IMPLEMENTASI DETEKSI KESALAHAN PADA PABRIK KIMIA SECARA *REAL-TIME* DENGAN *ON-THE-FLY SEMI-SUPERVISED LEARNING* BERBASIS *SUPPORT VECTOR MACHINES*

SKRIPSI

untuk memenuhi sebagian persyaratan

untuk memperoleh derajat Sarjana S-1

Program Studi Teknik Fisika

Diajukan oleh

Reza Andriady

16/399962/TK/44976

Kepada

DEPARTEMEN TEKNIK NUKLIR DAN TEKNIK FISIKA

FAKULTAS TEKNIK

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2020

PERNYATAAN BEBAS PLAGIASI

Saya yang bertanda tangan di bawah ini:

Nama	:	Reza Andriady
NIM	:	16/399962/TK/44976
Tahun terdaftar	:	2016
Program Studi	:	Teknik Fisika
Fakultas	:	Teknik

menyatakan bahwa dokumen ilmiah skripsi ini tidak terdapat bagian dari karya ilmiah lain yang telah diajukan untuk memperoleh gelar akademik di suatu lembaga Pendidikan Tinggi, dan juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang/lembaga lain, kecuali yang secara tertulis disitasi dalam dokumen ini dan disebutkan sumbernya secara lengkap dalam daftar pustaka.

Dengan demikian saya menyatakan bahwa dokumen ilmiah ini bebas dari unsur-unsur plagiasi dan apabila dokumen ilmiah Skripsi ini di kemudian hari terbukti merupakan plagiasi dari hasil karya penulis lain dan/atau dengan sengaja mengajukan karya atau pendapat yang merupakan hasil karya penulis lain, maka penulis bersedia menerima sanksi akademik dan/atau sanksi hukum yang berlaku.

Yogyakarta, 20 Juli 2020

Materai
Rp. 6000

Reza Andriady

NIM. 16/399962/TK/44976

HALAMAN PENGESAHAN

SKRIPSI

IMPLEMENTASI DETEKSI KESALAHAN PADA PABRIK KIMIA SECARA *REAL-TIME* DENGAN *ON-THE-FLY SEMI-SUPERVISED LEARNING* BERBASIS *SUPPORT VECTOR MACHINES*

oleh

Reza Andriady
16/399962/TK/44976

telah dipertahankan di depan Tim Penguji
pada tanggal *tanggal bulan tahun ujian*

Susunan Tim Penguji

Ketua Sidang

Nama Lengkap Ketua Sidang
NIP. XXXXXXXX XXXXXX X XXX

Penguji Utama

Nama Lengkap Penguji Utama
NIP. .....

Anggota Penguji

Nama Lengkap Anggota Penguji
NIP. .....

Diterima dan dinyatakan memenuhi
syarat kelulusan pada tanggal .....

Ketua Departemen Teknik Nuklir dan Teknik Fisika
Fakultas Teknik UGM

Nopriadi, S.T., M.Sc., Ph.D.
NIP. 19731119 200212 1 002

HALAMAN PERSEMBAHAN
HALAMAN TUGAS

KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI

UNIVERSITAS GADJAH MADA

FAKULTAS TEKNIK

DEPARTEMEN TEKNIK NUKLIR DAN TEKNIK FISIKA

Nama	:	Reza Andriady
NIM	:	16/399962/TK/44976
Pembimbing Utama	:	Dr.-Ing. Awang Noor Indra Wardana, S.T., M.T., M.Sc.
Pembimbing Pendamping	:	Nopriadi, S.T., M.Sc., Ph.D.
Judul Skripsi	:	Implementasi Deteksi Kesalahan Pada Pabrik Kimia Secara *Real-Time* Dengan *On-The-Fly Semi-Supervised Learning* Berbasis *Support Vector Machines*
Permasalahan	:	Perancangan dan pengujian program *fault detection* dengan *on-the-fly semi-supervised learning* menggunakan *support vector machines*

Pembimbing Utama

Dr.-Ing. Awang Noor Indra Wardana, S.T., M.T., M.Sc.
NIP. 197804012014041001

Pembimbing Pendamping

Nopriadi, S.T., M.Sc., Ph.D.
NIP. 19731119 200212 1 002

Mengetahui,
Ketua Departemen Teknik Nuklir dan Teknik Fisika
Fakultas Teknik UGM

Nopriadi, S.T., M.Sc., Ph.D.
NIP. 19731119 200212 1 002

... Karya ini kupersembahkan dengan bangga untuk diriku sendiri ...

… Tetapkan tujuan, mantapkan langkah…

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada rahmat Tuhan Yang Maha Esa, oleh karena berkat dan rahmatnya, penulis dapat menyelesaikan Tugas Akhir meskipun pada masa pandemi 2019-nCov. Juga karena pertolongan dan bimbingan-Nya, laporan Tugas Akhir dapat diselesaikan dengan sebaik-baiknya.

Penyusunan laporan tugas akhir berjudul "Implementasi Deteksi Kesalahan Pada Pabrik Kimia Secara *Real-Time* Dengan *On-The-Fly Semi-Supervised Learning* Berbasis *Support Vector Machines*" membahas pengembangan dan pengujian metode agar kesalahan proses pada pabrik kimia dapat dideteksi sedini mungkin, tanpa menggunakan data proses yang disimpan oleh pabrik, kontras dengan strategi deteksi kesalahan pada umumnya. Menggunakan support vector machines pada metode *semi-supervised learning*, data proses pabrik dapat diklasifikasi kedalam data normal dan data *salah*.

Tentu dalam pengerjaan tugas akhir dan penulisan laporan ini, penulis menerima banyak bantuan dari banyak pihak, oleh karena itu, penulis ingin mengucapkan terima kasih kepada: Dr.-Ing. Awang Noor Indra Wardana, S.T., M.T., M.Sc. selaku dosen pembimbing skripsi dan Bapak Nopriadi, S.T., M.Sc., Ph.D selaku dosen pemimbing pendamping, yang keduanya telah memberikan banyak bimbingan dan pengalaman pada penulis, sehingga penelitian dan penulisan tugas akhir dapat berjalan dengan lancer.

Penulis menyadari bahwa laporan tugas akhir ini jauh dari kata sempurna dengan banyak kekurangan dan kelemahan didalamnya, untuk itu penulis memohon maaf apabila terdapat kesalahan dalam penulisan laporan ini.

Demikianlah yang penulis dapat haturkan, penulis harap laporan ini dapat memberikan manfaat kepada pembaca.

Yogyakarta, Juli 2020

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMBANG DAN SINGKATAN

Lambang Romawi

Lambang	*Kuantitas*	*Satuan*
$\widetilde{K}$	Matriks kernel	-
$\hat{X}$	*X* setelah diproyeksikan menggunakan KPCA	-
$\check{x}$	Sampel pada *X* setelah diproyeksikan menggunakan KFDA	-
b	Ambang *hyperplane*	-
C	*Error penalty* SVM	-
L	Persamaan Langrange	-
l	Jumlah sampel pada sebuah kelas dari *X*	-
m	Jumlah sampel dari *X*	-
n	Jumlah variabel dari *X*	-
w	Arah normal *hyperplane*	-
x	Sampel data dari *X*	-
X	Himpunan data	-
Y	Label kelas dari tiap sampel pada *X*	-
y	Label kelas untuk sampel *x*	-
z	Matriks transformasi	-
K	Matriks kernel terpusat	-
$f(x)$	Fungsi *Hyperplane*	-

S	*Varians*	-
Cov	*Matriks kovarians*	-
c	Kelas dari *X*	-
f	Variabel pada X	-

Lambang Yunani

Lambang	*Kuantitas*	*Satuan*
α	*Langrange Multiplier*	-
ξ	Konstanta *slack* SVM	-
β	*Langrange Multiplier*	-
γ	Parameter gamma	
ϕ	*Kernel Mapping*	-
σ	Standar deviasi	-
ν	*Eigenvalue* dari KPCA	-
λ	*Eigenvector* dari KPCA	-
φ	*Eigenvector* dari KFDA	-

Subskrip

Lambang	*Deskripsi*
i	indeks koordinat
j	Indeks koordinat
KPCA	KPCA
KFDA	KFDA
m	Jumlah sampel dari *X*

n	Jumlah variabel dari *X*
k	indeks koordinat
c	Kelas dari *X*

Superskrip

Lambang	*Deskripsi*
d	Derajat polinomial

Singkatan

DAG	*Directed Acylic Graph*
DBSCAN	*Density Based Spectral Clustering of Applications With Noise*
FDD	*Fault Detection & Diagnosis*
KBT	Kereta Berkecepatan Tinggi
KFDA	*Kernel Fisher Discriminant Analysis*
PKM	Pengelompokan K-*Means*
KPCA	*Kernel Principal Component Analysis*
FBR	Fungsi Basis Radial
SVM	*Support Vector Machines*
PB	Positif Benar
NB	Negatif Benar

Implementasi Deteksi Kesalahan Pada Pabrik Kimia Secara *Real-Time* Dengan *On-The-Fly Semi-Supervised Learning* Berbasis *Support Vector Machines*

Oleh:

Reza Andriady

16/399962/TK/44976

Diajukan kepada Departemen Teknik Nuklir dan Teknik Fisika Fakultas Teknik
Universitas Gadjah Mada pada tanggal ……
untuk memenuhi sebagian persyaratan untuk memperoleh derajat
Sarjana Program Studi Teknik Fisika

INTISARI

Deteksi kesalahan berbasis data pada pabrik kimia umumnya membutuhkan kumpulan data berjumlah besar berisikan sampel-sampel ketika pabrik beroperasi secara normal dan ketika terdapat kesalahan pada proses pabrik, yang kemudian tiap sampel diberi label yang sesuai oleh ahli. Penelitian ini akan merancang dan menguji performa dari metode deteksi kesalahan dengan *on-the-fly semi-supervised learning* menggunakan metode *cluster-then-label*, di mana pelatihan model deteksi kesalahan dapat dilakukan sembari pabrik beroperasi, sehingga cocok untuk diterapkan apabila pabrik belum memiliki kumpulan data. Metode yang dirancang akan menggunakan data referensi dan penyangga, yang mana data referensi adalah sampel-sampel pertama dalam jumlah yang telah ditentukan sebelumnya sebagai acuan data normal bagi program, dan data penyangga berisi sampel yang masuk setelahnya. Digunakan metode KFDA untuk
pra-pemrosesan, dan k-*means* untuk pengelompokan yang dilanjutkan oleh SVM untuk klasifikasi. Berdasarkan data latih, program dibangun dengan parameter yakni ukuran data referensi sebesar 150 sampel, ukuran penyangga sebesar 50, dan γ sebesar $5x10^{-5}$. Didapatkan hasil bahwa perlu dilakukan penyesuaian parameter terlebih dahulu agar didapatkan hasil yang optimum. Pada pengujian menggunakan data latih, parameter disesuaikan sehingga γ sebesar $5x10^{-4}$, sedangkan pada data validasi, parameter disesuaikan sehingga ukuran data referensi sebesar 350 sampel dan γ sebesar $5x10^{-3}$. Dengan penyesuaian parameter, deteksi kesalahan dapat dilakukan pada kedua data.

Kata kunci: deteksi kesalahan, *support vector machines*, *semi-supervised learning*, *on-the-fly learning*

Pembimbing Utama : Dr.-Ing. Awang Noor Indra Wardana, S.T., M.T., M.Sc.

Pembimbing Pendamping : Nopriadi, S.T., M.Sc., Ph.D.

Implementation of Support Vector Machines-Based Fault Detection Using On-The-Fly Semi-Supervised Learning Method On Chemical Plant

by

Reza Andriady

16/399962/TK/44976

Submitted to the Departement of Nuclear Engineering and Engineering Physics
Faculty of Engineering Universitas Gadjah Mada on *Month Date, year*
in partial fulfillment of the requirement for the Degree of
Bachelor of Engineering in Engineering Physics

ABSTRACT

A typical data-driven fault detection strategy on chemical plants is to train the model using collected process data under normal and faulty conditions, which were labeled accordingly by experts. This research tries to design and evaluate a fault detection method's performance using on-the-fly semi-supervised learning, so that the model training can be done simultaneously as the plant operates, fit for a plant not having a large labeled process data. The designed method uses reference and buffer data. Reference data are first samples with predetermined size, used to refer normal data to the method, whereas buffer data are samples collected after. Methods used are KFDA for pre-processing, k-means for *clustering*, and SVM for classification. From the training data used, values for parameters were chosen: 150 as reference size, 50 as buffer size, and $5x10^{-5}$ for γ. It was found that parameters should be tuned according to the data to yield optimum results, parameters were tuned for training data with $5x10^{-4}$ as γ, and for validation data with 350 as reference size and $5x10^{-3}$ as γ. After parameters were tuned, the program succeeded in detecting faults on both dataset.

Keywords: fault detection, support vector machines, semi-supervised learning, on-the-fly learning

Supervisor : Dr.-Ing. Awang Noor Indra Wardana, S.T., M.T., M.Sc.

Co-supevisor : Nopriadi, S.T., M.Sc., Ph.D.

BAB I
PENDAHULUAN

I.1. Latar Belakang

Dengan berkembangnya industri modern, kualitas produk dan keselamatan merupakan faktor krusial pada proses manufaktur, namun terjadinya kesalahan pada proses dapat mengancam kedua hal tersebut. Kesalahan didefinisikan sebagai perilaku abnormal pada proses yang berhubungan dengan kegagalan mesin, kelelahan mesin, atau gangguan ekstrim pada proses [1]. Kesalahan pada salah satu bagian pada proses apabila tidak segera dideteksi dan ditangani, dapat mengakibatkan kerugian ekonomi pada perusahaan berupa produk yang berkualitas rendah, biaya perbaikan peralatan yang besar, serta dapat membahayakan keselamatan operator [2], [3]. Mempertimbangkan hal – hal tersebut, perusahaan harus mencari cara untuk mendeteksi dan mencari sumber kesalahan sedini mungkin. Dengan deteksi kesalahan, kesalahan dalam proses – proses pabrik dapat dideteksi sedini mungkin.

Berdasarkan data proses yang telah dikumpulkan, deteksi kesalahan berbasis data historis (*data-driven*) dapat dilakukan. Deteksi kesalahan secara *data-driven* bekerja dengan cara mempelajari data yang dimiliki pabrik sehingga kemudian dapat mengklasifikasi data – data baru dari pabrik tersebut [2], [3]. Digunakan teknik pembelajaran mesin untuk melakukan deteksi kesalahan karena dapat mengolah data dengan cepat dan akurat meskipun dengan jumlah variabel proses yang besar. Deteksi kesalahan dengan pembelajaran mesin memungkinkan perusahaan untuk menghindari kerugian dengan cara memperingatkan sedini mungkin bahwa terjadi kesalahan pada proses.

Umumnya, pembangunan model mesin belajar untuk deteksi kesalahan menggunakan teknik *supervised learning* yang membutuhkan kumpulan data dalam jumlah yang besar, berisi dengan sampel ketika pabrik beroperasi secara normal dan sampel ketika terdapat kesalahan dalam proses pabrik, yang tiap sampelnya sudah diberi label oleh para ahli untuk memberikan keterangan apakah sebuah

sampel merupakan sampel normal atau sampel salah. *Supervised learning* dapat memberikan hasil yang relatif lebih akurat, namun proses pengumpulan data merupakan tugas yang sulit dan membutuhkan waktu yang banyak. Di sisi lain, model mesin belajar lain yaitu teknik *unsupervised learning*, tidak membutuhkan kumpulan data berlabel namun tidak dapat memberikan hasil pengelompokan yang akurat [4]. Kombinasi dari kedua teknik tersebut disebut dengan teknik *semi-supervised learning*, yang mampu memberikan prediksi terhadap kelas dari sebuah sampel yang masuk ke dalam model secara akurat, tanpa memerlukan kumpulan data berlabel terlebih dahulu. Karena tidak membutuhkan data berlabel, metode *semi-supervised learning* memungkinkan untuk melakukan pembangunan model dengan data yang dikumpulkan sembari pabrik beroperasi.

Salah satu metode klasifikasi yang sering digunakan untuk deteksi dan isolasi kesalahan pada lingkup produksi adalah *support vector machines* (SVM) [5]. SVM dilaporkan memiliki akurasi yang tinggi [5] serta memiliki performa yang baik bahkan dengan menggunakan data latih yang sedikit [6], [7]. Namun sebagai salah satu metode *supervised learning*, SVM juga membutuhkan data berlabel untuk dapat membangun model klasifikasinya. Salah satu ekstensi dari SVM, yakni *one-class* SVM merupakan metode *unsupervised learning*, yang digunakan untuk permasalahan satu kelas, di mana data latih berupa data positif akan digunakan untuk membangun sebuah *hyperplane* untuk memisahkan data positif dengan data negatif [8]. Terdapat pula teknik *semi-supervised learning* menggunakan SVM, yang mana SVM didahului oleh metode pengelompokan *unsupervised*, sehingga didapatkan label hasil pengelompokan yang kemudian digunakan untuk melatih SVM. Metode tersebut disebut dengan metode *cluster-then-label* [9]. Dalam metode *cluster-then-label*, data berlabel digunakan sebagai panduan pengelompokan sehingga karakteristik dari data berlabel diikutsertakan secara implisit dalam algoritma pengelompokan.

Pada penelitian ini, teknik *semi-supervised learning* akan dilakukan secara *on-the-fly*, sehingga pembangunan model deteksi kesalahan dapat dilakukan sembari program beroperasi. Dengan begitu, program deteksi kesalahan yang

dirancang pada penelitian ini dapat digunakan untuk melakukan deteksi kesalahan apabila kumpulan data berlabel belum tersedia.

I.2. Perumusan Masalah

Kekurangan dari strategi deteksi kesalahan dengan *supervised learning* adalah strategi tersebut membutuhkan kumpulan data dalam jumlah besar yang berisi nilai-nilai pengukuran sensor pada saat pabrik beroperasi dalam kondisi normal maupun salah, serta telah diberi label yang sesuai oleh ahli. Kumpulan data tersebut digunakan sebagai data latih oleh strategi deteksi kesalahan, namun, proses pengumpulan kumpulan data merupakan tugas yang membutuhkan waktu yang lama dan mahal. Selain itu, apabila terdapat sebuah sampel dengan jenis kesalahan yang tidak terdapat pada kumpulan data latih, maka strategi deteksi kesalahan akan mengalami kesulitan dalam mendeteksi kesalahan tersebut. Dengan demikian, dibutuhkan sebuah metode yang tidak membutuhkan kumpulan data berlabel dari proses pabrik, sehingga dapat mengatasi kekurangan dari strategi deteksi kesalahan dengan *supervised learning*, namun tetap memiliki akurasi yang tinggi.

I.2.1. Batasan Masalah

Berikut batasan masalah yang digunakan dalam penelitian ini:

1. Data yang diteliti terbatas pada data pabrik Tennessee Eastman Chemical Company dan data pabrik yang disediakan oleh Brooks. Kedua data yang digunakan didapatkan melalui gudang data *South African Council for Automation and Control*.
2. Pengujian dan metode-metode yang digunakan merupakan metode deteksi kesalahan secara *data-driven.*
3. Program dibuat dengan menggunakan bahasa pemrograman Python.

I.3. Tujuan Penelitian

Penelitian ini bertujuan untuk memperoleh rancangan dari program yang dapat mendeteksi kesalahan pada data proses pabrik secara *on-the-fly semi-supervised learning*, sehingga program dapat mengumpulkan data latih sembari

pabrik beroperasi dan data latih yang digunakan tidak memerlukan label dari para ahli. Selain itu, penelitian juga bertujuan untuk menguji dari program yang telah dirancang.

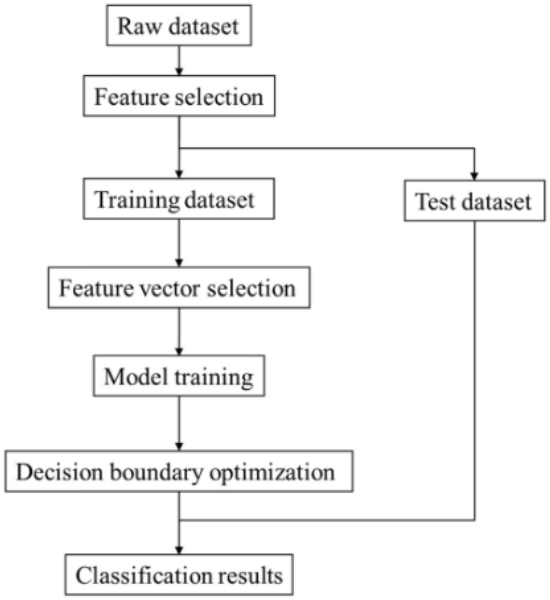
I.4. Manfaat Penelitian

Penelitian ini merancang kemudian menguji performa dari program yang dapat melakukan deteksi kesalahan secara *on-the-fly semi-supervised*, sehingga program yang dirancang diharapkan dapat melakukan deteksi kesalahan pada sebuah pabrik kimia tanpa memerlukan kumpulan data dari pabrik tersebut sebelumnya.

Deteksi kesalahan biasanya membutuhkan kumpulan data dari pabrik kimia tertentu yang sudah memiliki label. Memperoleh kumpulan data tersebut merupakan tugas yang sulit. Program dengan kemampuan *on-the-fly learning* dapat digunakan untuk melakukan mempelajari karakter dari data pabrik tanpa membutuhkan kumpulan data terlebih dahulu. Sehingga penelitian ini memiliki manfaat bagi industri supaya industri dapat menggunakan atau mengembangkan program yang telah dirancang untuk melakukan deteksi kesalahan apabila kumpulan data belum tersedia.

BAB II
TINJAUAN PUSTAKA

Sampai saat ini, *support vector machines* (SVM) sudah banyak diimplementasikan pada berbagai keperluan, baik regresi ataupun klasifikasi. Salah satu bidang mengimplementasikan SVM sebagai *classifier* adalah deteksi kesalahan.



Gambar II.1 Rangka kerja SVM untuk deteksi kesalahan pada KBT [2]

J. Liu, Y.-F. Li dan E. Zio [10] meneliti tentang implementasi rangka kerja SVM dalam mendeteksi kesalahan pada sistem pengereman kereta berkecepatan tinggi (KBT). Data yang di dalamnya terdapat kesalahan pada sistem pengereman sangatlah langka karena tingginya tingkat kehandalan sistem pengereman kereta berkecepatan tinggi, sehingga pada penelitian ini, SVM mengolah data yang sangat tidak berimbang dengan mayoritas data merupakan data pada kondisi normal. Rangka kerja SVM yang diusulkan dapat dibagi menjadi 4 tahap: seleksi fitur, seleksi vektor fitur, pembangunan model dan optimasi batas keputusan. Seleksi fitur dilakukan dengan cara menyisihkan variabel yang tidak relevan terhadap deteksi kesalahan untuk menyingkat waktu komputasi. Variabel yang disihkan

merupakan variabel yang menghasilkan nilai keterpisahan antar kelas dibawah batas yang ditetapkan. Dengan variabel yang tersisa, data kemudian diproyeksikan pada vektor fitur menggunakan metode kernel. Tahap seleksi vektor fitur kemudian menyisihkan vektor fitur yang tidak informatif terhadap deteksi kesalahan untuk menyingkat waktu komputasi. Menggunakan proyeksi data pada vektor fitur yang tersisa, SVM dibangun dan kemudian dioptimasi. Pada 15 dataset yang digunakan rangka kerja SVM yang ditawarkan menghasilkan 12 nilai F-*measurement* dan 9 nilai G-*mean* yang lebih tinggi dibandingkan 3 metode pembanding.

K.-Y. Chen, L.-S. Chen, M.-C. Chen dan C.-L. Lee [5] menguji SVM untuk mendeteksi kesalahan pada pembangkit listrik termal. Pra-proses dilakukan pada data untuk menghilangkan data yang inkonsisten dan terdapat derau serta untuk menormalisasi data. Setelah itu, seleksi fitur dilakukan untuk menyisihkan variabel yang tidak relevan dengan implementasi *metode correlation analysis* dan *decision tree algorithm*. Karena terdapat tahap eliminasi variabel, akan terjadi sedikit penurunan pada akurasi klasifikasi, tetapi penurunan tersebut berada didalam batas yang dapat diterima, dibandingkan dengan penurunan waktu komputasi yang dibutuhkan untuk mengkonstruksi SVM. Menggunakan variabel yang tersisa, SVM dikonstruksi dengan Fungsi Basis Radial (FBR) *kernel* dan kemudian dioptimalkan dengan mencari nilai parameter yang optimal. Hasil dari penelitian menunjukkan bahwa deteksi kesalahan menggunakan SVM mengalahkan performa metode *Linear Discriminant Analysis* (79.95%) dan *Back Propagation Neural Network* (85.57%), dengan SVM dapat mendeteksi kesalahan mengidentifikasi jenis kesalahan pada pembangkit listrik termal dengan akurasi diatas 91.93%.

Pada penelitian – penelitian diatas dapat dilihat bahwa dalam melakukan fungsi klasifikasi pada deteksi kesalahan, SVM sebagai *classifier* dilakukan setelah data diolah menggunakan proses reduksi dimensi. Reduksi dimensi merupakan tahap pengurangan variabel data untuk mengurangi waktu komputasi dalam konstruksi SVM. Reduksi dimensi dapat dilakukan dengan 2 cara, yaitu dengan seleksi fitur dan ekstraksi fitur. Seleksi fitur dilakukan dengan cara mengeliminasi variabel yang tidak informatif atau tidak memberikan informasi tambahan terhadap variabel lain. Seperti pada penelitian yang dilakukan pada yang sudah ditinjau, di

mana variabel yang memiliki nilai keterpisahan antar kelas dibawah batas yang ditentukan, dianggap sebagai variabel yang tidak relevan sehingga dapat dieliminasi. Lain halnya dengan reduksi dimensi menggunakan metode ekstraksi fitur yang membuat data diproyeksikan pada sebuah ruang fitur baru yang memiliki jumlah dimensi lebih sedikit dibanding jumlah variabel sebenarnya.

Z. M. Hira dan D. F. Gillies [11] melakukan penelitian untuk mengulas aplikasi seleksi fitur dan ekstraksi fitur pada data genetik. Pada penelitian ini, metode seleksi fitur dan ekstraksi fitur digunakan untuk mengurangi dimensi dari data genetik kanker, dan dibandingkan dengan cara meninjau akurasi SVM dalam mengklasifikasi data hasil reduksi dimensi tersebut. Penelitian ini membandingkan performa metode – metode populer pada seleksi fitur dan ekstraksi fitur. Terdapat total 4 metode yang digunakan dari seleksi fitur, yakni: ReliefF, *Information Gain*, *Information Gain Ratio*, dan χ^2-*statistics*. Sedangkan pada metode ekstraksi fitur digunakan metode PCA, *locally linear embedding* (LLE), dan Isomap. Penelitian ini menyimpulkan bahwa ekstraksi fitur menghasilkan akurasi SVM yang lebih tinggi serta dapat mengurangi overfitting dari SVM, namun seleksi fitur membutuhkan waktu komputasi yang lebih cepat serta menjaga karakteristik data sehingga dapat diinterpretasikan.

