jumlah kluster yang dihasilkan menjadi dua sedangkan pada DBSCAN dan Mean-Shift, digunakan keadaan dasar parameter DBSCAN yaitu *eps* = 0,5 dan *min_samples* = 5 serta keadaan parameter Mean-Shift yaitu *bandwidth* = *None*. Setelah dihasilkan kluster-kluster dari model-model di atas, dilakukan *post processing* dengan cara menggabungkan kluster untuk model *density based* yaitu DBSCAN dan Mean-Shift dengan menggunakan metode *distance based.* Setelah itu, setiap kluster yang ada dievaluasi dengan *groundtruth* yang ada. Dua kluster tiap model akan disematkan label N (0) atau Y (1) kemudian dibandingkan dengan label pada *groundtruth* yang

sebelumnya sudah dirubah menjadi angka 0 untuk N dan 1 untuk Y. Pada Tabel 4.1, terdapat hasil uji coba dari FCM pada kondisi dasar dengan metrik-metrik evaluasi yaitu *accuracy*, *precision*, *recall*, *f1-score*, dan *silhouette score*.

*Tabel 4.1 Hasil Uji Coba FCM pada Kondisi Dasar*

| Dataset | Accuracy | Precision | Recall | F1 Score | Silhouette Score |
|---|---|---|---|---|---|
| CM1 | 0,6875 | 1,0000 | 0,3750 | 0,5455 | 0,6530 |
| JM1 | 0,5916 | 0,8242 | 0,2329 | 0,3632 | 0,6381 |
| KC1 | 0,5932 | 0,7391 | 0,2881 | 0,4146 | 0,5368 |
| KC3 | 0,7143 | 1,0000 | 0,4286 | 0,6000 | 0,5401 |
| MC1 | 0,5000 | 0,5000 | 0,7143 | 0,5882 | 0,4654 |
| MC2 | 0,5000 | 0,5000 | 0,7778 | 0,6087 | 0,6551 |
| MW1 | 0,7000 | 0,6667 | 0,8000 | 0,7273 | 0,5979 |
| PC1 | 0,6364 | 0,8000 | 0,3636 | 0,5000 | 0,4718 |
| PC3 | 0,6538 | 0,7500 | 0,4615 | 0,5714 | 0,4704 |
| PC4 | 0,5143 | 0,5098 | 0,7429 | 0,6047 | 0,4711 |
| PC5 | 0,5217 | 0,7000 | 0,0761 | 0,1373 | 0,7851 |

Tabel 4.1 menunjukkan hasil model FCM dengan parameter n_cluster = 2 yang merepresentasikan dua kluster cacat dan tidak cacat. Perolehan akurasi tertinggi terlihat pada *dataset* KC3 yaitu 0,7143 , sedangkan nilai presisi tertinggi terdapat pada *dataset* CM1 dan KC3 yaitu 1. Pada *dataset* MW1, didapatkan *recall* tertingi yaitu berada pada nilai 0,8000 dan pada *dataset* MW1 didapatkan nilai F1 terbaik yaitu 0,7273. Silhouette score terbaik berada pada PC5 dengan nilai 0,7851.




*Gambar 4.1 Grafik Performa Model FCM pada Dataset*

Pada model DBSCAN, dilakukan *running* pada kondisi dasar untuk melihat performa dasar model. Model DBSCAN dasar tidak dapat menghasilkan dua kluster secara langsung sehingga dilakukan *post processing* untuk mendapatkan dua kluster dengan nilai yang dapat dievaluasi.

*Post processing* yang dilakukan pada skenario ini untuk model DBSCAN adalah *distance based*. Hasil pemrosesan tersebut terlihat pada tabel 4.2 dengan nilai sebagai berikut

*Tabel 4.2 Hasil Uji Coba DBSCAN pada Kondisi Dasar dengan Metode Distance Based*

| Dataset | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|
| CM1 | Distance Based | 0,6361 | 0,0471 | 0,0952 | 0,0630 | -0,2608 |
| JM1 | Distance Based | 0,6402 | 0,2167 | 0,2767 | 0,2431 | -0,1940 |
| KC1 | Distance Based | 0,6575 | 0,2797 | 0,2245 | 0,2491 | -0,1906 |
| KC3 | Distance Based | 0,6392 | 0,0750 | 0,0833 | 0,0789 | -0,2030 |
| MC1 | Distance Based | 0,6137 | 0,0162 | 0,3333 | 0,0308 | -0,2097 |
| MC2 | Distance Based | 0,5484 | 0,2273 | 0,1136 | 0,1515 | -0,0846 |
| MW1 | Distance Based | NaN | NaN | NaN | NaN | NaN |
| PC1 | Distance Based | 0,5096 | 0,0397 | 0,2182 | 0,0672 | -0,1489 |
| PC3 | Distance Based | 0,6277 | 0,1925 | 0,6308 | 0,2950 | -0,1216 |
| PC4 | Distance Based | 0,6031 | 0,0773 | 0,1705 | 0,1064 | -0,1540 |
| PC5 | Distance Based | 0,7090 | 0,3364 | 0,0786 | 0,1274 | -0,0864 |

Pada *dataset* MW1, semua poin data diklasifikasikan menjadi *outlier* sehingga menghasilkan nilai NaN atau *Not a Number*.

Menimbang performa model DBSCAN tanpa *post processing* sebelumnya, dilakukan *post processing* pada model Mean-Shift untuk membantu model dalam memprediksi cacat pada modul perangkat lunak. Metode *post processing* yang adalah *distance based*. Hasil uji coba Mean-Shift terlihat pada tabel 4.3.

*Tabel 4.3 Hasil Uji Coba Mean-Shift pada Kondisi Dasar dengan Metode Distance Based*

| Dataset | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|
| CM1 | Distance Based | 0,7982 | 0,2273 | 0,2381 | 0,2326 | 0,4663 |
| JM1 | Distance Based | 0,7877 | 0,4574 | 0,0900 | 0,1503 | 0,3877 |
| KC1 | Distance Based | 0,7496 | 0,5128 | 0,2041 | 0,2920 | 0,3294 |
| KC3 | Distance Based | 0,7577 | 0,2381 | 0,1389 | 0,1754 | 0,3973 |
| MC1 | Distance Based | 0,9319 | 0,0381 | 0,1111 | 0,0567 | 0,4755 |
| MC2 | Distance Based | 0,6774 | 0,8333 | 0,1136 | 0,2000 | 0,4283 |
| MW1 | Distance Based | 0,8280 | 0,2500 | 0,3600 | 0,2951 | 0,4458 |
| PC1 | Distance Based | 0,8601 | 0,1875 | 0,2182 | 0,2017 | 0,3649 |
| PC3 | Distance Based | 0,8424 | 0,1897 | 0,0846 | 0,1170 | 0,3383 |
| PC4 | Distance Based | 0,8220 | 0,2685 | 0,1648 | 0,2042 | 0,3654 |
| PC5 | Distance Based | 0,7420 | 0,6400 | 0,1048 | 0,1801 | 0,4806 |

Akurasi tertinggi terlihat pada MC1 dengan metode *post processing Distance Based* dengan nilai 0,9319.

### 4.1.2. Model dengan Feature Selection

Skenario ini melalui alur yang sama dengan skenario kondisi dasar model dengan tambahan seleksi fitur yang dikelompokkan sebagai berikut,

*Tabel 4.4 Kelompok Fitur Utama Skenario 2*

| Kelompok Fitur | Nama Fitur |
|---|---|
| McCabe | CYCLOMATIC_COMPLEXITY |
| | ESSENTIAL_COMPLEXITY |
| | DESIGN_COMPLEXITY |
| | BRANCH_COUNT |
| Halstead | NUM_OPERATORS |
| | NUM_OPERANDS |
| | NUM_UNIQUE_OPERANDS |
| | NUM_UNIQUE_OPERATORS |
| | HALSTEAD_LENGTH |
| | HALSTEAD_VOLUME |
| | HALSTEAD_LEVEL |
| | HALSTEAD_DIFFICULTY |
| | HALSTEAD_EFFORT |
| | HALSTEAD_PROG_TIME |
| | HALSTEAD_CONTENT |
| | HALSTEAD_ERROR_EST |
| Misc | LOC_CODE_AND_COMMENT |
| | LOC_EXECUTABLE |
| | LOC_COMMENTS |
| | LOC_BLANK |
| | LOC_TOTAL |

Dari kelompok fitur di atas, dibuat kombinasi kelompok sehingga kelompok yang dihasilkan adalah sebagai berikut,

*Tabel 4.5 Jenis Kombinasi Kelompok Fitur dan Pasangannya*

| Jenis Kombinasi | Pasangan Kombinasi |
|---|---|
| Tanpa kombinasi | McCabe |
| | Halstead |
| | Misc (*Miscellaneous*) |
| Kombinasi dua kelompok fitur | McCabe + Halstead |
| | McCabe + Misc |
| | Halstead + Misc |
| Kombinasi tiga kelompok fitur | McCabe + Halstead + Misc (*All*) |

Skenario ini memiliki alur yang sama dengan skenario satu, namun data yang dikonsumsi oleh model adalah data yang telah melewati seleksi fitur dengan menggunakan kelompok fitur pada Tabel 4.5. Selain itu, dilakukan empat macam metode *post processing* dengan menggabungkan kluster pada hasil kluster model *density based* seperti dijelaskan pada bagian 3.5.

Hasil pada skenario ini didaptkan dengan menggunakan model pada skenario 1 yang dijalankan dengan *dataset* yang telah melalui seleksi fitur. Pada FCM didapatkan hasil pada

tabel 4.6, 4.7, dan 4.8. Tabel 4.7 dan 4.8 berisi hasil fitur terbaik yaitu kelompok Halstead yang hasilnya dijabarkan bagian selanjutnya.

*Tabel 4.6 Hasil Uji Coba Model FCM dengan Menggunakan Kelompok Fitur Halstead*

| Dataset | Accuracy | Precision | Recall | F1 Score | Silhouette Score |
|---|---|---|---|---|---|
| CM1 | 0,6875 | 1.0 | 0,375 | 0,5455 | 0,7026 |
| JM1 | 0,5823 | 0,8193 | 0,2112 | 0,3358 | 0,6596 |
| KC1 | 0,6271 | 0,8 | 0,339 | 0,4762 | 0,5251 |
| KC3 | 0,7857 | 1.0 | 0,5714 | 0,7273 | 0,6054 |
| MC1 | 0,5714 | 0,6667 | 0,2857 | 0,4 | 0,4927 |
| MC2 | 0,5 | 0,5 | 0,7778 | 0,6087 | 0,6531 |
| MW1 | 0,7 | 0,6667 | 0,8 | 0,7273 | 0,6115 |
| PC1 | 0,6364 | 0,8 | 0,3636 | 0,5 | 0,451 |
| PC3 | 0,7115 | 0,7895 | 0,5769 | 0,6667 | 0,4836 |
| PC4 | 0,5429 | 0,5294 | 0,7714 | 0,6279 | 0,4988 |
| PC5 | 0,5217 | 0,7 | 0,0761 | 0,1373 | 0,7822 |

*Tabel 4.7 Hasil Uji Coba Model DBSCAN dengan Menggunakan Kelompok Fitur Halstead dengan Metode Post Processing Noise Vs Non Noise*

| Dataset | Accuracy | Precision | Recall | F1 Score | Silhouette Score |
|---|---|---|---|---|---|
| CM1 | 0,6942 | 0,2041 | 0,4762 | 0,2857 | 0,1974 |
| JM1 | 0,7851 | 0,4477 | 0,1247 | 0,1951 | 0,1024 |
| KC1 | 0,7573 | 0,5349 | 0,3129 | 0,3948 | -0,0036 |
| KC3 | 0,6031 | 0,2588 | 0,6111 | 0,3636 | **NaN** |
| MC1 | 0,9052 | 0,0429 | 0,1944 | 0,0704 | 0,0784 |
| MC2 | 0,5565 | 0,4127 | 0,5909 | 0,486 | **NaN** |
| MW1 | 0,532 | 0,1462 | 0,76 | 0,2452 | 0,0927 |
| PC1 | 0,8365 | 0,2308 | 0,4364 | 0,3019 | 0,1685 |
| PC3 | 0,8215 | 0,2041 | 0,1538 | 0,1754 | 0,277 |
| PC4 | 0,7937 | 0,2378 | 0,2216 | 0,2294 | 0,2902 |
| PC5 | 0,732 | 0,513 | 0,1725 | 0,2582 | 0,1246 |

*Tabel 4.8 Hasil Uji Coba Model Mean-Shift dengan Menggunakan Kelompok Fitur Halstead dengan Metode Post Processing Two Biggest Cluster*

| Dataset | Accuracy | Precision | Recall | F1 Score | Silhouette Score |
|---|---|---|---|---|---|
| CM1 | 0,8654 | 0,3333 | 0,0476 | 0,0833 | 0,5834 |
| JM1 | 0,778 | 0,3825 | 0,103 | 0,1623 | 0,4644 |
| KC1 | 0,7427 | 0,4638 | 0,1088 | 0,1763 | 0,5024 |
| KC3 | 0,8144 | 0,5 | 0,0278 | 0,0526 | 0,5843 |
| MC1 | 0,96 | 0,08 | 0,1111 | 0,093 | 0,506 |
| MC2 | 0,6532 | 0,5714 | 0,0909 | 0,1569 | 0,4485 |
| MW1 | 0,872 | 0,2941 | 0,2 | 0,2381 | 0,4325 |
| PC1 | 0,8837 | 0,2391 | 0,2 | 0,2178 | 0,4455 |
| PC3 | 0,8291 | 0,0833 | 0,0385 | 0,0526 | 0,4597 |
| PC4 | 0,8409 | 0,3088 | 0,1193 | 0,1721 | 0,5055 |

| Dataset | Accuracy | Precision | Recall | F1 Score | Silhouette Score |
|---|---|---|---|---|---|
| PC5 | 0,7302 | 0,5043 | 0,1288 | 0,2052 | 0,5133 |

Model terbaik ditentukan dari *composite score* setiap kasus dengan bobot 0,3, 0,4, dan 0,3 untuk metrik *precision*, *recall*, dan *f1 score*. Hasil *composite score* tiap kelompok fitur terlihat pada tabel 4.9, tabel 4.10, dan tabel 4.11 untuk tiap model.

*Tabel 4.9 Hasil Perhitungan Kelompok Fitur Terbaik dari Model FCM*

| Feature_Group | Precision | Recall | F1 | Silhouette Score | Composite_Score |
|---|---|---|---|---|---|
| all | 0,7263 | 0,4393 | 0,4942 | 0,5713 | 0,5419 |
| halstead | 0,752 | 0,468 | 0,523 | 0,5878 | **0,5697** |
| halstead_misc | 0,7616 | 0,4216 | 0,5014 | 0,5753 | 0,5476 |
| mccabe | 0,6739 | 0,3231 | 0,3681 | 0,6046 | 0,4418 |
| mccabe_halstead | 0,7286 | 0,4194 | 0,4741 | 0,598 | 0,5286 |
| mccabe_misc | 0,7413 | 0,4283 | 0,4823 | 0,5875 | 0,5384 |
| misc | 0,7615 | 0,4086 | 0,4885 | 0,6051 | 0,5384 |

*Tabel 4.10 Hasil Perhitungan Kelompok Fitur Terbaik dari Model DBSCAN*

| Feature_Group | Precision | Recall | F1 Score | Silhouette Score | Composite_Score |
|---|---|---|---|---|---|
| all | 0,1480 | 0,2744 | 0,1589 | -0,1743 | 0,2019 |
| halstead | 0,2529 | 0,3243 | 0,2298 | 0,1475 | 0,2745 |
| halstead_misc | 0,2123 | 0,4035 | 0,2399 | -0,1316 | 0,2971 |
| mccabe | 0,3647 | 0,2232 | 0,2307 | 0,5170 | 0,2679 |
| mccabe_halstead | 0,1963 | 0,3900 | 0,2191 | -0,0768 | 0,2806 |
| mccabe_misc | 0,2830 | 0,4812 | 0,3061 | 0,1956 | **0,3692** |
| misc | 0,3378 | 0,3703 | 0,3023 | 0,3894 | 0,3401 |

*Tabel 4.11 Hasil Perhitungan Kelompok Fitur Terbaik dari Model Mean-Shift*

| Feature_Group | Precision | Recall | F1 Score | Silhouette Score | Composite_Score |
|---|---|---|---|---|---|
| all | 0,3160 | 0,2543 | 0,2542 | 0,4072 | 0,2728 |
| halstead | 0,3496 | 0,1572 | 0,1836 | 0,4950 | 0,2228 |
| halstead_misc | 0,3470 | 0,2552 | 0,2566 | 0,4290 | 0,2831 |
| mccabe | 0,3327 | 0,1886 | 0,1991 | 0,5822 | 0,2349 |
| mccabe_halstead | 0,3110 | 0,1874 | 0,2075 | 0,4399 | 0,2305 |
| mccabe_misc | 0,3426 | 0,2727 | 0,2684 | 0,4916 | 0,2924 |
| misc | 0,3642 | 0,3034 | 0,2902 | 0,5328 | **0,3177** |

Keseluruhan nilai komposit berada pada rentang 0,2019 sampai 0,5697. Nilai terbaik pada model Fuzzy C-Means menggunakan kelompok fitur Halstead dengan nilai 0,5697, dan pada model DBSCAN, kelompok fitur McCabe Misc menjadi kelompok terbaik dengan nilai 0,3692. Pada model Mean-Shift nilai terbaik berada pada model Misc dengan nilai komposit sebesar

0,3177. Dengan hasil terkait, model Fuzzy C-Means dengan kelompok fitur Halstead menjadi kombinasi model dan kelompok fitur terbaik dengan nilai tertinggi sebesar 0,5697

### 4.1.3. Model dengan Hyperparameter Tuning

Pada skenario tiga, dilakukan *hyperparameter tuning* terhadap model terbaik skenario kedua. *Hyperparameter tuning* pada model terbaik yaitu FCM dengan kelompok fitur Halstead, menghasilkan peningkatan dalam performa model. Penyesuaian parameter seperti *eksponent fuzziness* (m), jumlah iterasi maksimum (max_iter), dan toleransi error (error) menunjukkan peningkatan pada metrik evaluasi utama yaitu *precision, recall,* dan *f1 score* pada beberapa *dataset*, namun pada *dataset* tertentu, tidak terdapat kenaikkan atau stagnan. Didapatkan parameter terbaik tertera pada tabel 4.12.

*Tabel 4.12 Hasil Parameter Terbaik pada Skenario Hyperparameter Tuning pada Model Terbaik Skenario 2*

| Dataset | Parameters |
|---|---|
| CM1 | 'error': 1e-05, 'm': 1.1, 'max_iter': 10 |
| JM1 | ‘error': 1e-05, 'm': 4.7, 'max_iter': 10 |
| KC1 | 'error': 1e-05, 'm': 3.5, 'max_iter': 10 |
| KC3 | 'error': 1e-05, 'm': 1.1, 'max_iter': 10 |
| MC1 | 'error': 1e-05, 'm': 1.1, 'max_iter': 10 |
| MC2 | 'error': 1e-05, 'm': 1.1, 'max_iter': 10 |
| MW1 | 'error': 1e-05, 'm': 1.1, 'max_iter': 10 |
| PC1 | 'error': 1e-05, 'm': 2.8, 'max_iter': 10 |
| PC3 | 'error': 1e-05, 'm': 4.4, 'max_iter': 10 |
| PC4 | 'error': 1e-05, 'm': 1.1, 'max_iter': 20 |
| PC5 | 'error': 1e-05, 'm': 2.9, 'max_iter': 10 |

Selain itu, terdapat beberapa *dataset* yang mengalami penurunan *precision*, seperti pada JM1, MC1, dan PC4, dengan penurunan pada rentang 0,02 sampai 0,14. Hasil *hyperparameter tuning* terlihat pada tabel 4.13 . Kenaikkan metrik evaluasi *recall* dan *f1 score* terjadi pada mayoritas *dataset* berkisar 0,01-0,3 dengan parameter yang berbeda-beda untuk tiap *dataset*. Metrik akurasi mengalami penurunan ketika meningkatkan nilai metrik evaluasi utama yaitu *precision*, *recall*, dan *f1 score*. Selain itu, nilai *silhouette score* cenderung menurun ketika hanya memfokuskan *hyperparameter tuning* pada metrik evaluasi utama dengan rentang penurunan mulai dari 0,02, namun terdapat beberapa *dataset* yang tidak menurun maupun naik nilainya.

*Tabel 4.13 Hasil Hyperparameter Tuning pada Model Terbaik Skenario 2*

| Dataset | Accuracy | Precision | Recall | F1 Score | Silhouette Score |
|---|---|---|---|---|---|
| CM1 | 0,6875 | 1,000 | 0,3750 | 0,5455 | 0,7026 |
| JM1 | 0,6273 | 0,7071 | 0,4348 | 0,5385 | 0,4582 |
| KC1 | 0,6271 | 0,7273 | 0,4068 | 0,5217 | 0,4828 |
| KC3 | 0,7857 | 1,000 | 0,5714 | 0,7273 | 0,6054 |
| MC1 | 0,5000 | 0,500 | 0,8571 | 0,6316 | 0,5024 |
| MC2 | 0,5556 | 0,5333 | 0,8889 | 0,6667 | 0,6223 |
| MW1 | 0,7000 | 0,6667 | 0,8000 | 0,7273 | 0,6115 |
| PC1 | 0,8182 | 0,8889 | 0,7273 | 0,8000 | 0,4242 |
| PC3 | 0,7500 | 0,8095 | 0,6538 | 0,7234 | 0,4505 |

| Dataset | Accuracy | Precision | Recall | F1 Score | Silhouette Score |
|---|---|---|---|---|---|
| PC4 | 0,5000 | 0,5000 | 0,9714 | 0,6602 | 0,6741 |
| PC5 | 0,6141 | 0,7561 | 0,3370 | 0,4662 | 0,5566 |

### 4.1.4. Model dengan PCA

Skenario empat menggunakan *dataset* awal yang direduksi sebagai dasar *feeder* model untuk skenario ini. Pereduksian bertujuan mengurangi dimensi yang digunakan dengan tetap mempertahankan poin data yang representatif bagi model-model yang digunakan. *Dataset* kondisi awal direduksi dengan PCA dengan variansi 0,95 sehingga didapatkan komponen

*Tabel 4.14 Hasil Reduksi Komponen dengan PCA*

| Dataset | Original Components | Components Reduced to | Reduction Percentage |
|---|---|---|---|
| CM1 | 37 | 11 | 70,3% |
| JM1 | 21 | 8 | 61,9% |
| KC1 | 21 | 7 | 66,7% |
| KC3 | 39 | 10 | 74,4% |
| MC1 | 38 | 14 | 63,2% |
| MC2 | 39 | 11 | 71,8% |
| MW1 | 37 | 11 | 70,3% |
| PC1 | 37 | 12 | 67,6% |
| PC3 | 37 | 12 | 67,6% |
| PC4 | 37 | 14 | 62,2% |
| PC5 | 38 | 14 | 63,2% |

Pada skenario ini, data akan diolah dengan model FCM pada kondisi dasar dengan jumlah *cluster* 2 (*n_cluster*) dan didapatkan hasil sebagai berikut.

