# HALAMAN JUDUL

**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 PLAGIARISME

**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 | Anggota Penguji |
|  |  |
| Nama Lengkap Penguji Utama  NIP. ..... | 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

# HALAMHALAMAN 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. | 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 kuppersembahkan untuk diriku sendiri …*

… *Tetapkan tujuan, mantapkan langkah*…

# KATA PENGANTAR

**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 fault detection pada umumnya. Menggunakan support vector machines pada metode semi-supervised learning, data proses pabrik dapat diklasifikasi kedalam data normal dan data faulty.

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 ISI

[HALAMAN JUDUL i](#_Toc57984693)

[PERNYATAAN BEBAS PLAGIARISME ii](#_Toc57984694)

[HALAMHALAMAN PERSEMBAHAN iv](#_Toc57984695)

[KATA PENGANTAR vii](#_Toc57984696)

[DAFTAR ISI ix](#_Toc57984697)

[DAFTAR TABEL xi](#_Toc57984698)

[DAFTAR GAMBAR xii](#_Toc57984699)

[DAFTAR LAMBANG DAN SINGKATAN xiii](#_Toc57984700)

[ABSTRACT xv](#_Toc57984701)

[BAB I PENDAHULUAN 16](#_Toc57984702)

[I.1. Latar Belakang 16](#_Toc57984703)

[I.2. Perumusan Masalah 18](#_Toc57984704)

[I.2.1. Batasan Masalah 19](#_Toc57984705)

[I.3. Tujuan Penelitian 19](#_Toc57984706)

[I.4. Manfaat Penelitian 19](#_Toc57984707)

[BAB II TINJAUAN PUSTAKA 21](#_Toc57984708)

[II.1. Isi Tinjauan Pustaka 21](#_Toc57984709)

[BAB III DASAR TEORI 29](#_Toc57984710)

[III.1. Semi-Supervised Learning 29](#_Toc57984711)

[III.2. Support Vector Machines (SVM) 30](#_Toc57984712)

[III.3. Metode Clustering 37](#_Toc57984713)

[III.3.1. K-means Clustering 37](#_Toc57984714)

[III.3.2. Density-based Spatial Clustering of Applications With Noise 39](#_Toc57984715)

[III.4. Kernel 41](#_Toc57984716)

[III.5. Kernel Fisher Discriminant Analysis 42](#_Toc57984717)

[III.6. Scaling 45](#_Toc57984718)

[III.7. Fault Detection and Diagnosis 46](#_Toc57984719)

[III.8. MQTT 48](#_Toc57984720)

[BAB IV PELAKSANAAN PENELITIAN 51](#_Toc57984721)

[IV.1. Alat dan Bahan Penelitian 51](#_Toc57984722)

[IV.2. Tata Laksana Penelitian 60](#_Toc57984723)

[IV.2.1. Pengujian KFDA dan SVM 61](#_Toc57984724)

[IV.2.2. Perancangan Program 62](#_Toc57984725)

[IV.2.3. Pembangunan Program 63](#_Toc57984726)

[IV.2.4. Pengujian Program Menggunakan Data Validasi 63](#_Toc57984727)

[BAB V Hasil dan Pembahasan 65](#_Toc57984728)

[V.1. Pengujian KFDA dan SVM 65](#_Toc57984729)

[V.1.1. Rangka kerja *fault detection* menggunakan KFDA dan SVM 65](#_Toc57984730)

[V.1.2. Pengujian KFDA dan SVM untuk *fault detection* 67](#_Toc57984731)

[V.2. Perancangan Program *Fault Detection* 70](#_Toc57984732)

[V.2.1. Pengujian Ukuran Referensi dan *Buffer* 70](#_Toc57984733)

[V.2.2. Pengujian Metode Clustering 75](#_Toc57984734)

[V.2.3. Pengujian K-Means Clustering 81](#_Toc57984735)

[V.3. Pembangunan Program 85](#_Toc57984736)

[V.3.1. Tahapan Program 85](#_Toc57984737)

[V.4. Pengujian Program Menggunakan Data Validasi 93](#_Toc57984738)

[V.4.1. Hasil Pengujian Program Menggunakan Data Validasi 93](#_Toc57984739)

[V.4.2. Penyesuaian Parameter Program Untuk Data Validasi 94](#_Toc57984740)

[V.4.3. Komparasi dengan strategi konvensional 98](#_Toc57984741)

[BAB VI KESIMPULAN DAN SARAN 99](#_Toc57984742)

[VI.1. Kesimpulan 100](#_Toc57984743)

[VI.2. Saran 99](#_Toc57984744)

[DAFTAR PUSTAKA 100](#_Toc57984745)

[LAMPIRAN 103](#_Toc57984746)

[LAMPIRAN A CONTOH LISTING PROGRAM *INPUT SERPENT* 104](#_Toc57984747)

[LAMPIRAN B TABEL DENSITAS MATERIAL 106](#_Toc57984748)

# DAFTAR TABEL

**DAFTAR TABEL**

[Tabel IV.IV.1. Perangkat Keras Penelitian 51](#_Toc57984749)

[Tabel IV.2 Perangkat Lunak Penelitian 51](#_Toc57984750)

[Tabel IV.3 Variabel pada Tennessee Eastman Process 56](#_Toc57984751)

[Tabel IV.4 Deskripsi Jenis Data pada Data Latih 57](#_Toc57984752)

[Tabel IV.5 Variabel pada Data Validasi 60](#_Toc57984753)

[Tabel V.1. *Confusion Matrix* dari KFDA-SVM Pada Personalan Multi-Kelas 69](#_Toc57984754)

[Tabel V.2. *Confuison Matrix* dari KFDA-SVM Pada Persoalan 2 Kelas 69](#_Toc57984755)

# DAFTAR GAMBAR

**DAFTAR GAMBAR**

[Gambar II.1 Framework SVM untuk deteksi kesalahan pada HST 22](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984756)

[Gambar II.2 Hasil *Feature Extraction* Menggunakan Berbagai Metode 27](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984757)

[Gambar III.1 Klasifikasi Menggunakan SVM 30](#_Toc57984758)

[Gambar III.2 Perbedaan One versus all dan One versus one 35](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984759)

[Gambar III.3 Penentuan Kelas Menggunakan DAG 36](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984760)

[Gambar III.4 Ilustrasi cara kerja metode DBSCAN 41](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984761)

[Gambar III.5 Skema Kerja MQTT 50](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984762)

[Gambar IV.1 Diagram P&ID dari Tennessee Eastman Process 54](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984763)

[Gambar IV.2 Diagram P&ID Data Validasi 58](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984764)

[Gambar V.1 Rangka Kerja Fault Detection 65](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984765)

[Gambar V.2 Hasil Pengujian Menggunakan KFDA-SVM 68](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984766)

[Gambar V.7 Pengujian Pada 72](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984767)

[Gambar V.7 Pengujian Pada 72](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984768)

[Gambar V.7 Pengujian Pada 72](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984769)

[Gambar V.7 Pengujian Pada 72](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984770)

[Gambar V.7 Pengujian Pada 72](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984771)

[Gambar V.11 Jumlah Faulty Sample Minimum 83](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984772)

[Gambar V.11 Nilai WCSS Minimum Pada Ukuran Referensi = 100 83](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984773)

[Gambar V.11 Nilai WCSS Minimum Pada Ukuran Referensi = 100 83](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984774)

[Gambar V.11 Nilai WCSS Minimum Pada Ukuran Referensi = 100 83](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984775)

[Gambar V.12 Akurasi Program Pada Data Latih Menggunakan Variasi Bobot Kelas Normal 88](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984776)

[Gambar V.13 Akurasi Program Pada Jenis *Fault* Menggunakan Variasi Bobot Kelas 89](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984777)

[Gambar V.14 Pengaruh Bobot Kelas Pada Akurasi Klasifikasi Data Normal 91](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984778)

[Gambar V.15 Akurasi Program Pada Data Validasi Menggunakan Ukuran Referensi = 150 Dengan Variasi Nilai Gamma 95](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984779)

[Gambar V.16 Akurasi Program Pada Data Validasi Menggunakan Ukuran Referensi = 300 Dengan Variasi Nilai Gamma 97](file:///C:\Users\Arzy\Desktop\TA_Arzy.docx#_Toc57984780)

# DAFTAR LAMBANG DAN SINGKATAN

**Lambang Romawi**

|  |  |  |
| --- | --- | --- |
| *Lambang* | *Kuantitas* | *Satuan* |
|  | Matriks kernel | - |
|  | *X* setelah diproyeksikan menggunakan KPCA | - |
|  | Sampel pada *X* setelah diproyeksikan menggunakan KFDA | - |
| *b* | *Threshold* *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 | - |
|  | Matriks kernel terpusat | - |
|  | 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 | - |
|  | *Langrange Multiplier* | - |
|  | *Kernel Mapping* | - |
|  | Standar deviasi | - |
|  | *Eigenvalue* dari KPCA | - |
| *λ* | *Eigenvector* dari KPCA | - |
|  | *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* |  |
|  | Derajat polinomial |  |

|  |  |
| --- | --- |
| **Singkatan** |  |
| SVM | *Support Vector Machines* |
| KFDA | *Kernel Fisher Discriminant Analysis* |
| FDD | *Fault Detection & Diagnosis* |
| KMC | *K-Means Clustering* |
| VMM | *Virtual Machine Manager* |
| HST | *High Speed Train* |
| RBF | *Radial Basis Function* |
| DAG | *Directed Acylic Graph* |
| KPCA | *Kernel Principal Component Analysis* |
| DBSCAN | *Density Based Spectral Clustering of Applications*  *With Noise* |

1. **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 Nuklir

**INTISARI**

*Fault detection* berbasis data pada pabrik kimia umumnya membutuhkan kumpulan data berjumlah besar berisikan sampel-sampel ketika pabrik beroperasi secara normal dan ketika terdapat *fault* pada proses pabrik, yang kemudian tiap sampel diberi label yang sesuai oleh ahli. Penelitian ini akan merancang dan menguji performa dari metode *fault detection* dengan *on-the-fly ­semi-supervised learning* menggunakan metode *cluster-than-label*, dimana pelatihan model *fault detection* dapat dilakukan sembari pabrik beroperasi, sehingga cocok untuk diterapkan apabila pabrik belum memiliki kumpulan data.

Metode yang dirancang akan menggunakan data referensi dan *buffer*, dimana data referensi adalah sampel-sampel pertama dalam jumlah yang telah ditentukan sebelumnya sebagai acuan data normal bagi program, dan data *buffer* berisi sampel yang masuk setelahnya. Digunakan metode KFDA untuk *pre-processing*, dan *k-means* untuk *clustering* yang dilanjutkan oleh SVM untuk *classification*. Dari data *training,* dihasilkan parameter yakni, ukuran data referensi sebesar 150 sampel, ukuran *buffer* sebesar 50, dan gamma sebesar 0.0085. Dihasilkan akurasi sebesar 77,17% untuk data *training* dan 63,03% untuk data validasi. Pada penelitian ini, penyetelan dilakukan pada parameter ukuran data referensi dan gamma. Apabila dilakukan penyesuaian parameter, akurasi data validasi mencapai 81,97%.

***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.

# ABSTRACT

**Implementation of Support Vector Machines-Based Fault Detection Using On-The-Fly Semi-Supervised Learning Method In 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 Nuclear Engineering

**ABSTRACT**

Common strategy of data-driven fault detection on chemical plants is to train the model using collected process data under normal andfaulty conditions which were labeled accordingly by experts. This research tries to design and evaluate the performance of a fault detection method using on-the-fly semi-supervised learning, so that the model training can be done while 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 for a reference of 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 are chosen: 150 as reference size, 50 as buffer size, and 0.0085 for gamma. Accuracy of 77,17% for training data and 63,03% for validation data are obtained using said parameter values. To yield optimum results, parameters should be tuned according to the data used. In this research, reference size and gamma are tuned, resulting in accuracy of 81,97% for validation data.

***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.

# PENDAHULUAN

## Latar Belakang

Pada pabrik yang modern, terjadi berbagai proses untuk menghasilkan barang jadi dengan kualitas tertentu menggunakan bantuan mesin berupa kumpulan sensor dan aktuator. Sensor, kontroler, aktuator bekerja sama untuk menghasilkan produk dengan kualitas dan kuantitas yang sudah ditentukan sebelumnya, dimana sensor memperhatikan nilai – nilai dari kondisi bahan lalu mengirimkan sinyal kepada kontroller. *Controller* kemudian mengirimkan perintah ke aktuator untuk mengolah produk berdasarkan nilai – nilai kondisi bahan. Variabel proses kemudian disimpan di dalam database tiap satuan waktu tertentu.