Penelitian yang dilakukan oleh L. H. Chiang, E. L. Russell dan R. D. Braatz [1] membandingkan beberapa metode ekstraksi fitur dalam deteksi kesalahan dan diagnosis kesalahan pada data proses Tennessee Eastman. Metode ekstraksi fitur yang dibandingkan adalah *fisher discriminant analysis* (FDA), *discriminant partial least squares* (DPLS), dan *principal component analysis* (PCA). Hasil reduksi dimensi dari ketiga metode tersebut akan diklasifikasi menggunakan karakterisasi T^2-*statistics* dan Q-*statistics* untuk dibandingkan akurasi klasifikasinya. Penelitian ini menyimpulkan bahwa metode PCA memilki rerata misklasifikasi paling besar dibandingkan metode lain, yaitu menghasilkan rerata klasifikasi sebesar 0,74, 0,61, dan 0,67. DPLS memiliki rerata misklasifikasi rata-rata 0,57 dan FDA memiliki rerata misklasifikasi paling kecil yaitu sebesar 0,21. PCA memiliki rerata misklasifikasi yang paling besar karena sebagai PCA membaca data normal dan data salah sebagai kelas yang sama, dan hanya berusaha memaksimalkan variansi

data. Sedangkan, FDA membaca data pada masing-masing kelas dan berusaha membuat ruang fitur sehingga proyeksi data memiliki variansi antar kelas yang besar. Pada studi ini peneliti juga menyimpulkan bahwa kemampuan FDA yang superior dalam diagnosis kesalahan merupakan kemampuan yang inheren, dan bukan dikarenakan model kriteria seleksi orde yang digunakan pada studi.

Menggunakan persamaan KFDA yang dikembangkan oleh Yang [12], Penelitian Zhu dan Song [13] menguji implementasi KFDA pada deteksi kesalahan menggunakan data proses Tennessee Eastman. Pada penelitian ini, data proses Tennessee Eastman diolah menggunakan PCA, KPCA, FDA, dan KFDA sebagai metode ekstraksi fitur sehingga data diproyeksikan kedalam ruang fitur 2 dimensi dan kemudian diklasifikasikan menggunakan *k-nearest neighbor* (KNN) dan *gaussian mixture model* (GMM) untuk mengidentifikasi apakah sebuah data merupakan data normal, kesalahan tipe 4, kesalahan tipe 8, atau kesalahan tipe 14. Penelitian menunjukkan bahwa PCA dan KPCA memproyeksikan data menjadi berpusat pada titik pusat grafik dan menghasilkan akurasi klasifikasi sebesar 60,13% dan 60,81% menggunakan KNN secara berurutan serta 64,31% dan 64% menggunakan GMM secara berurutan. FDA berhasil memisahkan kesalahan tipe 4 tetapi data normal, kesalahan tipe 8, dan kesalahan tipe 14 saling tumpang tindih. Ekstraksi fitur dengan FDA memiliki akurasi klasifikasi sebesar 58,19% menggunakan GMM dan 56,75% menggunakan KNN. KFDA berhasil memisahkan 4 kelas menjadi 4 kluster dengan catatan ada beberapa proyeksi kesalahan tipe 8 yang tumpang tindih dengan data normal. Secara keseluruhan, ekstraksi fitur dengan KFDA memiliki akurasi klasifikasi sebesar 92.75% menggunakan GMM dan 90,81% menggunakan KNN. Pada penelitian yang telah disebutkan, SVM mampu mendeteksi dan mengidentifikasi kesalahan pada sistem di mana ia diaplikasikan. Namun karena keterbatasan SVM di mana ia hanya memiliki performa baik pada data dapat dipisahkan secara linier, SVM tidak dapat mendeteksi kesalahan pada sistem dengan data yang tidak dapat dipisahkan secara linier. Pada penelitian tersebut, KFDA terbukti dapat memproyeksikan data yang tidak dapat dipisahkan secara linier menjadi 4 kluster yang terpisah pada ruang fitur 2 dimensi sehingga data tersebut menjadi terpisah secara linier.

Deteksi kesalahan secara data-driven menggunakan data deret waktu yang telah dikumpulkan untuk mempelajari karakter dari data normal dan data sehingga dapat kedua data tersebut dapat dipisahkan. Kelemahan dari supervised learning adalah metode tersebut membutuhkan data berlabel yang cukup pada tahap pembangunan model sebagai data latih, apabila sebuah data baru memiliki karakteristik yang berbeda dari data pada data latih, maka rawan terjadi misklasifikasi terhadap data baru tersebut. Pengumpulan data berlabel merupakan proses yang memakan waktu yang lama dan memakan biaya, yang mana data harus dikumpulkan dari setiap kondisi yang mungkin terjadi kemudian dilabelkan oleh ahli [4].

Selain metode *supervised*, terdapat pula metode *unsupervised* yang dapat melakukan karakterisasi data tanpa menggunakan label. Metode *unsupervised* merupakan metode yang tidak membutuhkan pengetahuan sebelumnya mengenai keluaran atau label dari sebuah data. Fungsi dari metode *unsupervised* adalah memberikan perkiraan keluaran berdasarkan struktur dari kumpulan data yang disediakan [14].

Penelitian pada [8] mengajukan metode deteksi kesalahan pada proses produksi wafer semikonduktor. Metode yang diajukan dapat melakukan deteksi kesalahan secara *unsupervised* dengan *on-the-fly learning* yang kemudian dipasang pada sistem proses produksi sehingga dapat mendeteksi kesalahan proses secara *real-time*. Metode tersebut menggunakan data-data histori dari proses untuk mendapatkan acuan data normal yang kemudian digunakan untuk membangun model seleksi fitur dan *classifier*. Metode tersebut menggunakan metode *unsupervised learning* yakni 1-Class SVM sebagai *classifier* yang hanya membutuhkan data dari 1 kelas untuk proses pembangunan model. Apabila data baru masuk ke dalam algoritma, data baru akan diolah menggunakan seleksi fitur kemudian diklasifikasikan oleh 1-Class SVM sebagai data salah atau data normal. Apabila data tersebut diklasifikasikan sebagai data normal, data tersebut akan dimasukkan kedalam kumpulan data referensi dan data paling tua pada kumpulan data referensi akan dikeluarkan atau disimpan berdasarkan skenario *sliding window* yang digunakan. Dengan terbentuknya data referensi baru, maka pembangunan

model seleksi fitur dan *classifier* akan dilakukan kembali untuk memperbarui model yang sudah dibangun. Terdapat 2 skenario *sliding window* yang diajukan, skenario pertama merupakan skenario apabila data baru masuk kedalam kumpulan data referensi, data paling tua tidak langsung dikeluarkan, namun apabila panjang kumpulan data sudah mencapai batas tertentu, maka beberapa data paling tua akan dikeluarkan secara bersamaan. Skenario kedua merupakan skenario *sliding window* apabila data paling tua dikeluarkan apabila terdapat data baru yang masuk. Metode yang diajukan pada penelitian ini mampu mendeteksi kesalahan pada 2 data industri dengan akurasi yang memuaskan, pada dataset 1 dihasilkan akurasi sebesar 95.65% pada kedua skenario, sedangkan pada dataset 2 dihasilkan akurasi sebesar 83.3 dan 91.67 pada masing-masing skenario. Penelitian yang dilakukan menunjukkan bahwa skenario *sliding window* pertama menghasilkan rerata alarm palsu yang lebih rendah, sedangkan skenario kedua menghasilkan rerate deteksi yang lebih tinggi.

Semi-supervised learning mengombinasikan kedua metode sebelumnya untuk saling melengkapi kekurangannya. *Supervised learning* membutuhkan data latih berlabel dalam jumlah besar, yang merupakan proses yang selain memakan biaya, juga membutuhkan waktu yang banyak. Sedangkan unsupervised learning tidak membutuhkan kumpulan data berlabel, sehingga dapat digunakan untuk mengolah data mentah, namun rawan terjadi misklasfikasi. *Semi-supervised* learning diajukan sebagai solusi dari kekurangan tersebut, metode semi-supervised learning dapat membangun model berdasarkan data dengan data berlabel dalam jumlah yang kecil, dan memperlakukan data lain sebagai data uji [4]**.** Penelitian [4] menyebutkan bahwa pembangunan model secara *semi-supervised* menggunakan data berlabel dan tidak berlabel menghasilkan akurasi yang lebih tinggi dibandingkan metode *supervised* maupun *unsupervised*. Salah satu metode *semi-supervised* adalah metode *cluster-then-label*, yaitu metode yang menggunakan metode pengelompokan untuk memberikan *pseudo-label* dari sebuah dataset, *pseudo-label* tersebut kemudian digunakan oleh metode *classifier* untuk menentukan label akhir dari tiap data dan untuk membentuk batas pemisah. Penelitian [15] menguji metode pengelompokan yang diikuti oleh metode

klasifikasi untuk mendeteksi anomali, yang mendapatkan hasil bahwa gabungan metode k-*medoid* dan metode SVM mendapatkan rata-rata akurasi sebesar 99.43% pada dataset kecil (10^4 sampel) dan 99.21% pada dataset besar (10^5 sampel).

Apabila karakteristik data normal dari sebuah proses telah diketahui, maka metode *cluster-then-label* dari *semi-supervised learning* dapat digunakan untuk menggolongkan data-data yang mirip kedalam kelompok-kelompok menggunakan metode pengelompokan dengan diberikan *pseudo-label*, dan diikuti oleh metode klasifikasi untuk memberikan label akhir dari kelompok-kelompok tersebut yang mempertimbangkan *pseudo-label* serta proyeksi data normal sebenarnya pada ruang fitur. Pada penelitian ini, tahap pengelompokan dan tahap klasifikasi dilakukan sebagai 2 proses independen, seperti pada penelitian [15] dan [9]. Penelitian ini akan menggunakan metode *cluster-then-label* untuk dapat melakukan deteksi kesalahan secara *real-time* dengan *on-the-fly learning*. Penelitian ini juga akan menggunakan metode *sliding window* yang digunakan pada penelitian [8] untuk mendeteksi kesalahan dengan membangun model pengelompokan dan klasifikasi ketika data baru masuk kedalam *sliding window*, namun pada penelitian ini, data yang diproses merupakan data yang terdapat pada *sliding window* karena digunakan metode pengelompokan terlebih dahulu. Untuk dapat melakukan *on-the-fly learning,* alih-alih menggunakan data historik untuk mendapatkan karakteristik data normal seperti pada penelitian [8], karakteristik data normal didapatkan melalui data-data awal yang masuk sebagai acuan data normal. Digunakan metode *kernel fisher discriminant analysis* sebagai metode ekstraksi fitur untuk menangkap karakteristik data normal serta memproyeksikan data normal pada sebuah persebaran yang berbeda dengan data salah. Setelah dilakukan pengelompokan menggunakan metode yang akan diuji pada penelitian ini, digunakan *support vector machines* sebagai metode *classifier*. *Support vecdtor machines* dipilih karena dapat menghasilkan akurasi yang tinggi meskipun pada data yang tidak seimbang. Hal tersebut menjadi salah satu faktor yang dipertimbangkan, mengingat program yang dirancang akan melakukan *real-time*, sehingga ketika diharapkan data salah dapat segera terdeteksi oleh program meskipun jumlah data salah relatif kecil dibandingkan jumlah data normal pada *sliding window*.

Penelitian	Metode		Sumber Data Latih	Mode Implementasi
	Kategori Metode	Metode yang digunakan		
Chen, Chen, Chen, dan Lee [5]	*Supervised*	SVM	Data histori	*Offline*
Albalate, Suchindranath, Suendermann, dan Minker [9]	*Semi-supervised*	*Cluster-then-label* (SVM)	Data histori	*Offline*
Hassan, Lambert-Lacroix, dan Pasqualini [8]	*Unsupervised*	*One-Class SVM*	Data histori dan *real-time*	*Online*
Hasil penelitian yang diharapkan	*Semi-supervised*	*Cluster-then-label* (SVM)	Data *real-time*	*Online*

BAB III
DASAR TEORI

III.1. Semi-Supervised Learning

Teknik pembelajaran mesin (*machine learning*) umumnya dibagi menjadi 2, yakni: *supervised learning* dan *unsupervised learning*. Metode *supervised* merupakan teknik pembelajaran mesin di mana model menggunakan kumpulan data yang telah disediakan sebelumnya untuk mempelajari keluaran dari tiap data, sehingga model kemudian dapat memprediksi keluaran dari sebuah data baru. Tugas dari metode *supervised* adalah untuk memetakan sebuah data sebagai masukan kedalam sebuah kelas sebagai keluaran. Berdasarkan kumpulan data dengan label yang telah disediakan, *supervised learning* akan mempelajari pola dari kumpulan data tersebut untuk membentuk sebuah model yang dapat mengklasifikasikan tiap sampel kedalam label yang sesuai [4]. Dengan model yang telah dibentuk, apabila model diberi masukan berupa sampel baru, maka model dapat memprediksi label dari sampel tersebut. Dalam kata lain, metode *supervised learning* merupakan metode yang menggunakan kebenaran yang telah disediakan (*ground truth*) untuk membentuk model.

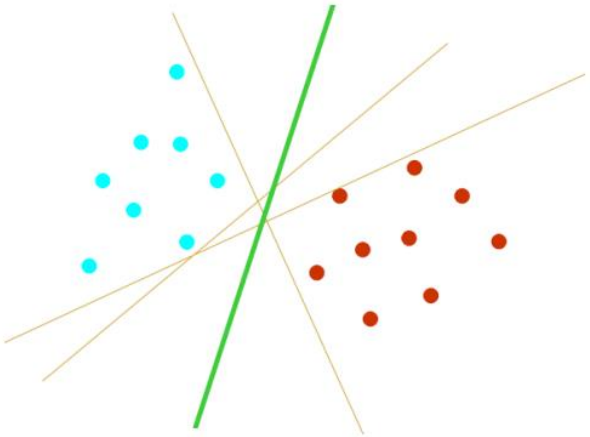
Sedangkan pada teknik pembelajaran mesin, metode *unsupervised* merupakan metode yang tidak membutuhkan pengetahuan sebelumnya mengenai keluaran dari data tersebut. Fungsi dari metode *unsupervised* adalah memberikan perkiraan keluaran berdasarkan struktur dari kumpulan data yang disediakan [14].

Semi-supervised learning mengombinasikan kedua metode sebelumnya untuk saling melengkapi kekurangannya. *Supervised learning* membutuhkan data latih berlabel dalam jumlah besar, yang merupakan proses yang selain memakan biaya, juga membutuhkan waktu yang banyak. Sedangkan *unsupervised learning* tidak membutuhkan kumpulan data berlabel, sehingga dapat digunakan untuk mengolah data mentah, namun rawan terjadi misklasfikasi. *Semi-supervised learning* diajukan sebagai solusi dari kekurangan tersebut, *semi-supervised learning* dapat digunakan pada kumpulan data dengan jumlah sampel yang kecil

dan dengan kelas yang tidak lengkap untuk dapat memeberikan label dari data baru [4].

III.2. Support Vector Machines (SVM)

Support vector machines (SVM) merupakan pendekatan statistik multivarian yang relatif baru dan menjadi populer karena memiliki hasil yang banyak diminati pada persoalan klasifikasi dan regresi [3]. Prinsip fundamental dari SVM adalah membentuk *hyperplane* (batas keputusan) antar kelas yang harus memiliki jarak maksimum antara *support vector* tiap kelas. *Support vector* sendiri merupakan data pada masing-masing kelas yang bertindak sebagai batas dari kelas tersebut.



Gambar III.1 Klasifikasi Menggunakan SVM [16]

Gambar III.1 menunjukkan beberapa titik data yang merupakan anggota dari 2 kelas, anggota dari kelas pertama disimbolkan dengan lingkaran berwarna biru, dan anggota dari kelas kedua disimbolkan dengan lingkaran berwarna merah. Meggunakan Gambar III.1 sebagai contoh, persoalan klasifikasi dapat diartikan sebagai persoalan untuk menemukan batas yang dapat memisahkan kedua kelas tersebut sehingga apabila sebuah data baru dimasukkan kedalam figur, kelas dari data baru dapat diprediksi berdasarkan letak data tersebut. Terdapat banyak kemungkinan garis batas yang dapat digunakan, pada Gambar III.1 terdapat beberapa contoh alternatif garis batas.

Garis batas terbaik ditentukan dengan mencari titik data terdekat pada kedua kelas dari garis batas tersebut kemudian mengukur batas. Titik data terdekat dengan garis batas disebut dengan *support vector*, dan batas merupakan jarak antara garis batas dengan *support vector*. Menurut algoritma SVM, garis batas yang memiliki batas paling besar merupakan garis batas terbaik untuk persoalan klasifikasi [16]. Pada Gambar III.1, garis tebal berwarna hijau merupakan garis batas terbaik karena memiliki batas yang lebih besar dibanding batas pada garis batas berwarna kuning. Pada sebuah figur 2 dimensi seperti Gambar III.1, SVM akan membentuk sebuah garis batas untuk memisahkan 2 kelas, sedangkan pada figur 3 dimensi, SVM akan membentuk sebuah bidang batas. Apabila data memiliki variabel lebih dari 3, maka figur tidak lagi dapat digambarkan, namun SVM tetap dapat mengklasifikasi data tersebut dengan membentuk sebuah *hyperplane*.

Dengan garis pemisah SVM yang telah dibentuk, maka apabila sebuah data baru yang belum diketahui kelasnya dimasukkan kedalam persamaan SVM tersebut, SVM dapat memberikan prediksi kelas dari data tadi berdasarkan nilai-nilai dari variabel data tersebut, di mana nilai-nilai variabel juga merupakan koordinat proyeksi data. Oleh karena kemampuan ini, SVM digolongkan sebagai teknik pembelajaran mesin secara *supervised*. Pada konteks SVM, SVM menggunakan data yang telah disediakan untuk menentukan posisi garis pemisah yang dapat memisahkan tiap kelas secara optimal, sehingga garis pemisah yang telah dibentuk kemudian dapat digunakan untuk memprediksi kelas dari data-data baru.

Diasumsikan bahwa himpunan data yang memiliki 2 kelas berada didalam matriks *X* dengan ukuran matriks *m* x *n*, *m* merupakan jumlah sampel yang diamati dan *n* merupakan jumlah variabel yang diamati. Tiap sampel pada *X* dinotasikan sebagai x_i yang merupakan vektor baris ke-*i* pada *X* dan berukuran ukuran *n*. Tiap sampel pada *X* diasumsikan merupakan anggota dari salah satu diantara 2 kelas, yaitu kelas positif dan kelas negatif. Sehingga sebuah vektor kolom *Y* berukuran *m* bertindak sebagai label kelas untuk tiap sampel, dengan baris ke-*i* pada *Y* atau dinotasikan sebagai (y_i) dapat memiliki 2 nilai, yaitu +1 dan -1. Apabila y_i bernilai +1, maka sampel x_i merupakan anggota dari kelas positif, begitu pula sebaliknya.

Diasumsikan bahwa data tersebut dapat dipisahkan menggunakan *hyperplane* dengan *n* dimensi, yang didefinisikan sebagai:

$$f(x) = w.x + b = 0 \tag{3.1}$$

w merupakan vektor berdimensi *m* dan *b* merupakan skalar. Parameter *w* dan *b* akan menentukan posisi dan orientasi dari *hyperplane* batas. Persoalan untuk menentukan *hyperplane* batas yang optimum merupakan sebuah persoalan optimasi, di mana *hyperplane* batas yang optimum harus memenuhi 2 persamaan berikut:

$$y_i f(x_i) = y_i(\langle w, x_i \rangle + b) \geq 1, i = 1, ..., m, \text{ dan} \tag{3.2}$$

$$\min_{w,b} = \frac{1}{2}\|w\|^2 \tag{3.3}$$

$\langle w, x \rangle$ menotasikan *inner product* dari vektor w dan vektor x Untuk memecahkan persoalan optimasi pada persamaan (3.3) dengan batasan persamaan (3.2), dapat digunakan teknik *Langrange Multiplier* sehingga persoalan tersebut menjadi:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i(y_i(\langle w, x_i \rangle + b) - 1) \tag{3.4}$$

Dengan α merupakan *Langrangian Multiplier*.

Pada keadaan riil, sering kali data tidak dapat diklasifikasikan secara sempurna, sehingga persamaan optimasi menjadi tidak dapat dipenuhi. Untuk mengatasi masalah ini, digunakan teknik *soft margin* dan persamaan ditulis kembali dengan menyisipkan variabel *slack* (ξ_i) [16], sehingga persamaan optimasi menjadi:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \tag{3.5}$$

$$L(w,b,\xi,\alpha,\beta) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i(y_i(\langle w,x_i\rangle + b) - 1 + \xi_i) - \sum_{i=1}^{m}\beta_i\xi_i \tag{3.6}$$

Dengan C disebut sebagai *error penalty*, serta α dan β merupakan *Langrangian Multiplier*. Dipandang persamaan minimalisasi sebagai *primal problem*.

$$\min_{w,b,\xi} \theta_p(w) = \min_{w,b,\xi}\max_{\alpha,\beta} L(w,b,\xi,\alpha,\beta) \tag{3.7}$$

Ketika memenuhi kondisi Kuhn-Tucker, maka persoalan primal dapat ditransformasikan menjadi bentuk *dual problem*:

$$\max_{\alpha,\beta} \theta_d(\alpha,\beta) = \max_{\alpha,\beta}\min_{w,b,\xi} L(w,b,\xi,\alpha,\beta) \tag{3.8}$$

Sehingga minimalisasi dari L dapat dilakukan dengan menyesuaikan nilai dari w, b, ξ. Pada nilai optimal, turunan dari L akan bernilai 0, sehingga:

$$\frac{\partial L}{\partial w} = 0, \quad \rightarrow \sum_{i=1}^{m}\alpha_i\beta_i = 0 \tag{3.9}$$

$$\frac{\partial L}{\partial b} = 0, \quad \rightarrow \sum_{i=1}^{m}\alpha_i\beta_i x_i = 0 \tag{3.10}$$

$$\frac{\partial L}{\partial b} = 0, \quad \rightarrow \alpha_i + \beta_i = C \tag{3.11}$$

Dengan memasukkan pesamaan (3.9), (3.10), (3.11) kedalam persamaan (3.6), maka dapatkan *dual quadratic optimization problem:*

$$\max_{\alpha} w(\alpha) = \max_{\alpha} \left\{ -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j \beta_i \beta_j \langle x_i, x_j \rangle + \sum_{k=1}^{m} \alpha_k \right\} \quad (3.12)$$

Yang memenuhi batasan:

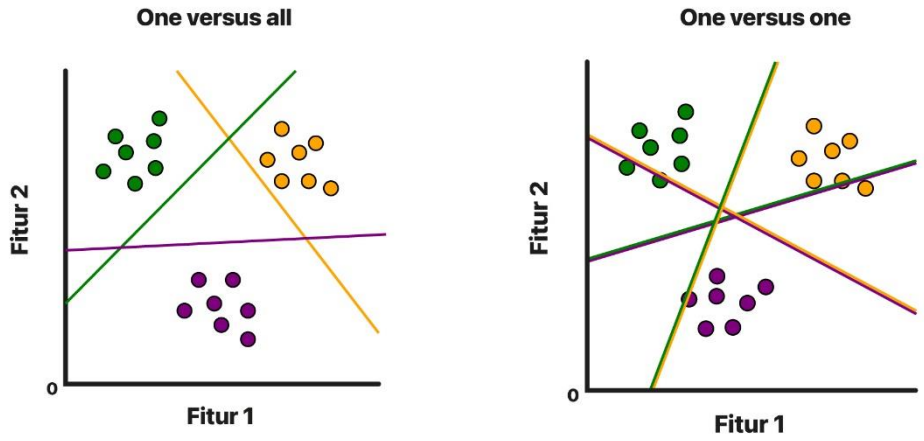
$$0 \leq \alpha_i \leq C, i = 1, \dots, m.$$
$$\sum_{i=1}^{m} \alpha_i \beta_i = 0 \quad (3.13)$$

Dengan menyelesaikan persoalan optimasi (3.12), maka nilai α_i akan didapatkan. Sehingga apabila persamaan (3.10) dapat disisipkan kedalam persamaan (3.1) menjadi [16]:

$$f(x) = w.x + b = \sum_{i=1}^{m} \alpha_i \beta_i \langle x_i, x_j \rangle + b \quad (3.14)$$

Persamaan konstruksi SVM diatas digunakan untuk membentuk garis batas yang mampu mengklasifikasikan himpunan data ke dalam 2 kelas. Pada persoalan yang membutuhkan klasifikasi ke dalam lebih dari 2 kelas, terdapat 2 jenis metode *multi-class* SVM yang dapat digunakan, yaitu metode yang menggunakan dan mengkombinasikan beberapa SVM biner untuk tiap pasangan kelas serta metode yang secara langsung mempertimbangkan semua data di tiap kelas. Sehingga pada SVM multi-kelas, metode yang digunakan antara membutuhkan konstruksi *hyperplane* yang jumlahnya berbanding lurus dengan jumlah kelas, atau dengan memformulasikan problematika optimasi yang lebih rumit. Terdapat banyak metode SVM multi-kelas yang berbasis klasifikasi biner, 3 diantaranya adalah metode *one vs one*, *one vs all*, dan DAGSVM. Eksperimen yang dilakukan oleh Hsu dan Lin [17] menunjukkan bahwa kombinasi klasifikasi biner, khususnya *one*

vs one dan *directed acyclic graph support vector machine* (DAGSVM), yang merupakan metode yang lebih cocok untuk penggunaan praktis dibandingkan metode-metode lain.