*Tabel 4.15 Hasil Uji Coba Model FCM dengan Dataset Reduksi PCA*

| Dataset | Accuracy | Precision | Recall | F1 Score | Silhouette Score |
|---|---|---|---|---|---|
| CM1 | 0,6250 | 1,0000 | 0,2500 | 0,4000 | 0,8121 |
| JM1 | 0,5016 | 0,6667 | 0,0062 | 0,0123 | 0,9566 |
| KC1 | 0,5763 | 0,9091 | 0,1695 | 0,2857 | 0,8347 |
| KC3 | 0,6429 | 1,0000 | 0,2857 | 0,4444 | 0,8443 |
| MC1 | 0,5714 | 0,5385 | 1,0000 | 0,7000 | 0,8700 |
| MC2 | 0,5000 | 0,5000 | 0,8889 | 0,6400 | 0,7916 |
| MW1 | 0,6000 | 0,5556 | 1,0000 | 0,7143 | 0,1302 |
| PC1 | 0,5455 | 1,0000 | 0,0909 | 0,1667 | 0,7850 |
| PC3 | 0,5000 | 0,5000 | 1,0000 | 0,6667 | 0,000 |
| PC4 | 0,5000 | 0,5000 | 0,0286 | 0,0541 | 0,7938 |
| PC5 | 0,5109 | 0,7500 | 0,0326 | 0,0625 | 0,9491 |

Didapatkan beberapa nilai yang lebih baik daripada nilai yang terdapat pada skenario 2, dengan mayoritas nilai *recall* dan *silhouette score* tertinggi berada pada skenario ini dan mayoritas nilai metrik akuarasi, *precision*, serta *f1 score* tertinggi berada pada skenario 2

### 4.1.5. Model *Supervised* dengan PCA

Pada skenario ini, dilakukan reduksi dimensi menggunakan PCA atau Principal Component Analysis dengan menggunakan model *supervised* sebagai model pengklasifikasi. Digunakan ragam CEV yang berada dalam rentang 95% hingga 99% dimana diperoleh sejumlah n_components yang sesuai pada tiap dataset.

*Tabel 4.16 Hasil Model Supervised Terbaik dengan PCA untuk Setiap Dataset*

| Dataset | Jumlah Komponen PCA | Model | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|---|---|
| CM1 | 16 | CatBoost | 0,9649 | 0,9344 | 1,0000 | 0,9661 |
| JM1 | 11 | Random Forest | 0,8097 | 0,7959 | 0,8331 | 0,8141 |
| KC1 | 9 | Random Forest | 0,7989 | 0,7680 | 0,8563 | 0,8098 |
| KC3 | 11 | CatBoost | 0,9844 | 0,9697 | 1,0000 | 0,9846 |
| **MC1** | **19** | **Random Forest** | **0,9948** | **0,9897** | **1,0000** | **0,9948** |
| MC2 | 10 | Random Forest | 0,8125 | 0,7500 | 0,9375 | 0,8333 |
| MW1 | 14 | CatBoost | 0,9333 | 0,8980 | 0,9778 | 0,9362 |
| PC1 | 16 | Random Forest | 0,9600 | 0,9259 | 1,0000 | 0,9615 |
| PC3 | 18 | CatBoost | 0,9108 | 0,8551 | 0,9892 | 0,9173 |
| PC4 | 20 | CatBoost | 0,9452 | 0,9333 | 0,9589 | 0,9459 |

Secaraa umum terlihat bahwa model Random Forest dan CatBoost menjadi model yang paling baik pada *dataset* yang ada. Nilai terbaik model sendiri berada pada Dataset MC1 dimana dengan menggunakan model Random Forest, dihasilkan akurasi paling tinggi diantara kasus *dataset* lain pada angka 0,9948 dengan presisi dan *recall* yang tertinggi yaitu 0,9897 dan 1,000 serta *f1-score* tertinggi yaitu 0,9948 (Ghaffaru, 2025).

## 4.2. Pembahasan

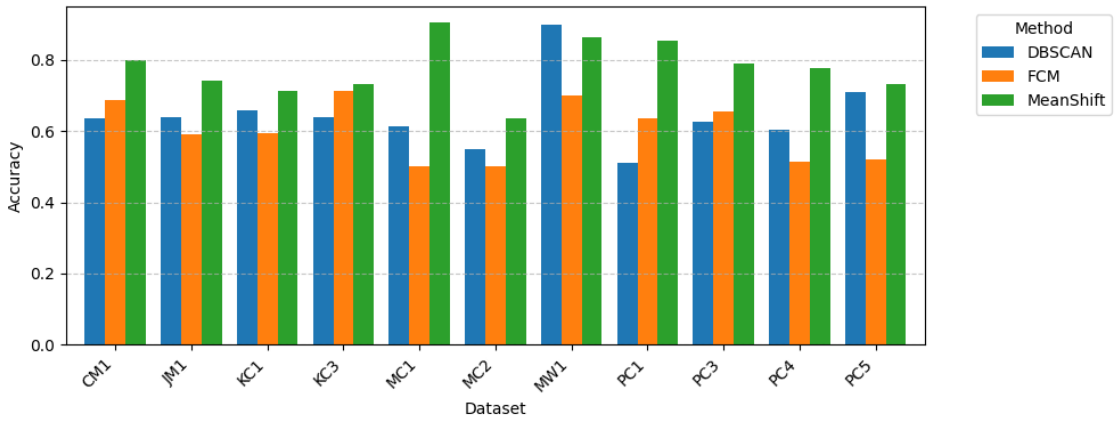
Pada subbab ini akan berisi pembahasan dan analisis secara mendalam dari seluruh hasil uji coba yang telah dilampirkan pada subbab sebelumnya.

### 4.2.1 Pembahasan Uji Coba Skenario 1

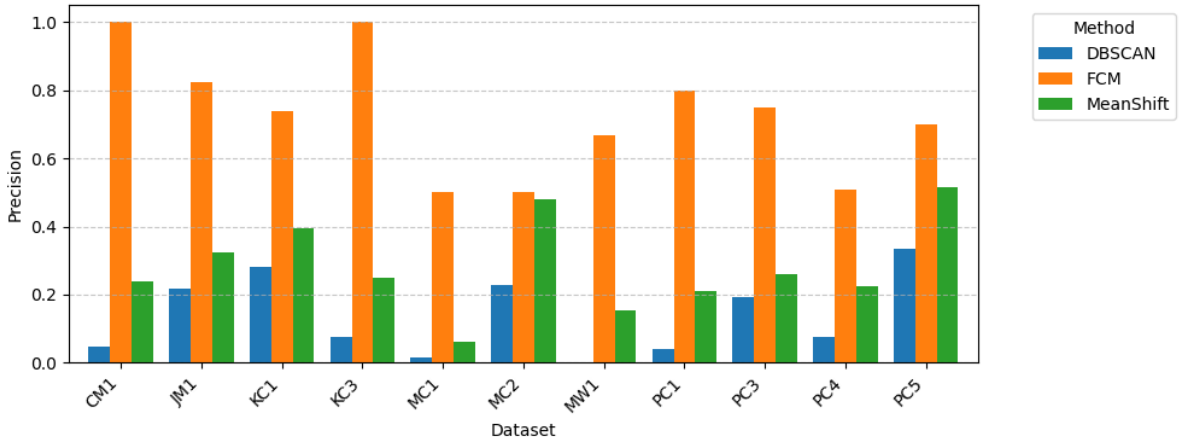
Skenario ini dilakukan dengan menjalankan model Fuzzy C-Means (FCM), DBSCAN, dan Mean-Shift dalam kondisi dasarnya. Pada FCM, data *train* diproses dengan *Synthetic Minority Oversampling Technique* (SMOTE) untuk mengatasi *imbalance* pada *dataset* dan dilakukan

*random undersampling* pada data *test* agar hasil yang didapatkan lebih seimbang. Pada DBSCAN dan Mean-Shift, data diproses langsung oleh model setelah standarisasi karena kedua model tidak melakukan *train* seperti model FCM dan mempelajari pola data untuk membentuk kluster. Data yang diproses oleh kedua model *density based* tersebut tidak di-*oversampling* karena hal ini dapat mempengaruhi hasil *clustering* kedua model dan menyebabkan kluster yang dihasilkan tidak representatif dikarenakan banyak data dari hasil *oversampling* yang akan mempengaruhi pengenalan pola pada model ini sehingga data tetap pada keadaan semula. Setelah itu, model Fuzzy C-Means, DBSCAN, dan Mean-Shift dibangun agar dapat digunakan. Pada Fuzzy C-Means, digunakan parameter n_cluster = 2 untuk membuat jumlah kluster yang dihasilkan menjadi dua sedangkan pada DBSCAN dan Mean-Shift, digunakan keadaan dasar parameter DBSCAN yaitu *eps* = 0,5 dan *min_samples* = 5 serta keadaan parameter Mean-Shift yaitu *bandwidth* = *None*. Setelah dihasilkan kluster-kluster dari model-model di atas, dilakukan *post processing* dengan cara menggabungkan kluster untuk model *density based* yaitu DBSCAN dan Mean-Shift dengan menggunakan metode *distance based.* Selanjutnya, model *density based* melalui *post processing* untuk mendapatkan dua kluster terbaik untuk memenuhi kebutuhan prediksi cacat. Metode *post processing* yang digunakan untuk skenario ini adalah *distance based* dengan mengelompokkan kluster berdasarkan *centroid* kluster yang dibuat oleh model sebelumnya kemudian titik akan di *assign* ke setiap *centroid* terdekat. Setelah itu, setiap kluster yang ada dievaluasi dengan *groundtruth* yang tersedia dari *dataset*. Dua kluster tiap model akan disematkan label N atau Y dalam bentuk angka 0 dan 1 kemudian dibandingkan dengan label pada *groundtruth* yang sebelumnya sudah dirubah menjadi angka 0 untuk N dan 1 untuk Y.

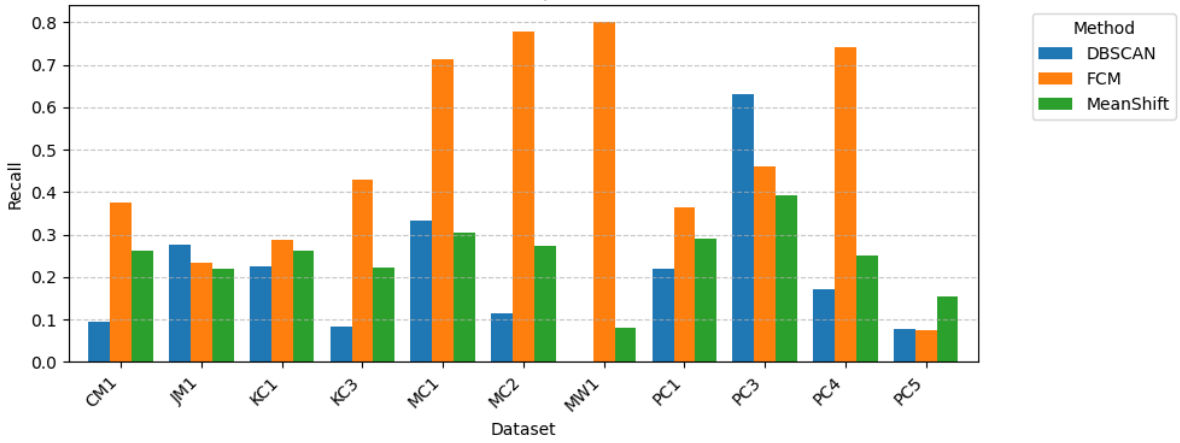
Analisis performa model dalam memprediksi cacat pada modul perangkat lunak menggunakan lima metrik yaitu *accuracy, precision, recall,* dan *f1-score* serta *Silhouette Score*. Nilai akurasi pada penelitian ini membantu untuk menentukan penyematan label dengan prediksi benar terbaik terlebih dahulu dikarenakan kluster yang dihasilkan oleh model *clustering* tidak memiliki makna tertentu sehingga nilai akurasi digunakan sebagai acuan untuk melihat prediksi paling akurat sebelum model dievaluasi dengan metrik lain untuk mengetahui performa model dalam memprediksi cacat. Ketika akurasi prediksi yang dihasilkan oleh model berada dibawah 50%, maka label akan ditukar. Penukaran label dalam penelitian ini berarti label cacat akan ditukar dengan label tidak cacat agar prediksi mencapai akurasi yang lebih baik.



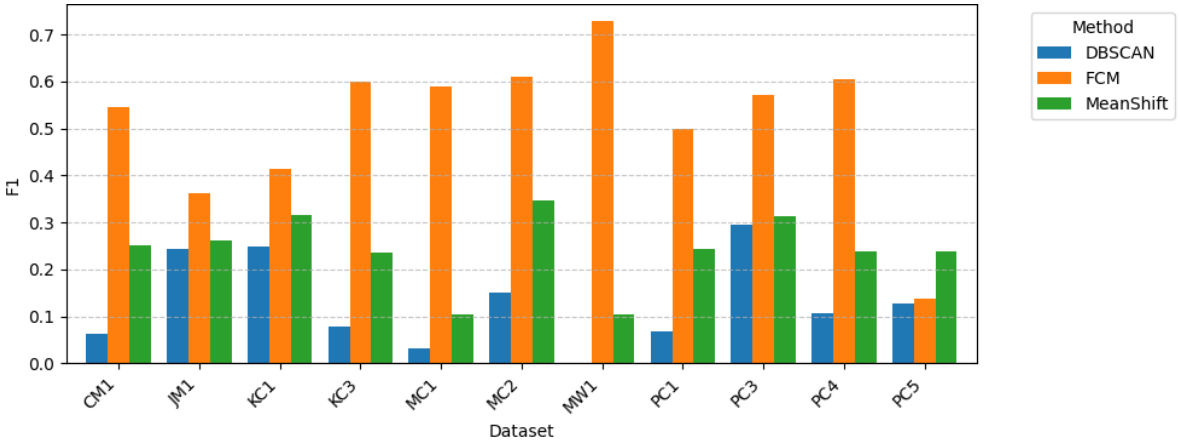

*Gambar 4.2 Grafik Komparasi Metrik Akurasi Setiap Metode pada Berbagai Dataset*



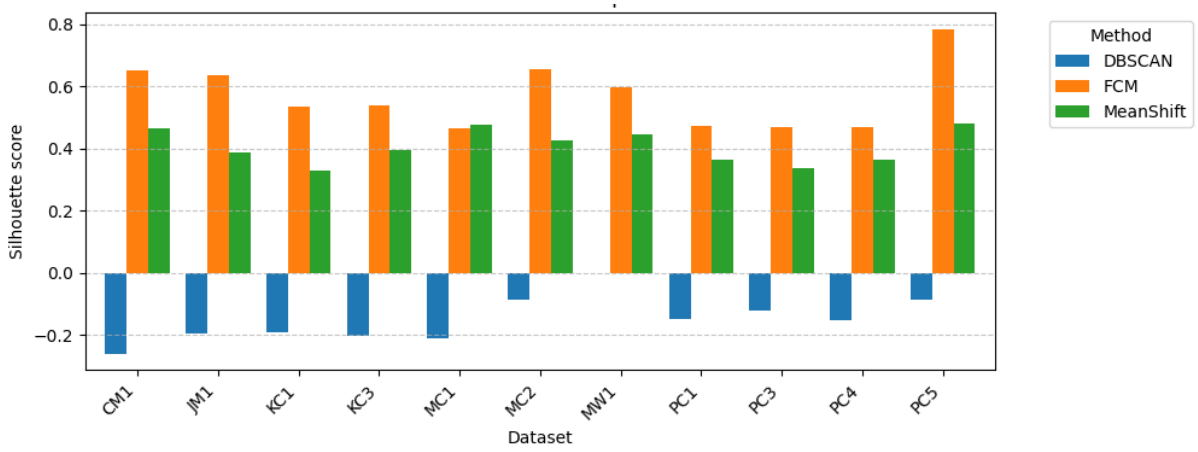

*Gambar 4.3 Grafik Komparasi Metrik Presisi Setiap Metode pada Berbagai Dataset*




*Gambar 4.4 Grafik Komparasi Metrik Recall Setiap Metode pada Berbagai Dataset*




*Gambar 4.5 Grafik Komparasi Metrik F1 Score Setiap Metode pada Berbagai Dataset*




*Gambar 4.6 Grafik Komparasi Metrik Silhouette Score Setiap Metode pada Berbagai Dataset*

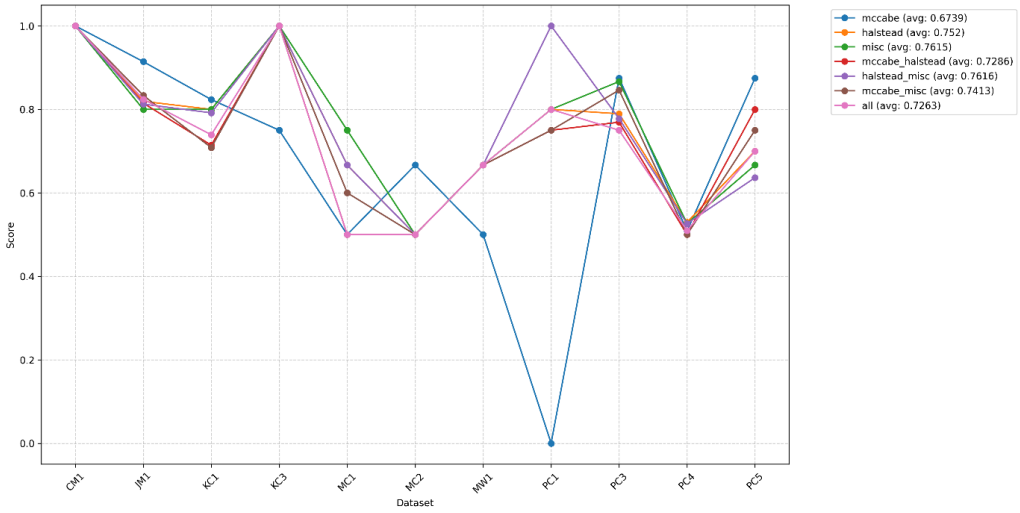
Terlihat pada gambar Gambar 4.2, Gambar 4.3, Gambar 4.4, Gambar 4.5, dan Gambar 4.6 bahwa FCM unggul di seluruh metriks kecuali akurasi. Keunggulan FCM pada metrik-metrik seperti *precision, recall, f1 score,* dan *silhouette score* menunjukkan keandalan FCM dibandingkan model-model lain dalam memprediksi cacat pada penelitian ini. Akurasi tertinggi berada pada model Mean-shift dimana pada model ini berhasil melakukan prediksi yang tepat namun berdasarkan sebaran data yang ada pada tiap *dataset* yang mana pada kasus ini, data memiliki sebaran yang mengikuti prinsip kepadatan model ini sehingga model ini dapat mengelompokkan poin data untuk diproses dengan metode *distance based* untuk mendapatkan dua kluster terbaik dan menyematkan label hasil *post procesing* pada tiap poin data.

Pengelompokkan model DBSCAN dan Mean-Shift yang berdasar pada kepadatan, tidak dapat memberikan jaminan pengelompokkan yang sesuai. Karakteristik modul cacat maupun tidak cacat tidak sejalan dengan similaritas serta densitas modul-modul perangkat lunak sehingga poin data yang dikelompokan dengan metode ini menghasilkan kluster-kluster yang memiliki karakteristik sama tetapi tidak selalu dapat memberikan jawaban dari pertanyaan apakah ada cacat didalamnya.