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 atau fault didefinisikan sebagai perilaku abnormal pada proses yang berhubungan dengan kegagalan mesin, kelelahan mesin, atau gangguan eksrim pada proses [1]. *Fault* 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 fault detection dikembangkan selama 20 tahun terakhir [2], baik yang berbasis model pabrik (*model-based*), maupun berbasis data historis (*data-driven*). Dengan fault detection, deteksi kesalahan dalam proses – proses pabrik yang kompleks dapat dilakukan sedini mungkin.

Berdasarkan database yang telah dikumpulkan, *fault detection* berbasis data historis (*data-driven*) dapat dilakukan. *Data-driven fault detection* bekerja dengan cara mempelajari database yang dimiliki pabrik sehingga kemudian dapat mengklasifikasi data – data baru dari pabrik tersebut. Kesulitan deteksi *fault* pada pabrik diakibatkan oleh jumlah variabel proses dan karakteristik data, dimana semakin banyak variabel proses yang terlibat, maka proses deteksi fault membutuhkan waktu semakin banyak. *Fault detection* sulit untuk dilakukan oleh manusia karena keterbatasan daya pikir untuk mengolah dan mengklasifikasikan data dengan jumlah variabel proses yang besar, sehingga digunakan *machine learning* karena dapat mengolah data dengan cepat dan akurat meskipun dengan jumlah variabel proses yang besar.

Salah satu metode klasifikasi yang sering digunakan adalah support vector machines (SVM) karena akurasi SVM lebih tinggi dibandikan metode *machine learning* tradisional lain pada berbagai aplikasi [3]. Meskipun SVM hanya akurat untuk data yang bersifat *linearly­ seperable* ­­atau dapat dipisahkan secara linier, data dari pabrik yang bersifat *­non-linearly seperable* dapat diolah terlebih dahulu menggunakan metode feature extraction kemudian diklasifikasikan menggunakan SVM. Dengan mempelajari kelas dari masing – masing data, feature extraction berfungsi untuk mencari proyeksi data dimana data menjadi *linearly seperable*, sehingga kemudian dapat diklasifikasikan menggunakan metode klasifikasi.

Perusahaan dalam usaha untuk mempertahankan dan meningkatkan kualitas produknya, perlu melakukan pengawasan yang ketat terhadap proses yang berjalan di dalam pabrik. *Fault* harus dideteksi supaya perusahaan dan pabrik dapat segera melakukan *counter-measure­* dari *fault*. *Fault detection* dengan *machine learning* memungkinkan perusahan untuk menghindari kerugian dengan cara memperingatkan sedini mungkin bahwa terjadi fault pada proses. Dengan menggunakan *fault detection*, perusahaan dapat meminimalisir resiko ekonomi dan keselamatan.

Data yang disimpan oleh sebuah pabrik merupakan kumpulan sampel tiap satu satuan waktu berisikan beberapa atau semua nilai pengukuran sensor terhadap kondisi bahan pada proses, sinyal kontrol pada aktuator, serta *output* dari actuator pada satu waktu. Kumpulan data ini dapat digunakan untuk mendeteksi apabila pada sebuah sampel baru terdapat kesalahan pada proses. Untuk mengklasifikasi apakah sebuah sampel termasuk normal atau *faulty*, dapat digunakan salah satu metode klasifikasi *machine learning* yaitu *support vector machines* (SVM)*.*

Umumnya, pembangunan model mesin belajar untuk fault *detection* menggunakan teknik *supervised learning* yangmembutuhkan kumpulan data dalam jumlah yang besar, berisi dengan sampel ketika pabrik beroperasi secara normal dan sampel ketika terdapat *fault* dalam proses pabrik, yang tiap sampelnya sudah diberi label oleh para ahli untuk memberikan keterangan apakah sebuah sampel merupakan sampel normal atau sampel *faulty*. Proses pengumpulan data ini 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 berjumlah besar. 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.

Pada penelitian ini, teknik *semi-supervised learning* akan dilakukan secara *on-the-fly*, sehingga pembangunan model *fault detection* dapat dilakukan sembari program beroperasi. Dengan begitu, program *fault detection* yang dirancang pada penelitian ini dapat digunakan untuk melakukan *fault detection* apabila kumpulan data berlabel belum tersedia.

## Perumusan Masalah

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

### 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 diambil pada sacac.org.za.
2. Pengujian dan metode-metode yang digunakan merupakan metode *fault detection* secara *data-driven.*
3. Pengujian dilakukan menggunakan bahasa pemrograman Python.

## Tujuan Penelitian

Penelitian ini bertujuan untuk memperoleh rancangan dari program yang dapat mendeteksi *fault* 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.

## Manfaat Penelitian

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

*Fault detection* 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 dimana industri dapat menggunakan atau mengembangkan program yang telah dirancang untuk melakukan *fault detection* apabila kumpulan data belum tersedia.

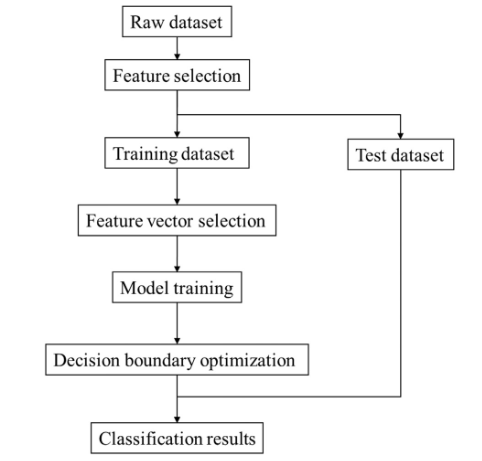
# TINJAUAN PUSTAKA

## Isi Tinjauan Pustaka

### SVM

Sampai saat ini, *support vector machines* (SVM) sudah banyak diimplementasikan pada berbagai keperluan, baik regresi ataupun klasifikasi. Maraknya implementasi SVM ini dikarenakan SVM mampu menghasilkan akurasi yang lebih tinggi dibandikan metode *machine learning* tradisional lain pada berbagai aplikasi [3]. Salah satu bidang mengimplementasikan SVM sebagai classifier adalah *fault detection*, dimana *fault detection* sendiri menjadi fokus oleh banyak peneliti selama 20 tahun terakhir [2]. Tidak sedikit penelitian *fault detection* yang mendapatkan hasil bahwa SVM dan pengembangannya mampu mendeteksi data faulty pada sistem yang diimplementasikan.

J. Liu, Y.-F. Li dan E. Zio [4] meneliti tentang implementasi *framework* SVM dalam mendeteksi kesalahan pada sistem pengereman kereta berkecepatan tinggi. Data dimana terdapat *fault* 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, dimana mayoritas data merupakan data pada kondisi normal. *Framework* SVM yang diusulkan dapat dibagi menjadi 4 tahap: *feature selection, feature vector selection, model construction* dan *decision boundary optimization*. *Feature selection* dilakukan dengan cara menyisihkan variabel yang tidak relevan terhadap *fault detection* 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 *feature vector* menggunakan metode kernel. Tahap *feature vector selection* kemudian menyisihkan *feature vector* yang tidak informatif terhadap *fault detection* untuk menyingkat waktu komputasi. Menggunakan proyeksi data pada *feature vector* yang tersisa, SVM dikonstruksi dan kemudian dioptimasi. Pada 15 dataset yang digunakan framework SVM yang ditawarkan menghasilkan 12 nilai F-*measurement* dan 9 nilai G-*mean value* yang lebih tinggi dibandingkan 3 metode *benchmark*.



Gambar II.1 Framework SVM untuk deteksi kesalahan pada HST

[4]

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 noise serta untuk menormalisasi data. Setelah itu, *feature selection* 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 *Radial Basis Function* (RBF) *kernel* dan kemudian dioptimalkan dengan mencari nilai parameter yang optimal. Hasil dari penelitian menunjukkan bahwa *fault detection* menggunakan SVM mengalahkan performa metode *Linear Discriminant Analysis* (79.95%) dan *Back Propagation Neural Network* (85.57%), dimana SVM dapat mendeteksi fault mengidentifikasi jenis fault pada pembangkit listrik termal dengan akurasi diatas 91.93%

### Metode Feature Extraction

Pada penelitian – penelitian diatas dapat dilihat bahwa dalam melakukan fungsi klasifikasi pada *fault detection*, 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 *feature selection* dan *feature extraction*. *Feature selection* 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, dimana 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 *feature extraction* dimana data diproyeksikan pada sebuah *feature space* baru yang memiliki jumlah dimensi lebih sedikit dibanding jumlah variabel sebenarnya.

Z. M. Hira and D. F. Gillies [6] melakukan penelitian untuk mengulas aplikasi *feature selection* dan *feature extraction* pada data genetik. Pada penelitian ini, metode *feature selection* dan *feature extraction* 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 *feature selection* dan *feature extraction*. Terdapat total 4 metode yang digunakan dari *feature selection*, yakni: ReliefF, *Information Gain*, *Information Gain Ratio*, dan X2-*statistics*. Sedangkan pada metode *feature extraction* digunakan metode PCA, *locally linear embedding* (LLE), dan Isomap. Penelitian ini menyimpulkan bahwa *feature extraction* menghasilkan akurasi SVM yang lebih tinggi serta dapat mengurangi overfitting dari SVM, namun feature selection membutuhkan waktu komputasi yang lebih cepat serta menjaga karakteristik data sehingga dapat diinterpretasikan.

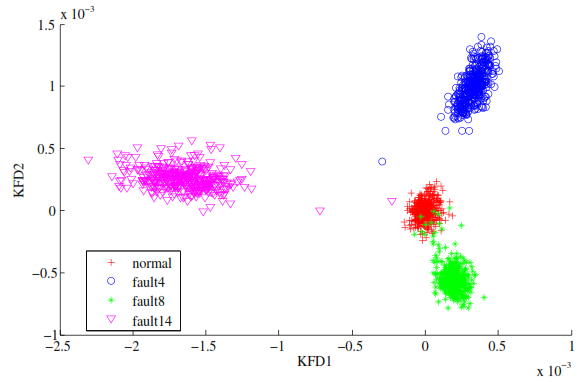
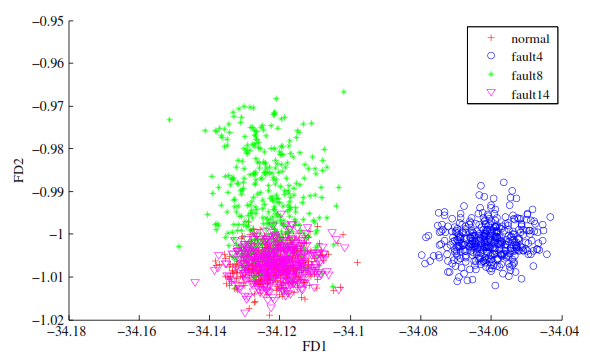
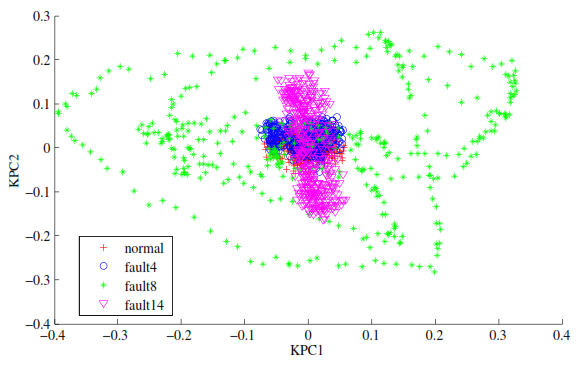
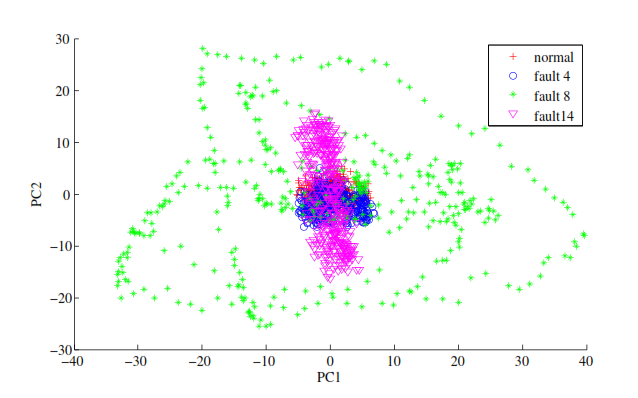
Penelitian yang dilakukan oleh L. H. Chiang, E. L. Russell dan R. D. Braatz membandingkan beberapa metode feature extraction dalam *fault detection* dan *fault diagnosis* pada data proses Tennessee Eastman. Metode *feature extraction* yang dibandingkan adalah *fisher discriminant analysis* (FDA), *discriminant partial least squares* (DPLS), dan *principal component analysis* (PCA). Hasil reduksi dimensi dari ketiga metode tersebut akan diklasifikasi menggunakan T2-*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 0.742, 0.609, dan 0.667 menggunakan klasifikasi T2-*statistics*, Q-*statistics*, serta kombinasi (T2-*statistics* dan Q-*statistics*) secara berurutan. DPLS memiliki rerata misklasifikasi rata-rata 0.56933 dan FDA memiliki rerata misklasifikasi paling kecil yaitu sebesar 0.206. PCA memiliki rerata misklasifikasi yang paling besar karena sebagai PCA membaca data normal dan data *faulty* sebagai kelas yang sama, dan hanya berusaha memaksimalkan variansi data. Di sisi lain, FDA membaca data pada masing-masing kelas dan berusaha membuat *feature space* dimana proyeksi data memiliki variansi antar kelas yang besar. Pada studi ini peneliti juga menyimpulkan bahwa kemampuan FDA yang superior dalam *fault diagnosis* merupakan kemampuan yang inheren, dan bukan dikarenakan model *order selection criteria* yang digunakan pada studi. [1]