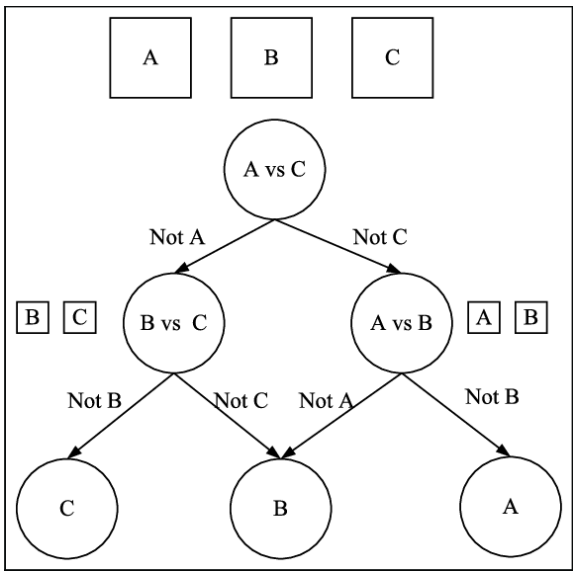
Gambar III.2 Perbedaan one versus all dan one versus one [17]

Pada Gambar III.2 diilustrasikan perbedaan metode *one versus all* dan *one versus one*. Metode *one vs one* akan membentuk *hyperplane* batas sebanyak $c((c-1)/2)$, dengan c merupakan jumlah kelas dan setiap *hyperplane* batas akan memisahkan tiap pasangan kelas. Setelah semua *hyperplane* dikonstruksi, penentuan kelas dari sebuah sampel bisa dilakukan dengan strategi voting atau disebut juga strategi *Max Wins* [17]. Apabila sebuah *hyperplane* menentukan sampel merupakan anggota dari kelas *i*, maka jumlah vote untuk kelas *i* bertambah 1, dan dilakukan sampai semua *hyperplane* mengklasifikasi sampel tersebut.

Berbeda dengan metode *one vs one*, metode *one vs all* akan membentuk *hyperplane* batas sebanyak *c*. Metode ini akan mengkonstruksi satu *hyperplane* untuk tiap kelas yang memisahkan kelas tersebut dengan kelas lainnya, sehingga *hyperplane* untuk kelas c_i memiliki hasil klasifikasi c_i atau *not* c_i. Penentuan kelas dari sebuah sampel dapat ditentukan dari hasil klasifikasi yang menyatakan bahwa sampel merupakan anggota dari kelas tersebut.

DAGSVM mirip dengan metode *one vs one* di mana dikonstruksi *hyperplane* untuk tiap pasangan kelas dengan jumlah $c((c-1)/2)$ *hyperplane*, namun DAGSVM memiliki sistem penentuan kelas yang berbeda. Penentuan kelas pada DAGSVM menggunakan *binary directed acyclic graph* (DAG) yang memiliki *node* sejumlah $c((c+1)/2)$ dan c tingkat, sehingga penentuan kelas dari sebuah sampel dilakukan secara bertingkat dengan mempertimbangkan hasil klasifikasi dari *node* sebelumnya. Pada DAG, node pada tingkat ke-c merupakan hasil klasifikasi dan *node* pada tingkat sebelumnya merupakan fungsi klasifikasi. Dengan demikian, DAGSVM mengimplimentasikan teori DAG kedalam klasifikasi SVM, dan mengimplementasikan simplifikasi *one vs one* SVM, sehingga menjaga efisiensi komputasi yang tinggi serta memiliki akurasi klasifikasi yang tinggi pula [18].

Seperti ditunjukan pada Gambar III.3, DAGSVM yang digunakan untuk mengklasifikasi data kedalam 3 kelas akan membentuk DAG dengan 6 *node* dan 3



Gambar III.3 Penentuan Kelas Menggunakan DAG [13]

tingkat. Tiap *node* menghasilkan 2 keluaran klasifikasi yang menyatakan bahwa sebuah sampel bukan anggota dari c_i, dan diteruskan ke *node* yang sesuai untuk diklasifikasi lagi. Hasil dari penentuan kelas sebuah sampel ditunjukkan oleh *node* pada tingkat terakhir pada DAG.

III.3. Metode Pengelompokan

Analisis pengelompokan merupakan teknik pembelajaran mesin secara *unsupervised* yang berguna untuk mengelompokkan data-data yang mirip dari sebuah kumpulan data ke dalam kelompok-kelompok. Pengelompokan oleh metode pengelompokan dapat berbasis densitas, jarak, maupun properti lain.

III.3.1. Pengelompokan K-means

Proses k-*means* merupakan proses yang membagi sebuah populasi data kedalam k kelompok berbasis sebuah sampel. Metode k-*means* dapat diprogram dengan mudah sehingga memudahkan untuk mengolah data yang besar [19]. Aplikasi proses k-*means* pada pengelompokan berdasarkan kemiripan disebut sebagai pengelompokan k-*means*, yang merupakan metode pengelompokan yang banyak digunakan. Pengelompokan k-*means* bertujuan untuk mencari nilai minimal dari rerata jarak kuadrat antara pusat kelompok dengan sebuah data pada kelompok tersebut atau disebut sebagai *within cluster sum of squares* (WCSS). Pada pengelompokan k-*means*, ditinjau konstanta *k* sebagai jumlah kelompok yang telah ditentukan, matriks *X* sebagai kumpulan data berukuran matriks *m* x *n* dengan fitur sebanyak *m* dan sampel sejumlah *n*, maka pengelompokan k-*means* akan memiliki *objective function (*ϕ*)* sebagai berikut [20]:

$$\phi = \sum_{j=1}^{k} \sum_{i=1}^{m} \left\| x_i^{(j)} - c_j \right\|^2 \tag{3.15}$$

Dengan c_j merupakan pusat kelompok ke-*j* dan $x_i^{(j)}$ merupakan sebuah sampel dari *X* yang termasuk kedalam c_j. Dari persamaan tersebut, diartikan bahwa J merupakan jumlah dari jarak kuadrat antara tiap pusat kelompok ke tiap sampel yang termasuk kedalam kelompok tersebut, dan pengelompokan k-*means* bertujuan untuk mencari nilai minimum dari ϕ. Algoritma dari pengelompokan k-*means* adalah sebagai berikut [20]:

1. Dipilih pusat kelompok (*c*) sebanyak *k* secara acak dari sampel-sampel pada kumpulan data yang disediakan.
2. Untuk tiap $j \in \{1, 2, .. k\}$ dan $i \in \{1, 2, .. m\}$, golongkan tiap sampel (x_i) dari *X* sebagai anggota dari kelompok ke-*j* (C_j) apabila x_i lebih dekat ke c_j daripada pusat kelompok lain.
3. Untuk tiap $j \in \{1, 2, .. k\}$, tetapkan c_j sebagai mean dari C_j.

4. Ulangi tahap 2 dan 3 hingga *C* tidak berubah.

Dengan algoritma yang iteratif seperti yang telah disebutkan, pengelompokan k-*means* akan mencari posisi untuk tiap c_j yang menghasilkan ϕ yang minimum.

Salah satu kesulitan dalam penggunaan metode pengelompokan adalah tidak ada jawaban pasti mengenai jumlah kelompok yang tepat. Seperti yang telah disebutkan sebelumnya, pada pengelompokan k-*means*, nilai *k* haruslah ditentukan terlebih dahulu sejak sebelum memulai algoritma. Untuk mengatasi hal tersebut, terdapat berbagai metode untuk mencari jumlah kelompok yang optimal, seperti *within cluster sum of squares* (WCSS).

$$WCSS = \sum_{j=1}^{k} \sum_{i=1}^{m} \left\| x_i^{(j)} - c_j \right\|^2 \tag{3.16}$$

Karena *objective function* (ϕ) dari pengelompokan k-*means* adalah untuk mencari nilai terkecil dari WCSS, maka temtu WCSS memiliki rumus yang sama dengan ϕ. WCSS dapat digunakan untuk mencari jumlah kelompok yang optimal dengan metode yang disebut dengan *elbow method*. *Elbow method* merupakan sebuah strategi sehingga penentuan jumlah kelompok optimal berdasarkan pembentukan siku pada grafik WCSS. Untuk menentukan siku pada sebuah grafik WCSS, maka dihitung WCSS dari sebuah kumpulan data menggunakan sebuah rentang nilai k. Kumpulan nilai WCSS tersebut kemudian digambarkan dalam sebuah grafik kurva, dan berdasarkan *elbow* method, jumlah kelompok optimal merupakan nilai *k* di mana kurva membentuk sebuah siku (*elbow*) [21]. Pembentukan siku pada kurva WCSS menandakan bahwa terjadi penurunan nilai WCSS yang relatif drastis pada WCSS dengan jumlah kelompok sebelumnya dibandingkan nilai WCSS dengan jumlah kelompok tersebut. Penurunan nilai WCSS diasumsikan yang drastis kemudian diasumsikan sebagai bukti bahwa jumlah kelompok optimum.

Meskipun pengelompokan k-*means* merupakan metode yang populer digunakan dalam berbagai aplikasi, namun yang membuat pengelompokan k-*means* sebagai metode yang populer adalah kecepatan dan kesederhanaannya, bukan akurasi [20]. Karena pada tahap 1 pusat kelompok pertama kali ditetapkan

secara random, maka untuk tiap percobaan pada sebuah kumpulan data yang sama, pengelompokan k-*means* dapat menghasilkan pengelompokan yang berbeda dengan akurasi yang berbeda pula. Penelitian [20] memperkenalkan metode di mana posisi pusat kelompok pertama tidak ditentukan secara acak, namun dengan iterasi tersendiri, metode tersebut disebut sebagai k-*means*++. Pada k-*means*++, D(x) merupakan jarak dari sebuah sampel menuju pusat kelompok terdekat yang telah ditentukan pada iterasi sebelumnya. Setelah pusat kelompok pertama dipilih dari sampel yang tersedia secara acak, pusat kelompok berikutnya dipilih dari sampel yang tersedia dengan probabilitas yang mana pemilihan sebuah sampel sebagai pusat kelompok proporsional terhadap jarak sampel tersebut menuju pusat kelompok terdekat, sehingga semakin jauh jarak sebuah sampel dari pusat kelompok, semakin mungkin sampel tersebut ditentukan sebagai pusat kelompok berikutnya. K-*means*++ memiliki algoritma sebagai berikut [20]:

1a. Tentukan pusat kelompok pertama c_1, dipilih secara acak dari sampel pada *X*.

1b. Tentukan pusat kelompok berikutnya c_i, yang dipilih secara acak dari sampel pada *X* dengan probabilitas $\frac{D(x)}{\sum_{x \epsilon X} D(x)^2}$

1c. Ulangi tahap 1b, hingga c_k telah ditentukan

2-4. Dijalankan sama seperti algortima k-*means*.

III.3.2. Density-based Spatial Clustering of Applications With Noise

Density-based Spatial Clustering of Applications With Noise (DBSCAN) melakukan penggolongan kumpulan data berdasarkan densitas dari proyeksi tiap sampel pada kumpulan data tersebut, sehingga sebagai hasil dari DBSCAN, tiap kelompok akan memiliki densitas yang berbeda dengan kelompok lainnya, strategi tersebut tentu saja berbeda dengan pengelompokan k-*means* yang melakukan penggolongan berdasarkan pusat kelompok terdekat.

DBSCAN dikembangkan oleh Martin Ester, Hans-Peter Kriegel, Jörg Sander, dan Xiaowei Xu sebagai solusi yang menjawab kebutuhan-kebutuhan dari aplikasi metode pengelompokan pada basis data spasial yang besar, yakni [22]:

1. Membutuhkan pengetahuan domain yang minimum dalam penentuan nilai parameter yang sesuai, karena perkiraan nilai parameter seringkali tidak diketahui

ketika berhadapan dengan sebuah basis data yang besar. DBSCAN hanya memerlukan 1 parameter yang perlu disetel, sehingga mudah bagi pengguna untuk menyesuaikan nilai parameter tersebut agar sesuai dengan karakteristik basis data yang diolah.

2. Dapat melakukan penggolongan dengan bentuk yang acak, dikarenakan bentuk kelompok pada basis data spasial dapat memiliki bentuk yang beragam. Karena DBSCAN melakukan penggolongan berdasarkan densitas, DBSCAN akan melakukan penggolongan dalam bentuk yang beragam untuk menyesuakan bentuk persebaran dari data dengan densitas yang sama.

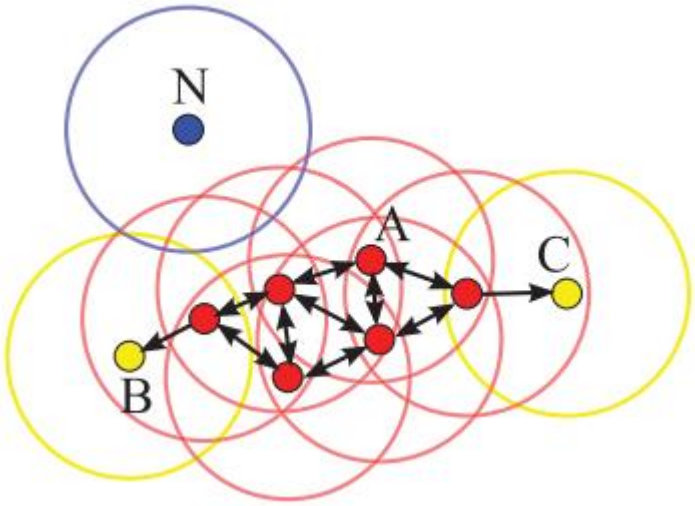
3. Memiliki efisiensi yang baik pada basis data yang besar, DBSCAN dilaporkan memiliki efisiensi yang baik meskipun pada basis data yang besar.

Algoritma dari metode DBSCAN dapat dijelaskan secara abstrak sebagai berikut [23]:

1. Tentukan titik pusat-titik pusat dari kumpulan data yang diolah. Sebuah sampel pada kumpulan data dianggap sebagai titik pusat apabila pada radius sebesar ε disekitar sampel tersebut terdapat sampel lain dengan jumlah yang melampaui ambang yang telah ditentukan. Titik pusat dan sampel yang berada didalam radius akan membentuk 1 kelompok kecil.

2. Gabungkan semua titik pusat-titik pusat yang terdapat pada radius satu sama lain ke dalam 1 kelompok yang lebih besar.

3. Untuk sampel yang bukan merupakan titik pusat dan tidak terdapat di dalam radius dari sebuah titik pusat, maka sampel tersebut dianggap sebagai derau, namun apabila sampel tersebut terdapat didalam radius dari sebuah titik pusat, maka sampel tersebut dianggap sebagai titik batas.



Gambar III.4 Ilustrasi cara kerja metode DBSCAN [23]

III.4. Kernel

Metode kernel adalah sebuah metode untuk memproyeksikan data ke ruang fitur berdimensi tinggi tanpa perlu mengkalkulasikan vektor fitur pada ruang tersebut secara eksplisit, melainkan dengan menghitung *inner product* dari tiap pasangan data pada ruang fitur semula, sehingga komputasi vektor fitur dapat dilakukan dengan lebih efisien [24]. Metode kernel dilakukan dengan cara mencari *inner product* dari pasangan sampel, atau secara umum dirumuskan dengan:

$$\widetilde{K}(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \tag{3.17}$$

Dengan x_i dan x_j merupakan sampel *X* ke-*i* dan ke-*j* dengan *i*,*j* = 1, 2, ..., *m*. Dengan demikian, $\widetilde{K}$ merupakan matriks persegi dengan ukuran *m* x *m*. Apabila $\phi(x_i)$ berdekatan dengan $\phi(x_j)$, maka nilai $\widetilde{K}$ akan relatif besar, sedangkan apabila berjauhan atau hampir ortogonal, maka nilai $\widetilde{K}$ akan relatif kecil. Sehingga dapat

dikatakan bahwa $\widetilde{K}$ merupakan kumpulan ukuran mengenai kemiripan tiap pasangan sampel.

Terdapat beberapa rumus jenis *mapping* kernel yang dapat digunakan, beberapa diantaranya adalah:

1. Fungsi Basis Radial (FBR) kernel:

$$\widetilde{K}(x_i, x_j) = \exp\left(\frac{\|x_i, -x_j\|^2}{2\sigma^2}\right) \tag{3.18}$$

2. Polynomial kernel:

$$\widetilde{K}(x_i, x_j) = \left(x_i{}^T x_j + a\right)^d \tag{3.19}$$

Dengan *d* merupakan derajat dari polinomial dan a ≥ 0 merupakan parameter *trade-off* antara pengaruh derajat tinggi dengan derajat rendah. Nilai σ merupakan deviasi standar dari kernel FBR yang merupakan sebuah fungsi gaussian yang digunakan sebagai ukuran kemiripan dari sebuah data. Invers dari σ adalah γ, nilai γ yang kecil mendefinisikan sebuah fungsi *gaussian* yang memiliki varians yang besar, sehingga 2 data akan disebut sebagai data yang mirip meskipun kedua data tersebut diproyeksikan secara terpisah.

III.5. Kernel Fisher Discriminant Analysis

Reduksi dimensi merupakan metode untuk memproyeksikan data ke ruang fitur dengan dimensi yang lebih kecil dibandingkan ruang fitur semula, atau dalam kata lain, reduksi dimensi digunakan untuk mengurangi jumlah fitur (dimensi) pada sebuah himpunan data. Ruang fitur sendiri merupakan sebutan bagi sebuah ruang di mana data diproyeksikan menurut nilai-nilai variabelnya. Metode ini penting karena dapat memfasilitasi visualisasi dalam sebuah figur (2 dimensi atau 3 dimensi), mengurangi waktu yang dibutuhkan untuk komputasi *classifier*, serta meningkatkan akurasi klasifikasi [25]. Terdapat 2 metode yang sering digunakan untuk melakukan reduksi dimensi, yaitu seleksi fitur dan ekstraksi fitur. Seleksi fitur mengurangi jumlah variabel dengan cara mengeliminasi variabel-variabel

yang tidak relevan, umumnya dengan mengeliminasi variabel yang memiliki nilai variansi dibawah ambang yang telah ditentukan. Berbeda dari seleksi fitur, ekstraksi fitur membentuk variabel-variabel baru sehingga data diproyeksikan dengan variansi sebesar mungkin pada ruang fitur baru. Variabel baru yang dibentuk merupakan kombinasi liner dari variabel-variabel semula [2]. Dengan memperhatikan variansi, kedua metode menjaga agar tidak banyak informasi yang hilang ketika digunakan ruang fitur yang berdimensi lebih rendah.

Salah satu metode ekstraksi fitur yang sering digunakan adalah *fisher discriminant analysis*, metode ini banyak digunakan dan terbukti memiliki performa baik dalam berbagai aplilkasi. Namun, karena batasannya pada persoalan-persoalan linier, FDA memiliki performa yang buruk pada data yang tidak dapat dipisahkan secara linier [26]. Beberapa peneliti kemudian mengembangkan FDA yang dapat memiliki performa baik pada data yang tidak dapat dipisahkan secara linier, salah satunya adalah dengan menggunakan metode kernel.

Yang [12] pada penelitiannya merumuskan KFDA sebagai metode 2 tahap, yaitu *kernel principal component analysis* (KPCA) dan kemudian dilanjutkan dengan FDA. Diasumsikan sebuah himpunan data bernotasi *X* dengan *n* jumlah variabel, maka pada tahap KPCA, dicari ruang fitur kernel dari himpunan data menggunakan persamaan dari metode kernel yang digunakan.

Matriks transformasi KPCA dapat dihitung dengan persamaan:

$$z_{KPCA} = \left(\frac{\nu_1}{\sqrt{\lambda_1}}, \dots, \frac{\nu_n}{\sqrt{\lambda_n}}\right)^T \tag{3.20}$$

Dengan ν dan λ merupakan *eigenvalue* dan *eigenvector* dari K, dan K merupakan matriks $\widetilde{K}$ yang terpusat. Menggunakan perkalian *dot product* dengan z_{KPCA}, *X* dapat diproyeksikan kedalam ruang fitur baru dan dinotasikan sebagai $\hat{X}$, yang mana $\hat{X}$ kemudian digunakan untuk membentuk matriks transformasi KFDA (z_{KFDA}) pada tahap kedua, yaitu tahap FDA. z_{KFDA} didapatkan menerapkan FDA pada z_{KPCA} sehingga didapatkan *eigenvector* (φ) dari $\langle S^{-1}, Cov \rangle$, dengan *Cov*

merupakan matriks kovarians dari mean antar kelas dan *S* merupakan variansi dari $\hat{X}$ pada tiap vektor fitur pada ruang fitur z_{KPCA}.

$$Cov = \frac{1}{m}\sum_{i=1}^{c} l_i(M_{c_i} - M_{\hat{X}})(M_{c_i} - M_{\hat{X}})^T \quad (3.21)$$

$$S = diag(\lambda_1, \lambda_2, \dots, \lambda_m) \quad (3.22)$$

Dengan *m* menotasikan jumlah sampel pada *X*, *c* menotasikan jumlah kelas, l_i menotasikan jumlah sampel pada kelas ke-*i*, M_{c_i} menotasikan mean dari kelas ke-*i*,dan M_0 menotasikan mean dari $\hat{X}$. Sebuah data baru bernotasi *x* dapat diproyeksikan pada ruang fitur KFDA dengan persamaan:

$$\check{x} = {z_{KFDA}}^T.({z_{KPCA}}^T.x) \quad (3.23)$$

Jumlah dimensi dari ruang fitur KFDA memiliki jumlah yang sama dengan jumlah *eigenvector* (φ) yang digunakan.

III.6. Penskalaan

Penskalaan merupakan metode yang biasanya dilakukan pada tahap pra-proses data. Dengan menggunakan penskalaan, maka nilai tiap variabel pada sebuah data diubah menjadi memiliki rentang yang sama sehingga semua variabel memiliki proporsi yang sama. Penskalaan dilakukan pada kumpulan data yang memiliki rentang fitur yang beragam, apabila pada sebuah kumpulan data tidak dilakukan penskalaan fitur, maka fitur yang berentang pada bilangan lebih besar akan mendominasi fitur yang memiliki rentang pada bilangan lebih kecil [27]. Dengan melakukan penskalaan, semua fitur akan memiliki rentang yang sama pada bilangan yang sama, sehingga tidak ada fitur yang mendominasi.

Diasumsikan himpunan data *X* yang memiliki *m* sampel dan *n* variabel, variabel pada *X* dinotasikan dengan *f*, dengan f_i merupakan vektor berukuran *m*. Hasil rentang nilai serta satuan-satuan yang digunakan bergantung pada penskalaan yang digunakan. Beberapa contoh dan persamaan dari *scaler*, adalah:

- *Standard Scaler*

Standard scaler cocok untuk data yang terdistribusi normal, namun tidak dianjurkan untuk data yang tidak terdistribusi normal. *Standard scaler* mengubah nilai tiap variabel menjadi memiliki mean 0 dan standar deviasi 1, sehingga nilai memiliki rentang -1 sampai 1. *Standard scaler* dapat digunakan dengan persamaan [27]:

$$x_i\ terskala = \frac{x_i - \overline{f_i}}{\sigma} \tag{3.24}$$

- *Max-min Scaler*

  Max-min scaler dapat digunakan apabila data tidak terdistribusi normal. *Max-min Scaler* mengubah nilai tiap variabel menjadi berentang 0 sampai 1, atau -1 atau 1 apabila terdapat nilai negatif. Namun *max-min scaler* sensitif terhadap *outlier*, sehingga apabila terdapat *outlier* pada data, disarankan menghilangkan data *outlier* tersebut atau menggunakan *scaler* lain. *Max-min scaler* dapat digunakan dengan persamaan [27]:

$$x_i\ terskala = \frac{x_i - \min(f_i)}{\max(f_i) - \min(f_i)} \tag{3.25}$$

- Robust Scaler

  Robust scaler mengubah data dengan cara sama dengan *Max-Min scaler*, tetapi *robust scaler* tidak sensitif terhadap *outlier*. Alih alih menggunakan nilai maksimal dan minimum, *robust scaler* menggunakan jangkauan antar kuartil sehingga persamaannya menjadi [28]:

$$x_i\ terskala = \frac{x_i - Q_1(f_i)}{Q_3(f_i) - Q_1(f_i)} \tag{3.26}$$

III.7. Deteksi dan Diagnosis Kesalahan

Kesalahan didefinisikan sebagai perilaku abnormal pada proses yang berhubungan dengan kegagalan mesin, kelelahan mesin, atau gangguan eksrim pada proses [1]. Apabila kesalahan terjadi dan tidak segera ditangani, kualitas produk atau bahkan juga keselamatan proses menjadi terancam. Kesalahan pada salah satu bagian pada proses apabila tidak segera dideteksi dan ditangani, dapat mengakibatkan kerugian ekonomi pada perusahaan berupa produk yang berkualitas rendah, biaya perbaikan peralatan yang besar, serta dapat membahayakan keselamatan operator. Mempertimbangkan hal – hal tersebut, perusahaan harus mencari cara untuk mendeteksi dan mencari sumber kesalahan sedini mungkin, sehingga metode deteksi kesalahan dikembangkan selama 20 tahun terakhir [3].

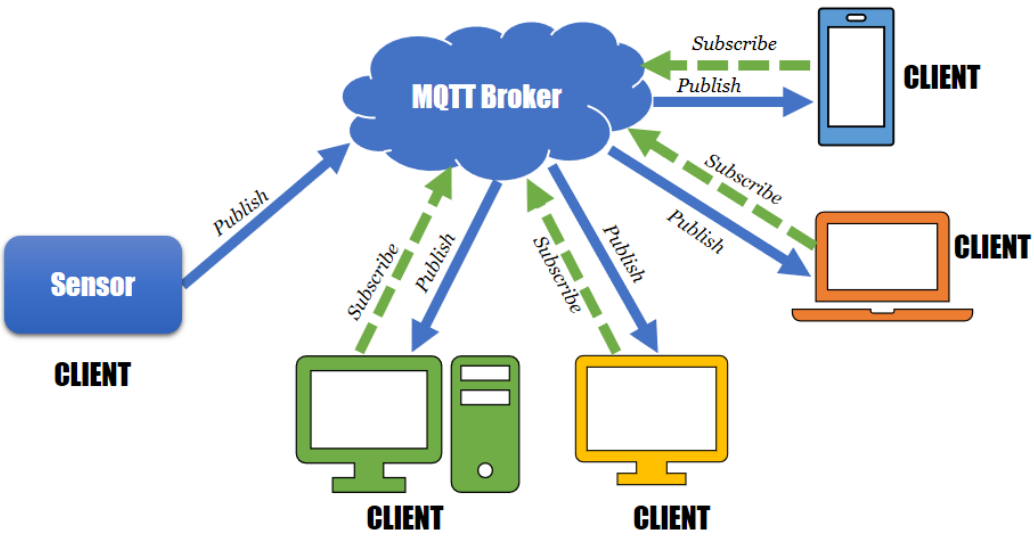
Deteksi dan diagnosis kesalahan merupakan upaya untuk meningkatkan keamanan dan kehandalan sebuah sistem dengan cara mendeteksi perilaku sistem yang abnormal. Keselamatan proses pada pabrik bergantung pada kondisi mesin-mesin pabrik yang digunakan, apabila terdapat kesalahan pada mesin, proses pabrik akan mengalami gangguan dan dapat mengancam kualitas produk, kondisi peralatan mesin, serta keselamatan operator pada pabrik. Deteksi kesalahan yang dilakukan dengan segera, dapat mendeteksi kesalahan pada kondisi mesin melalui data dari sensor dan aktuator, sehingga perusahaan dapat mengurangi resiko kerugian ekonomi dari biaya perbaikan mesin atau kualitas produk yang rendah.