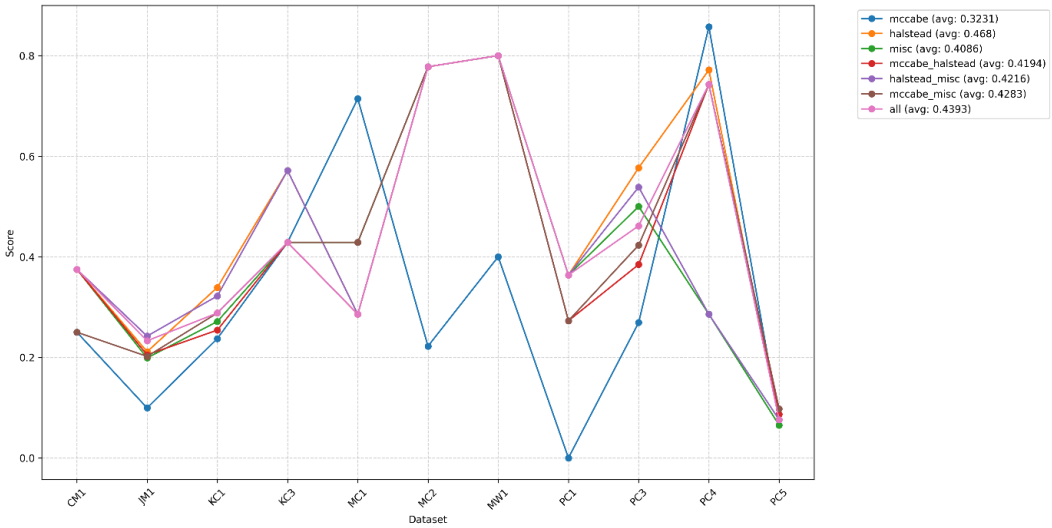
### 4.2.2 Pembahasan Uji Coba Skenario 2

Skenario ini memiliki alur yang sama dengan skenario satu, namun data yang dikonsumsi oleh model adalah data yang telah melewati seleksi fitur dengan menggunakan kelompok fitur pada Tabel 4.5. Selain itu, dilakukan empat macam metode *post processing* dengan menggabungkan kluster pada hasil kluster model *density based* seperti dijelaskan pada bagian 3.5. Terdapat tiga kelompok fitur utama serta tujuh kombinaasi kelompok fitur seperti terlampir pada subbab 4.1.2. Pengelompokan kelompok fitur utama didasarkan pada karakteristik dari tiap fitur. CYCLOMATIC_COMPLEXITY, ESSENTIAL_COMPLEXITY, dan DESIGN_COMPLEXITY merupakan metrik yang termasuk dalam metrik-metrik yang diusung oleh McCabe pada tahun 1976. BRANCH_COUNT merupakan jumlah cabang dalam sebuah modul perangkat lunak, lebih spesifiknya adalah semua *edge* yang keluar dari sebuah *decision node* sehingga sesuai dengan prinsip-prinsip McCabe yang memperhatikan *control flow* dari sebuah program (McCabe, 1976), salah satunya melalui cabang keputusan seperti BRANCH_COUNT ini. Metrik Halstead pada *dataset* terdiri dari NUM_OPERATORS,

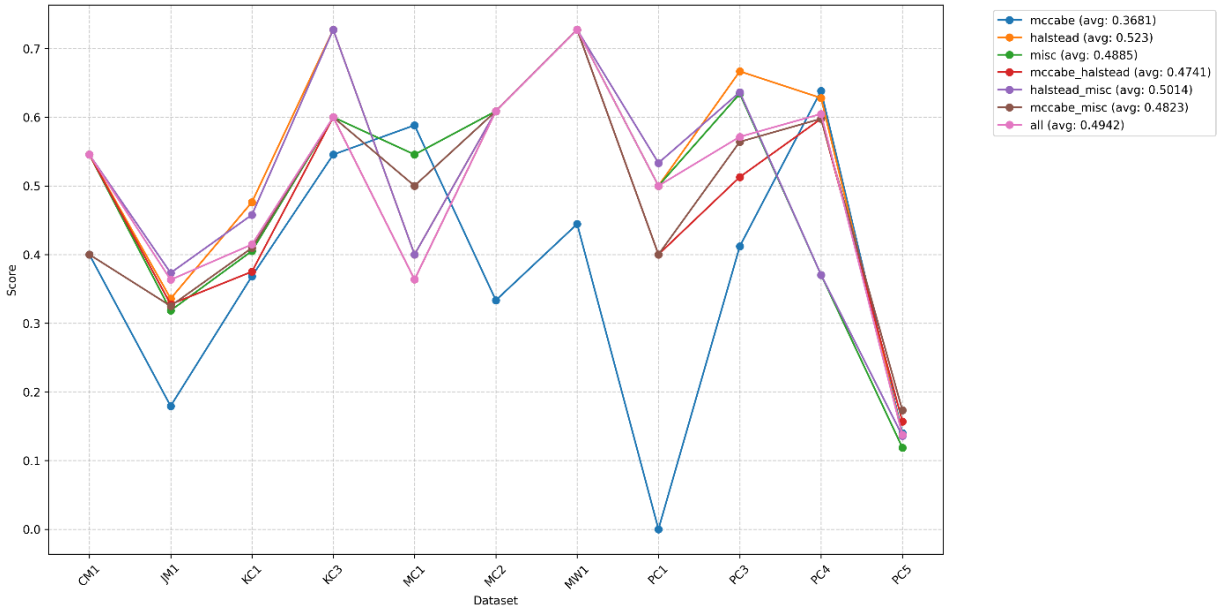
NUM_OPERANDS, NUM_UNIQUE_OPERANDS, NUM_UNIQUE_OPERATORS, HALSTEAD_LENGTH, HALSTEAD_VOLUME, HALSTEAD_LEVEL, HALSTEAD_DIFFICULTY, HALSTEAD_EFFORT, HALSTEAD_PROG_TIME, HALSTEAD_CONTENT, dan HALSTEAD_ERROR_EST. Setiap metrik pada kelompok fitur Halstead merupakan metrik yang mengusahakan pengukuran terhadap atribut yang dimiliki sebuah algoritma melalui penghitungan langsung terhadap operan dan operator maupun perhitungan kuantitatif parameter lain pada Halstead menggunakan nilai metrik penghitungan langsung (Halstead, 1977). Metrik *miscellanoues* pada penelitian ini merupakan berbagai metrik yang menyertai *dataset* sebagai pelengkap informasi sebuah modul perangkat lunak seperti LOC_CODE_AND_COMMENT, LOC_EXECUTABLE, LOC_COMMENTS, LOC_BLANK, dan LOC_TOTAL. Dari kelompok fitur utama tersebut, dilakukan kombinasi kelompok fitur seperti terlihat pada gambar 4.7. Hasil dari berbagai seleksi fitur pada model terbaik yaitu FCM terlihat pada grafik dibawah ini,




*Gambar 4.7 Grafik Perbandingan Performa Kelompok Fitur dengan Model FCM pada Metrik Presisi*



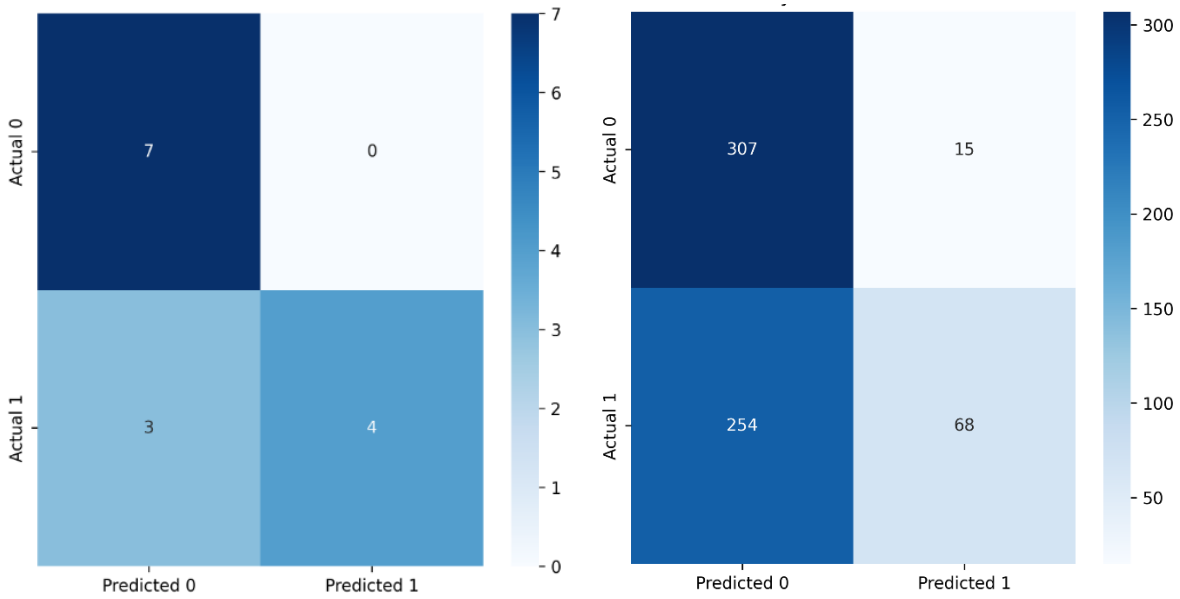

*Gambar 4.8 Grafik Perbandingan Performa Kelompok Fitur dengan Model FCM pada Metrik Recall*




*Gambar 4.9 Grafik Perbandingan Performa Kelompok Fitur dengan Model FCM pada Metrik F1 Score*

Terdapat 7 kelompok fitur dari proses seleksi fitur yang dilakukan. Model terbaik yang didapatkan sesuai hasil yang terdapat pada subab 4.1.2 adalah pada FCM dengan kelompok fitur Halstead. Terlihat pada grafik garis di atas bahwa Halstead cenderung konstan dibandingkan dengan kelompok fitur lain yang memiliki nilai yang tinggi sekali pada suatu *dataset* maupun rendah sekali pada *dataset* yang lain seperti pada kelompok McCabe dan All. Terlihat pula pada gambar 4.7, 4.8, dan 4.9 di atas bahwa model FCM memiliki nilai metrik yang konstan dalam memprediksi cacat pada data dengan nilai metrik Halstead Kelompok fitur Halstead sendiri berisi metrik-metrik dari Halstead yang diberikan oleh NASA MDP *dataset*. Metrik dari McCabe dan *Miscellaneous* (misc) tidak seefektif dalam mendeteksi cacat perangkat lunak dengan algoritma Fuzzy C-Means karena beberapa alasan. Metrik McCabe, seperti *Cyclomatic Complexity*, *Essential Complexity*, dan *Branch Count*, hanya berfokus pada kompleksitas alur kontrol, tanpa mempertimbangkan operasi komputasi atau estimasi kesalahan seperti pada metrik Halstead. Hal ini menyebabkan kesalahan ketika mendeteksi cacat pada modul berkompleksitas struktural tinggi yang mana belum tentu lebih rentan cacat jika logika internalnya sederhana.

Sementara itu, Misc, seperti LOC_TOTAL dan LOC_EXECUTABLE, terlalu umum karena hanya mengukur ukuran fisik kode, bukan kompleksitas atau kualitas modul perangkat lunak itu sendiri. Kode yang panjang tidak selalu mengindikasikan kerentanan cacat jika dirancang dengan baik, sedangkan kode pendek dapat sangat rentah terhadap kesalahan jika logikanya rumit. Melihat hal ini, fitur Misc dirasa kurang bermakna dalam membedakan modul cacat dan non cacat, sehingga ketika dibandingkan dengan Halstead yang menggabungkan metrik seperti NUM_OPERATORS dan HALSTEAD DIFFICULTY untuk menangkap karakteristik kode yang lebih relevan dengan kemungkinan cacat, performa metrik Misc berada dibawah Halstead.




(a) (b)

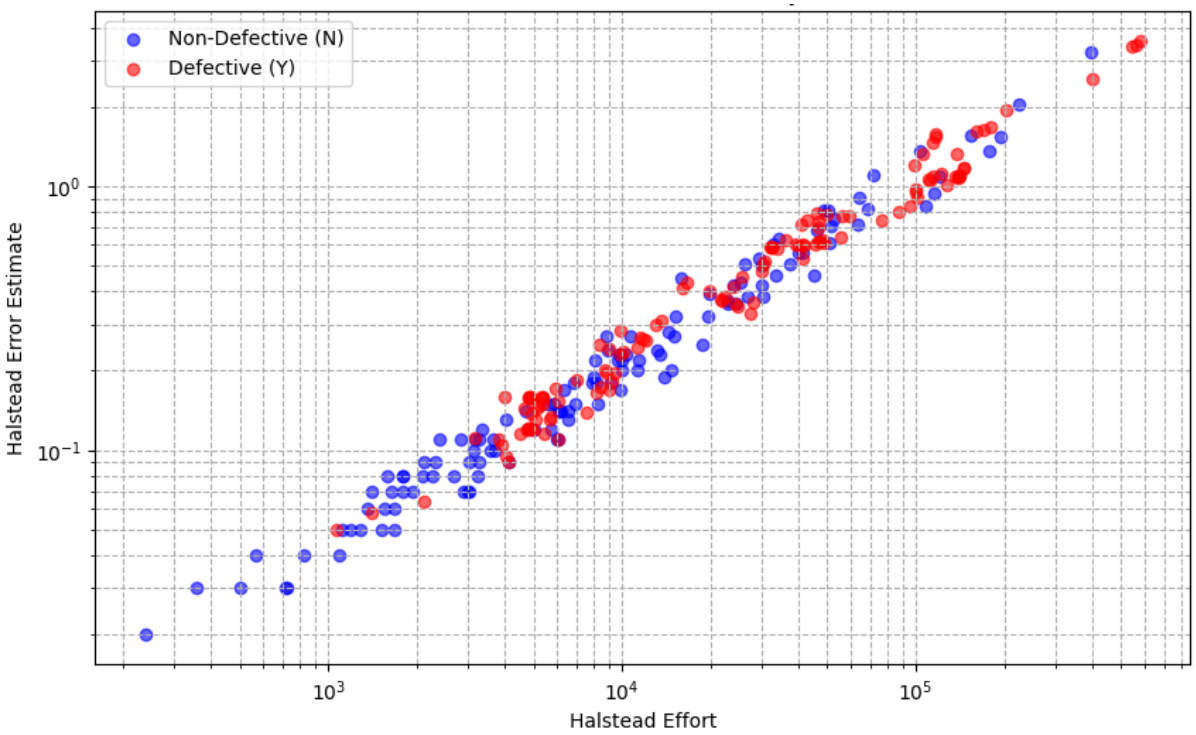
*Gambar 4.10 Confussion Matrix FCM untuk kelompok fitur Halstead. (a) Dataset KC3. (b) Dataset JM1*

Pada *confussion matrix dataset* KC3, kapabilitas model FCM dengan kelompok fitur Halstead dalam meminimalisir *False Negative* dan keandalannya dalam memprediksi kelas tidak cacat pada *dataset* tersebut. Namun, di sisi lain, terlihat pada *confussion matrix* JM1 bahwa model memiliki *False Negative* yang lebih tinggi dibandingkan dengan *True Positive*. Hal ini dapat disebabkan oleh sebaran data yang kurang variatif dan representatif terhadap berbagai kasus cacat yang terjadi pada *dataset* tersebut dalam kelompok fitur Halstead.

*Tabel 4.17 Contoh Studi Kasus pada Dataset KC3 Berdasarkan Confussion Matrix FCM Kelompok Fitur Halstead*

| Feature | True Positive | | | | False Negative | | |
|---|---|---|---|---|---|---|---|
| | Data 1 | Data 2 | Data 3 | Data 4 | Data 5 | Data 6 | Data 7 |
| HALSTEAD_CONTENT | 115,54 | 125,82 | 139,4 | 102,5 | 28,48 | 32,17 | 29,1 |
| HALSTEAD_DIFFICULTY | 19,78 | 44,77 | 16,46 | 38,09 | 9,5 | 13,68 | 10,5 |
| HALSTEAD_EFFORT | 45226,86 | 252155,78 | 37755,59 | 148686,82 | 2569,87 | 6017,65 | 3208,05 |
| HALSTEAD_ERROR_EST | 0,76 | 1,88 | 0,76 | 1,3 | 0,09 | 0,15 | 0,1 |
| HALSTEAD_LENGHT | 367 | 829 | 357 | 597 | 59 | 88 | 65 |
| HALSTEAD_LEVEL | 0,05 | 0,02 | 0,06 | 0,03 | 0,11 | 0,07 | 0,1 |

| Feature | True Positive | | | | False Negative | | |
|---|---|---|---|---|---|---|---|
| | Data 1 | Data 2 | Data 3 | Data 4 | Data 5 | Data 6 | Data 7 |
| HALSTEAD_PROG_TIME | 2512,6 | 14008,65 | 2097,53 | 8260,37 | 142,77 | 334,31 | 178,23 |
| HALSTEAD_VOLUME | 2285,98 | 5632,57 | 2294,18 | 3903,88 | 270,51 | 440 | 305,53 |
| NUM_OPERANDS | 135 | 308 | 144 | 219 | 19 | 31 | 21 |
| NUM_OPERATORS | 232 | 521 | 213 | 378 | 40 | 57 | 44 |
| NUM_UNIQUE_OPERANDS | 58 | 86 | 70 | 69 | 12 | 17 | 13 |
| NUM_UNIQUE_OPERATORS | 17 | 25 | 16 | 24 | 12 | 15 | 13 |
| GROUNDTRUTH_LABEL | Y | Y | Y | Y | Y | Y | Y |
| PREDICTIVE_LABEL | Y | Y | Y | Y | N | N | N |




*Gambar 4.11 Scatter Plot Halstead Effort Vs Halstead Error Estimate pada Data Train Dataset KC3*

Tabel 4.17 berisi data studi kasus pada *dataset* KC3 yang poin datanya diklasifikasikan menjadi *True Positive* dan *False Negative* untuk mengetahui karakteristik poin data dengan *groundtruth* positif (cacat). Kolom pada tabel terkait menunjukkan tujuh data yang memiliki label *groundtruth* "Y" atau positif cacat dan kemudian dibagi menjadi dua bagian yaitu data dengan kelas *True Positive* dan *False Negative*. Di sisi lain, kolom pertama sebuah baris data berisi nama kolom fitur yang dimiliki setiap poin data sehingga setiap baris data berisi nilai-nilai sebuah fitur pada tiap data. Terlihat nilai-nilai yang dimiliki oleh poin data dengan kelas *False Negative* mempunyai sebaran nilai yang berbeda dengan poin data dengan kelas *True Positive*. Seperti pada HALSTEAD_EFFORT, nilai fitur yang dimiliki poin data pada kelas *True Positive* berada pada lingkup puluhan hingga ratusan ribu, namun pada poin data kelas *False Negative* memiliki nilai pada rentang ribuan saja. Tidak hanya pada fitur tersebut, tetapi kesenjangan nilai terjadi pada hampir seluruh fitur.

Hal ini dapat disebabkan oleh data latih berlabel cacat pada model FCM dengan fitur Halstead yang tidak representatif terhadap berbagai kemungkinan cacat modul perangkat lunak KC3. Hasil terkait mengindikasikan variansi modul cacat yang tinggi sehingga dibutuhkan *dataset* yang dapat mengimbangi variansi tinggi tersebut serta data yang melingkupi berbagai kemungkinan cacat dalam konteks fitur Halstead. Pada *scatter plot* Gambar 4.11, terlihat *data train* pada *dataset* KC3 memiliki persebaran data cacat yang terkumpul pada suatu bagian dengan persebaran data yang padat untuk HALSTEAD_EFFORT diatas pertengahan 1000 sampai 10000 dan HALSTEAD_ERROR_ESTIMATE pada nilai diatas 0,1 ke atas. Meskipun terdapat data di rentang 1000 sampai 5000 pada HALSTEAD_EFFORT dan data pada rentang di bawah 0,1 pada HALSTEAD_ERROR_ESTIMATE, jumlah data yang cacat pada rentang tersebut tidak sebanyak jumlah data yang tidak cacat. Pada model lain yang hasil evaluasinya berada dibawah FCM yaitu DBSCAN dan Mean-Shift, hal ini juga dapat disebabkan oleh *dataset* yang kurang bervariasi dalam melingkupi semua kemungkinan yang dapat terjadi dalam sebuah kasus cacat pada modul perangkat lunak. Selain itu, data yang tidak seimbang dan tidak terjadinya pembangkitan data karena alasan potensi pembiasan pola data, membuat model ini tidak dapat bekerja secara maksimal.

```
1. int find_max(int a, int b) {
2.     if (a > b) {
3.         return a;
4.     } else {
5.         return b;
6.     }
7. }
```

*Gambar 4.12 Contoh Kode Tidak Cacat dalam Pencarian Nilai Maksimum dari Dua Parameter Masukan*

```
1. int find_max(int a, int b) {
2.     if (a < b) {
3.         return a;
4.     } else {
5.         return b;
6.     }
7. }
```

*Gambar 4.13 Contoh Kode Cacat dalam Pencarian Nilai Maksimum dari Dua Parameter Masukan*

Pada **Error! Reference source not found.** dan Gambar 4.13 terdapat kode sebuah program yang menerima dua parameter masukan. Kedua kode ini merepresentasikan sebuah program dengan keadaan berbeda. Kedua kode tersebut memiliki perbedaan pada baris kedua dimana terdapat perbedaan tanda pembanding lebih besar dan lebih kecil. Pada Gambar 4.12, kode

berjalan dengan baik dan berhasil menunjukkan nilai maksimum dari dua parameter masukan, sedangkan pada Gambar 4.13, kode berjalan namun tidak menghasilkan luaran nilai maksimum. Dari kedua contoh kode tersebut, didapatkan metrik kompleksitas Halstead pada kode tersebut dengan jumlah sebagai berikut,

*Tabel 4.18 Jumlah Metrik Kompleksitas Kode Tidak Cacat*

| Metric Feature | Jumlah |
|---|---|
| HALSTEAD_CONTENT | 7,59 |
| HALSTEAD_DIFFICULTY | 7,0 |
| HALSTEAD_EFFORT | 372,05 |
| HALSTEAD_ERROR_EST | 0,0177 |
| HALSTEAD_LENGHT | 16 |
| HALSTEAD_LEVEL | 0,143 |
| HALSTEAD_PROG_TIME | 20,67 |
| HALSTEAD_VOLUME | 53,15 |
| NUM_OPERANDS | 6 |
| NUM_OPERATORS | 10 |
| NUM_UNIQUE_OPERANDS | 3 |
| NUM_UNIQUE_OPERATORS | 7 |

*Tabel 4.19 Jumlah Metrik Kompleksitas Kode Cacat*

| Metric Feature | Jumlah |
|---|---|
| HALSTEAD_CONTENT | 7,59 |
| HALSTEAD_DIFFICULTY | 7,0 |
| HALSTEAD_EFFORT | 372,05 |
| HALSTEAD_ERROR_EST | 0,0177 |
| HALSTEAD_LENGHT | 16 |
| HALSTEAD_LEVEL | 0,143 |
| HALSTEAD_PROG_TIME | 20,67 |
| HALSTEAD_VOLUME | 53,15 |
| NUM_OPERANDS | 6 |
| NUM_OPERATORS | 10 |
| NUM_UNIQUE_OPERANDS | 3 |
| NUM_UNIQUE_OPERATORS | 7 |

Terlihat pada contoh kasus ini bahwa kedua kode dengan jumlah metrik kompleksitas Halstead yang sama (disebutkan pada Tabel 4.18 dan Tabel 4.19), dapat memiliki status kecacatan yang berbeda. Pada **Error! Reference source not found.**, ditampilkan kode tanpa kondisi cacat untuk mencari nilai maksimal dari dua masukan yang diberikan. Di sisi lain, **Error! Reference source not found.** menunjukkan kode dengan kondisi cacat logika ketika kode tersebut

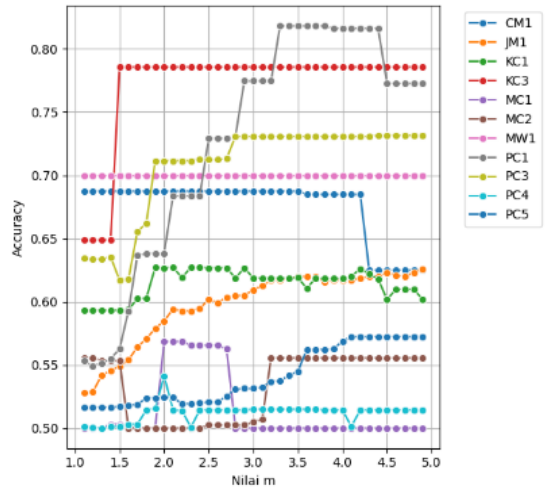
seharusnya mencari nilai maksimal antara dua masukan yang diberikan. Hal ini menunjukkan kemungkinan yang dapat terjadi dalam penggunaan metrik kompleksitas perangkat lunak Halstead dimana sebuah kode atau modul perangkat lunak dengan jumlah atau *value* metrik yang sama, dapat memiliki kondisi yang berbeda.