Menggunakan persamaan KFDA yang dikembangkan oleh Yang, **[7]** melakukan penelitian untuk menguji implementasi KFDA pada *fault detection* menggunakan data proses Tennessee Eastman. Pada penelitian ini, data proses Tennessee Eastman diolah menggunakan PCA, KPCA, FDA, dan KFDA sebagai metode *feature extraction* sehingga data diproyeksikan kedalam *feature space* 2 dimensi dan kemudian diklasifikasikan menggunakan *k-nearest neighbor* (KNN) dan *gaussian mixture model* (GMM) untuk mengidentifikasi apakah sebuah data merupakan data normal, *fault type* 4*, fault type* 8*,* atau *fault type* 14.Penelitian menunjukkan bahwa PCA dan KPCA memproyeksikan data menjadi berpusat pada titik pusat grafik dan menghasilkan akurasi klasifikasi sebesar 60.125% dan 60.8125% menggunakan KNN secara berurutan serta 64.3125% dan 64% menggunakan GMM secara berurutan. FDA berhasil memisahkan *fault type* 4tetapi data normal, *fault type* 8*,* dan *fault type* 14saling tumpang tindih. *Feature extraction dengan* FDA memiliki akurasi klasifikasi sebesar 58.1875% menggunakan GMM dan 56.75% menggunakan KNN. KFDA berhasil memisahkan 4 kelas menjadi 4 kluster dengan catatan ada beberapa proyeksi *fault type* 8 yang tumpang tindih dengan data normal. Secara keseluruhan, *feature extraction dengan* KFDA memiliki akurasi klasifikasi sebesar 92.75% menggunakan GMM dan 90.8125% menggunakan KNN.Pada penelitian yang telah disebutkan, SVM mampu mendeteksi dan mengidentifikasi kesalahan pada sistem dimana ia diaplikasikan. Namun karena keterbatasan SVM dimana ia hanya memiliki performa baik pada persoalan linier, SVM tidak dapat mendeteksi kesalahan pada sistem dengan data yang non-linier. Pada penelitian Zhu dan Song [1], KFDA terbukti dapat memproyeksikan data nonlinier menjadi 4 kluster yang terpisah pada *feature space* 2 dimensi.

Gambar . Hasil *Feature Extraction* Menggunakan Berbagai Metode

(b) KPCA

(a) PCA



(c) FDA

(d) KFDA

**[1]**

### Metode On-The-Fly Fault Detection

Fault detection 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 *training set*, apabila sebuah data baru memiliki karakteristik yang berbeda dari data pada *training set*, maka rawan terjadi misklasifikasi terhadap data baru tersebut. Pengumpulan data berlabel merupakan proses yang memakan waktu yang lama dan memakan biaya, dimana data harus dikumpulkan dari setiap kondisi yang mungkin terjadi kemudian dilabelkan oleh ahli [2].

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. Salah satu metode *unsupervised* yakni clustering umumnya digunakan untuk memberikan pemahaman tambahan pada sebuah data, metode ini banyak digunakan pada berbagai aplikasi seperti segmentasi customer, analitik data besar, analitik suara, dan deteksi penipuan [3].

Penelitian pada [4]mengajukan metode *fault detection* pada proses produksi wafer semikonduktor. Metode yang diajukan dapat melakukan *fault detection* 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 *feature selection* 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 *feature selection* kemudian diklasifikasikan oleh 1-Class SVM sebagai data *faulty* 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 *feature selection* dan *classifier* akan dilakukan kembali untuk memperbarui model yang sudah dibangun. Terdapat 2 skenario *sliding window* yang diajukan, skenario pertama merupakan skenario dimana 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* dimana 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, dimana 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 *false alarm rate* yang lebih rendah, sedangkan skenario kedua menghasilkan *detection rate* yang lebih tinggi.

*Semi-supervised learning* mengombinasikan kedua metode sebelumnya untuk saling melengkapi kekurangannya. Dimana *supervised learning* membutuhkan data training 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, dimana metode semi-supervised learning membangun model berdasarkan data berlabel dalam jumlah yang kecil, dan memperlakukan data lain sebagai data uji [2]**.** Penelitian [2] 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 *clustering* untuk memberikan *pseudo-label* dari sebuah dataset dimana *pseudo-label* tersebut kemudian digunakan oleh metode *classifier* untuk menentukan label akhir dari tiap data dan untuk membentuk batas pemisah. Penelitian [5]menguji metode *clustering* yang diikuti oleh metode *classification* untuk mendeteksi anomali, dimana gabungan metode k-*medoid* dan metode SVM mendapatkan rata-rata akurasi sebesar 99.43% pada dataset kecil (10000 sampel) dan 99.21% pada dataset besar (100000 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 *clustering* dengan diberikan *pseudo-label*, dan diikuti oleh metode *classification* untuk memberikan label akhir dari kelompok-kelompok tersebut yang mempertimbangkan *pseudo-label* serta proyeksi data normal sebenarnya pada *feature-space*. Pada penelitian ini, tahap *clustering* dan tahap *classification* dilakukan sebagai 2 proses independen, seperti pada penelitian [5] dan [6]. Penelitian ini akan menggunakan metode *cluster-then-label* untuk dapat melakukan *fault detection* secara *real-time* dengan *on-the-fly learning*. Penelitian ini juga akan menggunakan metode *sliding window* yang digunakan pada penelitian [4]untuk mendeteksi kesalahan dengan membangun model *clustering* dan *classifier* ketika data baru masuk kedalam *sliding window*, namun pada penelitian ini, data yang diproses merupakan data yang terdapat pada *sliding window* karena digunakan metode *clustering* terlebih dahulu. Untuk dapat melakukan *on-the-fly learning,* alih-alih menggunakan data historik untuk mendapatkan karakteristik data normal seperti pada penelitian [4], karakteristik data normal didapatkan melalui data-data awal yang masuk sebagai acuan data normal.Digunakan metode kernel fisher discriminant analysis sebagai metode *feature extraction* untuk menangkap karakteristik data normal serta memproyeksikan data normal pada sebuah persebaran yang berbeda dengan data *faulty*. Setelah dilakukan *clustering* 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 *fault* dapat segera terdeteksi oleh program meskipun jumlah data *fault* relatif kecil dibandingkan jumlah data normal pada *sliding window*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Peneliti | Metode | | Sumber Data Latih | Mode Implementasi |
| Kategori Metode | Metode yang digunakan |
| Chen, Chen, Chen, dan Lee [7] | *Supervised* | SVM | Data histori | *Offline* |
| Albalate, Suchindranath, Suendermann, dan Minker | *Semi-supervised* | *Cluster-then-label* (SVM) | Data histori | *Offline* |
| Hassan, Lambert-Lacroix, dan Pasqualini [4] | *Unsupervised* | *One-Class SVM* | Data histori dan *real-time* | *Online* |
| Hasil penelitian yang diharapkan | *Semi-supervised* | *Cluster-then-label* (SVM) | Data *real-time* | *Online* |

# DASAR TEORI

## Semi-Supervised Learning

Teknik mesin belajar (*machine learning*) umumnya dibagi menjadi 2, yakni: *supervised learning* dan *unsupervised learning*. Metode *supervised* merupakan teknik *machine learning* dimana model menggunakan kumpulan data yang telah disediakan sebelumnya untuk mempelajari output dari tiap data, sehingga model kemudian dapat memprediksi *output* 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. 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 [2].

Sedangkan pada teknik *machine learning*, metode *unsupervised* merupakan metode yang tidak membutuhkan pengetahuan sebelumnya mengenai output dari data tersebut. Fungsi dari metode *unsupervised* adalah memberikan perkiraan *output* berdasarkan struktur dari kumpulan data yang disediakan. Metode *clustering* umumnya digunakan untuk memberikan pemahaman tambahan pada sebuah data, metode ini banyak digunakan pada berbagai aplikasi seperti segmentasi *customer*, analitik data besar, analitik suara, dan deteksi penipuan [3].

*Semi-supervised learning* mengombinasikan kedua metode sebelumnya untuk saling melengkapi kekurangannya. Dimana *supervised learning* membutuhkan data *training* 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 dapat memeberi label dari data. *Semi-supervised learning* membangun model berdasarkan data berlabel dalam jumlah yang kecil, dan memperlakukan data lain sebagai data uji [2].

## 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. [12]. Metode klasifikasi berbasis SVM memiliki sifat generalisasi yang lebih baik dikarenakan SVM berbasis pada prinsip struktur minimalisir resiko [13]. 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 titik data pada sebuah kelas yang bertindak sebagai batas dari kelas tersebut.



[13]

Gambar . Klasifikasi Menggunakan SVM

Gambar 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 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 tersebut terdapat beberapa contoh alternatif garis batas.

Garis batas terbaik ditentukan dengan mencari titik data terdekat pada kedua kelas dari garis batas tersebut kemudian mengukur *margin*. Titik data terdekat dengan garis batas disebut dengan *support vector*, dan *margin* merupakan jarak antara garis batas dengan *support vector*. Menurut algoritma SVM, garis batas yang memiliki *margin* paling besar merupakan garis batas terbaik untuk persoalan klasifikasi. Pada gambar 1, garis tebal berwarna hijau merupakan garis batas terbaik karena memiliki margin yang lebih besar dibanding margin pada garis batas berwarna kuning. Pada sebuah figur 2 dimensi seperti gambar 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, dimana nilai-nilai variabel juga merupakan koordinat proyeksi data. Oleh karena kemampuan ini, SVM digolongkan sebagai teknik *machine learning* secara *supervised*. Pada konteks SVM, SVM menggunakan data yang telah disediakan untuk menentukan posisi garis pemisah yang dapat memisahkan tiap kelas secara optimal, dimana 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 *xi* 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, dimana baris ke-*i* pada *Y* atau dinotasikan sebagai (*yi*) dapat memiliki 2 nilai, yaitu +1 dan -1. Apabila *yi*bernilai +1, maka sampel *xi*merupakan anggota dari kelas positif, begitu pula sebaliknya. Diasumsikan bahwa data tersebut dapat dipisahkan menggunakan *hyperplane* dengan *n* dimensi, yang didefinisikan sebagai:

|  |  |  |
| --- | --- | --- |
|  |  | (3.1) |

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

|  |  |  |
| --- | --- | --- |
|  | , *i* = 1, ..., *m*, dan | (3.2) |
|  |  | (3.3) |

menotasikan *inner product* dari vektor dan vektor Untuk memecahkan persoalan optimasi pada persamaan (3) dengan batasan persamaan (2), dapat digunakan teknik *Langrange Multiplier* dimana persoalan tersebut menjadi:

|  |  |  |
| --- | --- | --- |
|  |  | (3.4) |

*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* (*ξi*), sehingga persamaan optimasi menjadi

|  |  |  |
| --- | --- | --- |
|  |  | (3.5) |
|  |  | (3.6) |

Dimana *C* disebut sebagai *error penalty*, serta dan merupakan *Langrangian Multiplier*.[12]. Dipandang persamaan minimalisasi sebagai *primal problem*.

|  |  |  |
| --- | --- | --- |
|  |  | (3.7) |

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

|  |  |  |
| --- | --- | --- |
|  |  | (3.8) |

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

|  |  |  |
| --- | --- | --- |
|  |  | (3.9) |
|  |  | (3.10) |
|  |  | (3.11) |

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

|  |  |  |
| --- | --- | --- |
|  |  | (3.12) |

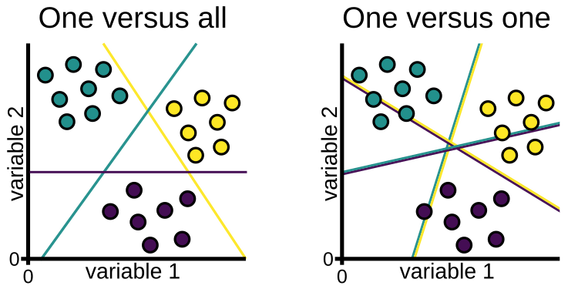
Yang memenuhi batasan:

|  |  |  |
| --- | --- | --- |
|  |  | (3.13) |

Dengan menyelesaikan persoalan optimasi (12), maka nilai akan didapatkan. Sehingga apabila persamaan (10) dapat disisipkan kedalam persamaan (1) menjadi:

|  |  |  |
| --- | --- | --- |
|  |  | (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. Eksperimen yang dilakukan oleh Hsu dan Lin menunjukkan bahwa kombinasi klasifikasi biner, khususnya *one vs one* dan *directed acyclic graph support vector machine* (DAGSVM), merupakan metode yang lebih cocok untuk penggunaan praktis dibandingkan metode-metode lain [14].