Deteksi kesalahan dan diagnosis kesalahan merupakan 2 persoalan yang berbeda, yang mana deteksi kesalahan merupakan persoalan untuk mendeteksi data salah dari sebuah sistem, sedangkan diagnosis kesalahan merupakan persoalan untuk mengidentifikasi jenis kesalahan berdasarkan karakteristik kesalahan tersebut. Kedua persoalan ini dapat dipecahkan dengan metode klasifikasi sehingga deteksi kesalahan merupakan klasifikas 2 kelas sedangkan diagnosis kesalahan merupakan klasifikasi multi-kelas. Pada prakteknya, deteksi kesalahan dan diagnosis kesalahan dapat dilakukan secara bersamaan sebagai persoalan klasifikasi multi-kelas.

Terdapat 2 pendekatan FDD yang dapat digunakan, yakni pendekatan *data-driven* dan pendekatan *model-based*. Pendekatan *model-based* membutuhkan pengetahuan mengenai model proses dalam bentuk persamaan matematis, berdasarkan persamaan matematis tersebut, ketergantungan antara variabel yang dapat diukur [29] dan batas-batas kesalahan dapat dideteksi. Berbeda halnya dengan pendekatan *data-driven* di mana batas-batas kesalahan dibentuk tidak melalui pengetahuan mengenai model matematis proses, melainkan menggunakan data deret waktu yang dihimpun selama proses dijalankan [2].

Pada pendekatan *data-driven*, diasumsikan bahwa data historis proses yang berjumlah besar telah tersedia, baik data kondisi normal maupun data salah [3]. Data historis proses merupakan himpunan data deret waktu yang didapatkan dari nilai keluaran sensor dan aktuator. Data historis proses tersebut kemudian digunakan untuk membuat model klasifikasi, sehingga apabila data baru dimasukkan kedalam model klasifikasi, maka didapatkan sebuah keluaran berupa label dari data baru tersebut.

III.8. *Message Queue Telemetry Transport*



Gambar III.5 Skema Kerja MQTT [20]

Protokol *Message Queue Telemetry Transport* (MQTT) merupakan protocol pesan yang ringan (*lightweight*) dengan arsitektur *publish*/*subscribe* yang

digunakan diatas protokol komunikasi TCP/IP. Dengan arsitektur tersebut, MQTT didesain untuk bersifat terbuka dan mudah untuk diaplikasikan, serta dapat menangani ribuan *client* hanya dengan menggunakan 1 server. Karakteristik tersebut membuat MQTT ideal untuk digunakan dalam kondisi sulit, seperti apabila *bandwidth* jaringan rendah atau dengan perangkat yang memiliki kapabilitas komputasi dan memori yang terbatas [30]. Protokol MQTT memudahkan komunikasi *machine-to-machine*, yaitu komunikasi antar perangkat yang menjadi semakin umum dengan berkembangnya teknologi.

Untuk komunikasi dengan protokol MQTT dapat dilakukan, maka dibutuhkan perangkat sebagai MQTT *client* dan MQTT *broker*. MQTT *client* merupakan perangkat yang terhubung dengan server pengiriman pesan, dan menggunakan *topic* untuk mem-*publish* pesan sehingga *client* lain menerima pesan tersebut. MQTT *client* juga dapat *subscribe* pada sebuah *topic* sehingga dapat menerima pesan apabila sebuah informasi telah di-*publish* pada *topic* tersebut. MQTT *broker* merupakan bertindak sebagai sebuah *server* yang mengimplementasikan protokol MQTT sehingga mampu memfasilitasi komunikasi antar berbagai MQTT *client*.

BAB IV
PELAKSANAAN PENELITIAN

IV.1. Alat dan Bahan Penelitian

IV.1.1. Alat Penelitian

Tabel IV.1. Perangkat Keras Penelitian

Perangkat Keras	
Nama Alat	Spesifikasi
Komputer *Desktop*	OS: Windows 10 Pro Processor: Intel Core i5-4460 @3.20 GHz RAM: 12 GB

Tabel IV.2 Perangkat Lunak Penelitian

Perangkat Lunak	
Nama Alat	Versi
Visual Studio Code	1.47
Eclipse Mosquitto	3.1.1
Python	3.8.6
Eclipse Paho MQTT	1.5.1
Scikit-learn	0.23.2
Numpy	1.19.0
Pandas	1.1.3

Visual Studio Code merupakan perangkat lunak *code editor* yang bersifat *open source* dan lintas platform milik Microsoft. Visual Studio Code memiliki beragam fitur yang memudahkan penulis untuk membuat dan menyunting *code*, seperti fitur IntelliSense yang dapat membantu pembuatan *code*, *bracket-matching* yang dapat menyeimbangkan tanda kurung, *auto-identation*, *syntax highlighting* serta banyak fitur lain. Karena bersifat *open-source*, pengguna juga dapat mengunduh *extension* buatan pengguna lain untuk mengoptimasi dan personalisasi

Visual Studio Code. Pada penelitian ini, Visual Studio Code digunakan untuk membuat dan menyunting program untuk mengolah data dengan bahasa pemrograman *Python*.

Perangkat lunak yang digunakan untuk mengelola salah satu Port pada komputer personal sebagai MQTT Broker sehingga dapat membuka komunikasi antar MQTT Client.

Bahasa pemrograman Python diciptakan oleh Guido Van Rossun pada tahun 1989 sebagai proyek *open-source*, yang berarti semua orang dapat berkontribusi untuk memperkaya kemampuan dari Python dalam bentuk *modules* dan *packages*. Sebagai proyek *open source*, Python mendukung *modules* dan *packages*, sehingga seseorang dapat menggunakan program yang telah dibuat orang lain.

Pada penelitian ini, digunakan beberapa pustaka dengan fungsi masing-masing, yakni Eclipse Paho MQTT, scikit-learn, Numpy, Pandas. Eclipse Paho MQTT memungkinkan program yang dibangun untuk terhubung dengan MQTT Broker sebagai MQTT Client, sehingga dapat mengirimkan dan menerima informasi pada atau dari program lain yang juga terhubung pada MQTT Broker. Pada penelitian ini, digunakan 2 program sebagai MQTT Client, 1 untuk mengirimkan data dan 1 untuk menerima dan mengolah data.

Pustaka scikit-learn merupakan pustaka untuk mengolah data menggunakan model-model pembelajaran mesin dan algoritma pendukungnya. Dengan menggunakan pustaka scikit-learn, pengolahan data dapat dengan mudah dilakukan karena scikit-learn memiliki berbagai model pembelajaran mesin untuk berbagai keperluan, seperti klasifikasi, regresi, dan pengelompokan. Selain itu, pengguna juga dapat mengatur berbagai parameter dari model yang digunakan. Didalam scikit-learn juga terdapat berbagai algoritma reduksi dimensi, seleksi model, dan pra-pemrosesan untuk mendukung akurasi dari model pembelajaran mesin yang tersedia. Dengan berbagai algoritma tersebut, selain pengguna dapat mempersiapkan data sebelum diolah oleh model pembelajaran mesin dengan algoritma reduksi dimensi dan pra-pemrosesan, pengguna juga dapat menguji parameter yang optimal menggunakan seleksi model. Pustaka scikit-learn dibangun diatas pustaka Numpy sehingga dapat digunakan bersamaan.

Numpy merupakan pustaka *open-source* untuk kalkulasi ilmiah yang terdapat pada Python. Numpy menawarkan berbagai macam fungsi matematis yang komprehensif sehingga menjadi ketergantungan dari banyak pustaka-pustaka Python. Meski dengan fungsi yang berbagai macam, Numpy dapat digunakan dengan mudah oleh banyak kalangan dikarenakan sintaksnya yang sederhana, serta memiliki kecepatan operasi yang tinggi.

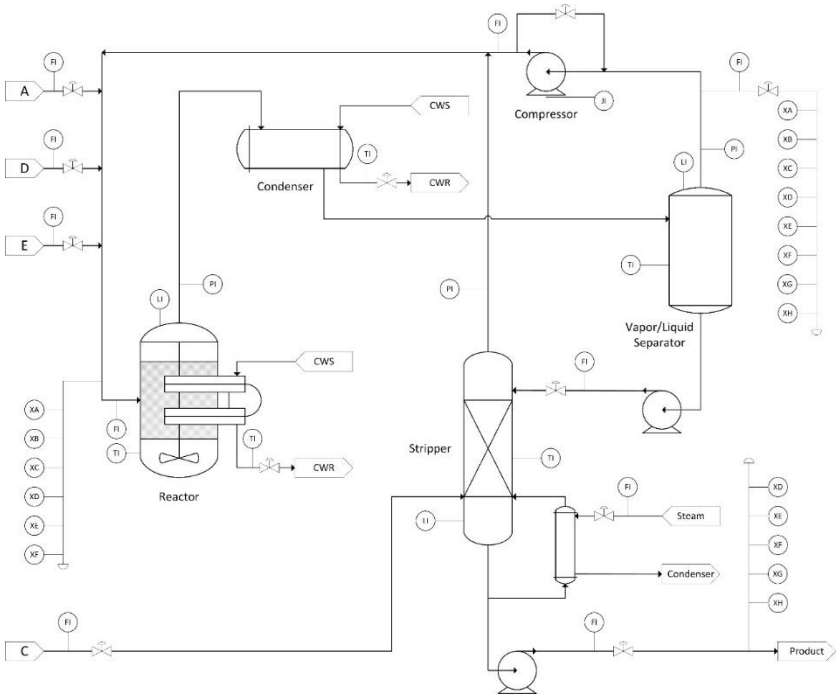
Pandas merupakan pustaka *open-source* pada Python yang digunakan untuk analisis dan manipulasi data. Pandas mampu membaca dan mengubah data-data pada berbagai format seperti CSV, teks, Microsoft Excel, dan database SQL. Pandas memiliki jenis obyek berupa DataFrame yang dapat dimanipulasi dengan mudah, manipulasi yang disebut melingkupi namun tidak terbatas pada: *indexing*, *slicing*, *merging*, *reshaping*, *pivoting*, dan pengolahan data yang tidak lengkap.

IV.1.2. Bahan Penelitian

Digunakan 2 kumpulan data pada penelitian ini, yakni kumpulan data dari Tennessee Eastman Chemical Company dan data dari penelitian Brooks. Kumpulan data dari Tennessee Eastman Chemical Company memiliki 52 variabel dan digunakan untuk membangun program deteksi kesalahan pada penelitian ini, sedangkan kumpulan data dari penelitian Brooks memiliki 20 variabel digunakan sebagai validasi dari program yang sudah dibangun.

1. Data Latih

Data Tennessee Eastman Chemical Company merupakan kumpulan data simulasi dari proses pabrik yang telah dipublikasikan oleh Eastman Chemical Company pada tahun 1991 sehingga dapat digunakan oleh publik pada berbagai topik yang relevan, seperti desain strategi kendali, optimasi, kendali prediktif, edukasi, dan topik-topik lain [31]. Berdasarkan proses pabrik dan data-data yang tertera pada publikasi tersebut, seorang peneliti dapat mensimulasikan proses pabrik Tennessee Eastman dan mengolah data yang didapatkan pada topik yang diinginkan. Pada penelitian ini, kumpulan data dari Tennessee Eastman akan digunakan untuk menentukan parameter-parameter pada program yang akan dirancang.



Gambar IV.1 Diagram P&ID dari Tennessee Eastman Process [31]

Gambar IV.1 merupakan diagram perpipaan dan instrumentrasi dari Tennessee Eastman Chemical Company yang menggambarkan baik alur proses maupun peralatan kendali dari pabrik kimia tersebut. Proses Tennessee Eastman memiliki 5 unit operasi utama, yakni:

1. Reaktor: Reaktan (A, D, E) diumpankan dalam bentuk gas ke dalam reaktor, di mana reaktan kemudian akan bereaksi. Produk akan keluar dari reaktor dalam bentuk uap bersamaan dengan umpan yang tidak bereaksi.

2. Kondensor: Produk dari reaktor masuk ke dalam kondensor yang menyebabkan produk akan dikondensasikan menjadi campuran uap dan cairan.
3. Pemisah (*separator*): Campuran uap dan cairan dipisahkan didalam *separator* menjadi komponen uap dan cairan dan dimasukkan kedalam proses yang berbeda. Produk sampingan (F) dan bahan inert (B) dibersihkan oleh pemisah dalam bentuk uap.
4. Kompresor daur ulang: kompresor akan menerima komponen uap dari separator untuk kemudian dikirimkan kembali kedalam reaktor sebagai umpan.
5. Pemisah (*stripper*): Komponen cairan dari *separator* akan masuk ke *stripper* untuk dipisahkan dari sisa reaktan menggunakan umpan C dan menghasilkan produk (G, H), dengan rasio massa G dan H dari produk ditentukan oleh permintaan dari pasar atau keterbatasan kapasitas.

Tennessee Eastman Chemical Company memiliki 52 variabel atau fitur yang tediri dari 22 variabel proses, 11 variabel yang dimanipulasi, serta 19 variabel dari pengukuran komposisi. Tabel IV.3 berisi nama dan satuan dari tiap variable pada proses Tennessee Eastman Chemical Company.

Tabel IV.3 Variabel pada Tennessee Eastman Process

No	Deskripsi Variabel	Satuan	No	Deskripsi Variabel	Satuan
Variabel Proses					
1	Umpan A (aliran 1)	ksmc h^{-1}	12	Level *seperator*	%
2	Umpan D (aliran 2)	kg h^{-1}	13	Tekanan *seperator*	kPa gauge
3	Umpan E (aliran 3)	kg h^{-1}	14	Laju aliran bawah *separator* (aliran 10)	$m^3 h^{-1}$
4	Total Umpan (aliran 4)	ksmc h^{-1}	15	Level *stripper*	%
5	Aliran daur ulang (aliran 8)	ksmc h^{-1}	16	Tekanan *stripper*	kPa gauge
6	Laju umpan reaktor (aliran 6)	ksmc h^{-1}	17	Laju aliran bawah *stripper* (aliran 11)	$m^3 h^{-1}$
7	Tekanan reaktor	kPa gauge	18	Suhu *stripper*	°C
8	Level reaktor	%	19	Laju aliran uap *stripper*	kg h^{-1}
9	Suhu reaktor	°C	20	Usaha kompresor	kW
10	Laju pembersihan (aliran 9)	ksmc h^{-1}	21	Suhu keluaran pendingin reaktor	°C
11	Suhu *seperator*	°C	22	Suhu keluaran pendingin *separator*	°C
Variable yang dimanipulasi					
23	Katub aliran umpan D (aliran 2)	kg h^{-1}	29	Katub aliran cairan *separator* (aliran 10)	$m^3 h^{-1}$
24	Katub aliran umpan E (aliran 3)	kg h^{-1}	30	Katub aliran cairan *stripper* (aliran 11)	$m^3 h^{-1}$
25	Katub aliran umpan A (aliran 1)	ksmc h^{-1}	31	Katub aliran uap *stripper*	%
26	Katub aliran umpan total (aliran 4)	ksmc h^{-1}	32	Katub pendingin reaktor	$m^3 h^{-1}$
27	Katub kompresor daur ulang	%	33	Katub pendingin kondensor	$m^3 h^{-1}$
28	Katub pembersihan (aliran 9)	%			
Pengukuran Komposisi					
34	Komponen A (aliran 6)	mol%	44	Komponen E (aliran 9)	mol%
35	Komponen B (aliran 6)	mol%	45	Komponen F (aliran 9)	mol%
36	Komponen C (aliran 6)	mol%	46	Komponen G (aliran 9)	mol%
37	Komponen D (aliran 6)	mol%	47	Komponen H (aliran 9)	mol%
38	Komponen E (aliran 6)	mol%	48	Komponen D (aliran 11)	mol%
39	Komponen F (aliran 6)	mol%	49	Komponen E (aliran 11)	mol%
40	Komponen A (aliran 9)	mol%	50	Komponen F (aliran 11)	mol%
41	Komponen B (aliran 9)	mol%	51	Komponen G (aliran 11)	mol%
42	Komponen C (aliran 9)	mol%	52	Komponen H (aliran 11)	mol%
43	Komponen D (aliran 9)	mol%			

Digunakan kumpulan data sebanyak 2420 sampel dengan rincian data normal sebanyak 500 sampel, data salah tipe 2 sebanyak 480 normal, data salah tipe 5 sebanyak 480 sampel, data salah tipe 7 sebanyak 480 sampel, dan data salah tipe 14 sebanyak 480 sampel. Rincian data yang digunakan dapat dilihat pada Tabel IV.4.

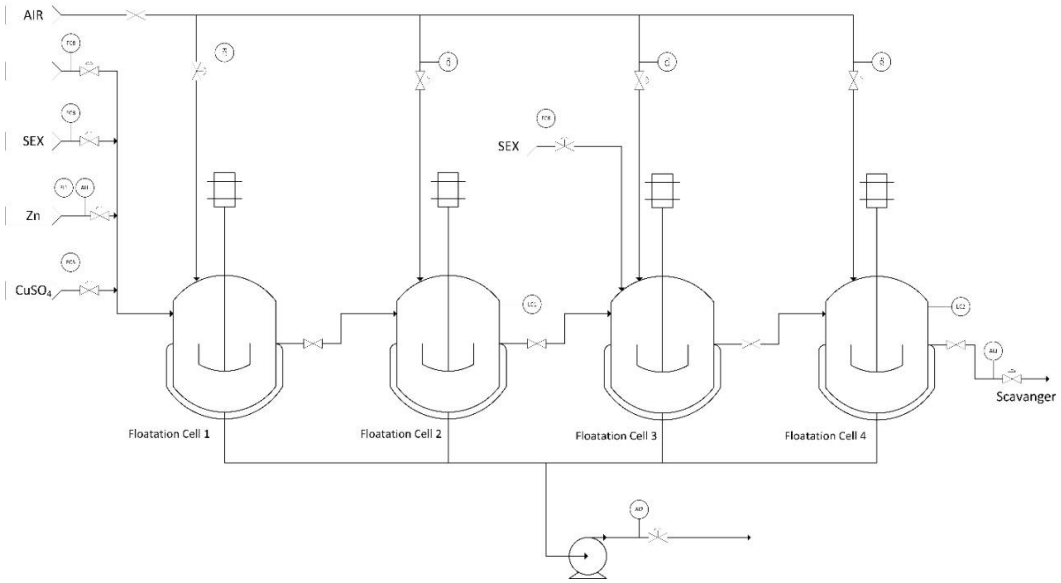
Tabel IV.4 Deskripsi Jenis Data pada Data Latih

Jenis Data	Deskripsi
Data normal	Pabrik beroperasi dalam keadaan standar
Data salah tipe 2	Perubahan komposisi B, rasio A/C konstan
Data salah tipe 5	Perubahan suhu pada masukan pendingin kondensor
Data salah tipe 6	Penurunan aliran umpan A
Data salah tipe 7	Penurunan tekanan umpan C

Data salah tipe 2 hingga tipe 7 didapatkan dengan cara memberikan perubahan berupa *step* pada variabel proses tertentu. Dengan memberikan gangguan berupa perubahan tertentu pada variabel, data yang didapatkan tentu akan berbeda dengan apabila digunakan keadaan normal. Program deteksi kesalahan yang dirancang diharapkan dapat menangkap perbedaan dari data tersebut, dan mengklasifikasikan data dengan sesuai.

2. Data Validasi

Salah satu penelitian yang dilakukan oleh Brooks pada tahun 2018 bertujuan untuk menguji performa metode *data-driven* dari sebuah paket perangkat lunak komersil untuk melakukan deteksi kesalahan dan rekonstruksi data pada sebuah kumpulan data sensor pabrik [32]. Pada penelitian ini, data tersebut akan digunakan untuk menguji program yang telah dibangun berdasarkan data Tennessee Eastman Process untuk melihat apakah program bersifat universal.



Gambar IV.2 Diagram P&ID Data Validasi [32]

Gambar IV.2 merupakan diagram perpipaan dan instrumentasi dari kumpulan data pabrik pada penelitian oleh *Brooks*. Diagram menjelaskan alur proses dan instrumentasi dari pabrik tersebut. Proses terdiri dari 4 *flotation cell* yang berfungsi untuk memisahkan bijih seng sulfida dari mineral lain. 4 sel tersebut membentuk apa yang disebut sebagai *rougher bank*. Pada tiap sel, dengan mengalirkan air bersamaan dengan bijih yang telah dihancurkan ke dalam umpan, *rougher bank* akan melakukan potongan kasar dan memisahkan komponen yang mengapung sehingga terbentuk *slurry*. Cairan surfaktan berupa sodium ethyl xanthate (SEX) dicampurkan kedalam *slurry* untuk menurunkan tegangan permukaan, bersamaan dengan $CuSO_4$ dan naphthalene sulphonate (NS). Udara dialirkan kedalam *slurry* untuk membentuk gelembung udara, partikel-partikel yang terikat dengan gelembung udara akan mengapung ke permukaan dan membentuk buih. Buih akan dipisahkan sebagai konsentrat dan diproses secara lebih lanjut melalui *pump box*. Sisa *slurry* akan masuk ke sel selanjutnya di mana *slurry* akan diproses seperti pada sel sebelumnya.

Data dari proses ini dihasilkan oleh 14 sensor dengan penjelasan yang dapat dilihat pada Tabel IV.5, dengan AI1, AI2, dan AI3 masing-masing menghasilkan 3 pengukuran persen konsentrasi dari Zn, Pb, dan Fe, sehingga kumpulan data memiliki 20 hasil pengukuran. Data memiliki 6029 sampel dengan pengumpulan data selama 6 minggu dengan rerata pencuplikan 10

menit. Diamati bahwa terdapat beberapa kejadian di mana satu atau lebih sensor mengalami kegagalan, sehingga tidak menghasilkan hasil pengukuran, ditandai dengan nilai 0. Penelitian [32] bertujuan untuk mendeteksi kegagalan sensor tersebut dan melakukan rekonstruksi data berdasarkan data yang sudah dipelajari sebelumnya, namun pada penelitian ini, program yang dirancang harus dapat mendeteksi kegagalan sensor berdasarkan perbedaan dari data referensi yang telah dikumpulkan pada awal program dijalankan.

Pada penelitian [32], digunakan 301 sampel kontiyu sebagai data latih dan 701 sampel kontiyu sebagai data uji. Data kontiyu tersebut dimulai dari tanggal 29 September 2015 pukul 02.00 hingga 1 Oktober 2015 pukul 04.00 untuk data latih, data tersebut dipilih karena memiliki rentang panjang yang tidak terdapat kegagalan sensor. Sedangkan untuk data uji, digunakan data dimulai dari 18 Oktober 2015 pukul 03.32 hingga 23 Oktober 2015 pukul 00:12, yang banyak terjadi kegagalan sensor AI3 dengan rentang yang panjang. Pada penelitian ini, kelompok data latih digunakan untuk mengumpulkan data referensi, kelompok data uji digunakan untuk menguji apakah program dapat mendeteksi kesalahan.

Tabel IV.5 Variabel pada Data Validasi

No	*Tag*	Deskripsi Variabel	Satuan
1	LC1	Ketinggian cairan pada sel 2	mm
2	LC2	Ketinggian cairan pada sel 4	mm
3	FC1	Aliran udara yang masuk ke sel 1	m_n^3/h
4	FC2	Aliran udara yang masuk ke sel 2	m_n^3/h
5	FC3	Aliran udara yang masuk ke sel 3	m_n^3/h
6	FC4	Aliran udara yang masuk ke sel 4	m_n^3/h
7	FC5	Aliran tambahan tembaga sulfat	m^3/h
8	FC6	Aliran Sodium ethyl xanthate ke sel 1	m^3/h
9	FC7	Aliran Sodium ethyl xanthate ke sel 3	m^3/h
10	FC8	Aliran Naphtalene sulfat ke sel 1	m^3/h
11	FI1	Aliran volumetric umpan	m^3/h
12	AI1	*Analyzer* XRF	% Zn, Pb, Fe
13	AI2	*Analyzer* XRF konsentrat	% Zn, Pb, Fe
14	AI3	*Analyzer XRF tails*	% Zn, Pb, Fe

IV.2. Tata Laksana Penelitian

Penelitian ini dilakukan dengan beberapa tahapan, yaitu pengujian KFDA dan SVM sebagai metode deteksi kesalahan, perancangan program, pembangunan program, pengujian program menggunakan data latih dan pengujian program menggunakan data validasi.

IV.2.1. Pengujian KFDA dan SVM

Tahap ini dilakukan dengan menggunakan rangka kerja yang dibangun melalui studi literatur untuk menguji performa KFDA-SVM dalam mendeteksi kesalahan pada data pabrik. Pada tahap ini dilakukan pelatihan model dan pengujian model yang diiterasi untuk mencari nilai optimal dari parameter KFDA sehingga didapatkan akurasi klasifikasi SVM yang optimal pula.

Sebagian besar dari algoritma ini merupakan tahap pelatihan model dan tahap pengujian model. Pelatihan model merupakan tahap untuk mengonstruksi *scaler*, matriks proyeksi KFDA, serta SVM menggunakan data latih untuk kemudian di uji

menggunakan data uji pada tahap pengujian model. Pada tahap ini akan diuji apakah rangka kerja KFDA-SVM dapat melakukan deteksi kesalahan secara *supervised* pada data latih yang digunakan.

IV.2.2. Perancangan Program

Tahap perancangan program dilakukan setelah ditemukan parameter pada rangka kerja yang menghasilkan akurasi yang optimal, tahap ini dilakukan untuk membuat alur program yang dapat mengimplementasikan dengan baik rangka kerja yang telah diuji. Tahap perancangan program menjadi penting mengingat perbedaan antara tahap pengujian sebelumnya dengan kondisi nyata, di mana pada tahap Pengujian KFDA dan SVM, data latih masih memiliki label yang digunakan untuk membentuk matrix transformasi KFDA dan *classifier*, sedangkan pada kondisi riil, data yang masuk tidak memiliki label.

Pada tahap ini, akan dilakukan perancangan program untuk mengimplementasikan KFDA-SVM yang telah diuji sebelumnya sehingga dapat melakukan deteksi kesalahan tanpa membutuhkan kumpulan data yang telah tersedia sebagai data latih. Perancangan program dilakukan dengan cara menguji metode pengelompokan yang sesuai untuk diimplementasikan bersama KFDA-SVM untuk membentuk metode *semi-supervised learning*.

Apabila metode pengelompokan yang sesuai telah ditentukan, pengujian selanjutnya dilakukan untuk mencari parameter yang paling optimum untuk rangka kerja yang dibangun. Diakhir tahap ini didapatkan akurasi rangka kerja yang ditawarkan untuk mendeteksi kesalahan pada data latih dengan berbagai pengaturan parameter.