### 4.2.3 Pembahasan Uji Coba Skenario 3

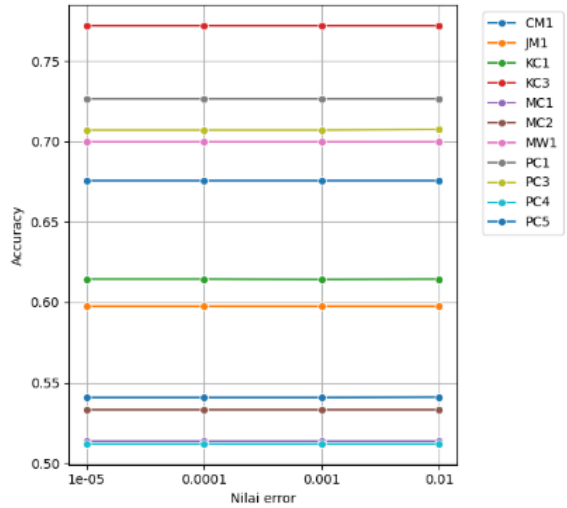
Pemilihan model dan kelompok fitur terbaik yang akan di *hyperparameter tune* berdasarkan metrik evaluasi dengan komposisi 0,3 untuk *precision*, 0,4 untuk *recall,* dan 0,3 untuk *f1 score*. Pembobotan ini dibebankan lebih pada *recall* untuk memberikan dampak lebih kritis pada *false negative* dibandingkan *false positive*. Bobot ini menekankan *recall* sebagai metrik utama yang mengindikasikan bahwa kemampuan model untuk mendeteksi semua kasus positif yang terjadi. Pada prediksi cacat pada perangkat lunak, keandalan ini lebih diutamakan daripada *false positive* (*precision*). Sementara itu, *f1 score* diberikan bobot yang setara dengan *precision* dan mempunya peran sebagai penyeimbang dan menjaga generalisasi model. Hal ini dapat terjadi karena harmonik *mean* pada *f1 score* yang menggabungkan *precision* dan *recall* secara proporsional (Ho, 2009). Pendekatan serupa dalam menjadikan *recall* yang diberi bobot lebih tinggi pada studi klasifikasi data tidak seimbang juga dilakukan untuk memitigasi bias terhadap kelas mayoritas (Provost, 2013). Pemanfaatan pembobotan seperti demikian disesuaikan dengan tujuan aplikasi, dalam konteks penelitian ini adalah untuk melakukan prediksi cacat terhadap modul-modul perangkat lunak dimana melewatkan kasus positif (*false negative*) berpotensi menimbulkan dampak yang lebih besar (Chandola, 2009).

Pada bagian ini dilakukan *hyperparameter tuning* terhadap model terbaik dengan menggunakan kelompok fitur terbaik. Pemilihan fitur terbaik didasarkan pada nilai *presisi*, *recall*, dan *f1 score* tiap kelompok fitur. Pemilihan metrik performa didasarkan pada kebutuhan prediksi cacat penilitian yang membutuhkan prediksi cacat (positif) yang tinggi. Metrik *precision* menyediakan kuantifikasi terhadap hasil prediksi positif cacat modul perangkat lunak dan metrik *recall* menyediakan kuantifikasi terhadap hasil prediksi yang diberikan kepada modul cacat nyata sehingga dapat menunjukkan keandalan model dalam memprediksi cacat pada modul perangkat lunak terkait. Selain *precision* dan *recall*, penggunaan *f1 score* dapat memberikan nilai dan gambaran harmonik antara metrik *precision* dan *recall* sehingga mendapatkan pandangan baru terhadap tiap kasus pada tiap model. Parameter yang disesuaikan agar mendapatkan performa yang optimal adalah m, max_iter, dan error. Parameter error diatur agar hasil pengelompokkan memiliki toleransi konvergensi yang presisi. Semakin kecil nilai error, semakin baik hasil *clustering* karena pengelompokkannya akan semakin akurat. Pada parameter m (*fuzziness exponent*), ditentukan seberapa fleksibel atau *fuzzy* batas antarcluster dalam pengelompokkan data. Fleksibilitas ini akan mempengaruhi derajat keanggotaan suatu poin data dalam sebuah kluster tertentu. Semakin besar nilai m, semakin fleksibel batas antarkluster itu sendiri dan makin cocok untuk data yang memiliki *overlap* tinggi. Pada lima tabel di bawah, terlihat pengaruh nilai parameter terhadap metrik-metrik performa yaitu *accuracy, precision, recall, f1-score,* dan *silhouette score* per *dataset*. Pada parameter *max_iter*, diatur nilai-nilai yang membatasi berapa kali algoritma ini dijalankan atau diiterasi sebelum akhirnya dihentikan. Pada parameter *error*, dihitung nilai konvergensi yang ditoleransi dalam nilai derajat keanggotaan. Setiap parameter dipadukan untuk mendapatkan model yang terbaik. Seperti pada hasil yang tercantum pada subbab 4.1.3, didapatkan nilai *m* yang bervariasi dimana

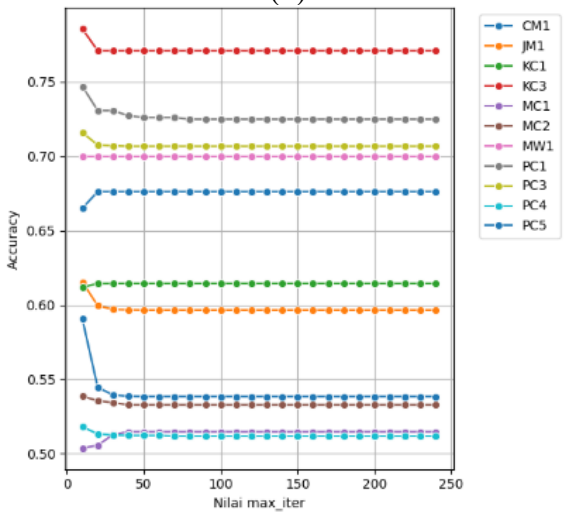
menunjukkan seberapa fleksibel nilai derajat keanggotaan suatu poin data dengan 1,1 fleksibilitas terendah dan 4,4 sebagai fleksibilitas tertinggi.




(a)

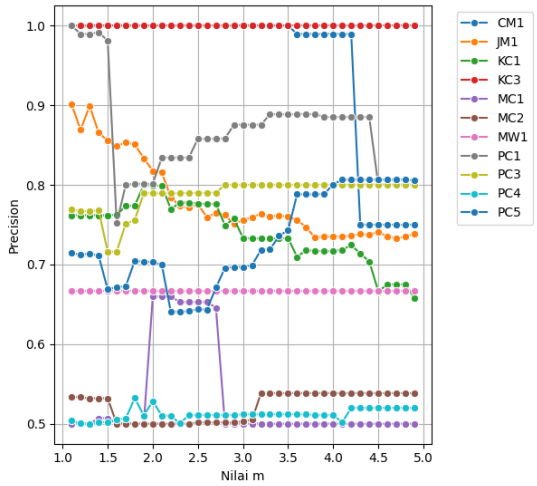



(b)

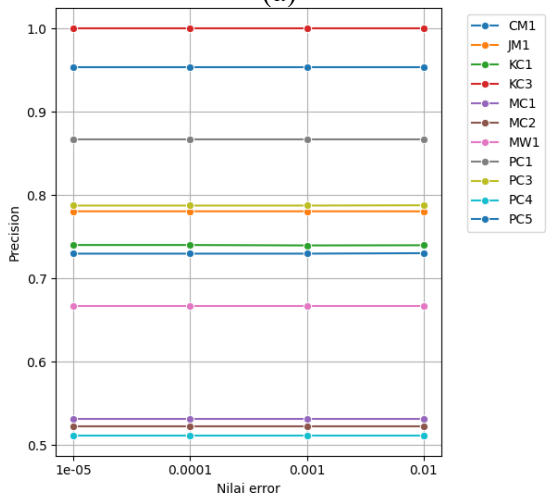



(c)

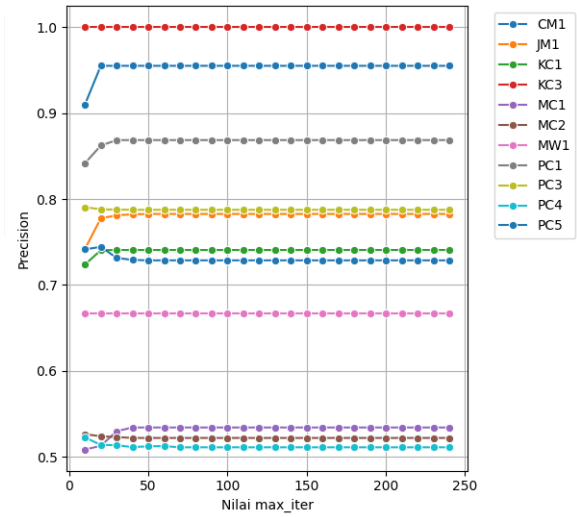
*Gambar 4.14 Grafik Pengaruh Parameter terhadap Accuracy per Dataset. (a) Parameter m (fuzziness exponent). (b) Parameter error (convergence tolerance). (c) Parameter max_iter*




(a)

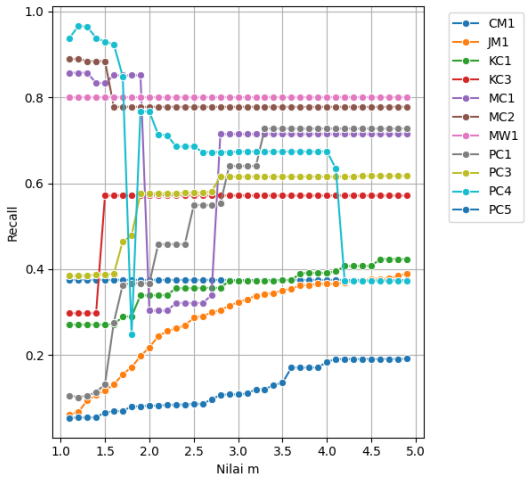



(b)

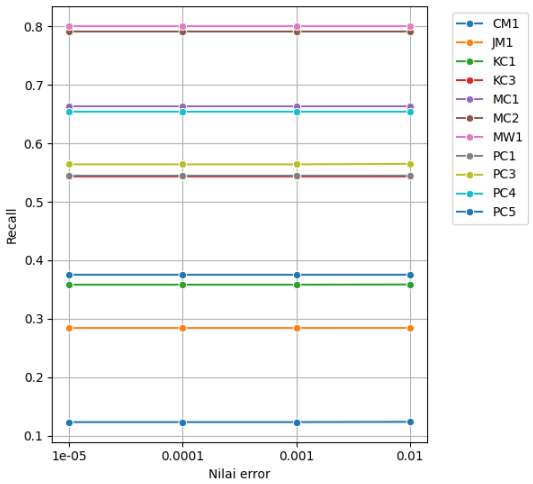



(c)

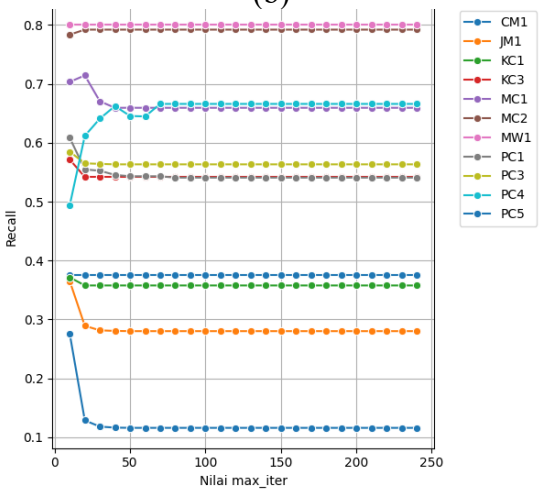
*Gambar 4.15 Grafik Pengaruh Parameter terhadap Precision per Dataset.(a) Parameter m (fuzziness exponent). (b) Parameter error (convergence tolerance). (c) Parameter max_iter*




(a)

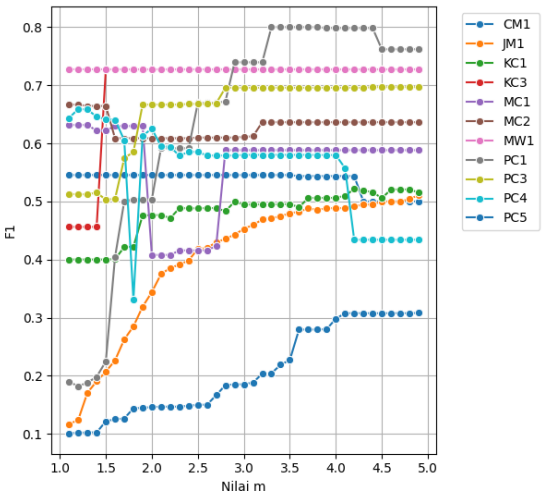



(b)

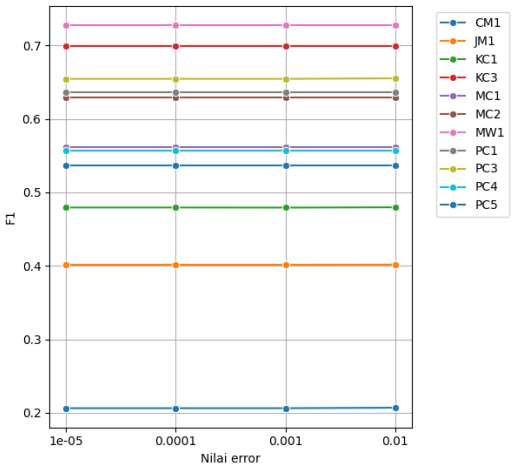



(c)

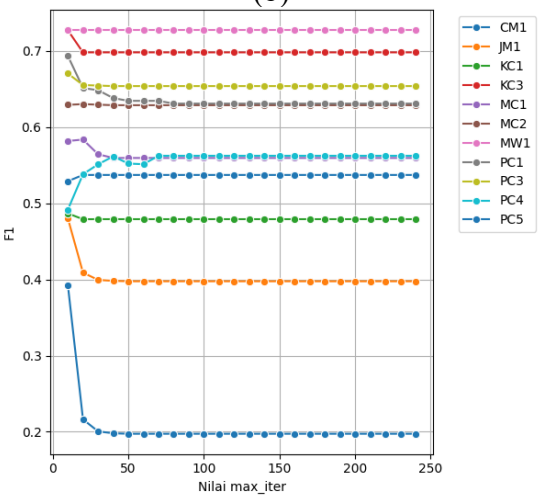
*Gambar 4.16 Grafik Pengaruh Parameter terhadap Recall per Dataset.(a) Parameter m (fuzziness exponent). (b) Parameter error (convergence tolerance). (c) Parameter max_iter*




(a)

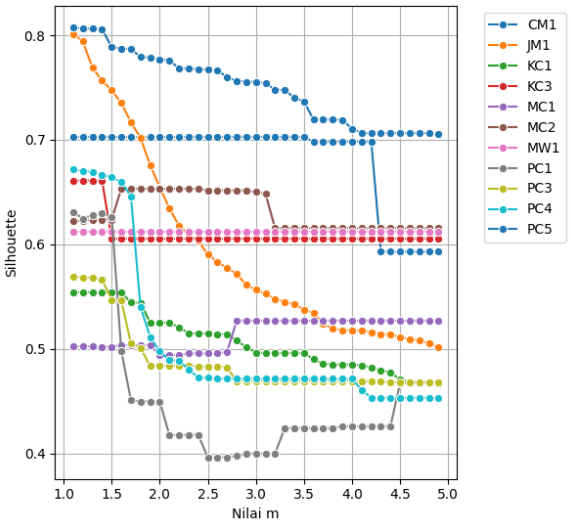



(b)

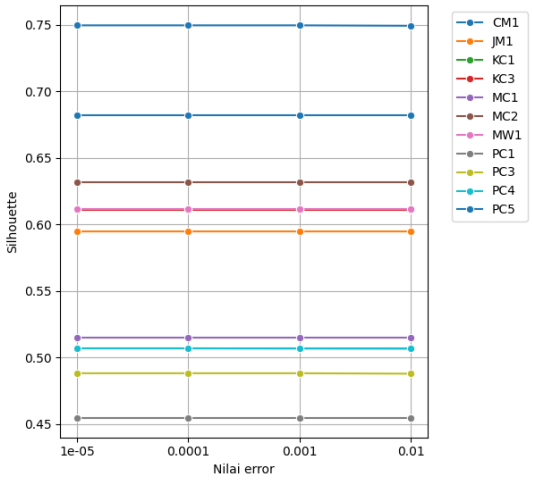



(c)

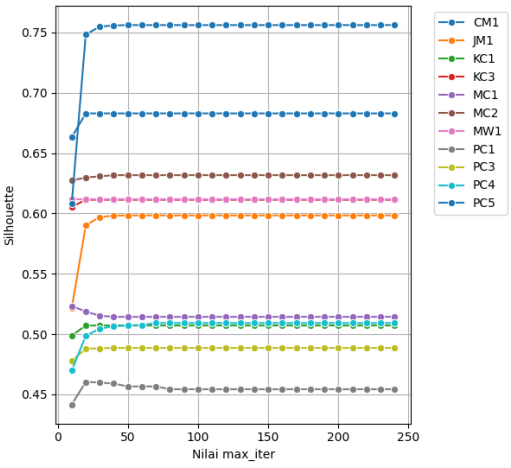
*Gambar 4.17 Grafik Pengaruh Parameter terhadap F1 Score per Dataset.(a) Parameter m (fuzziness exponent). (b) Parameter error (convergence tolerance). (c) Parameter max_iter*




(a)




(b)




(c)

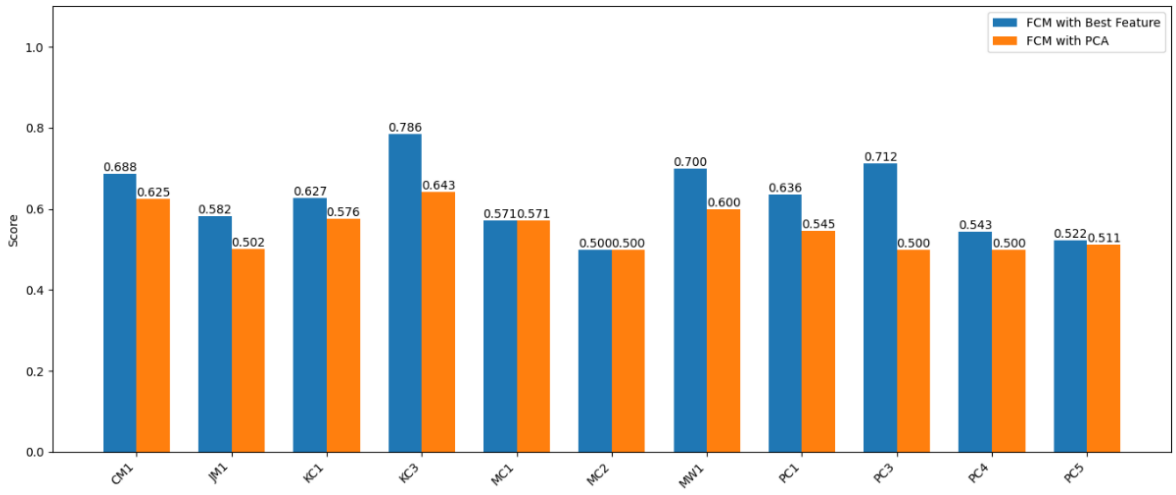
*Gambar 4.18 Grafik Pengaruh Parameter terhadap Silhouette Score per Dataset. (a) Parameter m (fuzziness exponent). (b) Parameter error (convergence tolerance). (c) Parameter max_iter*

Melalui gambar Gambar 4.14 Gambar 4.15 Gambar 4.16 Gambar 4.17, dan Gambar 4.18, terlihat pengaruh pemilihan parameter *m* atau *fuzziness exponent* yang mempengaruhi nilai

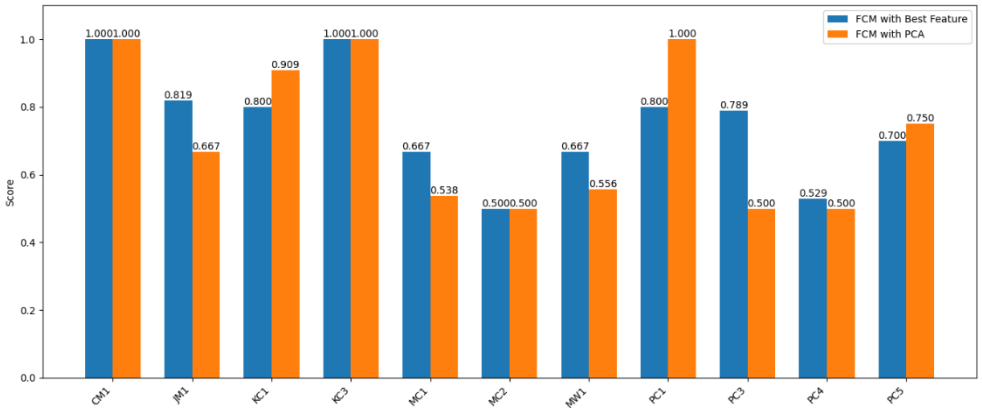
metrik evaluasi. Selain itu, terlihat parameter *max_iter* yang menunjukkan perubahan paling signifikan terhadap metrik evaluasi pada nilai 10 sampai 20, Parameter *error* sendiri tidak memberikan pengaruh yang signifikan terhadap metrik evaluasi *accuracy, precision, recall, f1 score,* dan *silhouette score*. Pada sebuah modul perangkat lunak, fitur-fitur *dataset* yang diberikan berpotensi mengalami *overlap* antara satu sama lain sehingga penetapan parameter seperti *m* atau *fuzziness exponent* yang tepat, memungkinkan penanganan kompleksitas dalam *dataset* seperti NASA MDP yang batas antar poin datanya rawan *overlap*. Pada parameter *max_iter*, dilakukan pembatasan jumlah maksimum iterasi agar mencapai konvergensi. Penggunaan *max_iter* akan menghindarkan model dari komputasi yang tidak perlu dilakukan sehingga proses akan terhenti meskipun kriteria error belum tercapai. Hal ini membantu komputasi model Fuzzy C-Means lebih efisien dan terlihat dengan cakupan pada nilai 10-20 memberikan nilai metrik evaluasi terbaik.

### 4.2.4 Pembahasan Uji Coba Skenario 4

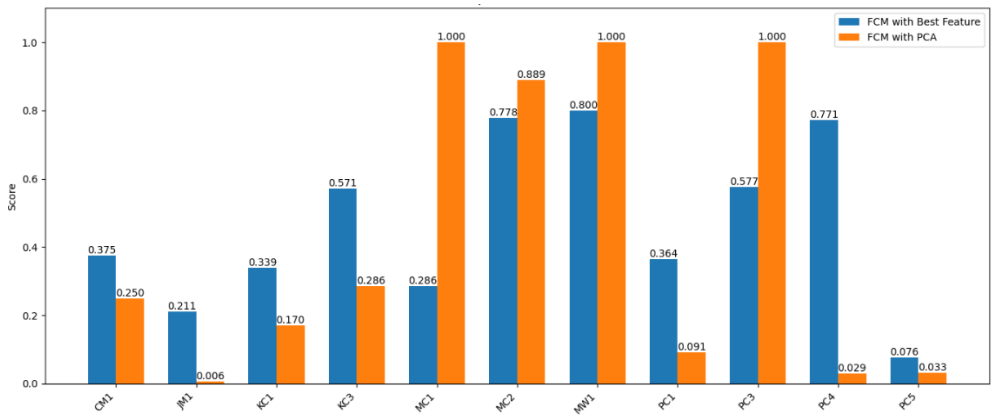
Pada skenario keempat, dilakukan pereduksian dimensi menggunakan *Principal Component Analysis* (PCA) guna menyederhanakan struktur dan komputasi model nantinya sebelum diterapkan pada model *clustering* yaitu FCM. PCA bertujuan mengurangi kompleksitas komputasi dengan mempertahankan variansi sesuai dengan *threshold* yang dipakai, pada penelitian ini sebesar 95%. Hasil reduksi menunjukkan penurunan jumlah fitur yang signifikan diatas 60%. Salah satu *dataset* mengalami pengurangan fitur dari 39 menjadi 10 fitur (tereduksi 74.4%). Reduksi yang besar pada *dataset* NASA MDP menunjukkan banyaknya fitur berkorelasi tinggi dan dapat direpresentasikan dalam dimensi yang lebih rendah tanpa kehilangan informasi penting. Sebelum dimasukan kedalam model, dilakukan *oversampling* terhadap data *train* serta *random undersampling* terhadap data *test* untuk mengatasi *imbalance* atau ketidakseimbangan dalam *dataset*.