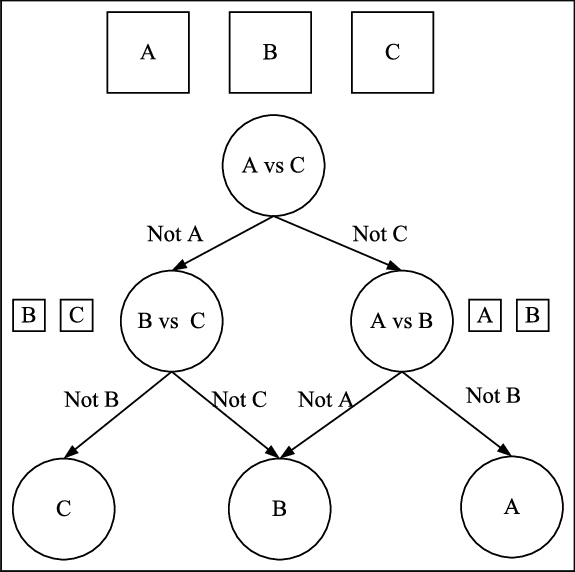
Gambar . Perbedaan One versus all dan One versus one

[25]

Terdapat banyak metode SVM multi-kelas yang berbasis klasifikasi biner, 3 diantaranya adalah metode *one vs one*, *one vs all*, dan DAGSVM. Metode *one vs one* akan membentuk *hyperplane* batas sebanyak *c*((*c*-1)/2), dimana *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.* 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 [14].

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 *ci* memiliki hasil klasifikasi *ci* atau *not ci*. 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* dimana 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 [15].



Gambar . Penentuan Kelas Menggunakan DAG

[15]

Seperti ditunjukan pada Gambar ., DAGSVM yang digunakan untuk mengklasifikasi data kedalam 3 kelas akan membentuk DAG dengan 6 *node* dan 3 tingkat. Tiap *node* menghasilkan 2 keluaran klasifikasi yang menyatakan bahwa sebuah sampel bukan anggota dari *ci*, 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 Clustering

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

### III.3.1. K-means Clustering

Proses K-*means* merupakan proses yang membagi sebuah populasi data kedalam k kelompok berbasis sebuah sampel. Proses tersebut mudah untuk diprogram dan ekonomis dalam konteks komputasi, sehingga memproses sampel besar pada sebuah komputer digital merupakan hal yang mungkin. Aplikasi dari proses K-*means* meliputi pengelompokan berdasarkan kemiripan (*clustering*), prediksi non-linier, memperkirakan distribusi multivariabel dan uji ketergantungan dari beberapa variabel secara non-parametrik [8].

Aplikasi proses k­-*means* pada pengelompokan berdasarkan kemiripan disebut sebagai k-*means clustering*, yang merupakan metode *clustering* yang banyak digunakan. K-*means clustering* bertujuan untuk mencari nilai minimal dari rerata jarak kuadrat antara pusat *cluster* dengan sebuah data pada *cluster* tersebut atau disebut sebagai *within cluster sum of squares* (WCSS). Pada K-*means clustering*, ditinjau konstanta *k* sebagai jumlah *cluster* yang telah ditentukan, matriks *X* sebagai kumpulan data berukuran matriks *m* x *n* dengan *feature* sebanyak *m* dan sampel sejumlah *n,* maka k­-*means clustering* akan memiliki objective function *(ϕ)* sebagai berikut:

|  |  |  |
| --- | --- | --- |
|  | *ϕ* | (3.155) |

Dimana *cj* merupakan pusat *cluster* ke-*j* dan *xi(j)* merupakan sebuah sampel dari *X* yang termasuk kedalam *cj*. Dari persamaan tersebut, diartikan bahwa J merupakan jumlah dari jarak kuadrat antara tiap pusat *cluster* ke tiap sampel yang termasuk kedalam *cluster* tersebut, dan K­-*means clustering* bertujuan untuk mencari nilai minimum dari *ϕ*. Algoritma dari K-*means clustering* adalah sebagai berikut:

1. Dipilih pusat *cluster (c)* sebanyak *k* secara acak dari sampel-sampel pada kumpulan data yang disediakan.

2. Untuk tiap *j ϵ {1, 2, .. k}* dan *i ϵ {1, 2, .. m},* golongkan tiap sampel *(xi)* dari *X* sebagai anggota dari *cluster* ke-*j* (*Cj)*apabila *xi* lebih dekat ke *cj* daripada pusat *cluster* lain.

3. Untuk tiap *j ϵ {1, 2, .. k}*, tetapkan *cj* sebagai mean dari *Cj*.

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

Dengan algoritma yang iteratif seperti yang telah disebutkan, K-*means clustering* akan mencari posisi untuk tiap *cj* yang menghasilkan *ϕ* yang minimum.

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

|  |  |  |
| --- | --- | --- |
|  | *WCSS* | (3.165) |

Karena *objective function (ϕ)* dari *k-means* *clustering* adalah untuk mencari nilai terkecil dari WCSS, maka temtu WCSS memiliki rumus yang sama dengan *ϕ*. WCSS dapat digunakan untuk mencari jumlah *cluster* yang optimal dengan metode yang disebut dengan *elbow method*. *Elbow method* merupakan sebuah strategi yang digunakan dimana penentuan jumlah *cluster* optimal didasarkan dari 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 *cluster* optimal merupakan nilai *k* dimana kurva membentuk sebuah siku (*elbow*). Pembentukan siku pada kurva WCSS menandakan bahwa terjadi penurunan nilai WCSS yang relatif drastis pada WCSS dengan jumlah *cluster* sebelumnya dibandingkan nilai WCSS dengan jumlah *cluster* tersebut. Penurunan nilai WCSS diasumsikan yang drastis kemudian diasumsikan sebagai bukti bahwa jumlah *cluster* optimum.

Meskipun k­-means clustering merupakan metode yang populer digunakan dalam berbagai aplikasi, namun yang membuat k-means clustering sebagai metode yang populer adalah kecepatan dan kesederhanaannya, bukan akurasi [9]. Karena pada tahap 1 pusat *cluster* pertama kali ditetapkan secara random, maka untuk tiap percobaan pada sebuah kumpulan data yang sama, *k*-*means clustering* dapat menghasilkan *clustering* yang berbeda dengan akurasi yang berbeda pula. Penelitian oleh David Arthur and Sergei Vassilvitskii kemudian memperkenalkan metode dimana posisi pusat *cluster* 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 *cluster* terdekat yang telah ditentukan pada iterasi sebelumnya. Setelah pusat *cluster* pertama dipilih dari sampel yang tersedia secara acak, pusat *cluster* berikutnya dipilih dari sampel yang tersedia dengan probabilitas dimana pemilihan sebuah sampel sebagai pusat *cluster* proporsional terhadap jarak sampel tersebut menuju pusat *cluster* terdekat, sehingga semakin jauh jarak sebuah sampel dari pusat *cluster*, semakin mungkin sampel tersebut ditentukan sebagai pusat *cluster* berikutnya. *K-means++* memiliki algoritma sebagai berikut:

1a. Tentukan pusat *cluster* pertama c1, dipilih secara acak dari sampel pada *X*.

1b. Tentukan pusat *cluster* berikutnya *ci*, yang dipilih secara acak dari sampel pada *X* dengan probabilitas

1c. Ulangi tahap 1b, hingga *ck* telah ditentukan

2-4. Dijalankan sama seperti algortima *k-means* [9]*.*

### 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 *cluster* akan memiliki densitas yang berbeda dengan *cluster* lainnya, strategi tersebut tentu saja berbeda dengan *k-means clustering* yang melakukan penggolongan berdasarkan pusat *cluster* terdekat. DBSCAN dikembangkan oleh Martin Ester, Hans-Peter Kriegel, Jörg Sander, dan Xiaowei Xu sebagai solusi yang menjawab kebutuhan-kebutuhan dari aplikasi metode *clustering* pada basis data spasial yang besar, yakni:

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 cluster 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.

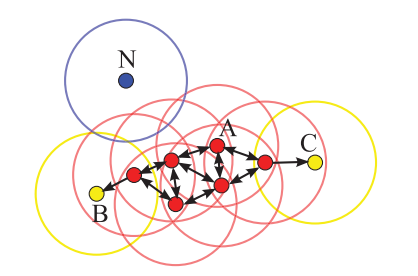
3. Memiliki efisiensi yang baik pada basis data yang besar, dimana DBSCAN dilaporkan memiliki efisiensi yang baik meskipun pada basis data yang besar.[10]

Algoritma dari metode DBSCAN dapat dijelaskan secara abstrak sebagai berikut:

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 *threshold* yang telah ditentukan. Titik pusat dan sampel yang berada didalam radius akan membentuk 1 *cluster kecil*.

2. Gabungkan semua titik pusat-titik pusat yang terdapat pada radius satu sama lain kedalam 1 *cluster* yang lebih besar.

3. Untuk sampel yang bukan merupakan titik pusat dan tidak terdapat didalam radius dari sebuah titik pusat, maka sampel tersebut dianggap sebagai *noise*, namun apabila sampel tersebut terdapat didalam radius dari sebuah titik pusat, maka sampel tersebut dianggap sebagai titik batas. [11]



Gambar . Ilustrasi cara kerja metode DBSCAN

[1]1]

## III.4. Kernel

Metode kernel adalah sebuah metode untuk memproyeksikan data ke *feature space* berdimensi tinggi tanpa perlu mengkalkulasikan *feature vector* pada *space* tersebut secara eksplisit, melainkan dengan menghitung *inner product* dari tiap pasangan data pada *feature space* semula, sehingga komputasi *feature vector* dapat dilakukan dengan lebih efisien [16]. Metode kernel dilakukan dengan cara mencari *inner product* dari pasangan sampel, atau secara umum dirumuskan dengan:

|  |  |  |
| --- | --- | --- |
|  |  | (3.17) |

Dimana dan merupakan sampel *X* ke-*i* dan ke-*j* dengan *i*,*j* = 1, 2, ..., *m*. Dengan demikian, merupakan matriks persegi dengan ukuran *m* x *m*. Apabila berdekatan dengan , maka nilai akan relatif besar, sedangkan apabila berjauhan atau hampir ortogonal, maka nilai akan relatif kecil. Sehingga dapat dikatakan bahwa merupakan kumpulan ukuran mengenai kemiripan tiap pasangan sampel. Untuk mengetahui apakah merupakan kernel yang *sah*, cukup dengan memastikan bahwa matrix kernel dari data *X* merupakan matriks yang simetris dan semi-definit positif [16].