IV.2.3. Pembangunan Program

Setelah pengujian dilakukan dan alur tahapan program dirancang, program dibangun dengan mengikuti rancangan program yang telah dibuat, sehingga data pabrik dapat dikirimkan satu per satu menggunakan protokol komunikasi MQTT untuk meniru kegiatan kumpulan sensor dari pabrik yang memiliki waktu cuplikan tertentu. Program dibangun menggunakan parameter-parameter optimum yang didapatkan pada tahap sebelumnya. Hasil akhir dari tahap ini adalah program

deteksi kesalahan yang mampu mendeteksi data salah dari sebuah pabrik secara *real-time* tanpa memerlukan tahapan pelatihan model yang eksplisit.

IV.2.4. Pengujian Program Menggunakan Data Latih

Setelah program selesai dibangun berdasarkan parameter-parameter yang didapatkan melalui pengujian sebelumnya, pengujian program menggunakan data latih dilakukan untuk menguji performa program sekaligus melakukan finalisasi terhadap fitur-fitur yang diperlukan program untuk melakukan deteksi kesalahan.

IV.2.5. Pengujian Program Menggunakan Data Validasi

Dengan program yang dibangun menggunakan data latih, performa dari program diuji menggunakan data validasi, menggunakan parameter-parameter yang didapatkan melalui tahap perancangan program.

IV.3. Rencana Analisis Hasil Penelitian

Berdasarkan nilai-nilai variabel proses dari tiap sampel pada data yang digunakan, program yang dirancang akan mencoba untuk mengklasifikasikan tiap sampel ke dalam 2 kelas, yaitu kelas normal dan kelas salah. Label hasil klasifikasi dari program kemudian akan dibandingkan dengan label data yang sebenarnya. Perbandingan ini dilakukan pada 3 tahap, yaitu tahap Pengujian KFDA dan SVM, tahap Perancangan Program, dan tahap Pengujian Program menggunakan Data Validasi.

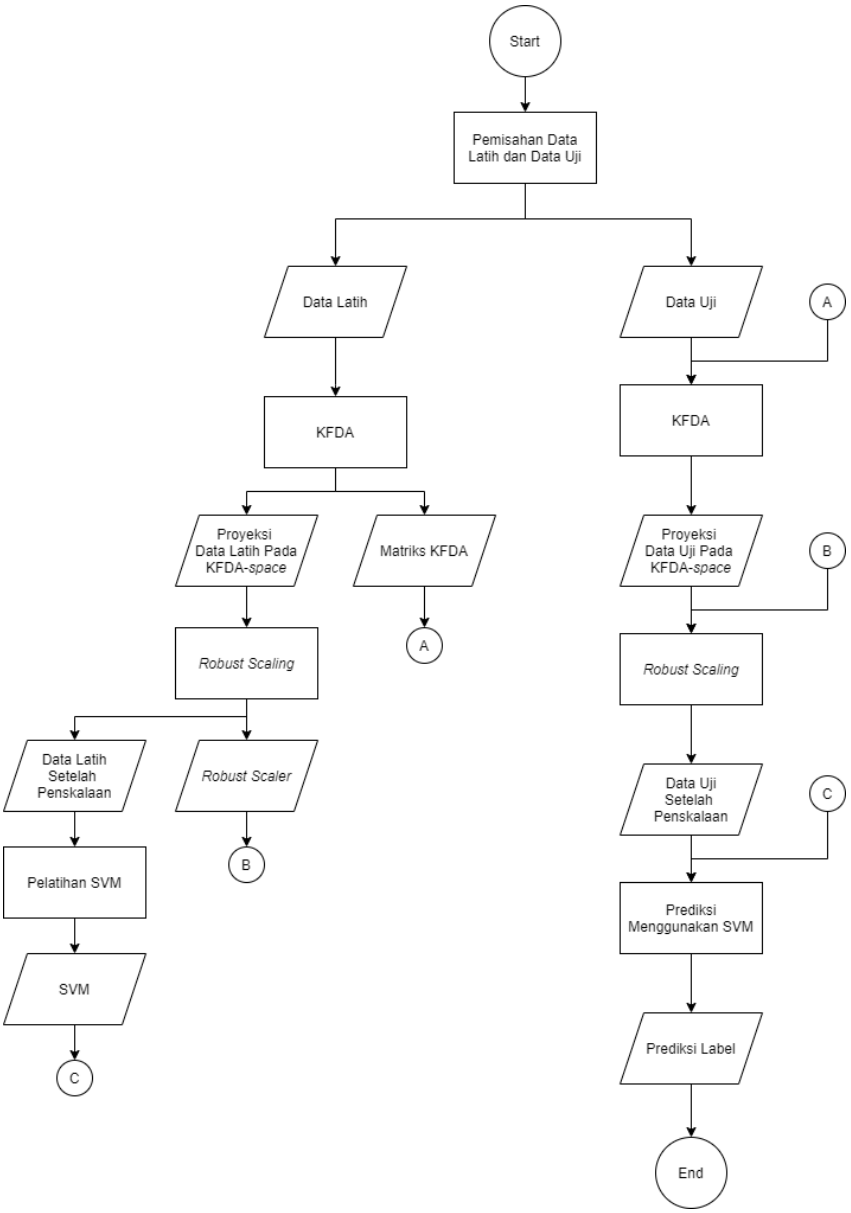
Pada tahap pengujian program, akan didapatkan hasil berupa prediksi label tiap sampel pada data uji. Hasil prediksi label kemudian dibandingkan dengan label yang sebenarnya untuk menghitung akurasi deteksi kesalahan menggunakan rangka kerja pada berbagai kombinasi nilai parameter. Perhitungan akurasi dapat dilakukan dengan cara membagi jumlah prediksi label yang benar dengan jumlah total prediksi, atau akurasi $= \frac{correct\ predictions}{total\ predictions}$. Hasil dari pengujian tersebut kemudian akan dirangkum serta digunakan untuk menentukan kombinasi parameter pada tahap perancangan program. Hasil perangkuman dapat digunakan untuk menentukan kinerja rangka kerja dalam deteksi kesalahan serta parameter yang optimal untuk digunakan sebagai parameter yang ditentukan sebelumnya pada program.

BAB V
Hasil dan Pembahasan

V.1. Pengujian KFDA dan SVM

V.1.1. Rangka kerja deteksi kesalahan menggunakan KFDA dan SVM

Rangka kerja deteksi kesalahan merupakan rangka kerja untuk mendeteksi sampel salah dari kumpulan sampel yang dimasukkan sebagai masukan. Dengan menggunakan metode klasifikasi pembelajaran mesin, deteksi kesalahan dapat dilakukan dengan membentuk *hyperplane* pemisah antara data normal dan data salah. Untuk mempermudah komputasi *hyperplane* pemisah, digunakan metode ekstraksi fitur untuk mengurangi jumlah fitur dari data tanpa menghilangkan informasi penting. Pada penelitian ini, ekstraksi fitur akan digunakan untuk menurunkan jumlah fitur kumpulan data menjadi 2 agar dapat digambarkan pada grafik 2 dimensi. Dengan hanya terdapat 2 fitur, *hyperplane* pemisah yang harus dikalkulasi oleh metode klasifikasi hanya berupa garis pemisah. Penurunan jumlah fitur menjadi 2 walaupun memberikan kemudahan dalam penggambaran pada grafik, namun memberikan tantangan dalam klasifikasi kesalahan. Metode ekstraksi fitur yang digunakan harus dapat memproyeksikan sampel normal dan sampel salah pada planar 2 dimensi secara terpisah. Tugas tersebut dapat dilakukan dengan menggunakan ekstraksi fitur secara *supervised*, yakni metode ekstraksi fitur yang melibatkan label dari data dalam proses pelatihan.



Gambar V.1 Rangka Kerja KFDA-SVM

Berdasarkan studi literatur yang telah dilakukan, dibentuk rangka kerja seperti yang tertera pada Gambar V.1 untuk melakukan deteksi kesalahan menggunakan SVM. Sebelum mulai melakukan deteksi kesalahan, label data yang digunakan dimanipulasi terlebih dahulu. Pada data yang digunakan, terdapat 5 kelas yang menandakan 1 kelas normal dan 4 jenis kesalahan. Pada penelitian ini, hanya digunakan 2 kelas, yaitu kelas normal dan kelas salah, yang secara berurutan memiliki label 0 dan 1. Manipulasi label dilakukan dengan cara mengubah label

pada sampel normal menjadi 0, dan mengubah label pada sampel semua jenis kesalahan menjadi 1, sehingga tugas klasifikasi SVM menjadi klasifikasi biner.

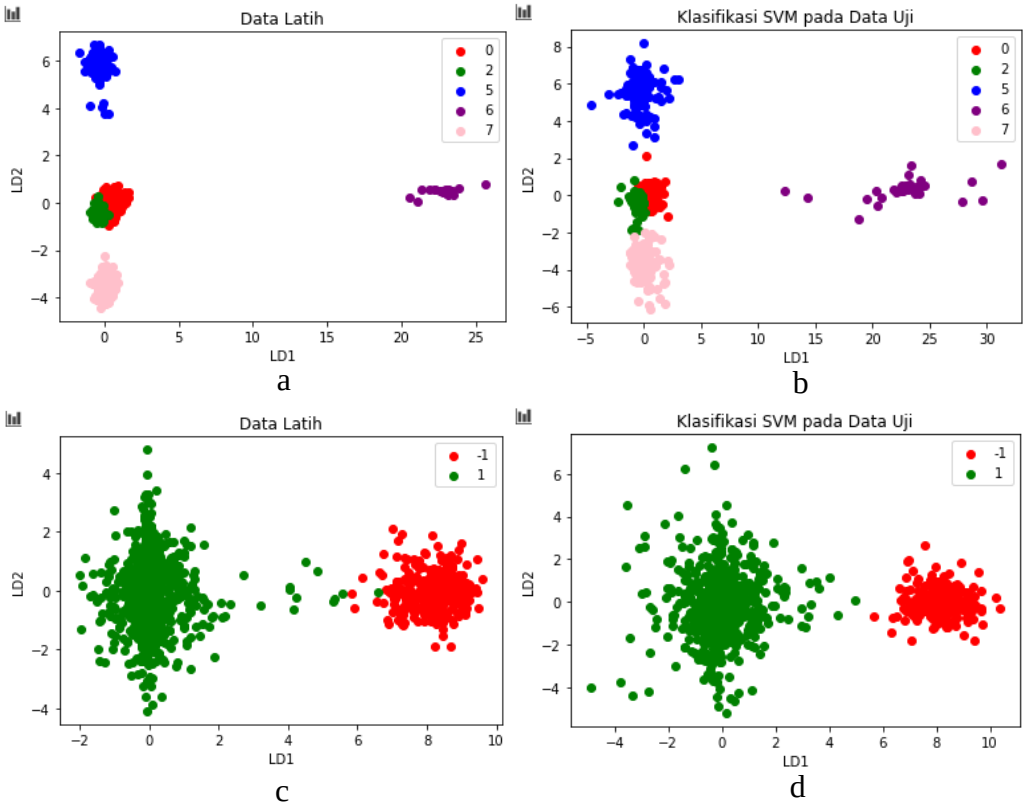
Data kemudian dipisahkan secara acak menjadi 2 set, yakni data latih dan data uji, yang kemudian akan diolah pada 3 tahap berikutnya, yakni penskalaan, ekstraksi fitur, dan klasifikasi. Tiap tahap dilakukan sebanyak 2 kali, yaitu untuk data latih dan untuk data uji. Data latih digunakan untuk membentuk atau melatih transformator dan prediktor yang akan diaplikasikan pada data latih dan data uji, kemudian data uji akan digunakan untuk menilai kinerja rangka kerja dalam melakukan deteksi kesalahan. Data latih kemudian diproyeksikan pada KFDA-*space* pada tahap berikutnya. Tahap ini juga memiliki 2 keluaran, yakni proyeksi data latih pada KFDA-*space* serta transformator berdasarkan data latih yang digunakan untuk memproyeksikan data uji pada KFDA-*space*. Kemudian, data latih akan ditransformasi menggunakan *robust scaler* untuk mengubah skala data latih. Keluaran dari tahap ini berupa data latih yang sudah ditransformasikan serta obyek *scaler* berdasarkan data latih yang kemudian akan digunakan untuk mentransformasikan data uji. Pada tahap berikutnya, proyeksi data latih yang telah diskalakan digunakan untuk membentuk prediktor SVM berdasarkan label dan posisi dari setiap sampel pada data latih. Prediktor kemudian digunakan untuk memprediksi label dari proyeksi data uji pada KFDA-*space*.

Pada rangka kerja yang dibentuk, digunakan *robust scaler* untuk penskalaan, KFDA untuk ekstraksi fitur, dan *support vector machines* untuk klasifikasi. Karena data yang akan digunakan tidak dapat dipastikan akan terdistribusi normal, *robust scaler* dipilih karena dapat melakukan penskalaan pada data bagaimanapun distribusinya. Selain itu, *robust scaler* juga tidak sensitif terhadap *outlier* karena menggunakan nilai jangkauan antar kuartil, tidak seperti *min-max scaler* yang sensitif terhadap *outlier* karena menggunakan nilai minimum dan maksimum. Pada penelitian ini kemudian digunakan *Kernel Fisher Discriminant Analysis* (KFDA) sebagai ekstraksi fitur. KFDA merupakan metode ekstraksi fitur secara *supervised*, sehingga KFDA akan mempertimbangkan label pada sampel untuk kemudian membentuk proyeksi yang memiliki variansi antar kelas yang maksimum dan variansi dalam kelas yang minimum. Dengan menggunakan KFDA, ekstraksi fitur

menjadi 2 fitur dengan proyeksi data antar kelas yang terpisah menjadi mungkin dilakukan.

V.1.2. Pengujian KFDA dan SVM untuk deteksi kesalahan

Rangka kerja ini merupakan inti dari program yang akan dirancang, di mana program akan mengaplikasikan rangka kerja tersebut supaya dapat mendeteksi kesalahan secara *real-time*. Pengujian rangka kerja deteksi kesalahan dilakukan dengan menggunakan dataset Tennessee Eastman Process yang berisi data normal dan data beberapa tipe kesalahan. Dataset kemudian dipisah menjadi data latih dan data uji dengan perbandingan 6:4. Untuk pembuatan matriks transformasi KFDA, digunakan nilai γ sebesar $5x10^{-5}$ untuk kernel FBR. SVM yang dilatih juga merupakan Kernel SVM yang menggunakan kernel FBR pula. Pengujian yang dilakukan memberikan hasil seperti pada Gambar V.2.



Gambar V.2 Hasil Pengujian Menggunakan KFDA-SVM

Gambar (a) pada Gambar V.2 merupakan *scatter plot* dari proyeksi data latih pada KFDA *subspace*. Data latih merupakan data yang digunakan untuk membuat matriks transformasi KFDA dan melatih SVM. Data latih digunakan

untuk melatih SVM sebanyak 2 kali, 1 untuk klasifikasi multi-kelas, dan 1 untuk klasifikasi 2 kelas. Dapat dilihat pada *scatter plot* (a) bahwa tiap kelas pada data latih terproyeksi pada KFDA *subspace* relatif secara terpisah, sehingga tidak banyak terdapat sampel pada 1 kelas yang terproyeksi pada kluster kelas lain. Pada *scatter plot* (b) dicoba menggunakan SVM yang dilatih menggunakan data latih untuk memprediksi label dari tiap kelas pada data latih. Dengan proyeksi data yang relatif terpisah, klasifikasi SVM pada data uji menghasilkan akurasi 99,69% pada persoalan multi-kelas.

Gambar (c) dan (d) merupakan hasil prediksi label data latih dan uji menggunakan persoalan 2 kelas. *Confusion matrix* untuk klasifikasi label data uji secara multikelas dapat dilihat pada tabel V.1 sedangkan untuk klasifikasi 2 kelas dapat dilihat pada tabel V.2.

Tabel V.1. *Confusion Matrix* dari KFDA-SVM Pada Personalan Multi-Kelas

		Label Sebenarnya				
		0	**2**	**5**	**6**	**7**
Label Prediksi	**0**	180	0	0	0	0
	2	7	180	0	0	0
	5	1	0	209	0	0
	6	0	0	0	186	0
	7	2	5	0	0	182
Akurasi		**96,97%**				

Tabel V.2. *Confusion Matrix* dari KFDA-SVM Pada Persoalan 2 Kelas

		Label Sebenarnya	
		-1	**1**
Label Prediksi	**-1**	196	0
	1	3	769
Akurasi		**99,69%**	

Dari tabel V.1 dapat dilihat bahwa misklasifikasi banyak terjadi antara label 0 dan label 2 dan 7. Pada Gambar (c) dan (d) pun dapat dilihat bahwa pada ujung label 0, terdapat beberapa sampel label 2 dan label 7, mengindikasikan bahwa data normal, salah tipe 2 dan tipe 7 memiliki karakteristik data yang mirip. Meskipun demikian, kedua hasil klasifikasi, baik multi kelas maupun 2 kelas memiliki akurasi yang sangat memuaskan, yaitu 96,97% untuk klasifikasi multi kelas, dan 99,69% untuk klasifikasi 2 kelas. Dengan nilai akurasi klasifikasi yang tinggi, rangka kerja KFDA-SVM dinilai cocok untuk digunakan pada program deteksi kesalahan yang akan dirancang pada tahap berikutnya.

V.2. Perancangan Program Deteksi Kesalahan

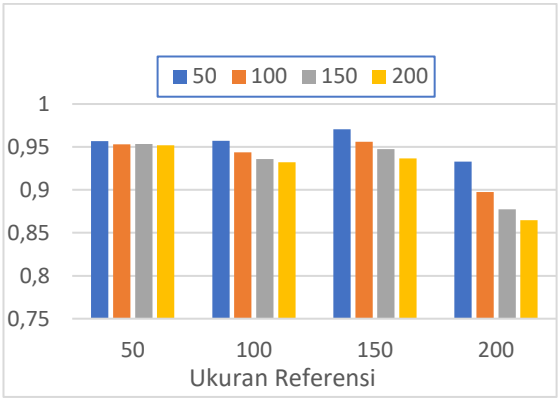
Pada tahap ini, dilakukan beberapa pengujian untuk menentukan parameter-parameter program yang optimal. Parameter yang akan diuji antara lain ukuran sampel referensi dan penyangga dan metode pengelompokan yang optimal.

V.2.1. Pengujian Ukuran Referensi dan Penyangga

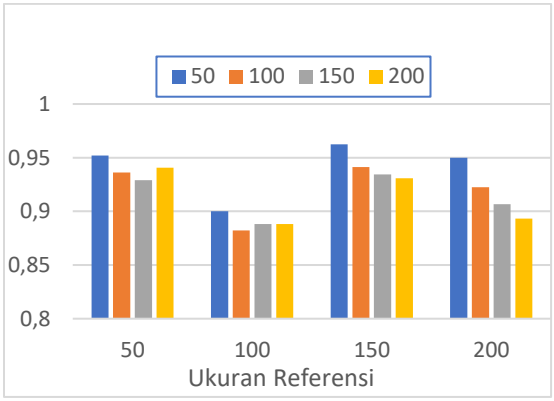
Pengujian terhadap ukuran referensi dan ukuran penyangga dilakukan supaya ditemukan ukuran yang optimum untuk mendapatkan akurasi terbaik. Data referensi adalah kelompok sampel yang diterima diawal program digunakan, kelompok sampel ini akan dianggap sebagai referensi data normal dan disimpan, serta kemudian juga digunakan untuk membentuk matriks KFDA. Data referensi ini diperlukan karena dengan adanya data referensi normal, apabila terjadi sebuah sampel diproyeksikan terlalu jauh dari proyeksi data referensi, maka program deteksi kesalahan akan mengklasifikasikan sampel tersebut sebagai data salah.

Sedangkan penyangga merupakan kelompok sampel yang diterima setelah referensi terisi, dengan penyangga akan secara terus menerus diisi oleh sampel baru yang masuk dan membuang sampel paling tua apabila penyangga sudah penuh, hal tersebut sesuai dengan strategi *sliding windows*.

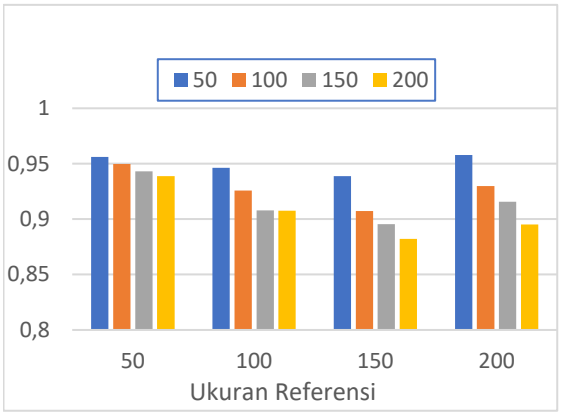
Akurasi dari deteksi kesalahan akan bervariasi tergantung pada ukuran referensi dan penyangga, dikarenakan matriks KFDA dibentuk oleh data referensi sehingga penggunaan ukuran referensi akan berbeda akan menghasilkan proyeksi sampel yang berbeda pula. Selain itu, data normal dan data penyangga digunakan untuk melatih SVM sebagai *classifier*, penggunaan ukuran data normal dan data penyangga yang berbeda, akan menghasilkan garis klasifikasi yang berbeda pula, karena pelatihan SVM menggunakan seluruh sampel untuk menentukan garis klasifikasi.



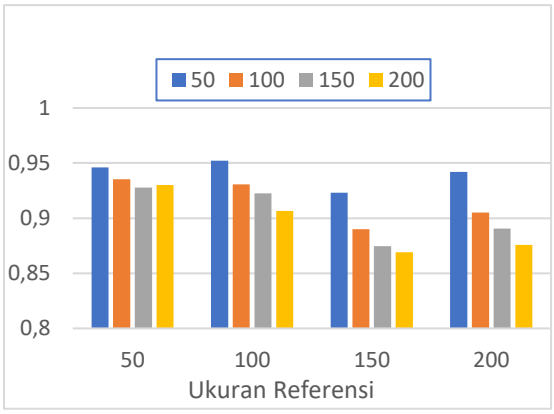
Gambar V.3 Pengujian Akurasi Pada Nilai $\gamma = 0{,}0001$



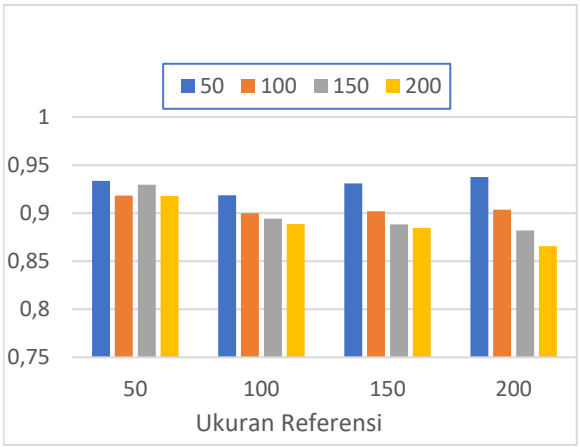
Gambar V.4 Pengujian Akurasi Pada Nilai $\gamma = 0{,}00005$



Gambar V.6 Pengujian Akurasi Pada Nilai $\gamma = 0{,}00001$



Gambar V.7 Pengujian Akurasi Pada Nilai $\gamma = 0{,}000005$



Gambar V.5 Pengujian Akurasi Pada Nilai $\gamma = 0{,}000001$

Pengujian dilakukan dengan memvariasikan ukuran referensi, ukuran penyangga, dan nilai γ untuk memprediksi label sampel pada setiap jenis kesalahan. Pengujian terhadap setiap jenis kesalahan dilakukan sebanyak 3 kali dengan kelompok sampel yang berbeda. Hal tersebut dilakukan dengan cara membagi tiap jenis kesalahan menjadi 3 kelompok dan melakukan pengujian deteksi kesalahan untuk tiap kelompok. Hal tersebut dilakukan untuk memastikan bahwa tiap kelompok sampel pada tiap jenis kesalahan dapat terwakili pada pengujian. Hasil dari perhitungan akurasi untuk tiap kelas kemudian dihitung rata-ratanya, sehingga hanya didapatkan 1 nilai akurasi untuk tiap kombinasi nilai γ, ukuran sampel referensi, dan ukuran sampel penyangga. Hasil pengujian dapat dilihat pada Gambar V.3, V.4, V.5, V.6, dan V.7.

Akurasi prediksi SVM bervariasi bergantung pada ukuran sampel referensi, ukuran sampel penyangga dan nilai γ yang digunakan. Pada hampir seluruh hasil pengujian, dapat dilihat tren bahwa akurasi prediksi SVM semakin menurun dengan bertambahnya ukuran sampel penyangga, dan semakin menurun pula dengan bertambahnya ukuran sampel referensi. Akurasi tertinggi sebesar 97,04% diraih dengan menggunakan nilai γ sebesar 10^{-4}, ukuran sampel referensi sebesar 150, dan ukuran sampel penyangga sebesar 50. Sedangkan akurasi tersendah sebesar 86,92% diraih dengan menggunakan nilai γ sebesar $5x10^{-6}$, ukuran sampel referensi sebesar 150, dan ukuran sampel penyangga sebesar 200.

Pada pengujian ukuran referensi dan penyangga yang dilakukan, diamati bahwa pada γ bernilai besar, seluruh data non-referensi akan diproyeksikan 1 daerah yang sempit sedangkan proyeksi data referensi memiliki persebaran yang besar, sehingga membuat klasifikasi oleh SVM sulit mendapatkan nilai yang akurat. Pada nilai γ yang kecil, proyeksi data normal akan memiliki persebaran yang luas pula, mengikuti persebaran dari proyeksi data referensi, sedangkan data salah akan diproyeksikan diluar persebaran data referensi.

Selain itu, ukuran data referensi berpengaruh pada keterpisahan antara proyeksi data normal dan data salah. Pada nilai γ yang besar, data penyangga akan di proyeksikan secara bertumpukan didalam persebaran data referensi, peningkatan ukuran data referensi akan memperbesar persebaran dari proyeksi data normal, tapi

data salah tetap diproyeksikan secara terpusat didalam persebaran data normal tadi. Sedangkan pada nilai γ lebih kecil, seiring meningkatnya ukuran data referensi, proyeksi data normal dan data salah akan semakin terpisah. Kombinasi nilai γ dan ukuran data referensi perlu diperhatikan untuk mendapat hasil optimum, pada nilai γ yang kecil dengan ukuran data referensi yang kecil, didapatkan hasil bahwa data normal justru diproyeksikan terpisah dari data referensi, sehingga data normal diklasifikasikan sebagai data salah oleh SVM. Selain itu, meskipun pada nilai γ yang kecil dengan ukuran data referensi yang besar didapatkan hasil bahwa data salah semakin terpisah dari data normal dan data referensi, penggunaan ukuran data referensi yang terlalu besar justru mengakibatkan data salah diproyeksikan dekat dengan atau bahkan didalam persebaran data normal.