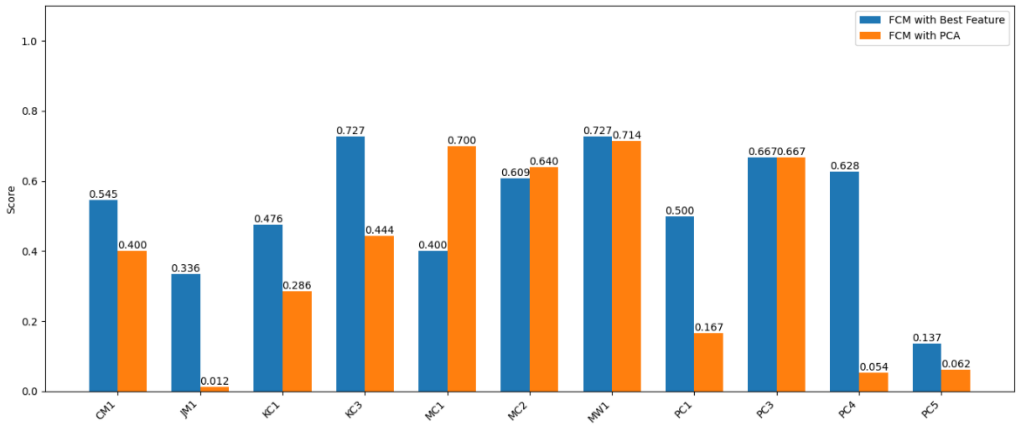

*Gambar 4.19 Grafik Perbandingan Model FCM dengan Kelompok Fitur Terbaik dan Model FCM dengan Dataset Tereduksi dari PCA pada Metrik Akurasi*



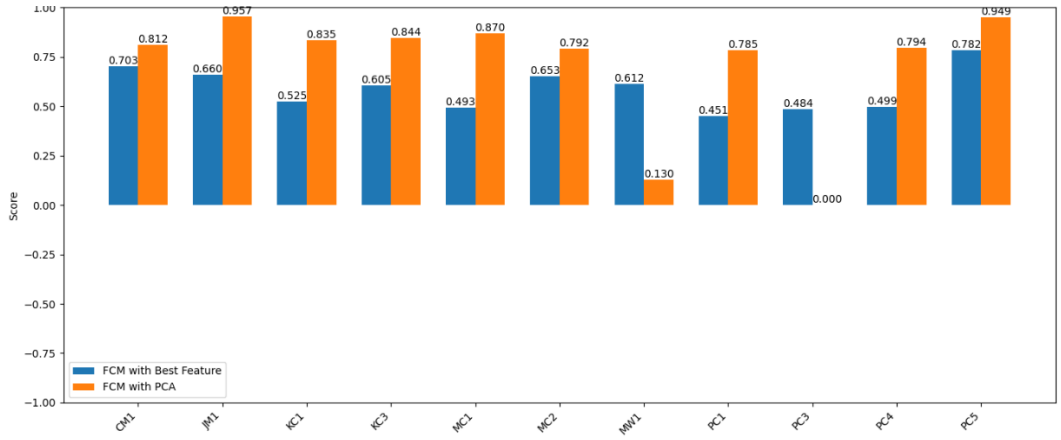

*Gambar 4.20 Grafik Perbandingan Model FCM dengan Kelompok Fitur Terbaik dan Model FCM dengan Dataset Tereduksi dari PCA pada Metrik Presisi*




*Gambar 4.21 Grafik Perbandingan Model FCM dengan Kelompok Fitur Terbaik dan Model FCM dengan Dataset Tereduksi dari PCA pada Metrik Recall*




*Gambar 4.22 Grafik Perbandingan Model FCM dengan Kelompok Fitur Terbaik dan Model FCM dengan Dataset Tereduksi dari PCA pada Metrik F1 Score*




*Gambar 4.23 Grafik Perbandingan Model FCM dengan Kelompok Fitur Terbaik dan Model FCM dengan Dataset Tereduksi dari PCA pada Metrik Silhouette Score*

*Dataset* yang telah direduksi oleh PCA kemudian dievaluasi oleh Fuzzy C-Means (FCM) dan menghasilkan performa yang bervariasi. Hasil skenario dua (menggunakan kelompok fitur terbaik Halstead) dan PCA pada skenario memperlihatkan bahwa skenario kelompok fitur terbaik unggul dalam hal akurasi, *precision*, dan *f1 score* di sebagian *dataset*. Temuan ini menunjukkan bahwa pemilihan fitur terbaik mampu mempertahankan informasi penting yang relevan dan representatif untuk mendeteksi cacat modul perangkat lunak dengan lebih tepat. Skenario dua lebih baik dalam menghindari *false positive*, namun cenderung menurun performa *recall*-nya ketika dihadapkan dengan skenario empat.

*Dataset* pada skenario dua hanya menggunakan metrik-metrik dari kelompok fitur Halstead dalam pendekatannya dan ketika dibandingkan dengan skenario ini yang menggunakan data hasil reduksi PCA dari berbagai komponen fitur yang disediakan oleh *dataset* NASA MDP (dengan jumlah awal komponen yang berbeda pula tiap *dataset*-nya), model FCM dengan data reduksi PCA lebih baik dalam memprediksi *false negative*. Keunggulan signifikan dalam *recall* ini menunjukkan keandalan *dataset* tereduksi dari PCA mampu mendeteksi lebih banyak cacat sehingga minim *false negative*. Selain itu, mayoritas *dataset* tereduksi dengan PCA memiliki nilai *silhouette score* yang lebih baik daripada kelompok fitur Halstead. Hal ini menunjukkan keandalan PCA dalam membentuk kluster yang lebih jelas dan terpisah. Namun, pada *dataset* KC3, data pada skenario reduksi *dataset* PCA hanya dapat menghasilkan satu kluster saja sehingga *recall* yang dihasilkan sempurna karena semua data *test* dianggap sebagai cacat (*positive case*). Oleh karena itu, jika dilihat dari nilai saja, *dataset* PCA terlihat lebih baik dari *dataset* kelompok fitur Halstead, namun jika melihat keandalannya dalam membentuk kluster, kasus KC3 menjadi pertimbangan keandalan *dataset* tereduksi PCA. Selain daripada itu, banyaknya informasi yang digunakan oleh PCA sebelum direduksi, dapat menjadi alasan mengapa *dataset* menjadi lebih representatif dan lebih dapat dipelajari polanya oleh model FCM.

### 4.2.5 Pembahasan Skenario 5 serta Model terbaik

Fuzzy C-Means (FCM) terbukti sebagai model *unsupervised* terbaik dalam mendeteksi cacat perangkat lunak berdasarkan hasil metrik evaluasi, namun ketika dibandingkan dengan

model *supervised* pada skenario 5, hasilnya lebih rendah daripada hasil pada *model supervised*. Model Random Forest sebagai model terbaik pada skenario 5 bekerja dengan menggunakan label ketika proses pelatihan sehingga tujuannya sesuai dengan prediksi cacat yang ingin dituju. Hal ini menyebabkan hasil model-model *supervised* skenario 5 lebih baik daripada model *unsupervised* FCM, DBSCAN, dan Mean-Shift. Namun, melihat *gap* penelitian serta pentingnya pengembangan model *unsupervised* untuk *Unsupervised Defect Prediction* (UDP) model yang memungkinkan pendeteksian cacat pada data tidak berlabel (Xu et al, 2021), maka model pada penelitian ini dapat dipertimbangkan untuk digunakan. Model FCM unggul dalam menangani ambiguitas data melalui sistem keanggotaan *fuzzy* yang sesuai dengan *dataset* seperti NASA MDP dengan karakteristik *overlap* tinggi. Kelompok fitur Halstead menjadi pilihan paling optimal karena metriknya, seperti HALSTEAD_EFFORT dan HALSTEAD_ERROR_EST, secara langsung mengukur kompleksitas dan potensi kesalahan, sehingga lebih representatif untuk identifikasi cacat. Disisi lain, terdapat NUM_OPERATORS dan HALSTEAD DIFFICULTY pada kelom pok fitur Halstead untuk menangkap karakteristik kode yang lebih relevan dengan kemungkinan cacat. Selain itu, FCM konsisten menghasilkan *silhouette score* tertinggi, menunjukkan kemampuan pembentukan kluster yang lebih terdefinisi dibandingkan model lain.

DBSCAN dan Mean-Shift menunjukkan keterbatasan signifikan dalam konteks deteksi cacat perangkat lunak. DBSCAN, meski unggul dalam mendeteksi *noise*, tetapi pada penelitian ini gagal menghasilkan kluster yang bermakna pada *dataset* tidak seimbang, seperti MW1 yang mengklasifikasikan semua data sebagai *outlier*. Sementara itu, Mean-Shift yang adaptif terhadap kepadatan data memiliki *recall* rendah (seperti pada 0,0707 pada JM1), mengindikasikan kesulitan mendeteksi cacat sebenarnya. Kedua algoritma ini juga sangat bergantung pada parameter seperti *eps* (DBSCAN) dan *bandwidth* (Mean-Shift), yang memerlukan penyesuaian (*tuning*) untuk setiap kasus agar dapat menyesuaikan dengan berbagai kemungkinan yang ada pada *dataset*. Hasil ini memperkuat temuan sebelumnya bahwa algoritma berbasis kepadatan kurang cocok untuk data dengan distribusi label tidak seimbang. Temuan ini sejalan dengan penelitian Xu et al. (2021) yang menemukan bahwa model kelompok PBC menjadi salah satu yang perlu mendapatkan perhatian lebih untuk diteliti karena performanya yang kompetitif.

Analisis fitur mengungkap bahwa metrik Halstead lebih efektif dibandingkan McCabe atau *miscellaneous*. Hal ini disebabkan oleh kemampuannya menangkap aspek kuantitatif seperti usaha pengembangan HALSTEAD_EFFORT dan estimasi kesalahan HALSTEAD_ERROR_EST, yang berkorelasi kuat dengan cacat perangkat lunak. Sebaliknya, metrik McCabe, seperti CYCLOMATIC_COMPLEXITY, hanya fokus pada alur kontrol, sehingga kurang sensitif terhadap cacat yang bersifat algoritmik (komputatif). Kombinasi fitur Halstead dan McCabe memang meningkatkan F1-Score hingga 0,5697, tetapi kelompok fitur Halstead tunggal tanpa kombinasi dengan kelompok fitur lain lebih stabil secara keseluruhan. Sehingga nilai model terbaik adalah Fuzzy C-Mean dengan kelompok fitur Halstead dengan nilai *recall* 0,8 dengan kenaikan pada setiap *dataset* dengan nilai berkisar 3-25%.

# BAB 5
# KESIMPULAN DAN SARAN

Bab ini memuat hal-hal mengenai kesimpulan dan saran selama penelitian dilangsungkan. Bahasan pada kesimpulan meliputi rangkuman dari metodologi, hasil, dan pembahasan yang diperoleh selama penelitian dalam menjawab rumusan permasalahan. Bagian saran akan berisi saran dan masukan yang dapat dilakukan pada pengembangan penelitian lebih lanjut.

## 5.1. Kesimpulan

Penelitian ini menghasilkan berbagai temuan melalui seluruh alur yang dilakukan meliputi perancangan metodologi, implementasi metode, uji coba skenario-skenario yang ada serta evaluasi performa model. Kesimpulan penelitian ini meliputi:

1. Fuzzy C-Means (FCM) Merupakan Algoritma Terbaik
   FCM secara konsisten menunjukkan performa superior dalam mendeteksi cacat perangkat lunak dibandingkan DBSCAN dan Mean-Shift. Keunggulan FCM terletak pada kemampuan menangani ambiguitas data melalui sistem keanggotaan fuzzy yang fleksibel sehingga data yang *overlap* dapat masuk kedalam kluster yang lebih sesuai. Pencapaian nilai metrik evaluasi *recall* tertinggi pada kelompok fitur Halstead dengan nilai 0,8 pada *dataset* MW1. *Silhouette score* tertinggi 0,7851 pada *dataset* PC5 yang menunjukkan kualitas kluster yang kohesif dan terpisah dengan baik. Keterbatasan algoritma berbasis kepadatan DBSCAN efektif mendeteksi *noise* namun gagal membentuk kluster yang baik, terlihat pada nilai *silhouette score* yang mayoritas berada dibawah 0 atau memiliki nilai negatif. Pada kasus MW1 semua data diklasifikasikan sebagai *outlier* sehingga bisa dikatakan DBSCAN tidak berhasil membentuk kluster dengan *dataset* NASA MDP yang mana dapat disebabkan oleh parameter yang terlalu kecil. Mean-Shift adaptif terhadap kepadatan data, namun memiliki *recall* rendah (paling rendah 0,07) sehingga kesulitan mendeteksi cacat sebenarnya dikarenakan kebergantungannya dengan parameternya (*bandwidth*) yang harus disesuaikan dengan karakter *dataset* agar mencapai bentuk optimalnya. Pengaruh signifikan *hyperparameter tuning* merupakan optimasi parameter FCM (eksponen fuzziness [*m*], toleransi konvergensi [*error*], iterasi maksimum [*max_iter*]) meningkatkan performa model. Rata-rata *F1-score* naik 10,8% setelah *tuning*. Nilai *m* = 1,1 dan *max_iter* = 10 menjadi parameter optimal bagi sebagian besar dataset dengan kenaikan terbesar terdapat pada nilai *recall* PC3 yang mencapai peningkatan sebesar 20%.
2. Pada seleksi fitur dengan kelompok fitur, Halstead menjadi metrik paling efektif diantara kelompok fitur lain. Dengan basis teori Halstead (seperti HALSTEAD_EFFORT, HALSTEAD_ERROR_EST, dan NUM_OPERATORS) dalam mendeteksi cacat pada modul perangkat lunak, ditunjukkan Halstead lebih representatif dibanding metrik McCabe (fokus alur kontrol) atau *miscellaneous* (ukuran fisik kode) karena mengukur kompleksitas komputasi dan estimasi kesalahan secara

langsung, meningkatkan rata-rata metrik *precision* sebesar 0,25 dan *F1-score* sebesar 0,08 pada FCM.

3. Penggunaan PCA pada skenario keempat berhasil mereduksi dimensi fitur dengan mempertahankan 95% variansi data. Evaluasi menunjukkan model FCM dengan data hasil reduksi PCA unggul dalam metrik *recall* dan *silhouette score*, menandakan kemampuan lebih baik dalam deteksi cacat serta membentuk kluster yang lebih jelas karena nilai *false negative* yang lebih rendah. Sementara itu, model dengan kelompok fitur Halstead menunjukkan performa lebih baik dalam *accuracy*, *precision*, dan *f1-score* yang berarti menunjukkan keunggulan penghindaran *false positive* yang lebih baik. Skenario ini menunjukkan PCA memberikan kluster yang variatif, namun pada kasus KC3, hanya dapat menghasilkan satu nilai.
4. Metode *supervised* secara umum memberikan performa yang lebih tinggi yaitu dengan perbedaan 0,2 - 0,3 dalam prediksi cacat perangkat lunak karena memanfaatkan label dalam pelatihan secara langsung pada proses pelatihannya. Namun, metode *unsupervised*, terutama FCM, tetap memiliki nilai penting untuk dikembangkan lebih lanjut, khususnya dalam skenario deteksi cacat pada data yang tidak berlabel.

## 5.2. Saran

Berdasarkan hasil penelitian dan analisis yang dihasilkan selama penelitian ini, terdapat beberapa saran yang dapat dilakukan pada penelitian di kemudian hari sebagai tindak lanjut penelitian yaitu:

1. Penanganan ketidakseimbangan dengan cara lain seperti menggunakan *synthethic generation* seperti CTABGAN.
2. Pengembangan suatu model prediksi cacat dapat dikembangkan sesuai dengan kebutuhan dan ketersediaan data sehingga ketika terdapat data berlabel pada suatu projek, dapat dipertimbangkan penggunaan model *supervised* untuk mendeteksi cacat.

# DAFTAR PUSTAKA


Abbineni, J., & Thalluri, O. (2018). *Software defect detection using machine learning techniques*. Dalam *Proceedings of the 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)* (hal. 471–475). IEEE. https://doi.org/10.1109/ICOEI.2018.8553830

Ahmed, M. R., Ali, M. A., Ahmed, N., Zamal, M. F., & Shamrat, F. M. (2020). The impact of software fault prediction in real-world application: An automated approach for software engineering. *Proceedings of the 2020 International Conference*. https://doi.org/10,1145/3379247.3379278

Alahmari, Asmaa, Jamal, Amani, & Elazhary, Hanan. (2021). Comparative Study of Common Density-based Clustering Algorithms. *1-6*. 10,1109/NCCC49330,2021.9428832.

Aljohani, A. (2024). Optimizing patient stratification in healthcare: A comparative analysis of clustering algorithms for EHR data. *International Journal of Computational Intelligence Systems, 17*, 173. https://doi.org/10,1007/s44196-024-00568-8

Analytics India Magazine. (n.d.). Hands-on tutorial on mean shift clustering algorithm. *Analytics India Magazine*. Retrieved February 18, 2025, from https://analyticsindiamag.com/ai-trends/hands-on-tutorial-on-mean-shift-clustering-algorithm/

Aquil, Mohammad Amimul Ihsan, & Wan Ishak, Wan Hussain. (2020). Predicting Software Defects using Machine Learning Techniques. *International Journal of Advanced Trends in Computer Science and Engineering, 9*, 6609. 10,30534/ijatcse/2020/352942020,

Aquil, M. A. I., & Wan Ishak, W. H. (2020). Predicting software defects using machine learning techniques. *International Journal of Advanced Trends in Computer Science and Engineering, 9*(5), 6609–6615. https://doi.org/10,30534/ijatcse/2020/352942020

Aulia, G. P., Widiharih, T., & Utami, I. T. (2023). PENERAPAN TEXT MINING DAN FUZZY C-MEANS CLUSTERING UNTUK IDENTIFIKASI KELUHAN UTAMA PELANGGAN PDAM TIRTA MOEDAL KOTA SEMARANG. *Jurnal Gaussian, 12*(1), 126-135. https://doi.org/10,14710/j.gauss.12.1.126-135

Bepery, Chinmay, Bhadra, Shaneworn, Rahman, Md.Mahbubur, Sarkar, Mihir, & Hossain, Mohammad. (2021). Improved Mean Shift Algorithm for Maximizing Clustering Accuracy. *Journal of Engineering Advancements, 2*, 1-6. 10,38032/jea.2021.01.001.

Bezdek, J. C., Ehrlich, R., & Full, W. (1984). FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences, 10*(2–3), 191–203. https://doi.org/10,1016/0098-3004(84)90020-7

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys, 41*(3), Article 15, 58 pages. https://doi.org/10,1145/1541880,1541882

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2011, June 9). *SMOTE: Synthetic Minority Over-sampling Technique* (arXiv:1106.1813) [Preprint]. arXiv. https://doi.org/10.48550/arXiv.1106.1813

Dada, E., Oyewola, D., Joseph, S., Dauda, A., Bassi, S., & Baba, A. (2021). Ensemble machine learning model for software defect prediction. *Artificial Intelligence, Advances in Machine Learning, 2*(1), 11–21.

Demirović, D. (2019). An implementation of the mean shift algorithm. *Image Processing On Line, 9*, 251–268. https://doi.org/10,5201/ipol.2019.255

Dewi, C., Siam, E. P., Wijayanti, G. A., Putri, M., Aulia, N., & Nooraeni, R. (2021). Comparison of DBSCAN and K-Means Clustering for Grouping the Village Status in Central Java 2020, *Jurnal Matematika, Statistika Dan Komputasi, 17*(3), 394–404. https://doi.org/10,20956/j.v17i3.11704

Ezugwu, A. E., Ikotun, A. M., Oyelade, O. O., Abualigah, L., Agushaka, J. O., Eke, C. I., & Akinyelu, A. A. (2022). A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence, 110*, 10474. https://doi.org/10,1016/j.engappai.2022.10474

GeeksforGeeks. (n.d.). Pengelompokan dalam pembelajaran mesin. *GeeksforGeeks*. Diambil pada 25 Januari 2025, dari https://www.geeksforgeeks.org/clustering-in-machine-learning/

Ghaffari, M. (2024). Ambiguity-driven fuzzy C-means clustering: how to detect uncertain clustered records. *Applied Intelligence*. https://doi.org/10,1007/s10489-016-0759-1

Halstead, M. H. (1977). *Elements of software science*. Elsevier.

Hung, M.-C., & Yang, D.-L. (2001). An efficient Fuzzy C-Means clustering algorithm. *Proceedings of the 2001 IEEE International Conference on Data Mining*, 225–232. https://doi.org/10,1109/ICDM.2001.989523

Jiang, Y., Cukic, B., & Menzies, T. (2007). *Fault prediction using early lifecycle data*. In Proceedings of the 18th IEEE International Symposium on Software Reliability Engineering (ISSRE '07) (pp. 237–246). IEEE. https://doi.org/10,1109/ISSRE.2007.24

Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374*(2065), 20150202. https://doi.org/10,1098/rsta.2015.0202

Krasner, H. (2020). The Cost of Poor Software Quality in the US: A 2020 Report. *Consortium for Information & Software Quality (CISQ)*. https://www.it-cisq.org/the-cost-of-poor-software-quality-in-the-us-a-2020-report/

McCabe, T. J. (1976). A complexity measure. *IEEE Transactions on Software Engineering, 2*(4), 308-320, https://doi.org/10,1109/TSE.1976.233837

McCabe, T. J., & Butler, C. W. (1989). Design complexity measurement and testing. *Communications of the ACM, 32*(12), 1415–1425. https://doi.org/10,1145/76380,76382

Miftahurrahmi, S., Zilrahmi, Amalita, N., & Mukhti, T. O. (2024). DBSCAN Method in Clustering Provinces in Indonesia Based on Crime Cases in 2022. *UNP Journal of Statistics and Data Science, 2*(3), 330–337. https://doi.org/10,24036/ujsds/vol2-iss3/203

Naeem, Samreen, Ali, Aqib, Anam, Sania, & Ahmed, Munawar. (2023). An Unsupervised Machine Learning Algorithms: Comprehensive Review. *IJCDS Journal, 13*, 911-921. 10,12785/ijcds/130172.