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

1. Radial Basis Function (RBF) kernel:

|  |  |  |
| --- | --- | --- |
|  |  | (3.18) |

2. Polynomial kernel:

|  |  |  |
| --- | --- | --- |
|  |  | (3.19) |

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

## III.5. Kernel Fisher Discriminant Analysis

*Dimension reduction* merupakan metode untuk memproyeksikan data ke *feature space* dengan dimensi yang lebih kecil dibandingkan *feature space* semula, atau secara sederhana, *dimension reduction* digunakan untuk mengurangi jumlah variabel (dimensi) pada sebuah himpunan data. *Feature space* sendirimerupakan sebutan bagi sebuah ruang atau *space* dimana 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 [17]. Terdapat 2 metode yang sering digunakan untuk melakukan *dimension reduction*, yaitu *feature selection* dan *feature extraction*. *Feature selection* mengurangi jumlah variabel dengan cara mengeliminasi variabel-variabel yang tidak relevan, umumnya dengan mengeliminasi variabel yang memiliki nilai variansidibawah *threshold* yang telah ditentukan. Berbeda dari *feature selection*, *feature extraction* membentuk variabel-variabelbaru sehingga data diproyeksikan dengan variansi sebesar mungkin pada *feature space* baru. Variabel baru yang dibentuk merupakan kombinasi liner dari variabel-variabel semula [6]. Dengan memperhatikan variansi, kedua metode menjaga agar tidak banyak informasi yang hilang ketika digunakan *feature space* yang berdimensi lebih rendah.

Salah satu metode *feature extraction* 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 persoalan non-linier [9]. Beberapa peneliti kemudian mengembangkan FDA yang dapat memiliki performa baik pada persoalan nonlinier, salah satunya adalah dengan menggunakan metode kernel.

Yang 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 *kernel feature space* dari himpunan data menggunakan persamaan dari metode kernel yang digunakan. [9]

Matrikstransformasi KPCA dapat dihitung dengan persamaan:

|  |  |  |
| --- | --- | --- |
|  |  | (3.20) |

Dimana ν dan λ merupakan *eigenvalue* dan *eigenvector* dari K, dan K merupakan matriks yang terpusat. Menggunakan perkalian *dot product* dengan , *X* dapat diproyeksikan kedalam *feature space* baru dan dinotasikan sebagai , yang mana kemudian digunakan untuk membentuk matriks transformasiKFDA pada tahap kedua, yaitu tahap FDA. didapatkan menerapkan FDA pada sehingga didapatkan *eigenvector* dari , dimana *Cov* merupakan matriks kovarians dari mean antar kelas dan *S* merupakan variansi dari pada tiap *feature vector* pada *feature space* .

|  |  |  |
| --- | --- | --- |
|  |  | (3.21) |
|  |  | (3.22) |

Dimana *m* menotasikan jumlah sampel pada *X*, *c* menotasikan jumlah kelas, menotasikan jumlah sampel pada kelas ke-*i*, menotasikan mean dari kelas ke-*i­,*dan menotasikan mean dari . Pada persamaan (20) dapat dilihat bahwa tiap eigenvalue merupakan nilai variansi dari data *X* yang diproyeksikan pada *eigenvector* sebagai *feature vector.* Sebuah data baru bernotasi *x* dapat diproyeksikan pada *feature space* KFDA dengan persamaan:

|  |  |  |
| --- | --- | --- |
|  |  | (3.23) |

Jumlah dimensi dari *feature space* KFDA memiliki jumlah yang sama dengan jumlah *eigenvector* yang digunakan.

## III.6. Scaling

*Scaling* merupakan metode yang biasanya dilakukan pada tahap pra-proses data. Dimana nilai tiap variabel pada sebuah data diubah menjadi memiliki rentang yang sama sehingga semua variabel memiliki proporsi yang sama. *Scaling* dilakukan pada kumpulan data yang memiliki rentang *feature* yang beragam. Ditinjau sebuah kumpulan data yang memiliki 2 *feature,* 1 *feature* memiliki rentang pada bilangan puluhan, misalkan 15 hingga 80, dan 1 *feature* memiliki rentang pada bilangan ribuan, misalkan 3000 hingga 3050. Apabila pada kumpulan data tersebut tidak dilakukan *feature scaling*, maka *feature* yang berentang pada bilangan ribuan akan mendominasi karena bilangannya lebih besar, meskipun memiliki rentang yang lebih kecil dibandingkan *feature* lainnya. Dengan melakukan *scaling*, kedua *feature* tadi akan memiliki rentang yang sama, sehingga tidak ada *feature* yang mendominasi.

Diasumsikan himpunan data *X* yang memiliki *m* sampel dan *n* variabel, variabel pada *X* dinotasikan dengan *f* dimana *fi*merupakan vektor berukuran *m*. Hasil rentang nilai serta satuan-satuan yang digunakan bergantung pada *scaling* 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:

|  |  |  |
| --- | --- | --- |
|  |  | (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:

|  |  |  |
| --- | --- | --- |
|  |  | (3.25) |

* Robust Scaler

*Robust Scaler* mengubah data dengan hasil 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:

|  |  |  |
| --- | --- | --- |
|  |  | (3.26) |

## III.7. Fault Detection and Diagnosis

Pabrik merupakan sebuah situs industri yang bertujuan untuk memproses sebuah bahan sehingga nilai bahan tersebut meningkat atau untuk menciptakan kekayaan baru (Conceptual Design of Chemical Plant). Pabrik menggunakan peralatan-peralatan khusus untuk memanufaktur atau memproses bahan serta untuk memonitor proses, sehingga hasil akhir yang didapatkan sesuai dengan yang hasil yang direncanakan berdasarkan rancangan proses. Peralatan industri yang digunakan pada sebuah pabrik bergantung pada bahan yang diproses serta kekayaan yang ingin dibuat, agar proses berjalan sesuai rencana, digunakan sistem kendali yang terdiri dari 3 perangkat, *sensor*, *actuator*, dan *controller*.

Sebuah *controller* merupakan otak dari sistem kendali yang mengirimkan sinyal perintah kepada *controller* berdasarkan sinyal yang dikirim oleh *sensor*. *Sensor* sendirimerupakan perangkat yang digunakan untuk membaca sebuah kondisi fisis bahan, dan mengirimkan *output* berupa sinyal yang mendeskripsikan nilai dari kondisi fisis tersebut kepada *controller*. Sedangkan *actuator* merupakan perangkat yang mengendalikan perangkat pada proses berdasarkan sinyal perintah dari *controller*. Untuk dapat mengendalikan proses agar berjalan sesuai rencana, seorang operator memasukkan *setpoint* berupa nilai yang diharapkan dari sebuah variabel pada proses. Sensor terus menerus akan mendeteksi nilai sebenarnya dari variabel tersebut, sehingga *controller* dapat mengetahui nilai *error*. *Error* merupakan perbedaan nilai *setpoint* dengan nilai sebenarnya. Kemudian *controller* akan mengirimkan sinyal perintah kepada *actuator* untuk mengonpensasi *error* tersebut.

Kesalahan atau *fault* didefinisikan sebagai perilaku abnormal pada proses yang berhubungan dengan kegagalan mesin, kelelahan mesin, atau gangguan eksrim pada proses [12]. Apabila *fault* terjadi dan tidak segera ditangani, kualitas produk atau bahkan juga keselamatan proses menjadi terancam. *Fault* 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 *fault detection* dikembangkan selama 20 tahun terakhir [2].

*Fault detection and diagnosis* (FDD) atau 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.

*Fault detection* dan *fault diagnosis* merupakan 2 persoalan yang berbeda, yang mana *fault detection* merupakan persoalan untuk mendeteksi data *faulty* dari sebuah sistem, sedangkan *fault diagnosis* merupakan persoalan untuk mengidentifikasi jenis *fault* berdasarkan karakteristik *fault* tersebut. Kedua persoalan ini dapat dipecahkan dengan metode klasifikasi dimana *fault detection* merupakan klasifikas 2 kelas sedangkan *fault diagnosis* merupakan klasifikasi multi kelas. Pada prakteknya, *fault detection* dan *fault diagnosis* 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 [18] dan batas-batas *fault* dapat dideteksi. Berbeda halnya dengan pendekatan *data-driven* dimana batas-batas *fault* dibentuk tidak melalui pengetahuan mengenai model matematis proses, melainkan menggunakan data *time-series* yang dihimpun selama proses dijalankan [19].

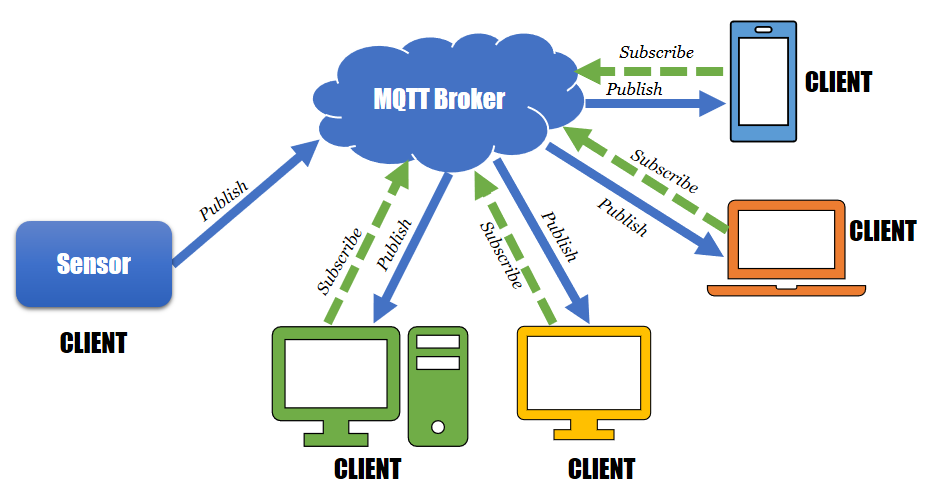
Pada pendekatan *data-driven*, diasumsikan bahwa data historis proses yang berjumlah besar telah tersedia, baik data kondisi normal maupun data kondisi *faulty*. Data historis proses merupakan himpunan data deret waktu yang didapatkan dari nilai *output* sensor dan *actuator*. Terdapat banyak cara yang dapat digunakan untuk mentransformasikan data sehingga data tersebut dapat digunakan sebagai *input* oleh sistem diagnosis. Tahap ini disebut dengan *feature extraction* dari data historis proses, dan dilakukan untuk memfasilitasi diagnosis data baru [20].

## III.8. MQTT

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 pada dimana *bandwidth* jaringan rendah atau dengan perangkat yang memiliki kapabilitas komputasi dan memori yang terbatas. Protokol MQTT memudahkan komunikasi *machine-to-machine*, yaitu komunikasi antar perangkat yang menjadi semakin umum dengan berkembangnya teknologi. Protokol MQTT *for Sensors* (MQTT-S) memungkinkan terhubungnya perangkat-perangkat yang awalnya tidak dapat menggunakan MQTT karena tidak memiliki kemampuan jaringan TCP/IP. MQTT-S memperluas cakupan protokol MQTT sehingga perangkat *sensor* dan *actuator* yang beroperasi menggunakan baterai dapat beroperasi dengan protokol MQTT untuk melakukan transfer data dua arah [21]

Berikut merupakan fitur protokol MQTT:

Gambar . Skema Kerja MQTT



[24]

###### Publish/subscribe *message pattern* yang mampu menyediakan distribusi pesan dari satu *client* ke banyak *client*.

###### *Messagging transport* yang dapat dilakukan tidak bergantung pada bentuk *payload* yang dikirimkan.

###### Menggunakan TCP/IP sebagai konektivitas dasar jaringan.

###### Terdapat tiga level *Qualities of Service* (QoS) dalam penyampaian pesan :

* 1. “*At most once*”, dimana pesan dikirim dengan upaya terbaik dari jaringan TCP/IP. Kehilangan pesan atau terjadi duplikasi dapat terjadi.
  2. “*At least once*”, dimana dapat dipastikan pesan tersampaikan walaupun duplikasi dapat terjadi.
  3. “*Exactly once*”, dimana pesan dapat dipastikan tiba tepat satu kali.

###### Menggunakan *overhead* pengiriman yang kecil (2 byte)

###### Memiliki mekanisme yang akan memberitahu *client* yang terhubung apabila terdapat diskoneksi *client* yang abnormal dengan fitur *Last Will* dan *Testament* [22].

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* [21].