Baik persebaran maupun keterpisahan proyeksi data, keduanya disebabkan oleh nilai kemiripan antara data normal dan data salah terhadap data referensi. Dengan nilai γ yang semakin besar, proyeksi sampel pada penyangga yang memiliki kemiripan dengan sampel referensi akan semakin terpusat, sedangkan proyeksi sampel penyangga yang berbeda dengan sampel referensi akan semakin tersebar. Namun dengan nilai γ yang terlalu besar, pengaruh dari sampel-sampel pada data referensi menjadi terlalu besar, yang menyebabkan data penyangga dinilai sangat mirip dengan data referensi sehingga data penyangga diproyeksikan pada daerah yang sangat sempit didalam persebaran data referensi. Dengan semakin besarnya ukuran sampel referensi, maka sebuah sampel pada penyangga akan semakin mungkin untuk memiliki kemiripan dengan sampel referensi, sehingga data normal akan diproyeksikan dengan persebaran yang mirip dengan data referensi, sedangkan data salah akan diproyeksikan secara terpisah. Namun dengan ukuran sampel referensi yang terlalu besar, data salah akan memiliki kemiripan dengan data referensi. Sehingga proyeksi data salah akan mendekati persebaran data referensi dan data normal.

V.2.2. Pengujian Metode Pengelompokan

Saat program deteksi kesalahan dijalankan, program deteksi kesalahan harus mampu melakukan pelatihan SVM menggunakan data pada data referensi dan penyangga. Karena SVM merupakan sebuah metode klasifikasi *supervised*, SVM

membutuhkan label dari data yang digunakan untuk dapat membuat garis klasifikasi. Sedangkan pada program yang dirancang, pelatihan SVM dilakukan secara *on-the-fly*, sehingga program harus mempelajari data yang masuk tanpa bantuan data historis. Untuk dapat melakukan pelatihan SVM secara *on-the-fly*, dibutuhkan metode yang dapat memberi label kepada data yang terdapat pada penyangga. Label ini kemudian akan digunakan oleh SVM untuk membuat garis klasifikasi. Salah satu cara melabelkan sampel yang dapat dilakukan adalah melalui metode pengelompokan, metode ini disebut dengan metode *cluster-then-label*, salah satu teknik *semi-supervised learning*. Pada bagian ini, akan dilakukan pengujian yang membandingkan performa metode pengelompokan, yaitu metode pengelompokan k-*means* dan metode DBSCAN.

Pengelompokan k-*means* dipilih untuk diuji karena pada nilai γ yang kecil dan ukuran data referensi yang besar, data salah diproyeksikan terpisah dari data normal, namun tersebar. Dengan menggunakan pengelompokan k-*means*, program dapat melakukan pengelompokan pada gabungan data referensi dan data penyangga. pengelompokan k-*means* akan melakukan pengelompokan terhadap data berdasarkan posisinya, sehingga apabila sebuah kumpulan data terletak jauh dari data referensi, maka pengelompokan k-*means* akan memberikan label yang berbeda kepada 2 kelompok tersebut.

DBSCAN melakukan pengelompokkan sebuah data tergantung pada kepadatan dari sebuah daerah pada data tersebut. DBSCAN akan menggolongkan sebuah sampel dengan menghitung jumlah sampel lain yang terdapat didalam radius tetentu disekitar sampel tersebut, radius tersebut merupakan salah satu parameter dari DBSCAN. Kelemahan dari DBSCAN adalah apabila data salah diproyeksikan secara terlalu tersebar, maka DBSCAN dapat hanya menangkap sebagian dari data salah atau justru salah menggolongkan sebagian data normal sebagai data salah karena kepadatannya.

Pada pengujian ini, pengelompokan dilakukan dengan pertama-tama menggabungkan data referensi dengan data penyangga menjadi sebuah data gabungan. pengelompokan k-means kemudian dilakukan pada data gabungan tersebut dan didapatkan keluaran berupa label dari tiap sampel berdasarkan

kelompoknya. Salah satu kekurangan dari penggunaan pengelompokan k-*means* adalah jumlah kelompok harus ditentukan sejak awal penggunaan metode, dikarenakan pengelompokan k-*means* tidak dapat mengetahui jumlah kelompok yang optimum untuk mengelompokkan data.

Pengujian dilakukan untuk membandingkan performa kelompok k-*means* dan DBSCAN pada berbagai skenario, yaitu variasi nilai γ, ukuran data referensi, serta jumlah data salah pada penyangga. Dengan begitu, dapat dilihat performa pada skenario yang menguntungkan dan tidak menguntungkan bagi tiap metode.

Table V.1 Hasil Pengujian Metode Pengelompokan

Sampel salah		**Positif Benar dan Negatif Benar**				**F-*Score***	
5		**PKM**		**DBSCAN**		**PKM**	**DBSCAN**
γ	Ukuran Referensi	PB	NB	PB	NB		
10^{-4}	50	4	43	5	22	0,73	0,30
	100	2	44	5	8	0,50	0,21
	150	1	44	5	10	0,29	0,22
	200	5	44	5	23	0,91	0,31
$5x10^{-5}$	50	4	44	5	12	0,80	0,23
	100	1	45	5	5	0,33	0,20
	150	4	45	5	10	0,89	0,22
	200	5	38	5	45	0,59	1,00
10^{-5}	50	3	15	0	45	0,16	0,00
	100	5	39	5	44	0,63	0,91
	150	5	45	5	44	1,00	0,91
	200	4	44	4	20	0,80	0,24
$5x10^{-6}$	50	3	45	0	45	0,75	0,00
	100	3	45	5	4	0,75	0,20
	150	3	36	5	23	0,35	0,31
	200	4	45	5	31	0,89	0,42
10^{-6}	50	5	45	0	42	1,00	0,00
	100	3	45	5	16	0,75	0,26
	150	5	40	4	40	0,67	0,57
	200	0	26	4	42	0,00	0,67

Sampel salah		**Positif Benar dan Negatif Benar**				**F-*Score***	
10		**PKM**		**DBSCAN**		**PKM**	**DBSCAN**
γ	Ukuran Referensi	PB	NB	PB	NB		
10^{-4}	50	7	38	10	22	0,74	0,53
	100	6	40	10	8	0,75	0,38
	150	2	39	10	10	0,31	0,40
	200	9	39	10	22	0,90	0,53
$5x10^{-5}$	50	8	39	10	12	0,84	0,42
	100	4	40	10	4	0,57	0,36
	150	7	40	10	8	0,82	0,38
	200	9	38	10	40	0,86	1,00
10^{-5}	50	8	39	0	40	0,84	0,00
	100	9	35	10	39	0,75	0,95
	150	9	40	10	39	0,95	0,95
	200	7	40	9	19	0,82	0,45
$5x10^{-6}$	50	8	40	0	40	0,89	0,00
	100	6	40	10	3	0,75	0,35
	150	8	40	10	18	0,89	0,48
	200	5	40	10	28	0,67	0,63
10^{-6}	50	10	40	0	37	1,00	0,00
	100	4	40	10	14	0,57	0,43
	150	5	40	9	36	0,67	0,78
	200	10	20	9	37	0,50	0,82

Sampel salah		**Positif Benar dan Negatif Benar**				**F-*Score***	
15		**PKM**		**DBSCAN**		**PKM**	**DBSCAN**
γ	Ukuran Referensi	PB	NB	PB	NB		
10^{-4}	50	9	34	15	16	0,72	0,61
	100	10	35	15	7	0,80	0,52
	150	5	34	15	10	0,48	0,55
	200	13	34	15	18	0,90	0,64
$5x10^{-5}$	50	13	34	15	11	0,90	0,56
	100	7	35	15	3	0,64	0,48
	150	8	35	15	8	0,70	0,53
	200	11	35	15	35	0,85	1,00
10^{-5}	50	12	35	0	35	0,89	0,00
	100	14	30	15	34	0,82	0,97
	150	14	35	15	34	0,97	0,97
	200	9	35	14	19	0,75	0,62
$5x10^{-6}$	50	13	35	0	35	0,93	0,00
	100	10	35	15	3	0,80	0,48
	150	12	35	15	18	0,89	0,64
	200	9	35	15	23	0,75	0,71
10^{-6}	50	15	35	0	32	1,00	0,00
	100	4	35	15	9	0,42	0,54
	150	9	35	14	31	0,75	0,85
	200	14	35	14	32	0,97	0,88

Sampel salah		**Positif Benar dan Negatif Benar**				**F-*Score***	
20		**PKM**		**DBSCAN**		**PKM**	**DBSCAN**
γ	Ukuran Referensi	PB	NB	PB	NB		
10^{-4}	50	12	29	20	13	0,73	0,70
	100	15	30	20	6	0,86	0,63
	150	9	29	20	9	0,60	0,66
	200	14	29	20	17	0,80	0,75
$5x10^{-5}$	50	17	30	20	9	0,92	0,66
	100	10	30	20	3	0,67	0,60
	150	9	30	20	7	0,62	0,63
	200	16	30	20	30	0,89	1,00
10^{-5}	50	20	30	0	30	1,00	0,00
	100	19	27	20	29	0,90	0,98
	150	19	30	20	29	0,97	0,98
	200	14	30	19	17	0,82	0,73
$5x10^{-6}$	50	17	30	0	30	0,92	0,00
	100	15	30	20	3	0,86	0,60
	150	15	30	20	17	0,86	0,75
	200	12	30	20	19	0,75	0,78
10^{-6}	50	20	30	0	28	1,00	0,00
	100	13	30	20	8	0,79	0,65
	150	14	30	19	27	0,82	0,90
	200	19	30	19	28	0,97	0,93

Dari pengujian yang dilakukan, didapatkan hasil sesuai yang tertera pada Tabel V.1, yang berisikan jumlah positif benar (PB) dan negatif benar (NB) menggunakan metode PKM dan DBSCAN pada berbagai nilai γ dan ukuran referensi dengan jumlah data salah sebanyak 5, 10, 15, dan 20 secara berturutan, dengan PB dan NB yang tertera pada tabel-tabel tersebut hanya dari penggolongan data penyangga, tidak termasuk data referensi.

Semakin kecilnya nilai γ memiliki pengaruh positif bagi metode PKM, semakin kecil nilai γ yang digunakan, akurasi pengelompokan dengan PKM semakin meningkat, disebabkan oleh semakin terpisahnya data salah dari data *normal* seiring meningkatnya nilai γ yang digunakan. PKM hanya dapat mengelompokkan data yang terpisah secara linier, sedangkan pada nilai γ yang besar, data salah diproyeksikan didalam persebaran data normal, sehingga PKM kesulitan untuk memisahkan data salah dari data normal, sifat PKM tersebut mengakibatkan metode PKM sulit untuk mendapatkan F-*score* bernilai sempurna. Namun pada pengujian yang dilakukan didapatkan hasil bahwa perubahan penggunaan nilai γ tidak memiliki pengaruh yang jelas terhadap nilai F-*score* yang dihasilkan oleh metode PKM maupun DBSCAN, dikarenakan proyeksi dari data akan sangat beragam bergantung pada ukuran referensi dan γ yang digunakan.

Pada DBSCAN, F-*score* yang dihasilkan apabila digunakan γ sebesar $5x10^{-5}$ selalu lebih rendah dibandingkan apabila digunakan γ sebesar 10^{-4}. Namun pada jumlah data salah dan ukuran referensi yang sama, dapat dilihat bahwa F-*score* dari DBSCAN meningkat dan meraih nilai maksimum apabila digunakan nilai γ sebesar 10^{-5}. F-*score* kemudian akan menurun lagi pada nilai γ sebesar $5x10^{-6}$ dan meningkat kembali pada nilai γ sebesar 10^{-6}. Hal tersebut menandakan bahwa DBSCAN sangat peka terhadap kepadatan hasil proyeksi data dari tiap γ yang digunakan, bukan hanya keterpisahan dari data normal dan data salah.

Ukuran data referensi juga mempengaruhi bagaimana data diproyeksikan, namun pada pengujian dengan nilai γ dan jumlah sampel salah yang sama, meningkatnya ukuran data referensi tidak memiliki pengaruh yang jelas bagi F-*score* PKM maupun DBSCAN.

Tidak ada perbedaan signifikan pada peningkatan jumlah data salah terhadap akurasi pengelompokan oleh metode DBSCAN. Meskipun terlihat ada peningkatan nilai F-*score*, namun hal tersebut dikarenakan meningkatnya jumlah data salah yang dapat dikelompokkan oleh DBSCAN. Pada PKM, pengaruh jumlah data salah baru terlihat pada γ yang kecil, dengan semakin besar jumlah data salah pada penyangga, F-*score* dari PKM meningkat dengan drastis karena dengan meningkatnya jumlah data salah, pusat dari tiap kelompok akan bergeser dan menghasilkan F-*score* yang lebih tinggi. Kondisi tersebut dapat dilihat pada nilai γ sebesar 10^{-5}.

Perlu diingat bahwa SVM mengklasifikasikan data dengan membentuk garis pemisah antar kelompok data yang memiliki label berbeda. Hasil pengelompokan oleh metode kelompok berupa label tiap sampel akan digunakan oleh SVM untuk mencari posisi dan orientasi garis pemisah yang optimum. Pada nilai γ yang lebih besar, data salah akan diproyeksikan didalam persebaran data referensi atau bahkan didalam persebaran data *normal*, sehingga meskipun DBSCAN dapat memberikan pengelompokan yang relatif akurat pada kondisi tersebut, namun SVM akan kesulitan untuk membuat garis pemisah untuk memisahkan data salah dari data normal secara akurat.

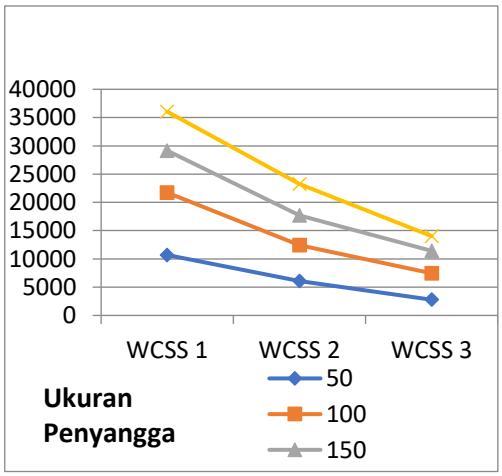
Pada pengujian yang telah dilakukan, didapatkan hasil bahwa penggunaan metode pengelompokan k-*means* menghasilkan F-*score* yang lebih tinggi dibandingkan metode DBSCAN pada 59 pengujian, terutama apabila digunakan jumlah sampel salah yang kecil. Sebagai hasil dari pengujian ini, metode pengelompokan k-*means* akan digunakan sebagai metode pengelompokan pada program yang dirancang.

V.2.3. Pengujian Pengelompokan K-Means

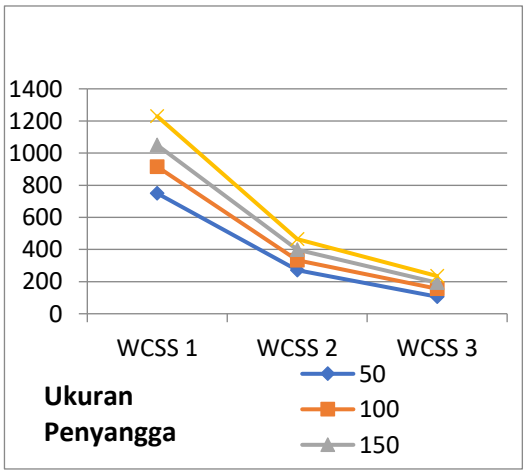
Pengujian dilakukan untuk mendapatkan sebuah nilai ambang untuk memulai metode pengelompokan k-*means* dan mengelompokkan data gabungan kedalam 2 kelompok. Seperti pada *elbow method*, pengujian akan membandingkan nilai *within cluster sum of squares* (WCSS) pada 1 hingga 3 kluster. *Elbow method* merupakan sebuah evaluasi secara visual, di mana seorang pengguna mengevaluasi grafik WCSS untuk menentukan jumlah kelompok optimal berdasarkan pembentukan sebuah siku pada grafik, sedangkan pada program yang dirancang, program harus

mampu mengevaluasi pembentukan siku secara otomatis dan mulai mengelompokkan data gabungan apabila terdapat siku pada penggunaan 2 kelompok. Sehingga pada tahap ini juga dilakukan pengujian implementasi metode pengelompokan k-*means* kedalam rangka kerja.

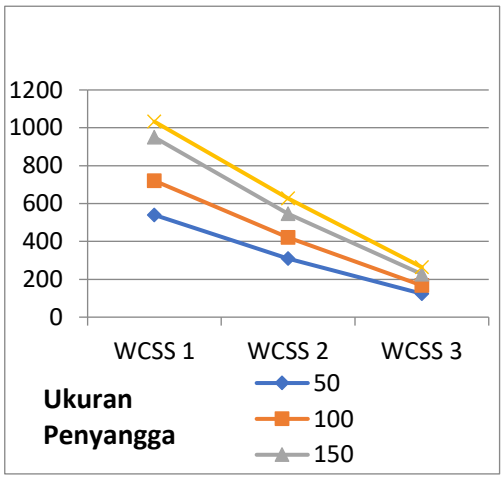
Pengujian dilakukan dengan memvariasikan ukuran referensi dan ukuran penyangga pada 50 hingga 200 pada γ sebesar $5x10^{-5}$. Data referensi akan diisi dengan sampel normal, dan data penyangga akan diisi dengan campuran sampel normal dan sampel salah. Kedua data tersebut kemudian digabungkan dan dilakukan metode pengelompokan k-*means* untuk dikelompokkan kedalam 2 kelompok, normal dan salah. Apabila pada data penyangga jumlah sampel salah terlalu sedikit, maka hasil dari pengelompokan k-*means* akan terlihat seperti sekedar membagi data gabungan menjadi 2 kelompok sama rata. Hal tersebut berarti pengelompokan k-*means* tidak memiliki akurasi tinggi pada komposisi data penyangga yang digunakan. Oleh sebab itu, pengujian ini juga digunakan untuk mengetahui berapa jumlah data salah pada data penyangga agar pengelompokan k-*means* dapat melakukan pelabelan secara akurat. Apabila akurasi pelabelan pengelompokan k-*means* melewati batas tertentu, maka akan dicatat ketiga nilai WCSS serta jumlah sampel salah yang digunakan. Dari pengujian tersebut didapatkan hasil seperti pada Gambar V.8, V.9, V.10 dan V.11.



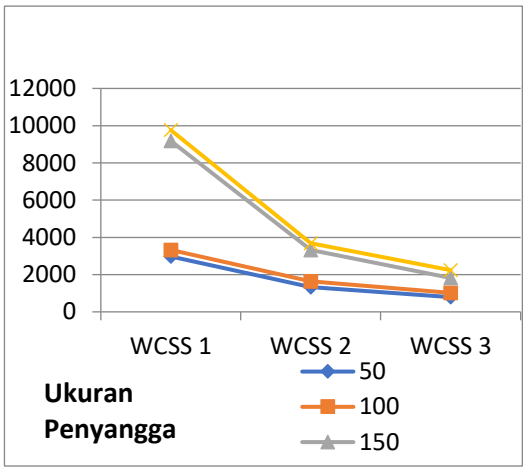
Gambar V.8 Nilai WCSS Minimum Pada Ukuran Referensi = 50



Gambar V.9 Nilai WCSS Minimum Pada Ukuran Referensi = 100

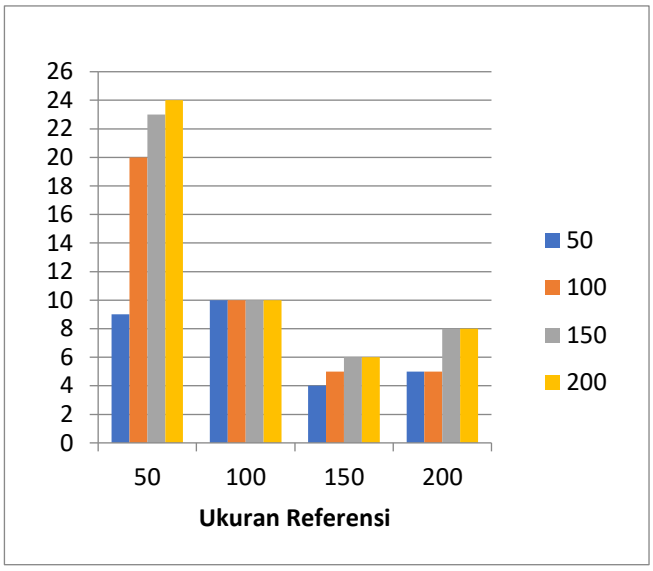


Gambar V.10 Nilai WCSS Minimum Pada Ukuran Referensi = 150



Gambar V.11 Nilai WCSS Minimum Pada Ukuran Referensi = 200

Tiap garis pada Gambar V.8, V.9, V.10, dan V.11 mewakili pengujian menggunakan penyangga dengan ukuran tertentu, dan tiap titik pada garis mewakili nilai WCSS apabila WCSS dihitung pada jumlah kluster yang bersangkutan. Variasi nilai minimum WCSS pada keempat Gambar utamanya disebabkan oleh perbedaan ukuran referensi serta perbedaan ukuran penyangga, ukuran referensi akan mempengaruhi proyeksi data dan ukuran penyangga akan mempengaruhi pengelompokan oleh PKM. Nilai WCSS meningkat seiring membesarnya ukuran referensi yang digunakan, hal tersebut dikarenakan data referensi diproyeksikan dengan persebaran yang besar, sehingga menghasilkan nilai WCSS yang besar.



Gambar V.12 Sampel Salah Minimum

Gambar V.12 menjelaskan jumlah sampel salah minimal dalam penyangga agar pengelompokan k-*means* dapat melakukan pelabelan secara akurat. Pada Gambar V.12 dapat dilihat bahwa dengan ukuran referensi yang semakin besar, dibutuhkan sampel salah pada data penyangga yang semakin sedikit, sedangkan seiring bertambahnya ukuran penyangga, jumlah sampel salah minimal ikut bertambah. Jumlah sampel salah minimal yang kecil akan memberikan dampak positif terhadap deteksi kesalahan, karena berarti pengelompokan k-*means* lebih peka terhadap data salah. Sedangkan pada pengujian menggunakan ukuran referensi sebesar 50, dibutuhkan jumlah sampel salah yang besar agar pengelompokan k-*means* dapat melakukan pelabelan secara akurat. Misalkan, pada ukuran penyangga sebesar 200, pengelompokan k-*means* membutuhkan data salah sebanyak 155 untuk dapat memberikan pelabelan secara akurat.

Pada *elbow method*, sebuah jumlah kelompok disebut sebagai jumlah kelompok optimal apabila grafik WCSS membentuk sebuah siku (*elbow*) pada jumlah kelompok tersebut. Mempertimbangkan metode tersebut, penentuan ambang dilakukan dengan cara mencari rasio beda nilai WCSS pada tiap pengujian, dengan rasio didapatkan dengan rasio = $(WCSS\ 1 - WCSS\ 2)/(WCSS\ 2 - WCSS\ 3)$. Rasio akan meningkat seiring dengan

bertambahnya jumlah sampel salah yang terdapat pada penyangga, sehingga apabila nilai ambang terlalu tinggi, berarti program akan membiarkan beberapa sampel salah masuk kedalam penyangga tanpa deteksi, pengelompokan k-*means* baru dijalankan apabila sampel salah sudah cukup banyak sehingga rasio mencapai ambang. Rasio yang paling kecil untuk tiap ukuran referensi ditetapkan sebagai ambang pada ukuran referensi tersebut agar pengelompokan k-*means* dapat segera dijalankan dengan jumlah sampel salah *yang* sedikit mungkin. Ambang yang didapatkan yakni: $ambang_{50}$ = 1.62, $ambang_{100}$ = 3.18, $ambang_{150}$ = 1.2, dan $ambang_{200}$ = 3.46.

V.2.3. Penentuan Parameter

Pada bagian ini, parameter yang akan digunakan sebagai parameter yang ditentukan dari program yang dirancang akan ditentukan. Berdasarkan pengujian pengaruh ukuran referensi, ukuran penyangga, dan nilai γ, didapatkan bahwa semakin besarnya nilai penyangga yang digunakan, akurasi dari SVM akan semakin menurun. Oleh karena itu, digunakan ukuran penyangga sebesar 50. Ukuran ini juga dirasa tepat untuk digunakan pada metode pengelompokan, karena dengan ukuran penyangga yang kecil, sebuah data salah yang masuk kedalam penyangga dapat lebih mudah terdeteksi dibandingkan apabila digunakan ukuran penyangga yang lebih besar. Hal tersebut dikarenakan pengelompokan k-*means* melakukan pengelompokan berdasarkan *mean* dari kumpulan data, sehingga akan kesulitan dalam mengelompokkan data yang tidak berimbang. Pada Gambar V.4 dapat dilihat bahwa pengujian menghasilkan akurasi tertinggi senilai 97,04% ketika digunakan ukuran referensi sebesar 150 dan ukuran penyangga sebesar 50, pada Gambar V.5, parameter yang sama mendapatkan akurasi tertinggi pula yakni sebesar 96,25%. Namun pada Tabel V.1, pengujian dengan nilai γ sebesar 10^{-4}, ukuran referensi sebesar 150 dan ukuran penyangga sebesar 50 menghasilkan F-*score* yang jauh lebih kecil dibandingkan apabila digunakan nilai γ sebesar $5x10^{-5}$. Pada program yang akan dirancang, digunakan parameter yang ditentukan berupa nilai γ sebesar $5x10^{-5}$, ukuran referensi sebesar 150, dan ukuran penyangga sebesar 50.

V.3. Pembangunan Program

V.3.1. Tahapan Program

Program dibangun menggunakan metode-metode yang telah diuji pada tahap sebelumnya, dan menggunakan parameter yang telah diuji sebagai parameter yang ditentukan. Program memiliki 4 tahapan utama, yakni pengolahan referensi, ekstraksi fitur, pengelompokan data, dan klasifikasi.