Neto, J.R., Souza, L.S., & Ribeiro, A.L. (2020). Comparative Analysis between the k-means and Fuzzy c-means Algorithms to Detect UDP Flood DDoS Attack on a SDN/NFV Environment. *International Conference on Web Information Systems and Technologies*.

Niño, Julio Omar Palacio. (2019). Evaluation Metrics for Unsupervised Learning Algorithms. 10,48550/arXiv.1905.05667.

Pitroda, Piyush, Donga, Bhavika, Domadiya, Dr. Dipti, & Domadiya, Hasmukh. (2024). Beyond The Basics: A Detailed Survey of Advanced Python Applications and Innovations. *International Journal for Research in Applied Science and Engineering Technology, 12*, 94-97. 10,22214/ijraset.2024.64457.

Powers, D. M. W. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. *Journal of Machine Learning Technologies, 2*(1), 37-63.

Provost, F., & Fawcett, T. (2013). *Data science for business: What you need to know about data mining and data-analytic thinking*. O'Reilly Media.

Purba, Oktaviana, Sitompul, Dian, Harahap, Tua, Saragih, Sri, & Siregar, Rizka. (2023). Application of Fuzzy C-Means Algorithm for Clustering Customers. *Hanif Journal of Information Systems, 1*, 26-36. 10,56211/hanif.v1i1.8.

Python Software Foundation. (2025). Python Logo. Diakses pada 25 Januari 2025, dari https://www.python.org/.

Qiao, W., & Shehu, A. (2022). Space partitioning and regression maxima seeking via a mean-shift-inspired algorithm. *Electronic Journal of Statistics, 16*(2), 5623–5658. https://doi.org/10,1214/22-EJS2056

Ramadhani, A. A. P., Nugroho, R. A., Faisal, M. R., Abadi, F., & Herteno, R. (2024). The impact of software metrics in NASA metric data program dataset modules for software defect prediction. *TELKOMNIKA: Telecommunication Computing Electronics and Control*, 22(4), 846–853.

Rawat, Mrinal & Dubey, Sanjay. (2012). Software Defect Prediction Models for Quality Improvement: A Literature Study. International Journal of Computer Science Issues. 9. 288-296.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics, 20*, 53-65.

Rpubs RStudio. Retrieved July 22, 2025, from https://rpubs.com/VicNP/UBL-SmoteClassif

Saatchi, R. (2024). Fuzzy Logic Concepts, Developments and Implementation. *Information, 15*(10), 656. https://doi.org/10,3390/info15100656

Sahoo, S. K., Pattanaik, P., & Mohanty, M. N. (2024). Modified fuzzy C-means clustering for anomaly detection in biomedical data. *International Journal of Intelligent Systems and Applications in Engineering, 12*(17s), 519–526.

Sambo, Ejika, Ijuo Ukpata, Sunday, Mary, Atiga, & John, Fumba. (2022). Impact of Product Quality on Customer Satisfaction and Loyalty.

Saphalya, Peta. (2022). Python- An Appetite for the Software Industry. *International journal of programming languages and applications, 12*(4):1-14. doi: 10,5121/ijpla.2022.12401

Sathyanarayanan, S. (2024). *Confusion Matrix-Based Performance Evaluation Metrics*. *African Journal of Biomedical Research, 27*(4S), 4023–4031. https://doi.org/10.53555/AJBR.v27i4S.4345

Siswantoro, M. Z. F. N., & Yuhana, U. L. (2023). *Software defect prediction based on optimized machine learning models: A comparative study*. *Teknika, 12*(2), 166–172. https://doi.org/10.34148/teknika.v12i2.634

Tao, H., Cao, Q., Chen, H., Li, Y., Niu, X., Wang, T., Geng, Z., & Shang, S. (2024). *Software defect prediction method based on clustering ensemble learning*. *IET Software*. Advance online publication. https://doi.org/10.1049/2024/6294422

Thakur, G., Priya, B., & Sharma, P. (2020). Optimal selection of network in heterogeneous environment based on fuzzy approach. *Journal of Mathematical and Computational Science, 10*, Article 4456. https://doi.org/10,28919/jmcs/4456

Vyawahare, H. R. (2022). Machine Learning: A Solution Approach for Complex Problems. *Indian Scientific Journal Of Research In Engineering And Management, 06*(04). https://doi.org/10,55041/ijsrem16123

Xu, Z., Li, L., Yan, M., Liu, J., Luo, X., Grundy, J., & Zhang, Y. (2021). A comprehensive comparative study of clustering-based unsupervised defect prediction models. *Journal of Systems & Software, 172*, 110862. https://doi.org/10,1016/j.jss.2020,110862

Zhou, S., Xu, H., Zheng, Z., Chen, J., Li, Z., Bu, J., Wu, J., Wang, X., Zhu, W., & Ester, M. (2024). A Comprehensive Survey on Deep Clustering: Taxonomy, Challenges, and Future Directions. *ACM Computing Surveys, 57*(3), 1–38. https://doi.org/10,1145/3689036

# LAMPIRAN

LAMPIRAN I

DBSCAN skenario I (tanpa *post processing*)

*jumlah kluster yang terbentuk tidak hanya dua atau bahkan semua titik dianggap pencilan

| Dataset | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|
| CM1 | NaN | NaN | NaN | NaN | -0.2608 |
| JM1 | NaN | NaN | NaN | NaN | -0.194 |
| KC1 | NaN | NaN | NaN | NaN | -0.1906 |
| KC3 | NaN | NaN | NaN | NaN | -0.203 |
| MC1 | NaN | NaN | NaN | NaN | -0.2097 |
| MC2 | 0.5484 | 0.2273 | 0.1136 | 0.1515 | NaN |
| MW1 | NaN | NaN | NaN | NaN | NaN |
| PC1 | NaN | NaN | NaN | NaN | -0.1489 |
| PC3 | NaN | NaN | NaN | NaN | -0.1216 |
| PC4 | NaN | NaN | NaN | NaN | -0.154 |
| PC5 | NaN | NaN | NaN | NaN | -0.0864 |

LAMPIRAN II

DBSCAN Skenario II

| Data set | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| CM1 | mccabe | Noise Vs Nonnoise | 0,8104 | 0,2222 | 0,1905 | 0,2051 | 0,529 |
| CM1 | mccabe | Two Largest Clusters | 0,8012 | 0,2326 | 0,2381 | 0,2353 | 0,529 |
| CM1 | mccabe | Distance Based | 0,8226 | 0,2778 | 0,2381 | 0,2564 | 0,529 |

| Data set | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| CM1 | mccabe | Silhouette Based | 0,8104 | 0,2222 | 0,1905 | 0,2051 | 0,529 |
| CM1 | halstead | Noise Vs Nonnoise | 0,6942 | 0,2041 | 0,4762 | 0,2857 | 0,1974 |
| CM1 | halstead | Two Largest Clusters | 0,6942 | 0,2041 | 0,4762 | 0,2857 | 0,1974 |
| CM1 | halstead | Distance Based | 0,6942 | 0,2212 | 0,5476 | 0,3151 | 0,1974 |
| CM1 | halstead | Silhouette Based | 0,6942 | 0,2041 | 0,4762 | 0,2857 | 0,1974 |
| CM1 | misc | Noise Vs Nonnoise | 0,7645 | 0,2464 | 0,4048 | 0,3063 | 0,3498 |
| CM1 | misc | Two Largest Clusters | 0,7645 | 0,2464 | 0,4048 | 0,3063 | 0,3498 |
| CM1 | misc | Distance Based | 0,7523 | 0,2532 | 0,4762 | 0,3306 | 0,3498 |
| CM1 | misc | Silhouette Based | 0,7645 | 0,2464 | 0,4048 | 0,3063 | 0,3498 |
| CM1 | mccabe_halstead | Noise Vs Nonnoise | 0,5199 | 0,1676 | 0,6905 | 0,2698 | -0,1742 |
| CM1 | mccabe_halstead | Two Largest Clusters | 0,5107 | 0,1685 | 0,7143 | 0,2727 | -0,1742 |
| CM1 | mccabe_halstead | Distance Based | 0,7431 | 0,0625 | 0,0714 | 0,0667 | -0,1742 |
| CM1 | mccabe_halstead | Silhouette Based | 0,5199 | 0,1676 | 0,6905 | 0,2698 | -0,1742 |
| CM1 | halstead_misc | Noise Vs Nonnoise | 0,5688 | 0,0769 | 0,2143 | 0,1132 | -0,1407 |
| CM1 | halstead_misc | Two Largest Clusters | 0,5688 | 0,0769 | 0,2143 | 0,1132 | -0,1407 |
| CM1 | halstead_misc | Distance Based | 0,5168 | 0,1882 | 0,8333 | 0,307 | -0,1407 |
| CM1 | halstead_misc | Silhouette Based | 0,5688 | 0,0769 | 0,2143 | 0,1132 | -0,1407 |
| CM1 | mccabe_misc | Noise Vs Nonnoise | 0,6575 | 0,2177 | 0,6429 | 0,3253 | NaN |
| CM1 | mccabe_misc | Two Largest Clusters | NaN | NaN | NaN | NaN | NaN |
| CM1 | mccabe_misc | Distance Based | NaN | NaN | NaN | NaN | NaN |

| Data set | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| CM1 | mccabe_ misc | Silhouette Based | NaN | NaN | NaN | NaN | NaN |
| CM1 | all | Noise Vs Nonnoise | 0,6514 | 0,0385 | 0,0714 | 0,05 | -0,2608 |
| CM1 | all | Two Largest Clusters | 0,6514 | 0,0385 | 0,0714 | 0,05 | -0,2608 |
| CM1 | all | Distance Based | 0,6361 | 0,0471 | 0,0952 | 0,063 | -0,2608 |
| CM1 | all | Silhouette Based | 0,6514 | 0,0385 | 0,0714 | 0,05 | -0,2608 |
| JM1 | mccabe | Noise Vs Nonnoise | 0,7943 | 0,5741 | 0,0577 | 0,1048 | 0,7196 |
| JM1 | mccabe | Two Largest Clusters | 0,7937 | 0,5519 | 0,0627 | 0,1125 | 0,7196 |
| JM1 | mccabe | Distance Based | 0,7876 | 0,4607 | 0,1017 | 0,1667 | 0,7196 |
| JM1 | mccabe | Silhouette Based | 0,7943 | 0,5741 | 0,0577 | 0,1048 | 0,7196 |
| JM1 | halstead | Noise Vs Nonnoise | 0,7851 | 0,4477 | 0,1247 | 0,1951 | 0,1024 |
| JM1 | halstead | Two Largest Clusters | 0,7803 | 0,4205 | 0,1377 | 0,2075 | 0,1024 |
| JM1 | halstead | Distance Based | 0,7413 | 0,2887 | 0,1632 | 0,2085 | 0,1024 |
| JM1 | halstead | Silhouette Based | 0,7851 | 0,4477 | 0,1247 | 0,1951 | 0,1024 |
| JM1 | misc | Noise Vs Nonnoise | 0,7917 | 0,5041 | 0,1507 | 0,2321 | 0,6508 |
| JM1 | misc | Two Largest Clusters | 0,7915 | 0,502 | 0,1532 | 0,2348 | 0,6508 |
| JM1 | misc | Distance Based | 0,7943 | 0,5469 | 0,0868 | 0,1499 | 0,6508 |
| JM1 | misc | Silhouette Based | 0,7917 | 0,5041 | 0,1507 | 0,2321 | 0,6508 |
| JM1 | mccabe_h alstead | Noise Vs Nonnoise | 0,764 | 0,4107 | 0,2996 | 0,3465 | -0,021 |
| JM1 | mccabe_h alstead | Two Largest Clusters | 0,7587 | 0,4003 | 0,3127 | 0,3511 | -0,021 |
| JM1 | mccabe_h alstead | Distance Based | 0,7089 | 0,2646 | 0,2215 | 0,2411 | -0,021 |

| Data set | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| JM1 | mccabe_halstead | Silhouette Based | 0,764 | 0,4107 | 0,2996 | 0,3465 | -0,021 |
| JM1 | halstead_misc | Noise Vs Nonnoise | 0,7356 | 0,3765 | 0,4057 | 0,3906 | -0,1434 |
| JM1 | halstead_misc | Two Largest Clusters | 0,7304 | 0,3734 | 0,4293 | 0,3994 | -0,1434 |
| JM1 | halstead_misc | Distance Based | 0,7012 | 0,273 | 0,2593 | 0,266 | -0,1434 |
| JM1 | halstead_misc | Silhouette Based | 0,7824 | 0,2848 | 0,0279 | 0,0508 | -0,1434 |
| JM1 | mccabe_misc | Noise Vs Nonnoise | 0,7731 | 0,4398 | 0,317 | 0,3684 | 0,3395 |
| JM1 | mccabe_misc | Two Largest Clusters | 0,7705 | 0,4333 | 0,3226 | 0,3698 | 0,3395 |
| JM1 | mccabe_misc | Distance Based | 0,7727 | 0,4144 | 0,2146 | 0,2828 | 0,3395 |
| JM1 | mccabe_misc | Silhouette Based | 0,7731 | 0,4398 | 0,317 | 0,3684 | 0,3395 |
| JM1 | all | Noise Vs Nonnoise | 0,6895 | 0,3408 | 0,5211 | 0,4121 | -0,194 |
| JM1 | all | Two Largest Clusters | 0,6839 | 0,3374 | 0,5329 | 0,4132 | -0,194 |
| JM1 | all | Distance Based | 0,6402 | 0,2167 | 0,2767 | 0,2431 | -0,194 |
| JM1 | all | Silhouette Based | 0,782 | 0,2114 | 0,0161 | 0,03 | -0,194 |
| KC1 | mccabe | Noise Vs Nonnoise | 0,7358 | 0,4133 | 0,1054 | 0,168 | 0,4946 |
| KC1 | mccabe | Two Largest Clusters | 0,741 | 0,4598 | 0,1361 | 0,21 | 0,4946 |
| KC1 | mccabe | Distance Based | 0,6936 | 0,3798 | 0,3333 | 0,3551 | 0,4946 |
| KC1 | mccabe | Silhouette Based | 0,7358 | 0,4133 | 0,1054 | 0,168 | 0,4946 |
| KC1 | halstead | Noise Vs Nonnoise | 0,7573 | 0,5349 | 0,3129 | 0,3948 | -0,0036 |
| KC1 | halstead | Two Largest Clusters | 0,7349 | 0,4663 | 0,3299 | 0,3865 | -0,0036 |
| KC1 | halstead | Distance Based | 0,5955 | 0,1562 | 0,1361 | 0,1455 | -0,0036 |

| Data set | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| KC1 | halstead | Silhouette Based | 0,7573 | 0,5349 | 0,3129 | 0,3948 | -0,0036 |
| KC1 | misc | Noise Vs Nonnoise | 0,7229 | 0,4195 | 0,2483 | 0,312 | 0,1925 |
| KC1 | misc | Two Largest Clusters | 0,7169 | 0,4064 | 0,2585 | 0,316 | 0,1925 |
| KC1 | misc | Distance Based | 0,7074 | 0,4122 | 0,3673 | 0,3885 | 0,1925 |
| KC1 | misc | Silhouette Based | 0,7229 | 0,4195 | 0,2483 | 0,312 | 0,1925 |
| KC1 | mccabe_halstead | Noise Vs Nonnoise | 0,6928 | 0,4146 | 0,5204 | 0,4615 | -0,0688 |
| KC1 | mccabe_halstead | Two Largest Clusters | 0,6549 | 0,3781 | 0,5646 | 0,4529 | -0,0688 |
| KC1 | mccabe_halstead | Distance Based | 0,6386 | 0,1649 | 0,1054 | 0,1286 | -0,0688 |
| KC1 | mccabe_halstead | Silhouette Based | 0,6859 | 0,1485 | 0,051 | 0,0759 | -0,0688 |
| KC1 | halstead_misc | Noise Vs Nonnoise | 0,6644 | 0,3924 | 0,5952 | 0,473 | -0,0841 |
| KC1 | halstead_misc | Two Largest Clusters | 0,6351 | 0,3657 | 0,602 | 0,455 | -0,0841 |
| KC1 | halstead_misc | Distance Based | 0,6334 | 0,3451 | 0,5 | 0,4083 | -0,0841 |
| KC1 | halstead_misc | Silhouette Based | 0,6644 | 0,3924 | 0,5952 | 0,473 | -0,0841 |
| KC1 | mccabe_misc | Noise Vs Nonnoise | 0,6885 | 0,3931 | 0,4252 | 0,4085 | 0,1392 |
| KC1 | mccabe_misc | Two Largest Clusters | 0,673 | 0,3812 | 0,4694 | 0,4207 | 0,1392 |
| KC1 | mccabe_misc | Distance Based | 0,6618 | 0,3173 | 0,2925 | 0,3044 | 0,1392 |
| KC1 | mccabe_misc | Silhouette Based | 0,6885 | 0,3931 | 0,4252 | 0,4085 | 0,1392 |
| KC1 | all | Noise Vs Nonnoise | 0,6265 | 0,3732 | 0,7007 | 0,487 | -0,1906 |
| KC1 | all | Two Largest Clusters | 0,5826 | 0,3491 | 0,7517 | 0,4768 | -0,1906 |
| KC1 | all | Distance Based | 0,6575 | 0,2797 | 0,2245 | 0,2491 | -0,1906 |

| Data set | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| KC1 | all | Silhouette Based | 0,6799 | 0,1518 | 0,0578 | 0,0837 | -0,1906 |
| KC3 | mccabe | Noise Vs Nonnoise | 0,8041 | 0,4615 | 0,3333 | 0,3871 | 0,2997 |
| KC3 | mccabe | Two Largest Clusters | 0,7938 | 0,4375 | 0,3889 | 0,4118 | 0,2997 |
| KC3 | mccabe | Distance Based | 0,5876 | 0,225 | 0,5 | 0,3103 | 0,2997 |
| KC3 | mccabe | Silhouette Based | 0,8041 | 0,4615 | 0,3333 | 0,3871 | 0,2997 |
| KC3 | halstead | Noise Vs Nonnoise | 0,6031 | 0,2588 | 0,6111 | 0,3636 | NaN |
| KC3 | halstead | Two Largest Clusters | NaN | NaN | NaN | NaN | NaN |
| KC3 | halstead | Distance Based | NaN | NaN | NaN | NaN | NaN |
| KC3 | halstead | Silhouette Based | NaN | NaN | NaN | NaN | NaN |
| KC3 | misc | Noise Vs Nonnoise | 0,7474 | 0,3617 | 0,4722 | 0,4096 | 0,2424 |
| KC3 | misc | Two Largest Clusters | 0,7165 | 0,3273 | 0,5 | 0,3956 | 0,2424 |
| KC3 | misc | Distance Based | 0,8196 | 0,5152 | 0,4722 | 0,4928 | 0,2424 |
| KC3 | misc | Silhouette Based | 0,7474 | 0,3617 | 0,4722 | 0,4096 | 0,2424 |
| KC3 | mccabe_halstead | Noise Vs Nonnoise | 0,6031 | 0,1525 | 0,25 | 0,1895 | NaN |
| KC3 | mccabe_halstead | Two Largest Clusters | NaN | NaN | NaN | NaN | NaN |
| KC3 | mccabe_halstead | Distance Based | NaN | NaN | NaN | NaN | NaN |
| KC3 | mccabe_halstead | Silhouette Based | NaN | NaN | NaN | NaN | NaN |
| KC3 | halstead_misc | Noise Vs Nonnoise | 0,5258 | 0,0484 | 0,0833 | 0,0612 | -0,0926 |
| KC3 | halstead_misc | Two Largest Clusters | 0,5258 | 0,0484 | 0,0833 | 0,0612 | -0,0926 |
| KC3 | halstead_misc | Distance Based | 0,5103 | 0,2243 | 0,6667 | 0,3357 | -0,0926 |

| Data set | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| KC3 | halstead_misc | Silhouette Based | 0,5258 | 0,0484 | 0,0833 | 0,0612 | -0,0926 |
| KC3 | mccabe_misc | Noise Vs Nonnoise | 0,6031 | 0,297 | 0,8333 | 0,438 | 0,0372 |
| KC3 | mccabe_misc | Two Largest Clusters | 0,5773 | 0,287 | 0,8611 | 0,4306 | 0,0372 |
| KC3 | mccabe_misc | Distance Based | 0,7216 | 0,125 | 0,0833 | 0,1 | 0,0372 |
| KC3 | mccabe_misc | Silhouette Based | 0,6031 | 0,297 | 0,8333 | 0,438 | 0,0372 |
| KC3 | all | Noise Vs Nonnoise | 0,6701 | 0,0333 | 0,0278 | 0,0303 | -0,203 |
| KC3 | all | Two Largest Clusters | 0,6701 | 0,0333 | 0,0278 | 0,0303 | -0,203 |
| KC3 | all | Distance Based | 0,6392 | 0,075 | 0,0833 | 0,0789 | -0,203 |
| KC3 | all | Silhouette Based | 0,6701 | 0,0333 | 0,0278 | 0,0303 | -0,203 |
| MC1 | mccabe | Noise Vs Nonnoise | 0,9518 | 0,0606 | 0,1111 | 0,0784 | 0,6478 |
| MC1 | mccabe | Two Largest Clusters | 0,9395 | 0,0543 | 0,1389 | 0,0781 | 0,6478 |
| MC1 | mccabe | Distance Based | 0,8868 | 0,0488 | 0,2778 | 0,083 | 0,6478 |
| MC1 | mccabe | Silhouette Based | 0,9518 | 0,0606 | 0,1111 | 0,0784 | 0,6478 |
| MC1 | halstead | Noise Vs Nonnoise | 0,9052 | 0,0429 | 0,1944 | 0,0704 | 0,0784 |
| MC1 | halstead | Two Largest Clusters | 0,8468 | 0,0386 | 0,3056 | 0,0685 | 0,0784 |
| MC1 | halstead | Distance Based | 0,8171 | 0,0401 | 0,3889 | 0,0727 | 0,0784 |
| MC1 | halstead | Silhouette Based | 0,9052 | 0,0429 | 0,1944 | 0,0704 | 0,0784 |
| MC1 | misc | Noise Vs Nonnoise | 0,8781 | 0,0684 | 0,4444 | 0,1185 | 0,3347 |
| MC1 | misc | Two Largest Clusters | 0,854 | 0,0601 | 0,4722 | 0,1066 | 0,3347 |
| MC1 | misc | Distance Based | 0,8494 | 0,0326 | 0,25 | 0,0577 | 0,3347 |