# PELAKSANAAN PENELITIAN

## IV.1. Alat dan Bahan Penelitian

**IV.1.1. Alat Penelitian**

Tabel IV... 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 |
| Laptop | OS: Windows 10 Home  Processor: Intel Core i5-8250U @1.6 GHz  RAM: 12 GB |

Tabel . 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*. Python merupakan bahasa pemrograman tingkat tinggi dengan proses komputasi yang dinamis dan memiliki struktur data tingkat tinggi namun untuk mudah dipelajari, sehingga menjadi bahasa pemrograman yang atraktif untuk *rapid prototyping* bagi banyak peneliti. Sebagai proyek *open source*, Python mendukung *modules* dan *packages*, dimana seseorang dapat menggunakan program yang telah dibuat orang lain. Dengan banyaknya orang yang menggunakan *python*, banyak pula *modules* dan *packages* yang telah diciptakan, sehingga seseorang dapat dengan mudah mencari dan menggunakan *modules* atau *packages* yang tersedia untuk menyelesaikan persoalan yang ia miliki.

Seringkali, seseorang memilih untuk menggunakan Python dikarenakan produktivitas yang ditawarkan. Pada Python tidak terdapat tahap *compilation*, sehingga siklus *edit-test-debug* dapat dilakukan dengan cepat, dimana proses *debug* pada Python juga dapat dilakukan dengan mudah. Python membutuhkan baris kode yang relatif sedikit, sehingga kurang rentan terhadap masalah dan lebih mudah untuk *debug*. Dengan segala fitur tadi, Python dapat digunakan untuk berbagai situasi, baik *online* maupun *offline* [23].

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 *machine learning* dan algoritma pendukungnya. Dengan menggunakan pustaka scikit-learn, pengolahan data dapat dengan mudah dilakukan karena scikit-learn memiliki berbagai model *machine learning* 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 *dimention reduction*, *model selection*, dan *pre-preprocessing* untuk mendukung akurasi dari model *machine learning* yang tersedia. Dengan berbagai algoritma tersebut, selain pengguna dapat mempersiapkan data sebelum diolah oleh model *machine learning* dengan algoritma *dimention reduction* dan *pre-processing*, pengguna juga dapat menguji *parameter yang optimal* menggunakan *model selection*. 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.

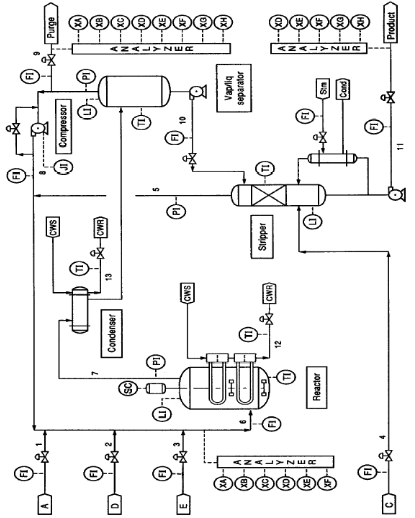
Pandas merupakan pustaka *open-source* pada Python yang digunakan untuk analisis dan manipulasi data. Pandas mampu membaca dan mengubah data-data pada berbagai format seperti CSV and teks, Microsoft Excel, dan database SQL. Pandas memiliki jenis obyek berupa DataFrame yang dapat dimanipulasi dengan mudah, 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 *fault detection* pada penelitian ini, sedangkan kumpulan data dari penelitian Brooks memiliki 20 variabel digunakan sebagai validasi dari program yang sudah dibangun.

1. Data Untuk Pembangunan Program

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. 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 . Diagram P&ID dari Tennessee Eastman Process

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, dimana reaktan kemudian akan bereaksi. Produk akan keluar dari reaktor dalam bentuk uap bersamaan dengan umpanyang tidak bereaksi.
2. Kondensor: Produk dari reaktor masuk ke dalam kondensor dimana 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). Dimana 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 . 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) | m3 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) | m3 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) | m3 h-1 |
| 24 | Katub aliran umpan E (aliran 3) | kg h-1 | 30 | Katub aliran cairan *stripper* (aliran 11) | m3 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 | m3 h-1 |
| 27 | Katub kompresor daur ulang | % | 33 | Katub pendingin kondensor | m3 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 *faulty* tipe 2 sebanyak 480 normal, data *faulty* tipe 5 sebanyak 480 sampel, data *faulty* tipe 7 sebanyak 480 sampel, dan data *faulty* tipe 14 sebanyak 480 sampel. Rincian data yang digunakan dapat dilihat pada Tabel IV.4.

Tabel . Deskripsi Jenis Data pada Data Latih

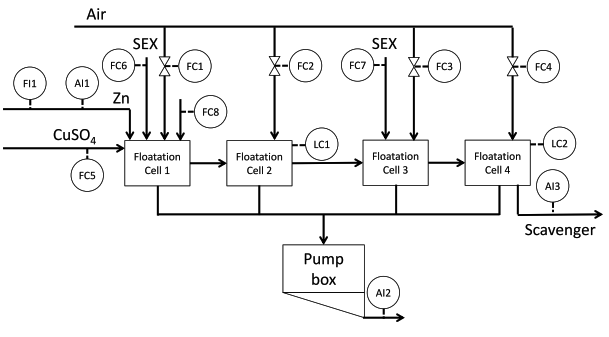
|  |  |
| --- | --- |
| Jenis Data | Deskripsi |
| Data Normal | Pabrik beroperasi dalam keadaan standar |
| Data *Faulty* tipe 2 | Perubahan komposisi B, rasio A/C konstan |
| Data *Faulty* tipe 5 | Perubahan suhu pada masukan pendingin kondensor |
| Data *Faulty* tipe 6 | Penurunan aliran umpan A |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

*Data Fault* 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 standar (tanpa diberikan gangguan). Program *fault detection* yang dirancang diharapkan dapat menangkap perbedaan dari data tersebut, dan mengklasifikasikan data dengan sesuai.

1. Data Untuk Validasi Program

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 *fault detection* dan rekonstruksi data pada sebuah kumpulan data sensor pabrik. Data pabrik yang disebutkan dapat diakses pada *web repository* dari *South African Council of Automation and Control* (SACAC). 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 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 kedalam feed, *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 CuSO4 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 dimana *slurry* akan diproses seperti pada sel sebelumnya.



Gambar . Diagram P&ID Data Validasi

Data dari proses ini dihasilkan oleh 14 sensor dengan penjelasan yang dapat dilihat pada table IV.5, dimana 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 dimana satu atau lebih sensor mengalami kegagalan, sehingga tidak menghasilkan hasil pengukuran, ditandai dengan nilai 0. Penelitian Brooks 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 *Brooks*, digunakan 301 sampel kontiyu sebagai data *training* dan 701 sampel kontiyu sebagai data *testing*. Data kontiyu tersebut dimulai dari tanggal 29 September 2015 pukul 02.00 hingga 1 Oktober 2015 pukul 04.00 untuk data *training*, data tersebut dipilih karena memiliki rentang panjang dimana tidak terdapat kegagalan sensor. Sedangkan untuk data *testing*, digunakan data dimulai dari 18 Oktober 2015 pukul 03.32 hingga 23 Oktober 2015 pukul 00:12, dimana banyak terjadi kegagalan sensor AI3 dengan rentang yang panjang. Pada penelitian ini, kelompok data *training* digunakan untuk mengumpulkan data referensi, kelompok data *testing* digunakan untuk menguji apakah program dapat mendeteksi *fault*. Selain itu, juga digunakan kelompok data lain sebanyak 700 sampel dimulai dari berakhirnya kelompok data *training*, kelompok data ini digunakan untuk menguji apakah program dapat mendeteksi data normal.

Tabel . 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 | mn3/h |
| **4** | FC2 | Aliran udara yang masuk ke sel 2 | mn3/h |
| **5** | FC3 | Aliran udara yang masuk ke sel 3 | mn3/h |
| **6** | FC4 | Aliran udara yang masuk ke sel 4 | mn3/h |
| **7** | FC5 | Aliran tambahan tembaga sulfat | m3/h |
| **8** | FC6 | Aliran Sodium ethyl xanthate ke sel 1 | m3/h |
| **9** | FC7 | Aliran Sodium ethyl xanthate ke sel 3 | m3/h |
| **10** | FC8 | Aliran Naphtalene sulfat ke sel 1 | m3/h |
| **11** | FI1 | Aliran volumetric umpan | m3/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 studi literatur, pengujian KFDA dan SVM sebagai metode *fault detection*, desain program, pembangunan program, dan pembuatan laporan.

Pengujian parameter KFDA dan klasifikasi menggunakan SVM dilaksanakan setelah melakukan studi terhadap penelitian-penelitian yang telah dilakukan sebelumnya sehingga pelaksaannya dapat dilakukan dengan tepat. Pada tahap tersebut, digunakan data pabrik yang tersedia di internet untuk menguji rangka kerja yang diajukan, yaitu *fault detection* menggunakan KFDA dan SVM. Tahap desain program kemudian dilakukan apabila rangka kerja yang diajukan memiliki akurasi yang memuaskan. Pembuatan laporan kemudian dilaksanakan dengan menjelaskan hasil analisis klasifikasi KFDA-SVM, proses perancangan dan pembuatan program *fault detection*, serta pengujian program *fault detection* yang telah dibuat.

### 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 *model training* dan *model testing* yang diiterasi untuk mencari nilai optimal dari parameter KFDA sehingga didapatkan akurasi klasifikasi SVM yang optimal pula.

Sebagian besar dari algoritma ini merupakan tahap *model training* dan tahap *model testing*. *Model training* merupakan tahap untuk mengonstruksi *scaler,* matriks proyeksi KFDA, serta SVM menggunakan data latih untuk kemudian di uji menggunakan data uji pada tahap *model testing*. Kedua tahap ini dilakukan secara berulang-ulang hingga ditemukan parameter KFDA yang optimal.

Sebelum masuk ke tahap *model training*, data pabrik kimia yang didapatkan melalui internet dibagi menjadi 2 bagian, yaitu data latih dan data uji. Pembagian data ini dilakukan secara random dengan perbandingan tertentu antara jumlah data uji dan data sampel, dimana sampel tiap kelas memiliki perbandingan yang sama. Sebagai contoh, apabila ditetapkan persentase antara data latih dan data uji adalah 80:20, maka persentase jumlah sampel tiap kelas antara data latih dan data uji juga 80:20. Hal ini dilakukan supaya data latih ataupun data uji memiliki jumlah sampel tiap kelas yang berimbang.

Scaling dilakukan agar data memiliki proporsi yang sama pada semua variabel, dimana algoritma untuk melakukan *scaling* disebut dengan *scaler*. *Scaler* dikonstruksi menggunakan data latih untuk mencari nilai dari satuan-satuan tertentu sehingga nilai variabel pada data latih dapat diubah rentangnya. Setelah *scaler* dikonstruksi, *scaler* digunakan untuk melakukan *scaling* pada data latih dan data uji.

Pada tahap *model training*, data latih dan parameter KFDA digunakan untuk mengonstruksi matriks proyeksi KFDA dengan menggunakan persamaan pada bab 3. Matriks proyeksi KFDA berfungsi untuk memproyeksikan data latih dan data uji kedalam KFDA *feature space*. Data latih yang telah diproyeksikan kedalam KFDA *feature space* akan digunakan untuk mengonstruksi *hyperplane* klasifikasi SVM.

Pada tahap *model testing*, data uji akan ditransformasikan seperti halnya data latih: data uji akan di-*scaling* menggunakan *scaler* serta diproyeksikan pada KFDA *feature space* menggunakan matriks proyeksi yang telah dikonstruksi pada tahap *model training*. Data latih kemudian akan diklasifikasikan menggunakan SVM yang telah dikonstruksi, apabila hasil klasifikasi belum optimal, maka parameter KFDA akan diubah dan tahap *model training* dijalankan kembali. Hasil klasifikasi belum optimal apabila proyeksi data uji pada KFDA *feature space* tidak memiliki keterpisahan antar kelas serta akurasi klasifikasi tidak memuaskan.

### 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, dimana pada tahap Pengujian KFDA dan SVM sebagai metode *fault detection*, data *training* 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 *fault detection* tanpa membutuhkan kumpulan data yang telah tersedia sebagai data *training*. Perancangan program dilakukan dengan cara menguji metode *clustering* yang sesuai untuk diimplementasikan bersama KFDA-SVM untuk membentuk metode *semi-supervised learning*.