1. Tahap Pengolahan Referensi

Tahap pengolahan referensi dimulai sejak pertama program dijalakan hingga didapatkan matriks transformasi KFDA. Pada tahap ini, program akan menyimpan sampel – sampel yang diterima melalui MQTT sebagai data referensi hingga ukuran data referensi terpenuhi. Apabila data referensi sudah penuh, data tersebut akan diolah menggunakan KFDA untuk mengekstraksi 2 fitur, keluaran dari KFDA adalah proyeksi data referensi pada ruang KFDA dan matriks KFDA yang kemudian akan digunakan untuk memproyeksikan data penyangga pada ruang KFDA. Proyeksi data referensi kemudian diolah menggunakan *robust scaler* untuk mengubah skala dari data referensi agar tidak ada fitur yang mendominasi fitur lainnya. Keluaran yang didapatkan adalah proyeksi data referensi yang sudah diubah skalanya dan persamaan *robust scaler* yang akan dipakai untuk mengubah skala pada data penyangga nantinya.

2. Tahap Ekstraksi Fitur

Apabila data referensi telah diolah, data yang masuk melalui MQTT kemudian dimasukkan kedalam data penyangga dan program akan menunggu hingga data penyangga sudah penuh. Apabila data penyangga sudah dipenuhi dengan sampel, data penyangga akan diolah menggunakan matriks KFDA dan *robust scaler* yang didapatkan dari tahap sebelumnya. Keluaran dari tahap ini berupa ekstraksi fitur data penyangga menjadi hanya 2 fitur sehingga dapat di gambarkan secara 2 dimensi. Saat sampel baru masuk melalui MQTT, program akan menambahkan sampel baru tersebut kedalam data penyangga dan membuang sampel paling lama pada data penyangga, sehingga ukuran data penyangga tetap seperti ukuran yang telah ditetapkan. Tahap ekstraksi fitur akan dijalankan setiap sampel baru masuk kedalam data penyangga.

3. Tahap Pengelompokan data

Tahap pengelompokan data merupakan tahap di mana dilakukan pemberian label untuk setiap sampel pada data penyangga, agar data penyangga dapat digunakan untuk melatih SVM. Pada tahapan ini, data referensi dan data penyangga digabung menjadi 1 gabungan data yang akan diolah menggunakan pengelompokan k-*means* untuk memprediksi label dari tiap sampel. Sebelum pengelompokan dilakukan, dilakukan pengujian terhadap rasio nilai WCSS data gabungan. Pengujian ini dilakukan dengan cara mencari nilai WCSS untuk 1 hingga 3 kelompok, dan dihitung rasio WCSS. pengelompokan k-*means* akan dijalankan apabila nilai rasio diatas nilai ambang yang digunakan. Hal tersebut dilakukan untuk menjaga akurasi dari program, atau dalam kata lain, untuk mencegah pengelompokan k-*means* dari menggolongkan data gabungan yang dipenuhi dengan data normal menjadi 2 kelompok yang akan diartikan oleh SVM sebagai data normal dan data salah. Keluaran dari pengelompokan k-*means* berupa label untuk tiap sampel pada data gabungan. Pada label data gabungan tersebut, label dari sampel-sampel data referensi diubah menjadi label normal untuk menanggulangi apabila terdapat sampel referensi yang terlabelkan sebagai data salah. Data gabungan dan label tersebut digunakan untuk melatih SVM pada tahap berikutnya.

4. Tahap Klasifikasi Data

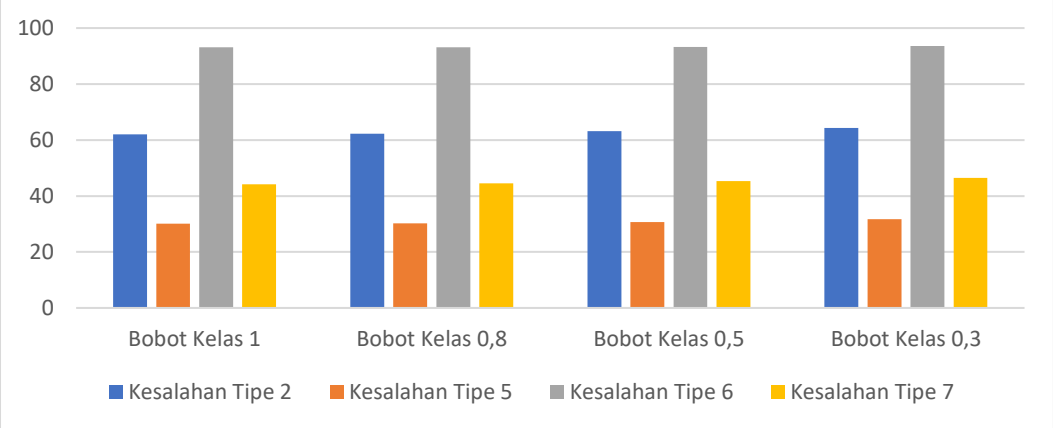
Pada tahap ini, SVM dilatih menggunakan data gabungan dan label yang didapatkan melalui tahap sebelumnya. SVM yang dilatih kemudian digunakan untuk memprediksi label dari setiap sampel, apakah sampel tersebut normal atau salah. Sama seperti tahap sebelumnya, tahap klasifikasi juga akan dilakukan setiap ada sampel baru yang masuk.

V.3.2. Pengujian Program Menggunakan Data Latih

Pada bagian ini, program yang telah dirancang diuji untuk melakukan deteksi kesalahan terhadap data latih dengan parameter yang telah ditentukan sebelumnya. Pengujian yang telah dilakukan dapat dilihat pada Gambar V.13. Pada pengujian ini, akan diuji pengaruh penggunaan bobot kelas dan strategi konvergensi SVM terhadap akurasi deteksi kesalahan. Bobot kelas menandakan tingkat prioritas data normal dibandingkan data salah, SVM akan menggunakan nilai bobot kelas tersebut

untuk menentukan batas pemisahnya. Dengan memperkecil bobot kelas data normal, maka SVM akan memprioritaskan data salah yang memiliki bobot kelas yang lebih tinggi, sehingga garis pemisah SVM akan dibuat lebih dekat dengan data normal. Sebagai konsekuensi, daerah data salah memiliki ruang lebih, sehingga apabila sebuah sample dengan jenis kesalahan lain terproyeksi lebih dekat dengan data normal, sampel tersebut akan diklasifikan sebagai sampel salah.

Pengukuran akurasi dilakukan dengan membandingkan prediksi label dari sampel pada data penyangga dengan label yang sebenarnya. Untuk setiap siklus klasifikasi akan didapatkan 1 nilai akurasi. Ketika semua sampel terkirim, nilai akurasi kumulatif akan dicari dengan menjumlahkan nilai akurasi yang didapat tiap siklus dan dibagi dengan jumlah siklus. Pengujian akan dilakukan dengan parameter yang telah ditentukan, yakni ukuran referensi 150 sampel, ukuran penyangga sebesar 50 sampel, dan γ sebesar $5x10^{-5}$. Hasil pengujian dapat dilihat pada Gambar V.13.

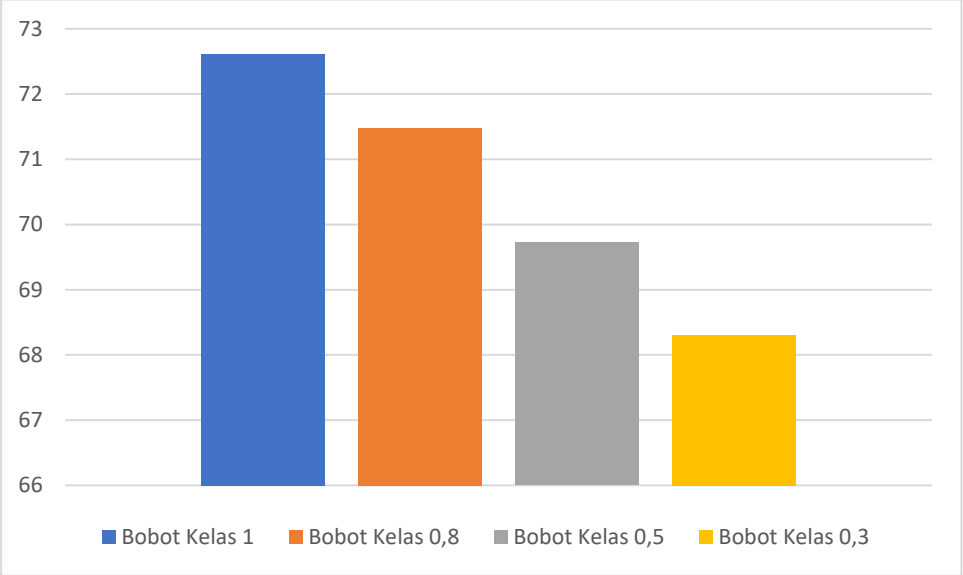


Gambar V.13 Akurasi Program Pada Jenis Kesalahan Dengan Variasi Bobot Kelas

Pada Gambar V.13 terlihat bahwa hampir semua mengujian memiliki kesulitan dalam mendeteksi data salah pada salah tipe 5 dan salah tipe 7. Hal tersebut dikarenakan proyeksi sampel salah tipe 5 dan salah tipe 7 bertumpukan dengan data referensi, sehingga program deteksi kesalahan mengalami kesulitan dalam memisahkan sampel salah tipe 5 dan salah tipe 7 dari sampel normal. Meskipun begitu, seiring mengecilnya bobot kelas data normal, garis pemisah SVM

akan semakin diletakkan melintasi data referensi karena lebih memprioritaskan data berlabel salah dan menyebabkan akurasi SVM meningkat seiring mengecilnya bobot kelas dari data normal.

Kesalahan tipe 5 dan kesalahan tipe 7 diproyeksikan pada daerah yang sama dengan data normal karena mereka memiliki data yang mirip. Ketika komposisi umpan B diubah yang kemudian menghasilkan data kesalahan tipe 5, bacaan sensor akan berosilasi pada rentang waktu tertentu. Osilasi tersebut dikarenakan pada sistem pabrik dikenakan masukan berupa step, yang kemudian menyebabkan sistem memberikan respon transien berupa osilasi hingga akhirnya mencapai keadaan tunak. Ketika sistem berosilasi, bacaan sensor akan naik-turun melintasi rentang yang dianggap normal oleh program. Ketika bacaan sensor menghasilkan data yang terlalu tinggi atau terlalu rendah dibandingkan data normal, maka data tersebut akan diproyeksikan diluar persebaran data normal dan dikategorikan sebagai data salah, namun ketika bacaan sensor menghasilkan data didalam rentang yang dianggap normal, maka data akan diproyeksikan didalam persebaran data normal dan terjadi misklasifikasi. Meskipun osilasi merupakan salah tersendiri, program yang dirancang tidak dapat mendeteksi osilasi, hanya mampu mendeteksi kesalahan berdasarkan proyeksi tiap sampel. Setelah data kesalahan *5* mencapai keadaan tunak, yakni ketika bacaan sensor tidak lagi berosilasi, data kesalahan *5* menghasilkan bacaan sensor yang mirip dengan data normal, yang menyebabkan data keadaan tunak kesalahan tipe 5 diproyeksikan didalam persebaran data normal dan kemudian menyebabkan misklasifikasi. Misklasifikasi data kesalahan tipe 7 memiliki penyebab yang sama dengan kesalahan tipe 5, namun pada data kesalahan tipe 7, kesalahan disebabkan oleh masukan *step* berupa penurunan tekanan umpan C.

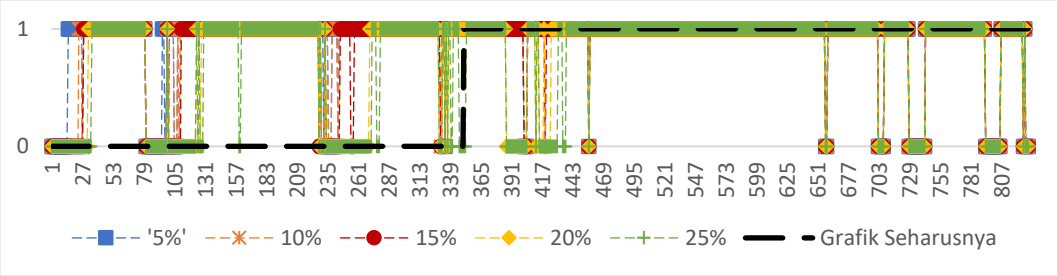


Gambar V.14 Pengaruh Bobot Kelas Pada Akurasi Klasifikasi Data Normal

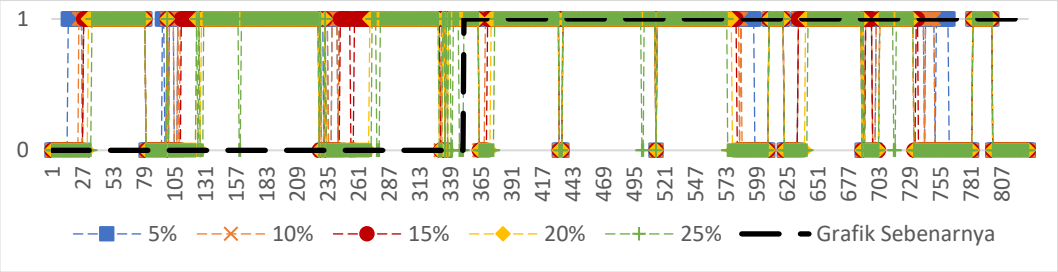
Gambar V.14 merupakan akurasi program dalam melakukan klasifikasi terhadap data normal dari pengujian program menggunakan data latih. Hasil tersebut didapatkan karena ambang WCSS yang digunakan terlalu kecil, sehingga pengelompokan k-*means* akan melakukan pelabelan data tanpa menunggu nilai rasio WCSS mencapai ambang yang kemudian menyebabkan pengelompokan dan klasifikasi akan langsung dijalankan karena program selalu berasumsi bahwa terdapat data salah. Dengan bobot data normal yang kecil, program deteksi kesalahan akan mengalami kesulitan dalam mengenali sampel normal, dan justru akan memprediksi sampel normal sebagai sampel salah. Hal ini merupakan akibat dari pelabelan sampel yang tidak akurat pada tahap pengelompokan k-*means*. Pada bobot kelas data normal yang rendah, SVM akan lebih peka terhadap sampel berlabel salah dan kurang memperdulikan data berlabel normal. Sebagai konsekuensi, SVM akan membuat garis pemisah yang melintasi kelompok data normal tersebut demi data berlabel salah. Sedangkan apabila data normal memiliki bobot kelas 1, bobot kelas data berlabel normal akan sama dengan bobot kelas data berlabel salah, alih-alih membuat garis pemisah yang menembus kelompok data berlabel normal, SVM justru akan membuat garis pemisah diujung kelompok normal, karena SVM tidak lagi lebih sensitif terhadap data berlabel salah, dan

sebagai hasilnya program deteksi kesalahan menghasilkan akurasi yang lebih tinggi untuk mendeteksi data normal.

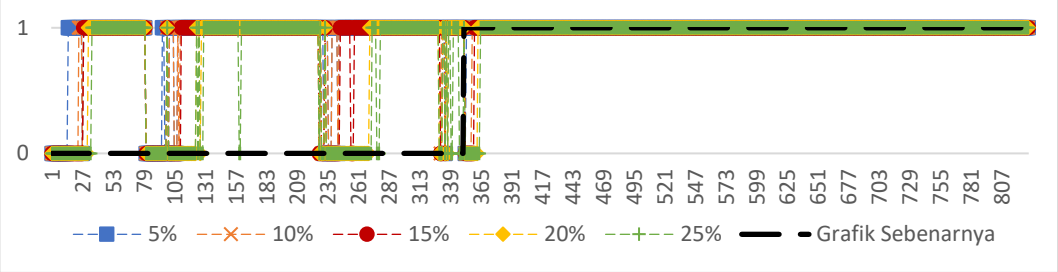
Pengujian dengan bobot kelas 0,3 dapat mencapai nilai akurasi yang lebih tinggi lagi apabila ambang rasio WCSS program sesuai dengan rasio WCSS ketika data kesalahan masuk ke dalam program. Meskipun begitu, penentuan ambang rasio WCSS yang tepat sulit untuk ditentukan, karena apabila rasio WCSS terlalu rendah, maka pengelompokan akan tetap dijalankan meskipun pada penyangga yang dipenuhi dengan data normal saja. Namun apabila rasio WCSS terlalu tinggi, maka apabila terdapat tipe kesalahan yang diproyeksikan didalam persebaran data normal seperti data kesalahan tipe 5 dan 7, maka persebaran tersebut akan menghasilkan rasio WCSS yang relatif kecil sehingga pengelompokan tidak akan dijalankan dan mengakibatkan kesalahan tidak terdeteksi oleh program. Meskipun rasio WCSS dipilih karena merupakan metode yang sering digunakan untuk mendeteksi jumlah kelas yang optimal untuk mengelompokkan sebuah data, namun penetapan ambang berupa rasio WCSS tidak efektif untuk digunakan pada program yang dirancang, karena data salah dapat diproyeksikan diluar maupun didalam persebaran data normal.



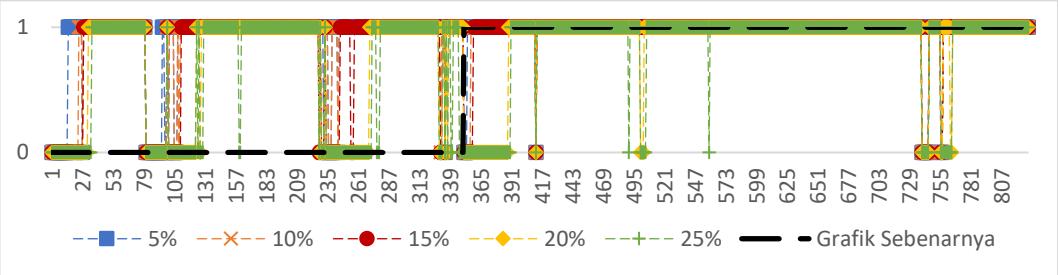
Gambar V.15 Hasil Kesimpulan Program Pada Variasi Persentasi Data Salah Pada Data Salah Tipe 2



Gambar V.16 Hasil Kesimpulan Program Pada Variasi Persentasi Data Salah Pada Data Salah Tipe 5



Gambar V.17 Hasil Kesimpulan Program Pada Variasi Persentasi Data Salah Pada Data Salah Tipe 6



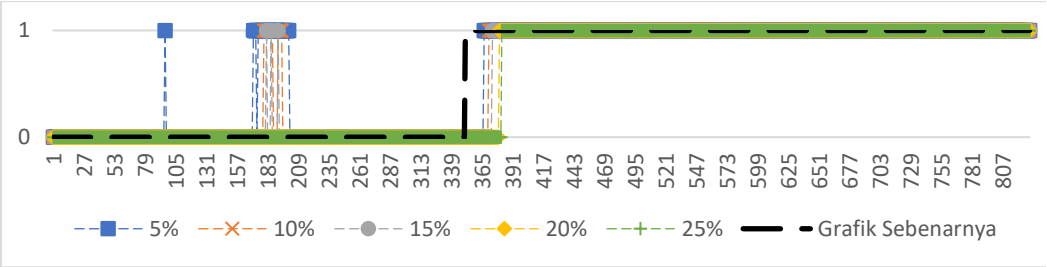
Gambar V.18 Hasil Kesimpulan Program Pada Variasi Persentasi Data Salah Pada Data Salah Tipe 7

Gambar V.15 hingga Gambar V.18 merupakan pengujian yang dilakukan untuk menilai kemampuan program dalam memberikan kesimpulan apakah terdapat kesalahan pada proses atau tidak. Garis “Grafik Sebenarnya” pada setiap

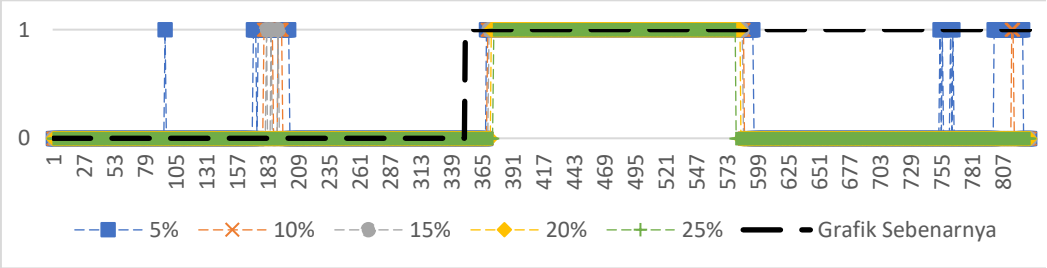
grafik tersebut menunjukkan bentuk grafik yang ideal, dimana program seharusnya memberikan kesimpulan bahwa proses normal (0) terlebih dahulu dan memberikan kesimpulan bahwa terdapat kesalaham pada proses sejak data ke 350. Pengujian ini dilakukan menggunakan bobot data normal sebesar 0, dengan variasi nilai ambang kesimpulan dari 5% hingga 25%. Nilai ambang kesimpulan merupakan persentase jumlah data salah pada penyangga yang digunakan untuk menyimpulkan apakah terdapat kesalahan pada proses. Nilai ambang kesimpulan sebesar 5% berarti program akan menyimpulan terdapat kesalahan proses apabila 5% data penyangga diklasifikasikan sebagai data salah.

Pada Gambar V.15 hingga Gambar V.18 dapat dilihat bahwa program pada berbagai nilai ambang memiliki kesulitan dalam memberikan kesimpulan yang sesuai dengan keadaan sebenarnya, terutama pada data normal. Gambar V.15 hingga V.18 menunjukkan bahwa program sering kali memberikan kesimpulan bahwa terdapat kesalahan proses yang salah, di mana pada data ke-1 hingga data ke-350, terdapat banyak kejadian program memberikan kesimpulan bahwa terdapat kesalahan proses dengan durasi yang panjang. Hasil tersebut tentu sesuai dengan pengujian pada Gambar V.13 yang memberikan hasil bahwa pengujian pada data

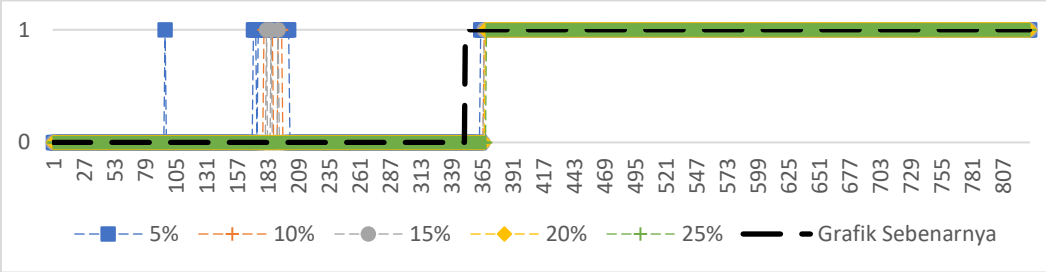
salah tipe 5 dan 7 menghasilkan akurasi yang rendah dibandingkan pada data salah tipe 2 dan 6.



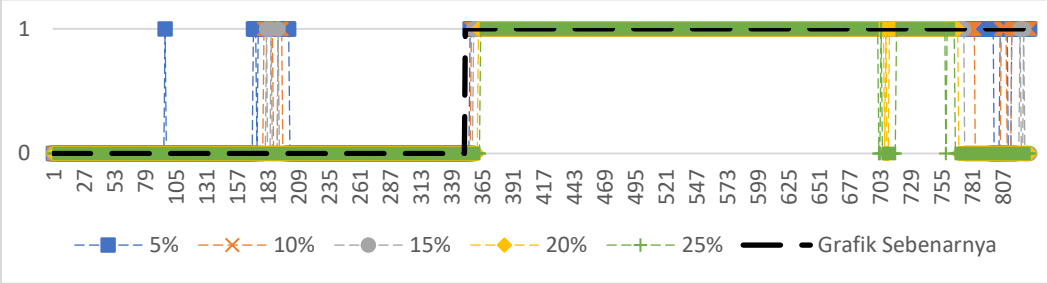
Gambar V.19 Hasil Kesimpulan Program Pada Variasi Persentasi Data Salah Pada Data Salah Tipe 2 Setelah Penyesuaian Parameter



Gambar V.20 Hasil Kesimpulan Program Pada Variasi Persentasi Data Salah Pada Data Salah Tipe 5 Setelah Penyesuaian Parameter



Gambar V.21 Hasil Kesimpulan Program Pada Variasi Persentasi Data Salah Pada Data Salah Tipe 6 Setelah Penyesuaian Parameter



Gambar V.22 Hasil Kesimpulan Program Pada Variasi Persentasi Data Salah Pada Data Salah Tipe 7 Setelah Penyesuaian Parameter

Gambar V.19 hingga Gambar V.22 merupakan pengujian untuk menilai kemampuan program dalam memberikan kesimpulan dengan nilai parameter yang telah disesuaikan. Pengujian tersebut menghasilkan hasil yang lebih mirip dengan Grafik Sebenarnya dibandingkan hasil pengujian pada Gambar V.15 hingga V.18. Pada Gambar V.19 dan V.21 dapat dilihat bahwa program dapat memberikan kesimpulan kesalahan proses dengan relatif akurat meskipun tidak secepat seharusnya. Pada Gambar V.20 dan V.22, meskipun kesimpulan kesalahan proses didapatkan dengan relatif akurat, namun terdapat beberapa kejadian dimana program memberikan kesimpulan bahwa tidak ada kesalahan proses sedangkan seharusnya ada. Selain itu, setelah penyesuaian parameter, program juga menjadi lebih akurat dalam memberikan kesimpulan bahwa tidak ada kesalahan proses, dapat dilihat kesimpulan program pada data dibawah 350. Meskipun penyesuaian parameter menghasilkan grafik yang relatif lebih baik dibandingkan pengujian menggunakan parameter sebelumnya, rentang terjadinya kesimpulan yang salah pada data dibawah 350 relatif besar.