| Data set | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| MC1 | misc | Silhouette Based | 0,8781 | 0,0684 | 0,4444 | 0,1185 | 0,3347 |
| MC1 | mccabe_halstead | Noise Vs Nonnoise | 0,7894 | 0,0525 | 0,6111 | 0,0967 | -0,102 |
| MC1 | mccabe_halstead | Two Largest Clusters | 0,6901 | 0,0419 | 0,7222 | 0,0791 | -0,102 |
| MC1 | mccabe_halstead | Distance Based | 0,7065 | 0,0281 | 0,4444 | 0,0529 | -0,102 |
| MC1 | mccabe_halstead | Silhouette Based | 0,8535 | 0,0155 | 0,1111 | 0,0272 | -0,102 |
| MC1 | halstead_misc | Noise Vs Nonnoise | 0,7039 | 0,0391 | 0,6389 | 0,0737 | -0,1583 |
| MC1 | halstead_misc | Two Largest Clusters | 0,6568 | 0,0352 | 0,6667 | 0,0669 | -0,1583 |
| MC1 | halstead_misc | Distance Based | 0,7126 | 0,0219 | 0,3333 | 0,041 | -0,1583 |
| MC1 | halstead_misc | Silhouette Based | 0,9062 | 0,0067 | 0,0278 | 0,0108 | -0,1583 |
| MC1 | mccabe_misc | Noise Vs Nonnoise | 0,8089 | 0,0531 | 0,5556 | 0,0969 | 0,178 |
| MC1 | mccabe_misc | Two Largest Clusters | 0,7572 | 0,0475 | 0,6389 | 0,0885 | 0,178 |
| MC1 | mccabe_misc | Distance Based | 0,8417 | 0,0341 | 0,2778 | 0,0608 | 0,178 |
| MC1 | mccabe_misc | Silhouette Based | 0,8089 | 0,0531 | 0,5556 | 0,0969 | 0,178 |
| MC1 | all | Noise Vs Nonnoise | 0,627 | 0,0374 | 0,7778 | 0,0714 | -0,2097 |
| MC1 | all | Two Largest Clusters | 0,5236 | 0,0305 | 0,8056 | 0,0587 | -0,2097 |
| MC1 | all | Distance Based | 0,6137 | 0,0162 | 0,3333 | 0,0308 | -0,2097 |
| MC1 | all | Silhouette Based | 0,8499 | 0,0039 | 0,0278 | 0,0068 | -0,2097 |
| MC2 | mccabe | Noise Vs Nonnoise | 0,7258 | 0,7273 | 0,3636 | 0,4848 | 0,5082 |
| MC2 | mccabe | Two Largest Clusters | 0,7258 | 0,7273 | 0,3636 | 0,4848 | 0,5082 |
| MC2 | mccabe | Distance Based | 0,7258 | 0,6786 | 0,4318 | 0,5278 | 0,5082 |

| Data set | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| MC2 | mccabe | Silhouette Based | 0,7258 | 0,7273 | 0,3636 | 0,4848 | 0,5082 |
| MC2 | halstead | Noise Vs Nonnoise | 0,5565 | 0,4127 | 0,5909 | 0,486 | NaN |
| MC2 | halstead | Two Largest Clusters | NaN | NaN | NaN | NaN | NaN |
| MC2 | halstead | Distance Based | NaN | NaN | NaN | NaN | NaN |
| MC2 | halstead | Silhouette Based | NaN | NaN | NaN | NaN | NaN |
| MC2 | misc | Noise Vs Nonnoise | 0,6371 | 0,4884 | 0,4773 | 0,4828 | NaN |
| MC2 | misc | Two Largest Clusters | NaN | NaN | NaN | NaN | NaN |
| MC2 | misc | Distance Based | NaN | NaN | NaN | NaN | NaN |
| MC2 | misc | Silhouette Based | NaN | NaN | NaN | NaN | NaN |
| MC2 | mccabe_halstead | Noise Vs Nonnoise | 0,5484 | 0,3 | 0,2045 | 0,2432 | NaN |
| MC2 | mccabe_halstead | Two Largest Clusters | NaN | NaN | NaN | NaN | NaN |
| MC2 | mccabe_halstead | Distance Based | NaN | NaN | NaN | NaN | NaN |
| MC2 | mccabe_halstead | Silhouette Based | NaN | NaN | NaN | NaN | NaN |
| MC2 | halstead_misc | Noise Vs Nonnoise | 0,5242 | 0,2414 | 0,1591 | 0,1918 | -0,1845 |
| MC2 | halstead_misc | Two Largest Clusters | 0,5242 | 0,2414 | 0,1591 | 0,1918 | -0,1845 |
| MC2 | halstead_misc | Distance Based | 0,5484 | 0,4231 | 0,75 | 0,541 | -0,1845 |
| MC2 | halstead_misc | Silhouette Based | 0,5242 | 0,2414 | 0,1591 | 0,1918 | -0,1845 |
| MC2 | mccabe_misc | Noise Vs Nonnoise | 0,6048 | 0,459 | 0,6364 | 0,5333 | NaN |
| MC2 | mccabe_misc | Two Largest Clusters | NaN | NaN | NaN | NaN | NaN |
| MC2 | mccabe_misc | Distance Based | NaN | NaN | NaN | NaN | NaN |

| Data set | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| MC2 | mccabe_misc | Silhouette Based | NaN | NaN | NaN | NaN | NaN |
| MC2 | all | Noise Vs Nonnoise | 0,5484 | 0,2273 | 0,1136 | 0,1515 | NaN |
| MC2 | all | Two Largest Clusters | NaN | NaN | NaN | NaN | NaN |
| MC2 | all | Distance Based | NaN | NaN | NaN | NaN | NaN |
| MC2 | all | Silhouette Based | NaN | NaN | NaN | NaN | NaN |
| MW1 | mccabe | Noise Vs Nonnoise | 0,82 | 0,2727 | 0,48 | 0,3478 | 0,3918 |
| MW1 | mccabe | Two Largest Clusters | 0,82 | 0,2727 | 0,48 | 0,3478 | 0,3918 |
| MW1 | mccabe | Distance Based | 0,668 | 0,1705 | 0,6 | 0,2655 | 0,3918 |
| MW1 | mccabe | Silhouette Based | 0,82 | 0,2727 | 0,48 | 0,3478 | 0,3918 |
| MW1 | halstead | Noise Vs Nonnoise | 0,532 | 0,1462 | 0,76 | 0,2452 | 0,0927 |
| MW1 | halstead | Two Largest Clusters | 0,512 | 0,1407 | 0,76 | 0,2375 | 0,0927 |
| MW1 | halstead | Distance Based | 0,656 | 0,1919 | 0,76 | 0,3065 | 0,0927 |
| MW1 | halstead | Silhouette Based | 0,532 | 0,1462 | 0,76 | 0,2452 | 0,0927 |
| MW1 | misc | Noise Vs Nonnoise | 0,728 | 0,2208 | 0,68 | 0,3333 | NaN |
| MW1 | misc | Two Largest Clusters | NaN | NaN | NaN | NaN | NaN |
| MW1 | misc | Distance Based | NaN | NaN | NaN | NaN | NaN |
| MW1 | misc | Silhouette Based | NaN | NaN | NaN | NaN | NaN |
| MW1 | mccabe_halstead | Noise Vs Nonnoise | 0,74 | 0,0238 | 0,04 | 0,0299 | -0,2337 |
| MW1 | mccabe_halstead | Two Largest Clusters | 0,772 | 0,0 | 0,0 | 0,0 | -0,2337 |
| MW1 | mccabe_halstead | Distance Based | 0,592 | 0,086 | 0,32 | 0,1356 | -0,2337 |

| Data set | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| MW1 | mccabe_halstead | Silhouette Based | 0,74 | 0,0238 | 0,04 | 0,0299 | -0,2337 |
| MW1 | halstead_misc | Noise Vs Nonnoise | 0,808 | 0,04 | 0,04 | 0,04 | -0,2125 |
| MW1 | halstead_misc | Two Largest Clusters | 0,836 | 0,0556 | 0,04 | 0,0465 | -0,2125 |
| MW1 | halstead_misc | Distance Based | 0,568 | 0,0337 | 0,12 | 0,0526 | -0,2125 |
| MW1 | halstead_misc | Silhouette Based | 0,808 | 0,04 | 0,04 | 0,04 | -0,2125 |
| MW1 | mccabe_misc | Noise Vs Nonnoise | 0,508 | 0,0545 | 0,24 | 0,0889 | NaN |
| MW1 | mccabe_misc | Two Largest Clusters | NaN | NaN | NaN | NaN | NaN |
| MW1 | mccabe_misc | Distance Based | NaN | NaN | NaN | NaN | NaN |
| MW1 | mccabe_misc | Silhouette Based | NaN | NaN | NaN | NaN | NaN |
| MW1 | all | Noise Vs Nonnoise | NaN | NaN | NaN | NaN | NaN |
| MW1 | all | Two Largest Clusters | NaN | NaN | NaN | NaN | NaN |
| MW1 | all | Distance Based | NaN | NaN | NaN | NaN | NaN |
| MW1 | all | Silhouette Based | NaN | NaN | NaN | NaN | NaN |
| PC1 | mccabe | Noise Vs Nonnoise | 0,8778 | 0,2407 | 0,2364 | 0,2385 | NaN |
| PC1 | mccabe | Two Largest Clusters | NaN | NaN | NaN | NaN | NaN |
| PC1 | mccabe | Distance Based | NaN | NaN | NaN | NaN | NaN |
| PC1 | mccabe | Silhouette Based | NaN | NaN | NaN | NaN | NaN |
| PC1 | halstead | Noise Vs Nonnoise | 0,8365 | 0,2308 | 0,4364 | 0,3019 | 0,1685 |
| PC1 | halstead | Two Largest Clusters | 0,8027 | 0,1938 | 0,4545 | 0,2717 | 0,1685 |
| PC1 | halstead | Distance Based | 0,8189 | 0,1304 | 0,2182 | 0,1633 | 0,1685 |

| Data set | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| PC1 | halstead | Silhouette Based | 0,8365 | 0,2308 | 0,4364 | 0,3019 | 0,1685 |
| PC1 | misc | Noise Vs Nonnoise | 0,8203 | 0,232 | 0,5273 | 0,3222 | 0,4107 |
| PC1 | misc | Two Largest Clusters | 0,8203 | 0,232 | 0,5273 | 0,3222 | 0,4107 |
| PC1 | misc | Distance Based | 0,8056 | 0,219 | 0,5455 | 0,3125 | 0,4107 |
| PC1 | misc | Silhouette Based | 0,8203 | 0,232 | 0,5273 | 0,3222 | 0,4107 |
| PC1 | mccabe_halstead | Noise Vs Nonnoise | 0,6303 | 0,1449 | 0,7273 | 0,2417 | -0,0559 |
| PC1 | mccabe_halstead | Two Largest Clusters | 0,6156 | 0,1399 | 0,7273 | 0,2346 | -0,0559 |
| PC1 | mccabe_halstead | Distance Based | 0,5479 | 0,1337 | 0,8364 | 0,2306 | -0,0559 |
| PC1 | mccabe_halstead | Silhouette Based | 0,6303 | 0,1449 | 0,7273 | 0,2417 | -0,0559 |
| PC1 | halstead_misc | Noise Vs Nonnoise | 0,5891 | 0,1478 | 0,8545 | 0,252 | NaN |
| PC1 | halstead_misc | Two Largest Clusters | NaN | NaN | NaN | NaN | NaN |
| PC1 | halstead_misc | Distance Based | NaN | NaN | NaN | NaN | NaN |
| PC1 | halstead_misc | Silhouette Based | NaN | NaN | NaN | NaN | NaN |
| PC1 | mccabe_misc | Noise Vs Nonnoise | 0,7054 | 0,1719 | 0,6909 | 0,2754 | 0,1796 |
| PC1 | mccabe_misc | Two Largest Clusters | 0,6996 | 0,1689 | 0,6909 | 0,2714 | 0,1796 |
| PC1 | mccabe_misc | Distance Based | 0,8159 | 0,093 | 0,1455 | 0,1135 | 0,1796 |
| PC1 | mccabe_misc | Silhouette Based | 0,7054 | 0,1719 | 0,6909 | 0,2754 | 0,1796 |
| PC1 | all | Noise Vs Nonnoise | 0,5228 | 0,0179 | 0,0909 | 0,0299 | -0,1489 |
| PC1 | all | Two Largest Clusters | 0,5302 | 0,0182 | 0,0909 | 0,0304 | -0,1489 |
| PC1 | all | Distance Based | 0,5096 | 0,0397 | 0,2182 | 0,0672 | -0,1489 |

| Data set | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| PC1 | all | Silhouette Based | 0,5228 | 0,0179 | 0,0909 | 0,0299 | -0,1489 |
| PC3 | mccabe | Noise Vs Nonnoise | 0,8338 | 0,2 | 0,1154 | 0,1463 | 0,5517 |
| PC3 | mccabe | Two Largest Clusters | 0,8319 | 0,2099 | 0,1308 | 0,1611 | 0,5517 |
| PC3 | mccabe | Distance Based | 0,867 | 0,3438 | 0,0846 | 0,1358 | 0,5517 |
| PC3 | mccabe | Silhouette Based | 0,8338 | 0,2 | 0,1154 | 0,1463 | 0,5517 |
| PC3 | halstead | Noise Vs Nonnoise | 0,8215 | 0,2041 | 0,1538 | 0,1754 | 0,277 |
| PC3 | halstead | Two Largest Clusters | 0,8167 | 0,1942 | 0,1538 | 0,1717 | 0,277 |
| PC3 | halstead | Distance Based | 0,8082 | 0,0135 | 0,0077 | 0,0098 | 0,277 |
| PC3 | halstead | Silhouette Based | 0,8215 | 0,2041 | 0,1538 | 0,1754 | 0,277 |
| PC3 | misc | Noise Vs Nonnoise | 0,8044 | 0,2957 | 0,4231 | 0,3481 | 0,404 |
| PC3 | misc | Two Largest Clusters | 0,8015 | 0,2932 | 0,4308 | 0,3489 | 0,404 |
| PC3 | misc | Distance Based | 0,83 | 0,304 | 0,2923 | 0,298 | 0,404 |
| PC3 | misc | Silhouette Based | 0,8044 | 0,2957 | 0,4231 | 0,3481 | 0,404 |
| PC3 | mccabe_halstead | Noise Vs Nonnoise | 0,698 | 0,2062 | 0,5077 | 0,2933 | -0,0813 |
| PC3 | mccabe_halstead | Two Largest Clusters | 0,6857 | 0,2 | 0,5154 | 0,2882 | -0,0813 |
| PC3 | mccabe_halstead | Distance Based | 0,7094 | 0,0849 | 0,1385 | 0,1053 | -0,0813 |
| PC3 | mccabe_halstead | Silhouette Based | 0,698 | 0,2062 | 0,5077 | 0,2933 | -0,0813 |
| PC3 | halstead_misc | Noise Vs Nonnoise | 0,5954 | 0,2143 | 0,8538 | 0,3426 | -0,1766 |
| PC3 | halstead_misc | Two Largest Clusters | 0,5926 | 0,2131 | 0,8538 | 0,341 | -0,1766 |
| PC3 | halstead_misc | Distance Based | 0,7958 | 0,3293 | 0,6308 | 0,4327 | -0,1766 |

| Data set | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| PC3 | halstead_misc | Silhouette Based | 0,5954 | 0,2143 | 0,8538 | 0,3426 | -0,1766 |
| PC3 | mccabe_misc | Noise Vs Nonnoise | 0,698 | 0,2513 | 0,7308 | 0,374 | NaN |
| PC3 | mccabe_misc | Two Largest Clusters | NaN | NaN | NaN | NaN | NaN |
| PC3 | mccabe_misc | Distance Based | NaN | NaN | NaN | NaN | NaN |
| PC3 | mccabe_misc | Silhouette Based | NaN | NaN | NaN | NaN | NaN |
| PC3 | all | Noise Vs Nonnoise | 0,51 | 0,0102 | 0,0308 | 0,0153 | -0,1216 |
| PC3 | all | Two Largest Clusters | 0,5147 | 0,0103 | 0,0308 | 0,0154 | -0,1216 |
| PC3 | all | Distance Based | 0,6277 | 0,1925 | 0,6308 | 0,295 | -0,1216 |
| PC3 | all | Silhouette Based | 0,51 | 0,0102 | 0,0308 | 0,0153 | -0,1216 |
| PC4 | mccabe | Noise Vs Nonnoise | 0,7898 | 0,0826 | 0,0511 | 0,0632 | 0,4046 |
| PC4 | mccabe | Two Largest Clusters | 0,7874 | 0,0877 | 0,0568 | 0,069 | 0,4046 |
| PC4 | mccabe | Distance Based | 0,7457 | 0,1899 | 0,2557 | 0,2179 | 0,4046 |
| PC4 | mccabe | Silhouette Based | 0,7898 | 0,0826 | 0,0511 | 0,0632 | 0,4046 |
| PC4 | halstead | Noise Vs Nonnoise | 0,7937 | 0,2378 | 0,2216 | 0,2294 | 0,2902 |
| PC4 | halstead | Two Largest Clusters | 0,7748 | 0,2074 | 0,2216 | 0,2143 | 0,2902 |
| PC4 | halstead | Distance Based | 0,7882 | 0,0654 | 0,0398 | 0,0495 | 0,2902 |
| PC4 | halstead | Silhouette Based | 0,7937 | 0,2378 | 0,2216 | 0,2294 | 0,2902 |
| PC4 | misc | Noise Vs Nonnoise | 0,7866 | 0,2944 | 0,3864 | 0,3342 | 0,2528 |
| PC4 | misc | Two Largest Clusters | 0,7866 | 0,2944 | 0,3864 | 0,3342 | 0,2528 |
| PC4 | misc | Distance Based | 0,8299 | 0,402 | 0,4659 | 0,4316 | 0,2528 |

| Data set | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| PC4 | misc | Silhouette Based | 0,7866 | 0,2944 | 0,3864 | 0,3342 | 0,2528 |
| PC4 | mccabe_halstead | Noise Vs Nonnoise | 0,648 | 0,1982 | 0,5057 | 0,2848 | -0,0756 |
| PC4 | mccabe_halstead | Two Largest Clusters | 0,5937 | 0,1756 | 0,5227 | 0,2629 | -0,0756 |
| PC4 | mccabe_halstead | Distance Based | 0,6811 | 0,1863 | 0,3864 | 0,2514 | -0,0756 |
| PC4 | mccabe_halstead | Silhouette Based | 0,648 | 0,1982 | 0,5057 | 0,2848 | -0,0756 |
| PC4 | halstead_misc | Noise Vs Nonnoise | 0,5984 | 0,2179 | 0,733 | 0,3359 | -0,1665 |
| PC4 | halstead_misc | Two Largest Clusters | 0,5685 | 0,2048 | 0,733 | 0,3201 | -0,1665 |
| PC4 | halstead_misc | Distance Based | 0,5457 | 0,1976 | 0,7443 | 0,3123 | -0,1665 |
| PC4 | halstead_misc | Silhouette Based | 0,8118 | 0,0299 | 0,0114 | 0,0165 | -0,1665 |
| PC4 | mccabe_misc | Noise Vs Nonnoise | 0,6567 | 0,2336 | 0,6477 | 0,3434 | 0,0678 |
| PC4 | mccabe_misc | Two Largest Clusters | 0,6386 | 0,2231 | 0,6477 | 0,3319 | 0,0678 |
| PC4 | mccabe_misc | Distance Based | 0,7087 | 0,156 | 0,25 | 0,1921 | 0,0678 |
| PC4 | mccabe_misc | Silhouette Based | 0,6567 | 0,2336 | 0,6477 | 0,3434 | 0,0678 |
| PC4 | all | Noise Vs Nonnoise | 0,5079 | 0,1937 | 0,8068 | 0,3124 | -0,154 |
| PC4 | all | Two Largest Clusters | 0,5378 | 0,0674 | 0,1818 | 0,0983 | -0,154 |
| PC4 | all | Distance Based | 0,6031 | 0,0773 | 0,1705 | 0,1064 | -0,154 |
| PC4 | all | Silhouette Based | 0,5079 | 0,1937 | 0,8068 | 0,3124 | -0,154 |
| PC5 | mccabe | Noise Vs Nonnoise | 0,7515 | 0,7342 | 0,1266 | 0,216 | 0,6233 |
| PC5 | mccabe | Two Largest Clusters | 0,7509 | 0,7143 | 0,131 | 0,2214 | 0,6233 |
| PC5 | mccabe | Distance Based | 0,7438 | 0,6935 | 0,0939 | 0,1654 | 0,6233 |

| Data set | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| PC5 | mccabe | Silhouette Based | 0,7515 | 0,7342 | 0,1266 | 0,216 | 0,6233 |
| PC5 | halstead | Noise Vs Nonnoise | 0,732 | 0,513 | 0,1725 | 0,2582 | 0,1246 |
| PC5 | halstead | Two Largest Clusters | 0,7243 | 0,474 | 0,179 | 0,2599 | 0,1246 |
| PC5 | halstead | Distance Based | 0,7037 | 0,369 | 0,1354 | 0,1981 | 0,1246 |
| PC5 | halstead | Silhouette Based | 0,732 | 0,513 | 0,1725 | 0,2582 | 0,1246 |
| PC5 | misc | Noise Vs Nonnoise | 0,7403 | 0,5616 | 0,179 | 0,2715 | 0,6666 |
| PC5 | misc | Two Largest Clusters | 0,7403 | 0,5616 | 0,179 | 0,2715 | 0,6666 |
| PC5 | misc | Distance Based | 0,7385 | 0,5524 | 0,1725 | 0,2629 | 0,6666 |
| PC5 | misc | Silhouette Based | 0,7403 | 0,5616 | 0,179 | 0,2715 | 0,6666 |
| PC5 | mccabe_halstead | Noise Vs Nonnoise | 0,7267 | 0,4912 | 0,3057 | 0,3769 | 0,1217 |
| PC5 | mccabe_halstead | Two Largest Clusters | 0,7237 | 0,4828 | 0,3057 | 0,3743 | 0,1217 |
| PC5 | mccabe_halstead | Distance Based | 0,7037 | 0,0926 | 0,0109 | 0,0195 | 0,1217 |
| PC5 | mccabe_halstead | Silhouette Based | 0,7267 | 0,4912 | 0,3057 | 0,3769 | 0,1217 |
| PC5 | halstead_misc | Noise Vs Nonnoise | 0,7267 | 0,4916 | 0,3188 | 0,3868 | 0,0429 |
| PC5 | halstead_misc | Two Largest Clusters | 0,7231 | 0,4821 | 0,3231 | 0,3869 | 0,0429 |
| PC5 | halstead_misc | Distance Based | 0,7107 | 0,4572 | 0,3734 | 0,4111 | 0,0429 |
| PC5 | halstead_misc | Silhouette Based | 0,7267 | 0,4916 | 0,3188 | 0,3868 | 0,0429 |
| PC5 | mccabe_misc | Noise Vs Nonnoise | 0,7468 | 0,5564 | 0,3122 | 0,4 | 0,4278 |
| PC5 | mccabe_misc | Two Largest Clusters | 0,7468 | 0,5564 | 0,3122 | 0,4 | 0,4278 |
| PC5 | mccabe_misc | Distance Based | 0,7468 | 0,5457 | 0,3777 | 0,4465 | 0,4278 |

| Data set | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| PC5 | mccabe_ misc | Silhouette Based | 0,7468 | 0,5564 | 0,3122 | 0,4 | 0,4278 |
| PC5 | all | Noise Vs Nonnoise | 0,7119 | 0,4643 | 0,4258 | 0,4442 | -0,0864 |
| PC5 | all | Two Largest Clusters | 0,7043 | 0,4506 | 0,4279 | 0,439 | -0,0864 |
| PC5 | all | Distance Based | 0,709 | 0,3364 | 0,0786 | 0,1274 | -0,0864 |
| PC5 | all | Silhouette Based | 0,7119 | 0,4643 | 0,4258 | 0,4442 | -0,0864 |