Apabila metode *clustering* 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 pabrik 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, dimana data latih 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 *fault detection* yang mampu mendeteksi data *faulty* dari sebuah pabrik secara real-time tanpa memerlukan tahapan *model training* yang eksplisit.

### IV.2.4. Pengujian Program Menggunakan Data Validasi

Setelah program dibangun, performa dari program diuji menggunakan Data Validasi, dengan parameter-parameter yang didapatkan melalui tahap perancangan program. Program juga dibandingkan dengan metode *fault detection* konvensional menggunakan KFDA-SVM, yang diberi masukan berupa data normal pada data validasi untuk membangun model KFDA-SVM kemudian diuji akurasinya menggunakan sisa data pada data validasi.

**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 *faulty.* 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 KFDA dan SVM, akan didapatkan hasil berupa prediksi label tiap sampel pada *testing* set. Hasil prediksi label kemudian dibandingkan dengan label yang sebenarnya untuk menghitung akurasi *fault detection* 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 . 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 *fault detection*, serta parameter yang optimal untuk digunakan.

Pada tahap Perancangan Program, hasil prediksi *fault detection* secara *real-time* akan dibandingkan dengan label data yang sebenarnya untuk menghitung akurasi program *fault detection* dengan rumus yang sama seperti yang digunakan pada tahap sebelumnya.

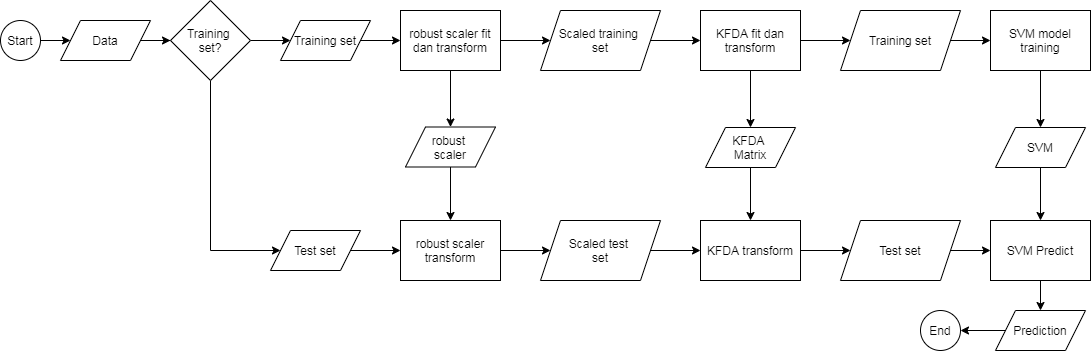
Setelah itu, pada tahap Pengujian Program, program yang dirancang diuji menggunakan data validasi, dan dibandingkan dengan strategi *fault detection* konvensional menggunakan KFDA-SVM.

# Hasil dan Pembahasan

## Pengujian KFDA dan SVM

### Rangka kerja *fault detection* menggunakan KFDA dan SVM

Rangka kerja *fault detection* merupakan rangka kerja untuk mendeteksi sampel *faulty* dari kumpulan sampel yang dimasukkan sebagai input. Dengan menggunakan metode klasifikasi *machine learning*, deteksi *fault* dapat dilakukan dengan membentuk *hyperplane* pemisah antara data normal dan data *faulty*. Untuk mempermudah komputasi *hyperplane* pemisah, digunakan metode *feature extraction* untuk mengurangi jumlah fitur dari data tanpa menghilangkan informasi penting. Pada penelitian ini, *feature extraction* 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 *fault*. Metode *feature extraction* yang digunakan harus dapat memproyeksikan sampel normal dan sampel *faulty* pada planar 2 dimensi secara terpisah. Tugas tersebut dapat dilakukan dengan menggunakan *supervised feature extraction*, yakni metode *feature extraction* yang melibatkan label dari data dalam proses *training*-nya.



Gambar . Rangka Kerja Fault Detection

Berdasarkan studi literatur yang telah dilakukan, dibentuk rangka kerja seperti yang tertera pada gambar 5.1 untuk melakukan *fault detection* menggunakan *support vector machines*. Sebelum mulai melakukan *fault detection*, label data yang digunakan dimanipulasi terlebih dahulu. Pada data yang digunakan, terdapat **5** kelas yang menandakan 1 kelas normal dan **4** jenis*fault.* Pada penelitian ini, hanya digunakan 2 kelas, yaitu kelas normal dan kelas *faulty*, 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 *fault* menjadi 1, sehingga tugas klasifikasi *support vector machines* menjadi klasifikasi biner, dimana hanya terdapat 2 label.

Data kemudian dipisahkan secara acak menjadi 2 set, yakni *training set* dan *testing set,* yang kemudian akan diolah pada 3 tahap berikutnya, yakni *scaling*, *feature extraction*, dan klasifikasi. Tiap tahap dilakukan sebanyak 2 kali, yaitu untuk *training set* dan untuk *testing set*. Testing set digunakan untuk membentuk atau melatih *transformator* dan *predictor* yang akan diaplikasikan pada *training set* dan *testing set*, sedangkan *testing set* akan digunakan untuk menilai kinerjarangka kerja dalam melakukan *fault detection*. Training set akan ditransformasi menggunakan *robust scaler* untuk mengubah skala *training set*. Keluaran dari tahap ini berupa *training set* yang sudah ditransformasikan serta obyek *transformator* berdasarkan *training set* yang kemudian akan digunakan untuk mentransformasikan *testing set*. Training set yang telah ditransformasikan kemudian diproyeksikan pada KFDA-*space* pada tahap berikutnya, sama seperti tahap *robust scaler,* tahap ini juga memiliki 2 keluaran, yakni proyeksi *training set* pada KFDA-*space* serta *transformator* berdasarkan *training set* yang digunakan untuk memproyeksikan *testing set* pada KFDA-*space*. Pada tahap berikutnya, proyeksi *training set* digunakan untuk membentuk *predictor* SVMberdasarkan label dan posisi dari setiap sampel pada *training set*. *Predictor* kemudian digunakan untuk memprediksi label dari proyeksi *testing set* pada KFDA-*space*.

Pada rangka kerja yang dibentuk, digunakan *robust scaler* untuk *scaling*, KFDA untuk *feature extraction*, dan *support vector machines* untuk klasifikasi. Karena data yang akan digunakan tidak dapat dipastikan akan terdistribusi normal, *robust scaler* dipilih karena dapat melakukan *scaling* pada data bagaimanapun distribusinya. Selain itu, *robust scaler* juga tidak sensitif terhadap *outlier* karena menggunakan nilai jangkauan antar kuartil, tidak seperti *min-max scaling* yang sensitif terhadap *outlier* karena menggunakan nilai minimum dan maksimum. Pada penelitian ini kemudian digunakan *Kernel Fisher Discriminant Analysis* (KFDA) sebagai *feature extraction*. KFDA merupakan metode *supervised feature* extraction, 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, *feature extraction* menjadi 2 fitur dengan proyeksi data antar kelas yang terpisah menjadi mungkin dilakukan.

## V.1.2. Pengujian KFDA dan SVM untuk *fault detection*

Rangka kerja ini merupakan inti dari program yang akan dirancang, dimana program akan mengaplikasikan rangka kerja supaya dapat mendeteksi *fault* secara *real-time*.Pengujian rangka kerja *fault detection* dilakukan dengan menggunakan dataset Tennessee Eastman Process yang berisi data normal dan data beberapa tipe *fault*. Dataset kemudian dipisah menjadi *training set* dan *testing set* dengan perbandingan 6:4. Untuk pembuatan matriks transformasi KFDA, digunakan nilai gamma sebesar 0.00005 untuk RBF kernel. SVM yang dilatih juga merupakan Kernel SVM yang menggunakan RBF kernel pula. Pengujian yang dilakukan memberikan hasil seperti pada gambar V.2.

Gambar (a) pada Gambar V.2 merupakan *scatter plot* dari proyeksi data latih pada KFDA *subspace*. Training set merupakan data yang digunakan untuk membuat matriks transformasi KFDA dan melatih SVM. Training set 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 training set terproyeksi pada KFDA *subspace* secara terpisah, dimana tidak terdapat sampel 1 kelas yang terproyeksi pada kluster kelas lain. Pada *scatter plot* (b) dicoba menggunakan SVM yang dilatih menggunakan *training set* untuk memprediksi label dari tiap kelas pada *training set*. Dengan proyeksi data yang begitu terpisah, klasifikasi SVM pada *training set* sendiri menghasilkan akurasi 100%, dapat dilihat dengan *scatter plot* (a) dan (b) yang persis sama.

Gambar (c) dan (d) merupakan hasil prediksi label *testing set* menggunakan SVM multi-kelas dan SVM 2 kelas. Dapat dilihat bahwa proyeksi dari *testing set* lebih tersebar dibandingkan proyeksi dari *training set* yang relatif lebih rapat, sehingga menimbulkan peluang untuk terjadi misklasifikasi prediksi label oleh SVM. *Confusion matrix* untuk klasifikasi label *testing set* secara multikelas dapat dilihat pada tabel V.1 sedangkan untuk klasifikasi 2 kelas dapat dilihat pada tabel V.2.

Gambar . Hasil Pengujian Menggunakan KFDA-SVM



a

b

c

d

Tabel .. *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 .. *Confuison 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 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. Kedua klasifikasi, baik multi kelas maupun 2 kelas memiliki akurasi yang sangat memuaskan, yaitu 99.45% untuk klasifikasi multi kelas, dan 99.17% untuk klasifikasi 2 kelas. Dengan nilai akurasi klasifikasi yang tinggi, rangka kerja KFDA-SVM dinilai cocok untuk digunakan pada program *fault detection* yang akan dirancang pada tahap berikutnya.

## Perancangan Program *Fault Detection*

Pada tahap ini, dilakukan beberapa pengujian untuk menentukan alur kerja dan parameter-parameter program yang optimal. Parameter yang akan diuji antara lain ukuran sampel referensi dan *buffer* dan nilai WCSS yang optimal.

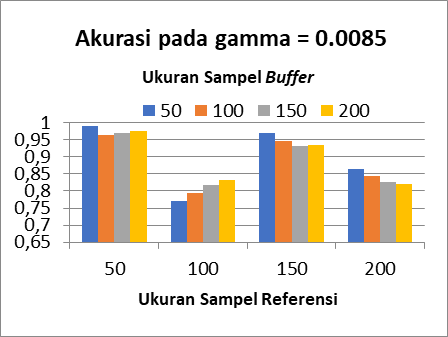
### Pengujian Ukuran Referensi dan *Buffer*

Pengujian terhadap ukuran referensi dan ukuran *buffer* 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 tanpa data referensi, akan sulit bagi program untuk memberikan batas antara sampel normal dan sampel *faulty*. Apabila *buffer* dipenuhi oleh sampel-sampel *faulty* maka bisa jadi program justru akan membagi sampel pada *buffer* menjadi 2 kelompok, yaitu normal dan *faulty* karena program tidak memiliki pengetahuan akan data normal. Selain itu, apabila terdapat pergerseran proyeksi sampel yang menjauhi kluster normal secara perlahan, tanpa adanya data referensi, program akan menganggap pergeseran tersebut sebagai perluasan kluster normal, dan akan rentan terjadi misklasifikasi. Dengan adanya data referensi normal, apabila terjadi pergeseran sampel ini terlalu jauh, maka program *fault detection* akan mengklasifikasikan sampel tersebut sebagai data *faulty*. Sedangkan *buffer* merupakan kelompok sampel yang diterima setelah referensi terisi, dimana *buffer* akan secara terus menerus diisi oleh sampel baru yang masuk. Dengan ukurannya yang terbatas, apabila *buffer* sudah penuh dengan sampel, sampel yang paling awal diterima akan dibuang ketika *buffer* menerima sebuah sampel baru.

Hal ini sesuai dengan salah satu metode pengolahan struktur data yakni FIFO (*first-in-first-out*) pada sebuah *queue*. FIFO merupakan metode pengolahan struktur data dimana obyek yang pertama masuk kedalam sebuah antrian (baik itu *queue* atau *stack*), akan diproses dan dikeluarkan pertama pula, sedangkan *queue* merupakan sebuah kumpulan obyek yang diatur secara berututan, dimana *queue* dapat dimodifikasi dengan menambahkan *obyek* di salah satu ujung atau menghapus *obyek* di ujung lainnya.