Tabel V.3 Jumlah Kesimpulan Keliru Pada Data Ke-171 Hingga Ke-201

Nilai Ambang				
5%	10%	15%	20%	25%
30	12	3	0	0

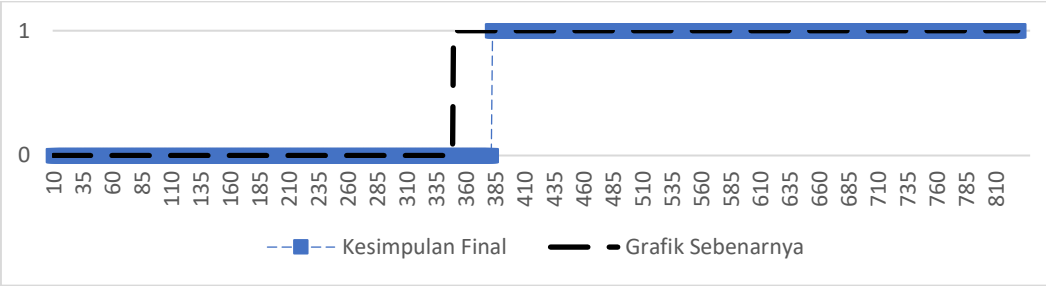
Tabel V.4 Jumlah Data Yang Dibutuhkan Untuk Menghasilkan Kesimpulan Terdapat Kesalahan Proses

	Nilai Ambang				
	5%	10%	15%	20%	25%
Data Salah Tipe 2	17	21	24	30	32
Data Salah Tipe 5	19	20	21	23	25
Data Salah Tipe 6	14	17	17	18	19
Data Salah Tipe 7	5	6	8	12	14

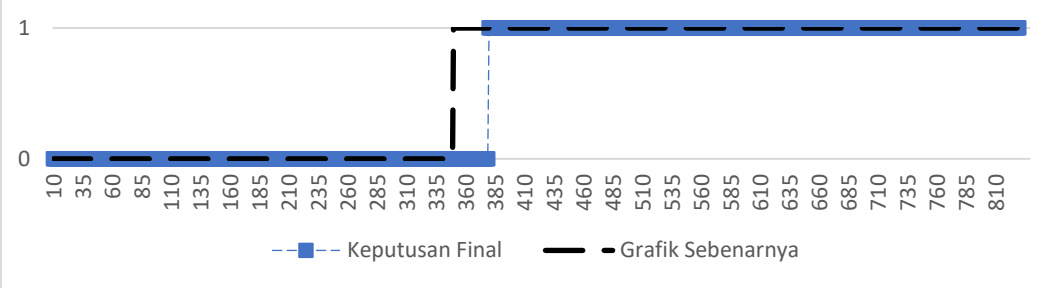
Tabel V.3 berisikan jumlah kesimpulan yang keliru pada data ke-171 hingga ke-201 pada berbagai nilai ambang, yakni ketika program menyimpulkan bahwa terdapat kesalahan proses, sedangkan sebenarnya tidak terdapat kesalahan proses. Ketika digunakan nilai ambang sebesar 5%, terdapat rentang kesimpulan keliru yang paling besar, yakni terjadi pada data ke-171 hingga ke-201. Sedangkan Tabel V.4 berisikan jumlah data salah yang dibutuhkan untuk mengubah kesimpulan program dari tidak ada kesalahan proses (0) ke terdapat kesalahan proses (1). Kedua tabel tersebut digunakan untuk membuat nilai ambang untuk menghasilkan kesimpulan final apakah terdapat kesalahan pada proses. Nilai ambang tersebut dibutuhkan untuk menanggulangi kesimpulan keliru apabila hanya digunakan nilai ambang berupa persentase data salah. Apabila program mendeteksi kesalahan dan telah mencapai nilai ambang keputusan final, maka program akan tetap memberikan keputusan bahwa terdapat kesalahan proses meskipun program sudah tidak mendeteksi kesalahan, berbeda dengan pada Gambar V.20 dan V.22 dimana program memberikan kesimpulan bahwa tidak terdapat kesalahan proses setelah memberikan kesimpulan bahwa terdapat kesalahan proses.

Pada Tabel V.3 diamati bahwa dengan semakin besarnya nilai ambang persentase data salah yang digunakan, maka jumlah kesimpulan keliru semakin berkurang. Namun, pada Tabel V.4 dapat diamati bahwa apabila nilai ambang persentase data salah terlalu besar, maka jumlah data salah yang dibutuhkan untuk mengubah kesimpulan program dari tidak ada kesalahan proses ke terdapat kesalahan proses akan semakin banyak. Berdasarkan Tabel V.3 dan V.4, program akan memberikan kesimpulan final bahwa terdapat kesalahan pada proses apabila

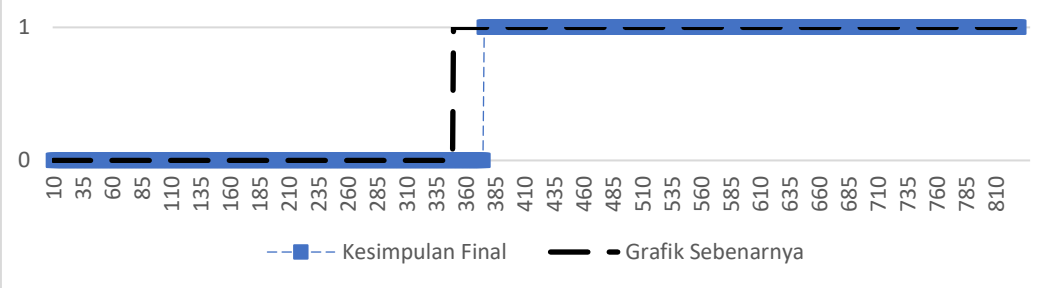
minimal terdapat 15% data salah pada data penyangga selama 10 masukan data. Nilai ambang tersebut dinilai sesuai untuk mengonpensasi kesimpulan yang keliru seperti pada Tabel V.3 namun masih cukup cepat untuk mendeteksi kesalahan proses seperti pada Tabel V.4. Berdasarkan nilai ambang keputusan final, didapatkan hasil sebagai berikut.



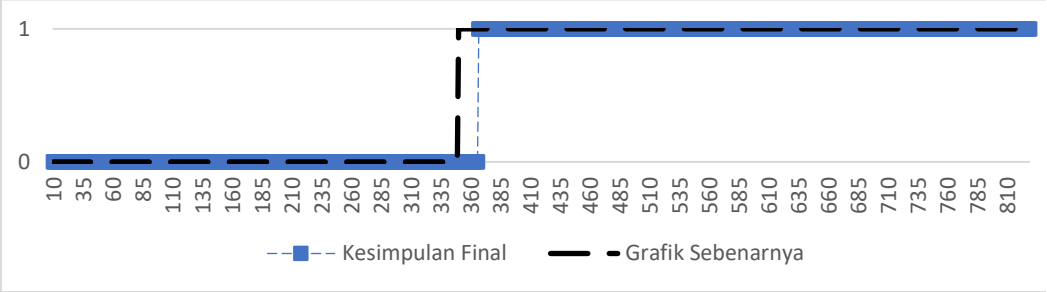
Gambar V.23 Hasil Kesimpulan Final Program Pada Data Salah Tipe 2



Gambar V.24 Hasil Kesimpulan Final Program Pada Data Salah Tipe 5



Gambar V.25 Hasil Kesimpulan Final Program Pada Data Salah Tipe 6

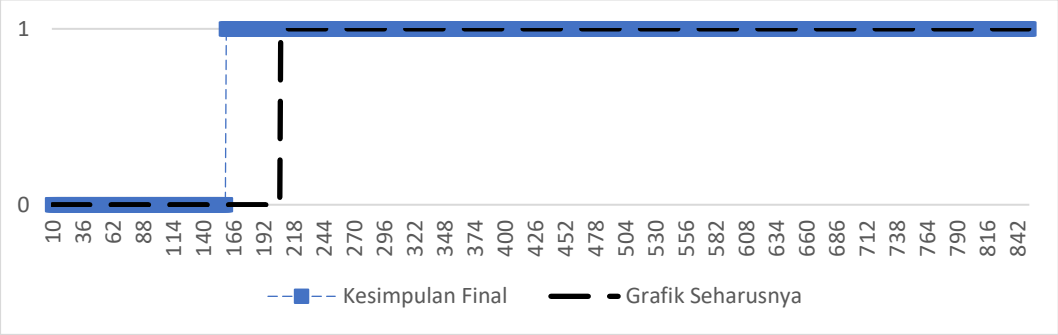


Gambar V.26 Hasil Kesimpulan Final Program Pada Data Salah Tipe 7

Gambar V.23 hingga Gambar V.26 merupakan hasil keputusan final pada program deteksi kesalahan. Dari keempat Gambar dapat dilihat bahwa program memberikan keputusan final setelah nilai ambang terpenuhi, sehingga misklasifikasi pada data sebelum 350 belum mengubah keputusan final dari tidak ada kesalahan proses (0) menjadi terdapat kesalahan proses (1). Dengan toleransi tersebut, program menjadi lebih akurat dalam memberikan keputusan final, namun memiliki konsekuensi pada jumlah data yang dibutuhkan untuk mengubah keputusan final.

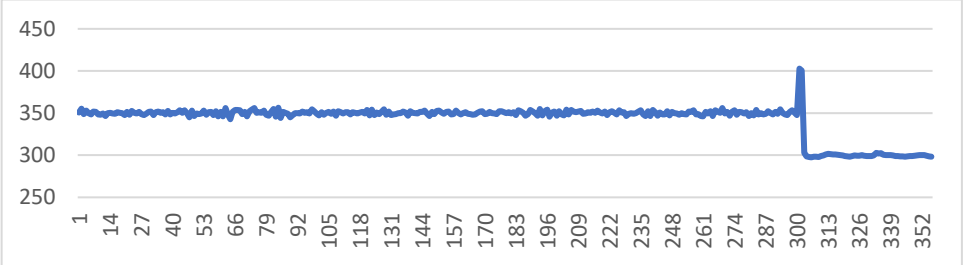
V.4. Pengujian Program Menggunakan Data Validasi

Tahap validasi dilakukan untuk menguji apakah program yang dirancang dengan parameter yang telah ditentukan sebelumnya dapat melakukan deteksi kesalahan pada data selain data latih. Pada tahap ini, program akan dijalankan untuk melakukan deteksi kesalahan pada data validasi menggunakan parameter-parameter yang telah ditentukan sebelumnya.

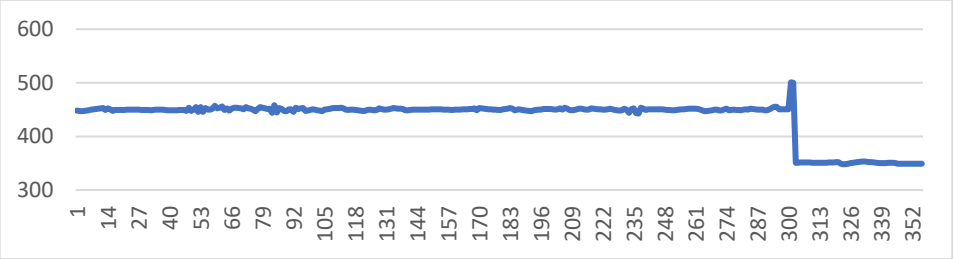


Gambar V.27 Hasil Kesimpulan Final Program Pada Data Validasi

Gambar V.27 merupakan hasil pengujian program pada data validasi. Pada pengujian yang dilakukan, program memberikan kesimpulan bahwa terdapat kesalahan terlalu awal. Hal tersebut dikarenakan terdapat data normal pada penyangga yang diproyeksikan secara terpisah dari data normal pada referensi, sehingga diklasifikasikan sebagai data salah. Misklasifikasi dari program berlangsung pada rentang yang besar, sehingga mencapai nilai ambang keputusan final dan menyebabkan program memberikan kesimpulan bahwa terdapat kesalahan yang terlalu awal. Dari Gambar V.27 dapat disimpulkan bahwa kesimpulan final bahwa terdapat kesalahan diambil karena misklasifikasi data normal. Misklasifikasi tersebut dikarenakan terdapat perbedaan antara data normal pada data latih, yaitu pada sampel ke-0 hingga sampel ke-300 dengan data normal pada data uji, yaitu pada sampel ke-301 hingga sampel ke-356 di dalam data validasi yang digunakan. Perbedaan tersebut terletak pada hasil bacaan sensor FC1 dan FC2.



Gambar V.28 Data Normal Pada Bacaan Sensor FC1

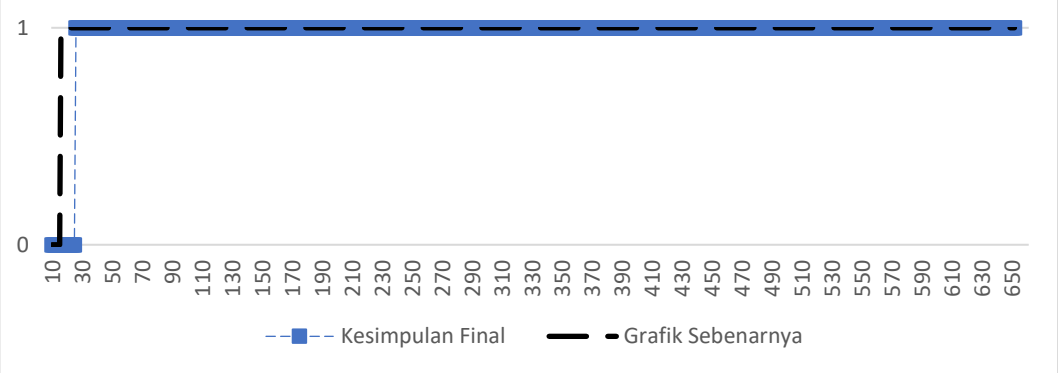


Gambar V.29 Data Normal Pada Bacaan Sensor FC2

Pada Gambar V.28 dan V.29, dapat diamati bahwa terdapat perbedaan signifikan pada data normal di dalam hasil bacaan sensor FC1 dan FC2. Perbedaan hasil bacaan sensor tersebut tidak terbatas pada data normal, melainkan juga terdapat pada data salah yakni setelah data ke-356, sehingga terdapat kemungkinan bahwa program tidak mendeteksi kegagalan sensor AI3, namun justru mendeteksi perbedaan hasil pengukuran sensor FC1 dan FC2.

Dengan perbedaan nilai hasil bacaan FC1 dan FC2 yang signifikan seperti pada Gambar V.28 dan V.29, program mengalami kesulitan untuk mendeteksi kegagalan sensor AI3. Program justru mengambil keputusan bahwa terdapat kesalahan ketika terdapat perubahan nilai hasil bacaan sensor FC1 dan FC2. Hal tersebut dikarenakan pada saat program mengumpulkan data referensi, 150 data referensi yang dikumpulkan sesuai dengan data ke-1 hingga ke-300, sehingga data setelah data ke-300 yang memiliki karakteristik yang berbeda akan dianggap sebagai data salah. Pada rentang nilai γ yang telah diuji, program tidak bisa mendeteksi kesalahan secara tepat dan justru mengambil kesimpulan bahwa terdapat kesalahan sebelum data salah masuk ke dalam program. Hal tersebut dapat diatasi dengan

memasukkan data ke-301 hingga ke-350 kedalam data referensi, sehingga program juga mengenali perubahan hasil bacaan sensor FC1 dan FC2 sebagai data normal.



Gambar V.30 Pengujian Program Pada Data Validasi Dengan Ukuran Referensi Sebesar 350 Sampel

Gambar V.30 merupakan hasil pengujian program pada data validasi dengan menggunakan ukuran referensi sebesar 350 sampel. Dengan menggunakan ukuran sampel yang lebih besar, program dapat memasukkan data ke-301 hingga ke-350 sehingga program dapat mengenali karakter data tersebut sebagai data normal. Pengujian yang dilakukan mendapatkan hasil bahwa program dapat melakukan deteksi kesalahan dengan tepat apabila data referensi yang masuk mewakili karakter data normal pada proses, sehingga ukuran referensi harus disesuaikan dengan jumlah data yang dibutuhkan untuk mendapatkan data referensi yang dapat mewakili karakter data normal pada proses. Pada pengujian tersebut, juga digunakan nilai γ sebesar $5x10^{-3}$, sehingga dapat disimpulkan bahwa untuk mendapatkan hasil yang optimal, parameter program perlu disesuaikan dengan data yang digunakan.

BAB VI
KESIMPULAN DAN SARAN

VI.1. Kesimpulan

Penelitian yang telah dilakukan bertujuan untuk merancang dan menguji performa dari program untuk melakukan deteksi kesalahan dengan fase pelatihan secara *on-the-fly*. Beberapa sampel pertama yang masuk akan digolongkan sebagai data referensi, yakni acuan data normal untuk program. Program yang dirancang menggunakan teknik *semi-supervised learning* di mana data yang masuk akan diolah oleh metode pengelompokan k-*means* untuk memberi label dari tiap data dan dilanjutkan dengan klasifikasi menggunakan SVM.

Pada program yang dirancang, sampel-sampel yang pertama masuk ke dalam program digunakan sebagai data referensi, yakni acuan data normal bagi program. Data referensi kemudian digunakan untuk membangun matriks KFDA yang digunakan untuk memperkecil jumlah fitur dari data program menjadi 2, sehingga data dapat diproyeksikan kedalam grafik 2 dimensi. Setelah itu, sampel-sampel baru akan dimasukkan kedalam penyangga untuk diolah. Terdapat 3 tahap utama ketika sampel baru masuk kedalam program, pra-pemrosesan, pengelompokan, klasifikasi. Setelah data diolah dengan *robust scaler*, sampel baru akan diproyeksikan kedalam ruang fitur KFDA menggunakan matriks KFDA. Sampel baru dan sampel lainnya didalam penyangga kemudian diolah dengan metode pengelompokan berupa pengelompokan k-*means* untuk mendapatkan label dari tiap sampel. Metode klasifikasi berupa SVM kemudian digunakan untuk membuat garis pemisah yang optimum antara data referensi dan data normal dengan data salah. Nilai ambang batas digunakan untuk mengambil kesimpulan apakah terdapat kesalahan pada proses berdasarkan hasil klasifikasi oleh SVM.

Program yang dirancang dapat mendeteksi kesalahan pada kedua kumpulan data yang digunakan. Walaupun demikian, harus dilakukan penyesuaian parameter terlebih dahulu agar didapatkan hasil yang optimum pada data latih dan data validasi, sehingga program belum dapat seutuhnya digunakan untuk deteksi kesalahan secara *on-the-fly*. Selain itu, pada pengujian menggunakan data validasi,

didapatkan pula bahwa program harus memiliki data yang mewakili karakter data normal sebagai data referensi untuk dapat melakukan deteksi kesalahan secara optimal. Walaupun parameter yang telah ditentukan sebelumnya tidak sesuai dengan data yang digunakan, parameter tersebut dapat digunakan sebagai titik mulai dalam pencarian parameter optimum untuk kedua data.

VI.2. Saran

Pada penelitian ini, dihasilkan program yang dapat melakukan deteksi kesalahan pada pabrik kimia tanpa menggunakan kumpulan data berlabel. Namun, program masih memerlukan penyesuaian parameter secara manual untuk mendapatkan hasil yang optimal. Diperlukan studi lebih lanjut untuk mencari parameter yang tepat agar didapatkan akurasi deteksi kesalahan yang optimum sembari program berjalan, sehingga program dapat seutuhnya digunakan untuk melakukan deteksi kesalahan tanpa melakukan penyesuaian parameter terlebih dahulu. Selain itu, juga diperlukan studi komparasi antara program yang diusulkan dengan metode-metode yang sudah tersedia sebelumnya.

DAFTAR PUSTAKA

[1] L. H. Chiang, E. L. Russell, and R. D. Braatz, "Fault diagnosis in chemical processes using Fisher discriminant analysis, discriminant partial least squares, and principal component analysis," *Chemom. Intell. Lab. Syst.*, vol. 50, no. 2, pp. 243–252, 2000, doi: 10.1016/S0169-7439(99)00061-1.

[2] Y. J. Yoo, "Data-driven fault detection process using correlation based clustering," *Comput. Ind.*, vol. 122, p. 103279, 2020, doi: 10.1016/j.compind.2020.103279.

[3] S. Yin, X. Gao, H. R. Karimi, and X. Zhu, "Study on support vector machine-based fault detection in Tennessee Eastman process," *Abstr. Appl. Anal.*, vol. 2014, 2014, doi: 10.1155/2014/836895.

[4] Y. C. A. P. Reddy, P. Viswanath, and B. E. Reddy, "Semi - supervised learning : a brief review," vol. 7, pp. 81–85, 2018.

[5] K. Y. Chen, L. S. Chen, M. C. Chen, and C. L. Lee, "Using SVM based method for equipment fault detection in a thermal power plant," *Comput. Ind.*, vol. 62, no. 1, pp. 42–50, 2011, doi: 10.1016/j.compind.2010.05.013.

[6] N. G. Lo, J. M. Flaus, and O. Adrot, "Review of Machine Learning Approaches in Fault Diagnosis applied to IoT Systems," *2019 Int. Conf. Control. Autom. Diagnosis, ICCAD 2019 - Proc.*, 2019, doi: 10.1109/ICCAD46983.2019.9037949.

[7] W. Liu, Z. Peng, and D. Cui, "A Review of Industrial Fault Diagnosis Based on Data-driven Methods," vol. 147, no. Ncce, pp. 889–893, 2018, doi: 10.2991/ncce-18.2018.148.

[8] A. Hajj Hassan, S. Lambert-Lacroix, and F. Pasqualini, "Real-Time Fault Detection in Semiconductor Using One-Class Support Vector Machines," *Int. J. Comput. Theory Eng.*, vol. 7, no. 3, pp. 191–196, 2015, doi: 10.7763/ijcte.2015.v7.955.

[9] A. Albalate, A. Suchindranath, D. Suendermann, and W. Minker, "A semi-supervised cluster-and-label approach for utterance classification," *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH 2010*, pp. 2510–2513, 2010.

[10] J. Liu, Y. F. Li, and E. Zio, "A SVM framework for fault detection of the braking system in a high speed train," *Mech. Syst. Signal Process.*, vol. 87, no. August 2016, pp. 401–409, 2017, doi: 10.1016/j.ymssp.2016.10.034.

[11] Z. M. Hira and D. F. Gillies, "A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data," *Comput. Math. Methods Med.*, vol. 2015, no. 1, pp. 2–4, 2015, [Online]. Available:

http://dx.doi.org/10.1155/2015/.

[12] J. Yang, Z. Jin, J. Y. Yang, D. Zhang, and A. F. Frangi, “Essence of kernel Fisher discriminant: KPCA plus LDA,” *Pattern Recognit.*, vol. 37, no. 10, pp. 2097–2100, 2004, doi: 10.1016/j.patcog.2003.10.015.

[13] Z. B. Zhu and Z. H. Song, “A novel fault diagnosis system using pattern classification on kernel FDA subspace,” *Expert Syst. Appl.*, vol. 38, no. 6, pp. 6895–6905, 2011, doi: 10.1016/j.eswa.2010.12.034.

[14] N. Amruthnath and T. Gupta, “Fault Diagnosis Using Clustering. What Statistical Test To Use for Hypothesis Testing?,” *arXiv*, vol. 6, no. 1, pp. 17–33, 2019, doi: 10.5121/mlaij.2019.6102.

[15] R. Chitrakar and H. Chuanhe, “Anomaly detection using Support Vector Machine classification with k-Medoids clustering,” *Asian Himalayas Int. Conf. Internet*, pp. 1–5, 2012, doi: 10.1109/AHICI.2012.6408446.

[16] R. G. Brereton and G. R. Lloyd, “Support Vector Machines for classification and regression,” 1998. doi: 10.1039/b918972f.

[17] C.-W. Hsu and C.-J. Lin, “A comparison of methods for multi-class support vector machines,” *Lect. Notes Comput. Sci.*, vol. 13, no. 2, pp. 415–425, 2002, doi: 10.1109/72.991427.

[18] Y. Du, W. Zhang, Y. Zhang, Z. Gao, and X. Wang, “Fault diagnosis of rotating machines for rail vehicles based on local mean decomposition - energy moment - directed acyclic graph support vector machine,” *Adv. Mech. Eng.*, vol. 8, no. 1, pp. 1–6, 2016, doi: 10.1177/1687814016629345.

[19] J. MacQueen, “SOME METHODS FOR CLASSIFICATION AND ANALYSIS OF MULTIVARIATE OBSERVATIONS,” 1967, doi: 10.1.1.308.8619.

[20] D. Arthur and S. Vassilvitskii, “k-means++: The Advantages of Careful Seeding,” in *SODA ’07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007, pp. 1027–1035.

[21] M. A. Syakur, B. K. Khotimah, E. M. S. Rochman, and B. D. Satoto, “Integration K-Means Clustering Method and Elbow Method for Identification of the Best Customer Profile Cluster,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 336, no. 1, 2018, doi: 10.1088/1757-899X/336/1/012017.

[22] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” 1996, [Online]. Available: http://www.dbs.informatik.uni-muenchen.de/ dbs/project/publikationen/veroeffentlichun- gen.html.

[23] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, “DBSCAN Revisited, Revisited,” *ACM Trans. Database Syst.*, vol. 42, no. 3, pp. 1–21, 2017, doi: 10.1145/3068335.

[24] A. Ng, “CS229 Lecture notes Margins : SVM,” *Intell. Syst. their Appl. IEEE*, vol. pt.1, no. x, pp. 1–25, 2012, [Online]. Available: http://www.stanford.edu/class/cs229/notes/cs229-notes3.pdf.

[25] L. J. P. Van Der Maaten, E. O. Postma, and H. J. Van Den Herik, “Dimensionality Reduction: A Comparative Review,” *J. Mach. Learn. Res.*, vol. 10, pp. 1–41, 2009, doi: 10.1080/13506280444000102.

[26] Q. Yang and K. A. Loparo, “Model-based and data driven fault diagnosis methods with applications to process monitoring,” *Case West. Reserv. Univ.*, no. May, pp. 1–191, 2004.

[27] X. Wan, “Influence of feature scaling on convergence of gradient iterative algorithm,” *J. Phys. Conf. Ser.*, vol. 1213, no. 3, 2019, doi: 10.1088/1742-6596/1213/3/032021.

[28] V. N. G. Raju, K. P. Lakshmi, V. M. Jain, A. Kalidindi, and V. Padma, “Study the Influence of Normalization/Transformation process on the Accuracy of Supervised Classification,” *Proc. 3rd Int. Conf. Smart Syst. Inven. Technol. ICSSIT 2020*, no. Icssit, pp. 729–735, 2020, doi: 10.1109/ICSSIT48917.2020.9214160.

[29] R. Isermann, “Model-based fault-detection and diagnosis - Status and applications,” *Annu. Rev. Control*, vol. 29, no. 1, pp. 71–85, 2005, doi: 10.1016/j.arcontrol.2004.12.002.

[30] V. Lampkin, W. T. Leong, L. Olivera, S. Rawat, N. Subrahmanyam, and R. Xiang, “Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry,” *IBM Redbooks*, p. 270, 2012, [Online]. Available: http://books.google.com/books?hl=en&lr=&id=F_HHAgAAQBAJ&oi=fnd&pg=PP1&dq=Building+Smarter+Planet+Solutions+with+MQTT+and+IBM+WebSphere+MQ+Telemetry&ots=2CE3veSYOD&sig=CR-rE8p9kZqMzAno6nzLy7G29qg.

[31] J. J. Downs and E. F. Vogel, “A Plant-wide Industrial Problem Process,” *Comput. Chem. Eng.*, vol. 17, no. 3, pp. 245–255, 1993.

[32] K. S. Brooks and M. Bauer, “Sensor validation and reconstruction: Experiences with commercial technology,” *Control Eng. Pract.*, vol. 77, no. March 2017, pp. 28–40, 2018, doi: 10.1016/j.conengprac.2018.04.003.

LAMPIRAN