LAMPIRAN III

MEANSHIFT Skenario II

*Pada kasus Meanshift, model berhasil melakukan pengelompokan sehingga metode *noise vs non noise* tidak cocok diterapkan karena hanya akan menjadi satu kluster saja

| Dataset | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| CM1 | mccabe | Distance Based | 0,7768 | 0,2182 | 0,2857 | 0,2474 | 0,5975 |
| CM1 | halstead | Two Biggest Cluster | 0,8654 | 0,3333 | 0,0476 | 0,0833 | 0,5834 |
| CM1 | halstead | Distance Based | 0,789 | 0,2453 | 0,3095 | 0,2737 | 0,5834 |
| CM1 | misc | Two Biggest Cluster | 0,8379 | 0,3778 | 0,4048 | 0,3908 | 0,5045 |
| CM1 | misc | Distance Based | 0,7859 | 0,25 | 0,3333 | 0,2857 | 0,5045 |
| CM1 | mccabe_halstead | Two Biggest Cluster | 0,841 | 0,3214 | 0,2143 | 0,2571 | 0,504 |
| CM1 | mccabe_halstead | Distance Based | 0,8471 | 0,3182 | 0,1667 | 0,2188 | 0,504 |
| CM1 | halstead_misc | Two Biggest Cluster | 0,8379 | 0,2963 | 0,1905 | 0,2319 | 0,4795 |
| CM1 | halstead_misc | Distance Based | 0,7982 | 0,26 | 0,3095 | 0,2826 | 0,4795 |
| CM1 | mccabe_misc | Two Biggest Cluster | 0,8073 | 0,2857 | 0,3333 | 0,3077 | 0,4791 |

| Dataset | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| CM1 | mccabe_misc | Distance Based | 0,8196 | 0,2927 | 0,2857 | 0,2892 | 0,4791 |
| CM1 | all | Two Biggest Cluster | 0,8257 | 0,2727 | 0,2143 | 0,24 | 0,4663 |
| CM1 | all | Distance Based | 0,7982 | 0,2391 | 0,2619 | 0,25 | 0,4663 |
| JM1 | mccabe | Two Biggest Cluster | 0,792 | 0,5061 | 0,1532 | 0,2352 | 0,6221 |
| JM1 | mccabe | Distance Based | 0,7944 | 0,5486 | 0,0875 | 0,1509 | 0,6221 |
| JM1 | halstead | Two Biggest Cluster | 0,778 | 0,3825 | 0,103 | 0,1623 | 0,4644 |
| JM1 | halstead | Distance Based | 0,7922 | 0,5182 | 0,0707 | 0,1245 | 0,4644 |
| JM1 | misc | Two Biggest Cluster | 0,786 | 0,4751 | 0,2364 | 0,3157 | 0,5889 |
| JM1 | misc | Distance Based | 0,785 | 0,4457 | 0,1222 | 0,1918 | 0,5889 |
| JM1 | mccabe_halstead | Two Biggest Cluster | 0,7807 | 0,4358 | 0,1706 | 0,2452 | 0,4259 |
| JM1 | mccabe_halstead | Distance Based | 0,764 | 0,3986 | 0,2562 | 0,3119 | 0,4259 |
| JM1 | halstead_misc | Two Biggest Cluster | 0,7842 | 0,4665 | 0,2333 | 0,311 | 0,409 |
| JM1 | halstead_misc | Distance Based | 0,7427 | 0,3318 | 0,2289 | 0,2709 | 0,409 |
| JM1 | mccabe_misc | Two Biggest Cluster | 0,7817 | 0,4623 | 0,2779 | 0,3472 | 0,5283 |
| JM1 | mccabe_misc | Distance Based | 0,7771 | 0,4223 | 0,1836 | 0,2559 | 0,5283 |
| JM1 | all | Two Biggest Cluster | 0,7782 | 0,4408 | 0,2308 | 0,3029 | 0,3877 |
| JM1 | all | Distance Based | 0,7412 | 0,3239 | 0,2202 | 0,2622 | 0,3877 |
| KC1 | mccabe | Two Biggest Cluster | 0,7461 | 0,4944 | 0,1497 | 0,2298 | 0,6277 |
| KC1 | mccabe | Distance Based | 0,7444 | 0,4694 | 0,0782 | 0,1341 | 0,6277 |

| Dataset | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| KC1 | halstead | Two Biggest Cluster | 0,7427 | 0,4638 | 0,1088 | 0,1763 | 0,5024 |
| KC1 | halstead | Distance Based | 0,716 | 0,3875 | 0,2109 | 0,2731 | 0,5024 |
| KC1 | misc | Two Biggest Cluster | 0,7108 | 0,382 | 0,2313 | 0,2881 | 0,4117 |
| KC1 | misc | Distance Based | 0,7246 | 0,3839 | 0,1463 | 0,2118 | 0,4117 |
| KC1 | mccabe_halstead | Two Biggest Cluster | 0,7375 | 0,4179 | 0,0952 | 0,1551 | 0,4018 |
| KC1 | mccabe_halstead | Distance Based | 0,722 | 0,4279 | 0,2925 | 0,3475 | 0,4018 |
| KC1 | halstead_misc | Two Biggest Cluster | 0,7435 | 0,4833 | 0,1973 | 0,2802 | 0,4341 |
| KC1 | halstead_misc | Distance Based | 0,7487 | 0,5122 | 0,1429 | 0,2234 | 0,4341 |
| KC1 | mccabe_misc | Two Biggest Cluster | 0,7255 | 0,4138 | 0,2041 | 0,2733 | 0,4941 |
| KC1 | mccabe_misc | Distance Based | 0,7229 | 0,4205 | 0,2517 | 0,3149 | 0,4941 |
| KC1 | all | Two Biggest Cluster | 0,7435 | 0,4828 | 0,1905 | 0,2732 | 0,3294 |
| KC1 | all | Distance Based | 0,7117 | 0,3949 | 0,2619 | 0,3149 | 0,3294 |
| KC3 | mccabe | Two Biggest Cluster | 0,7938 | 0,3333 | 0,1111 | 0,1667 | 0,5819 |
| KC3 | mccabe | Distance Based | 0,7835 | 0,4118 | 0,3889 | 0,4 | 0,5819 |
| KC3 | halstead | Two Biggest Cluster | 0,8144 | 0,5 | 0,0278 | 0,0526 | 0,5843 |
| KC3 | halstead | Distance Based | 0,7732 | 0,3824 | 0,3611 | 0,3714 | 0,5843 |
| KC3 | misc | Two Biggest Cluster | 0,8247 | 0,5417 | 0,3611 | 0,4333 | 0,4734 |
| KC3 | misc | Distance Based | 0,7268 | 0,3023 | 0,3611 | 0,3291 | 0,4734 |

| Dataset | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| KC3 | mccabe_halstead | Two Biggest Cluster | 0,7835 | 0,35 | 0,1944 | 0,25 | 0,4604 |
| KC3 | mccabe_halstead | Distance Based | 0,7938 | 0,3333 | 0,1111 | 0,1667 | 0,4604 |
| KC3 | halstead_misc | Two Biggest Cluster | 0,8196 | 0,5263 | 0,2778 | 0,3636 | 0,4197 |
| KC3 | halstead_misc | Distance Based | 0,768 | 0,3448 | 0,2778 | 0,3077 | 0,4197 |
| KC3 | mccabe_misc | Two Biggest Cluster | 0,7938 | 0,4231 | 0,3056 | 0,3548 | 0,4658 |
| KC3 | mccabe_misc | Distance Based | 0,768 | 0,3043 | 0,1944 | 0,2373 | 0,4658 |
| KC3 | all | Two Biggest Cluster | 0,8144 | 0,5 | 0,3333 | 0,4 | 0,3973 |
| KC3 | all | Distance Based | 0,732 | 0,25 | 0,2222 | 0,2353 | 0,3973 |
| MC1 | mccabe | Two Biggest Cluster | 0,8601 | 0,0353 | 0,25 | 0,0619 | 0,6653 |
| MC1 | mccabe | Distance Based | 0,8683 | 0,0526 | 0,3611 | 0,0919 | 0,6653 |
| MC1 | halstead | Two Biggest Cluster | 0,96 | 0,08 | 0,1111 | 0,093 | 0,506 |
| MC1 | halstead | Distance Based | 0,8596 | 0,0458 | 0,3333 | 0,0805 | 0,506 |
| MC1 | misc | Two Biggest Cluster | 0,8801 | 0,0733 | 0,4722 | 0,1269 | 0,5807 |
| MC1 | misc | Distance Based | 0,8478 | 0,0389 | 0,3056 | 0,069 | 0,5807 |
| MC1 | mccabe_halstead | Two Biggest Cluster | 0,9032 | 0,0678 | 0,3333 | 0,1127 | 0,4875 |
| MC1 | mccabe_halstead | Distance Based | 0,9247 | 0,0256 | 0,0833 | 0,0392 | 0,4875 |
| MC1 | halstead_misc | Two Biggest Cluster | 0,8888 | 0,067 | 0,3889 | 0,1143 | 0,4344 |
| MC1 | halstead_misc | Distance Based | 0,8601 | 0,0627 | 0,4722 | 0,1107 | 0,4344 |

| Dataset | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| MC1 | mccabe_misc | Two Biggest Cluster | 0,8504 | 0,0676 | 0,5556 | 0,1205 | 0,5616 |
| MC1 | mccabe_misc | Distance Based | 0,8817 | 0,0423 | 0,25 | 0,0723 | 0,5616 |
| MC1 | all | Two Biggest Cluster | 0,8776 | 0,0568 | 0,3611 | 0,0981 | 0,4755 |
| MC1 | all | Distance Based | 0,9037 | 0,0632 | 0,3056 | 0,1048 | 0,4755 |
| MC2 | mccabe | Two Biggest Cluster | 0,6855 | 0,6923 | 0,2045 | 0,3158 | 0,5652 |
| MC2 | mccabe | Distance Based | 0,6613 | 0,625 | 0,1136 | 0,1923 | 0,5652 |
| MC2 | halstead | Two Biggest Cluster | 0,6532 | 0,5714 | 0,0909 | 0,1569 | 0,4485 |
| MC2 | halstead | Distance Based | 0,6613 | 0,5385 | 0,3182 | 0,4 | 0,4485 |
| MC2 | misc | Two Biggest Cluster | 0,7097 | 0,7857 | 0,25 | 0,3793 | 0,533 |
| MC2 | misc | Distance Based | 0,6371 | 0,4762 | 0,2273 | 0,3077 | 0,533 |
| MC2 | mccabe_halstead | Two Biggest Cluster | 0,6694 | 0,6154 | 0,1818 | 0,2807 | 0,4379 |
| MC2 | mccabe_halstead | Distance Based | 0,6129 | 0,375 | 0,1364 | 0,2 | 0,4379 |
| MC2 | halstead_misc | Two Biggest Cluster | 0,6774 | 0,6429 | 0,2045 | 0,3103 | 0,4461 |
| MC2 | halstead_misc | Distance Based | 0,6048 | 0,4074 | 0,25 | 0,3099 | 0,4461 |
| MC2 | mccabe_misc | Two Biggest Cluster | 0,6935 | 0,6875 | 0,25 | 0,3667 | 0,4434 |
| MC2 | mccabe_misc | Distance Based | 0,6774 | 0,7 | 0,1591 | 0,2593 | 0,4434 |
| MC2 | all | Two Biggest Cluster | 0,6855 | 0,6471 | 0,25 | 0,3607 | 0,4283 |
| MC2 | all | Distance Based | 0,6371 | 0,48 | 0,2727 | 0,3478 | 0,4283 |

| Dataset | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| MW1 | mccabe | Two Biggest Cluster | 0,852 | 0,3125 | 0,4 | 0,3509 | 0,5488 |
| MW1 | mccabe | Distance Based | 0,78 | 0,1053 | 0,16 | 0,127 | 0,5488 |
| MW1 | halstead | Two Biggest Cluster | 0,872 | 0,2941 | 0,2 | 0,2381 | 0,4325 |
| MW1 | halstead | Distance Based | 0,876 | 0,25 | 0,12 | 0,1622 | 0,4325 |
| MW1 | misc | Two Biggest Cluster | 0,888 | 0,3333 | 0,12 | 0,1765 | 0,5548 |
| MW1 | misc | Distance Based | 0,824 | 0,2889 | 0,52 | 0,3714 | 0,5548 |
| MW1 | mccabe_halstead | Two Biggest Cluster | 0,832 | 0,1304 | 0,12 | 0,125 | 0,4431 |
| MW1 | mccabe_halstead | Distance Based | 0,808 | 0,2653 | 0,52 | 0,3514 | 0,4431 |
| MW1 | halstead_misc | Two Biggest Cluster | 0,884 | 0,3571 | 0,2 | 0,2564 | 0,4512 |
| MW1 | halstead_misc | Distance Based | 0,872 | 0,3333 | 0,28 | 0,3043 | 0,4512 |
| MW1 | mccabe_misc | Two Biggest Cluster | 0,856 | 0,3103 | 0,36 | 0,3333 | 0,486 |
| MW1 | mccabe_misc | Distance Based | 0,824 | 0,2121 | 0,28 | 0,2414 | 0,486 |
| MW1 | all | Two Biggest Cluster | 0,796 | 0,0938 | 0,12 | 0,1053 | 0,4458 |
| MW1 | all | Distance Based | 0,864 | 0,1538 | 0,08 | 0,1053 | 0,4458 |
| PC1 | mccabe | Two Biggest Cluster | 0,8822 | 0,2449 | 0,2182 | 0,2308 | 0,4589 |
| PC1 | mccabe | Distance Based | 0,8012 | 0,1 | 0,1818 | 0,129 | 0,4589 |
| PC1 | halstead | Two Biggest Cluster | 0,8837 | 0,2391 | 0,2 | 0,2178 | 0,4455 |
| PC1 | halstead | Distance Based | 0,9013 | 0,3 | 0,1636 | 0,2118 | 0,4455 |

| Dataset | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| PC1 | misc | Two Biggest Cluster | 0,863 | 0,2889 | 0,4727 | 0,3586 | 0,5447 |
| PC1 | misc | Distance Based | 0,8115 | 0,1651 | 0,3273 | 0,2195 | 0,5447 |
| PC1 | mccabe_halstead | Two Biggest Cluster | 0,8719 | 0,2419 | 0,2727 | 0,2564 | 0,3748 |
| PC1 | mccabe_halstead | Distance Based | 0,8807 | 0,1905 | 0,1455 | 0,1649 | 0,3748 |
| PC1 | halstead_misc | Two Biggest Cluster | 0,8601 | 0,25 | 0,3636 | 0,2963 | 0,3759 |
| PC1 | halstead_misc | Distance Based | 0,7923 | 0,1091 | 0,2182 | 0,1455 | 0,3759 |
| PC1 | mccabe_misc | Two Biggest Cluster | 0,8424 | 0,2347 | 0,4182 | 0,3007 | 0,4872 |
| PC1 | mccabe_misc | Distance Based | 0,8763 | 0,2542 | 0,2727 | 0,2632 | 0,4872 |
| PC1 | all | Two Biggest Cluster | 0,8557 | 0,2584 | 0,4182 | 0,3194 | 0,3649 |
| PC1 | all | Distance Based | 0,8542 | 0,2105 | 0,2909 | 0,2443 | 0,3649 |
| PC3 | mccabe | Two Biggest Cluster | 0,8234 | 0,2143 | 0,1615 | 0,1842 | 0,5698 |
| PC3 | mccabe | Distance Based | 0,849 | 0,2264 | 0,0923 | 0,1311 | 0,5698 |
| PC3 | halstead | Two Biggest Cluster | 0,8291 | 0,0833 | 0,0385 | 0,0526 | 0,4597 |
| PC3 | halstead | Distance Based | 0,8253 | 0,2545 | 0,2154 | 0,2333 | 0,4597 |
| PC3 | misc | Two Biggest Cluster | 0,8243 | 0,2993 | 0,3154 | 0,3071 | 0,5202 |
| PC3 | misc | Distance Based | 0,8329 | 0,3544 | 0,4308 | 0,3889 | 0,5202 |
| PC3 | mccabe_halstead | Two Biggest Cluster | 0,8139 | 0,1765 | 0,1385 | 0,1552 | 0,3994 |
| PC3 | mccabe_halstead | Distance Based | 0,83 | 0,1449 | 0,0769 | 0,1005 | 0,3994 |

| Dataset | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| PC3 | halstead_misc | Two Biggest Cluster | 0,8262 | 0,3008 | 0,3077 | 0,3042 | 0,3457 |
| PC3 | halstead_misc | Distance Based | 0,8148 | 0,2018 | 0,1692 | 0,1841 | 0,3457 |
| PC3 | mccabe_misc | Two Biggest Cluster | 0,7939 | 0,2331 | 0,2923 | 0,2594 | 0,4118 |
| PC3 | mccabe_misc | Distance Based | 0,8253 | 0,2545 | 0,2154 | 0,2333 | 0,4118 |
| PC3 | all | Two Biggest Cluster | 0,8205 | 0,2937 | 0,3231 | 0,3077 | 0,3383 |
| PC3 | all | Distance Based | 0,7882 | 0,2615 | 0,3923 | 0,3138 | 0,3383 |
| PC4 | mccabe | Two Biggest Cluster | 0,7937 | 0,0865 | 0,0511 | 0,0643 | 0,5326 |
| PC4 | mccabe | Distance Based | 0,7543 | 0,1304 | 0,1364 | 0,1333 | 0,5326 |
| PC4 | halstead | Two Biggest Cluster | 0,8409 | 0,3088 | 0,1193 | 0,1721 | 0,5055 |
| PC4 | halstead | Distance Based | 0,8598 | 0,4375 | 0,0398 | 0,0729 | 0,5055 |
| PC4 | misc | Two Biggest Cluster | 0,8071 | 0,2653 | 0,2216 | 0,2415 | 0,5151 |
| PC4 | misc | Distance Based | 0,8173 | 0,3542 | 0,3864 | 0,3696 | 0,5151 |
| PC4 | mccabe_halstead | Two Biggest Cluster | 0,8339 | 0,2603 | 0,108 | 0,1526 | 0,4491 |
| PC4 | mccabe_halstead | Distance Based | 0,7858 | 0,2073 | 0,1932 | 0,2 | 0,4491 |
| PC4 | halstead_misc | Two Biggest Cluster | 0,7984 | 0,2368 | 0,2045 | 0,2195 | 0,3924 |
| PC4 | halstead_misc | Distance Based | 0,7913 | 0,3057 | 0,3977 | 0,3457 | 0,3924 |
| PC4 | mccabe_misc | Two Biggest Cluster | 0,7858 | 0,25 | 0,2727 | 0,2609 | 0,4508 |
| PC4 | mccabe_misc | Distance Based | 0,7819 | 0,2179 | 0,2216 | 0,2197 | 0,4508 |

| Dataset | Feature Group | Method | Accuracy | Precision | Recall | F1 | Silhouette Score |
|---|---|---|---|---|---|---|---|
| PC4 | all | Two Biggest Cluster | 0,7866 | 0,2376 | 0,2443 | 0,2409 | 0,3654 |
| PC4 | all | Distance Based | 0,7772 | 0,2256 | 0,25 | 0,2372 | 0,3654 |
| PC5 | mccabe | Two Biggest Cluster | 0,7456 | 0,5894 | 0,1943 | 0,2923 | 0,6343 |
| PC5 | mccabe | Distance Based | 0,7491 | 0,626 | 0,179 | 0,2784 | 0,6343 |
| PC5 | halstead | Two Biggest Cluster | 0,7302 | 0,5043 | 0,1288 | 0,2052 | 0,5133 |
| PC5 | halstead | Distance Based | 0,7391 | 0,5714 | 0,1397 | 0,2246 | 0,5133 |
| PC5 | misc | Two Biggest Cluster | 0,7491 | 0,6051 | 0,2074 | 0,3089 | 0,6338 |
| PC5 | misc | Distance Based | 0,7355 | 0,5258 | 0,2227 | 0,3129 | 0,6338 |
| PC5 | mccabe_halstead | Two Biggest Cluster | 0,7468 | 0,581 | 0,2271 | 0,3265 | 0,4555 |
| PC5 | mccabe_halstead | Distance Based | 0,7344 | 0,5571 | 0,0852 | 0,1477 | 0,4555 |
| PC5 | halstead_misc | Two Biggest Cluster | 0,732 | 0,5145 | 0,155 | 0,2383 | 0,5309 |
| PC5 | halstead_misc | Distance Based | 0,745 | 0,6226 | 0,1441 | 0,234 | 0,5309 |
| PC5 | mccabe_misc | Two Biggest Cluster | 0,7414 | 0,5538 | 0,2249 | 0,3199 | 0,5994 |
| PC5 | mccabe_misc | Distance Based | 0,7285 | 0,4943 | 0,19 | 0,2744 | 0,5994 |
| PC5 | all | Two Biggest Cluster | 0,7397 | 0,5521 | 0,1965 | 0,2899 | 0,4806 |
| PC5 | all | Distance Based | 0,732 | 0,5145 | 0,155 | 0,2383 | 0,4806 |

## BIODATA PENULIS

Penulis dilahirkan di Surabaya, 07 Juni 2003, merupakan anak ketiga dari tiga bersaudara. Penulis telah menempuh pendidikan formal yaitu di TKK Santa Theresia Pandaan, SDK Panti Parama Pandaan, SMP Kristen Petra 4 Sidoarjo dan SMA Katolik Santo Louis 1 Surabaya. Setelah lulus dari SMA tahun 2021, Penulis mengikuti SBMPTN dan diterima di Departemen Teknik Informatika – FTEIC ITS pada tahun 2021 dan terdaftar dengan NRP 5025211152.

Di Departemen Teknik Informatika, Penulis sempat aktif di beberapa kegiatan yang diselenggarakan oleh Departemen, Himpunan Mahasiswa Teknik Informatika (HMTC) dan aktif sebagai asisten mata kuliah seperti Analisis dan Perancangan Sistem Informasi dan asisten praktikum pada Dasar Pemrograman dan Struktur Data .