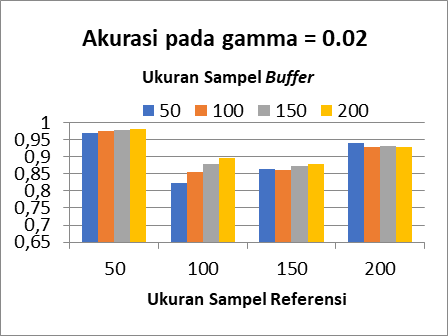
Akurasi dari *fault detection* akan bervariasi tergantung pada ukuran referensi dan *buffer,* dikarenakan matriks KFDA dibentuk oleh data referensi sehingga penggunaan ukuran referensi akan berbeda akan menghasilkan proyeksi sampel yang berbeda pula. Selain itu, data normal dan data *buffer* digunakan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

Pengujian dilakukan dengan memvariasikan ukuran referensi, ukuran *buffer*, dan nilai gamma untuk memprediksi label sampel pada setiap jenis *fault*. Pengujian terhadap setiap jenis *fault* dilakukan sebanyak 3 kali dengan kelompok *sampel* yang berbeda. Hal tersebut dilakukan dengan cara membagi tiap jenis *fault* menjadi 3 kelompok dan melakukan pengujian *fault detection* untuk tiap kelompok. Hal tersebut dilakukan untuk memastikan bahwa tiap kelompok sampel pada tiap jenis *fault* 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 gamma, ukuran sampel referensi, dan ukuran sampel *buffer*. Hasil pengujian dapat dilihat pada Gambar 5.3, 5.4, 5.5, 5.6, dan 5.7.



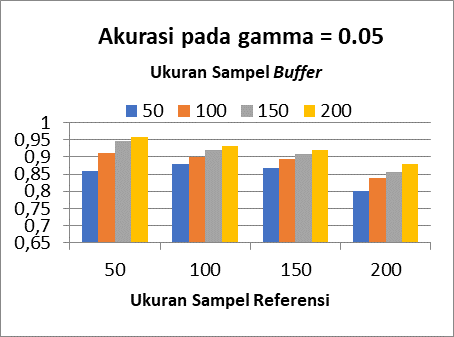
Gambar . Pengujian Pada

Gamma = 0.0085



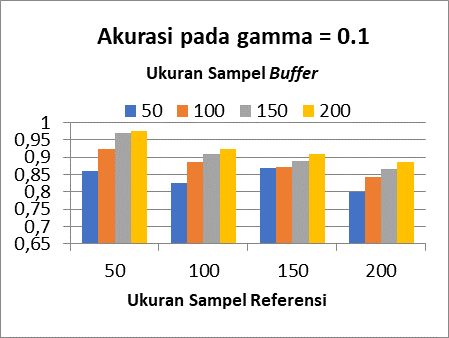
Gambar . Pengujian Pada

Gamma = 0.02



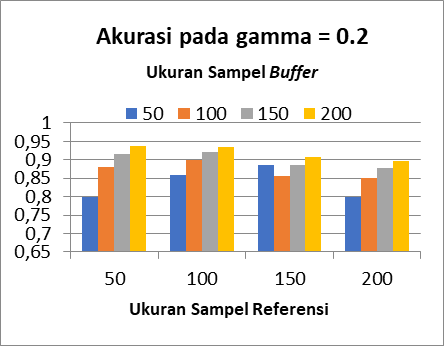
Gambar . Pengujian Pada

Gamma = 0.05



Gambar . Pengujian Pada

Gamma = 0.1



Gambar . Pengujian Pada

Gamma = 0.2

Akurasi prediksi SVM bervariasi bergantung pada ukuran sampel referensi, ukuran sampel *buffer* dan nilai gamma yang digunakan. Pada pengujian dengan gamma bernilai 0.05, 0.1, dan 0.2, dapat dilihat tren bahwa akurasi prediksi SVM semakin meningkat dengan bertambahnya ukuran sampel *buffer*, namun semakin menurun dengan bertambahnya ukuran sampel referensi. Pada pengujian menggunakan gamma bernilai 0.02, pengaruh dari ukuran sampel *buffer* tidak terlalu signifikan terhadap akurasi dari SVM, dilihat pada grafiknya yang relatif datar, kecuali pada ukuran referensi sebesar 100 yang membentuk tren yang sama seperti pada pengujian menggunakan nilai gamma sebesar 0.05. Pengujian tersebut juga menghasilkan akurasi yang paling kecil dibandingkan apabila menggunakan ukuran referensi lain pada nilai gamma yang sama, kecuali pada ukuran *buffer* sebesar 200. Pengujian menggunakan ukuran referensi 50 juga memiliki pola yang sama, namun dengan akurasi yang lebih tinggi, serta perbedaan akurasi antar ukuran *buffer* yang lebih kecil. Akurasi tertinggi sebesar 0.99 diraih dengan menggunakan nilai gamma sebesar 0.0085, ukuran sampel referensi sebesar 50, dan ukuran sampel *buffer* sebesar 50.

Pada pengujian ukuran referensi dan *buffer* yang dilakukan, diamati bahwa pada gamma 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 gamma yang kecil, proyeksi data normal akan memiliki persebaran yang luas pula, mengikuti persebaran dari proyeksi data referensi, sedangkan data *faulty* tetapakan diproyeksikan pada daerah yang sempit. Ukuran data referensi yang digunakan juga mempengaruhi persebaran dari data normal dan data *faulty*, dimana semakin besar ukuran data referensi, persebaran data normal dan data *faulty* akan semakin besar pula. Pengaruh ukuran data referensi terhadap persebaran data normal lebih besar dibandingkan terhadap persebaran data *faulty*, dimana dapat dilihat bahwa dengan ukuran data referensi sebesar 200 sampel, data normal memiliki persebaran yang mirip dengan data referensi, sedangkan data *faulty* masih relatif terpusat pada daerah yang sempit.

Selain persebaran proyeksi data, baik nilai gamma maupun ukuran data referensi berpengaruh pada keterpisahan antara proyeksi data normal dan data *faulty*. Pada nilai gamma yang besar, data *buffer* akan di proyeksikan secara bertumpukan didalam persebaran data referensi, peningkatan ukuran data referensi akan memperbesar persebaran dari proyeksi data normal, tapi data *faulty* tetap diproyeksikan secara terpusat didalam persebaran data normal tadi. Sedangkan pada nilai gamma lebih kecil, seiring meningkatnya ukuran data referensi, proyeksi data normal dan data *faulty* akan semakin terpisah. Kombinasi nilai gamma dan ukuran data referensi perlu diperhatikan untuk mendapat hasil optimum, pada nilai gamma 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 *faulty* oleh SVM. Selain itu, meskipun pada nilai gamma yang kecil dengan ukuran data referensi yang besar didapatkan hasil bahwa data *faulty* semakin terpisah dari data normal dan data referensi, penggunaan ukuran data referensi yang terlalu besar justru mengakibatkan data *faulty* diproyeksikan dekat dengan atau bahkan didalam persebaran data normal.

Baik persebaran maupun keterpisahan proyeksi data, keduanya disebabkan oleh ukuran kemiripan antara data normal dan data *faulty* terhadap data referensi. Dengan nilai gamma yang semakin besar, proyeksi sampel pada *buffer* yang memiliki kemiripan dengan sampel referensi akan semakin terpusat, sedangkan proyeksi sampel *buffer* yang berbeda dengan sampel referensi akan semakin tersebar. Namun dengan nilai gamma yang terlalu besar, pengaruh dari sampel-sampel pada data referensi menjadi terlalu besar, yang menyebabkan data *buffer* dinilai sangat mirip dengan data referensi sehingga data *buffer* diproyeksikan pada daerah yang sangat sempit didalam persebaran data referensi.

Dengan semakin besarnya ukuran sampel referensi, maka sebuah sampel pada *buffer* akan semakin mungkin untuk memiliki kemiripan dengan sampel referensi, sehingga data normal akan diproyeksikan dengan persebaran yang mirip dengan data referensi, sedangkan data *faulty* akan diproyeksikan secara terpisah. Namun dengan ukuran sampel referensi yang terlalu besar, data *faulty* akan memiliki kemiripan dengan data referensi. Sehingga proyeksi data *faulty* akan mendekati persebaran data referensi dan data normal.

### Pengujian Metode Clustering

Saat program *fault detection* dijalankan, program *fault detection* harus mampu melakukan *training* SVM menggunakan data pada data referensi dan *buffer.* 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*, dimana 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 *buffer*. Label ini kemudian akan digunakan oleh SVM untuk membuat garis klasifikasi. Salah satu cara melabelkan sampel yang dapat dilakukan adalah melalui metode *clustering*. Pada bagian ini, akan dilakukan pengujian yang membandingkan performa metode *clustering,* yaitu metode K-*means clustering* dan metode DBSCAN.

K­-*means clustering* dipilih untuk diuji karena pada nilai *gamma* yang kecil dan ukuran data referensi yang besar, data *faulty* akan diproyeksikan terpisah dari data normal, namun tersebar. Dengan menggunakan K-*means clustering*, program dapat melakukan *clustering* pada gabungan data referensi dan data *buffer*. K-*means clustering* akan melakukan pengelompokan terhadap data berdasarkan posisinya, itu berarti apabila sebuah kumpulan data terletak jauh dari data referensi, maka K-*means clustering* akan memberikan label yang berbeda kepada 2 kelompok tersebut. Pada pengujian ini, K-means clustering dilakukan dengan cara pertama-tama menggabungkan data referensi dengan data *buffer* menjadi sebuah data gabungan. K-means *clustering* kemudian dilakukan pada data gabungan tersebut dan didapatkan keluaran berupa label dari tiap sampel berdasarkan klasternya. Salah satu kekurangan dari penggunaan K-*means clustering* adalah jumlah *cluster* harus ditentukan sejak awal penggunaan metode, dikarenakan K-*means clustering* tidak dapat mengetahui jumlah klaster yang optimum untuk mengelompokkan data. Program *fault detection* yang dirancang akan menggolongkan sebuah data kedalam 2 kelompok, normal dan *faulty*. Tetapi apabila pada data hanya terdapat 1 jenis data, bisa itu hanya data normal atau data *faulty*, K-*means clustering* akan tetap menggolongkan data menjadi 2 kelompok, sehingga prediksi menjadi tidak akurat.

DBSCAN melakukan pengelompokkan sebuah data tergantung pada kepadatan dari sebuah daerah pada data tersebut. Metode *clustering* ini dipilih untuk diuji karena pada *gamma* kebanyakan hasil pengujian yang dilakukan sebelumnya, diamati bahwa data *faulty* diproyeksikan secara secara terpusat pada sebuah daerah tertentu, sehingga densitas klaster data *faulty* lebih besar dibandingkan densitas klaster data normal. 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 *faulty* diproyeksikan secara tersebar, maka DBSCAN dapat hanya menangkap sebagian dari data *faulty* atau justru salah menggolongkan sebagian data normal sebagai data *faulty* karena kepadatannya. Pengujian dilakukan untuk membandingkan performa K-*means clustering* dan DBSCAN pada berbagai skenario, yaitu variasi nilai gamma, ukuran data referensi, serta jumlah data *faulty* pada *buffer*. Dengan begitu, dapat dilihat performa pada skenario yang menguntungkan dan tidak menguntungkan bagi tiap metode.

Table . Pengujian Metode Clustering

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Faulty Sample** | **Data Size** | **True Positives and True Negatives** | | | | **F-Score** | |
| **5** | **50** | **KMC** | | **DB** | | **KMC** | **DB** |
| Gamma | Ref Size | TP | TN | TP | TN |
| 0,0001 | 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 |
| 0,00005 | 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 |
| 0,00001 | 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 |
| 0,000005 | 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 |
| 0,000001 | 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 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Faulty Sample** | **Data Size** | **True Positives and True Negatives** | | | | **F-Score** | |
| **10** | **50** | **KMC** | | **DB** | | **KMC** | **DB** |
| Gamma | Ref Size | TP | TN | TP | TN |
| 0,0001 | 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 |
| 0,00005 | 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 |
| 0,00001 | 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 |
| 0,000005 | 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 |
| 0,000001 | 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 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Faulty Sample** | **Data Size** | **True Positives and True Negatives** | | | | **F-Score** | |
| **20** | **50** | **KMC** | | **DB** | | **KMC** | **DB** |
| Gamma | Ref Size | TP | TN | TP | TN |
| 0,0001 | 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 |
| 0,00005 | 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 |
| 0,00001 | 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 |
| 0,000005 | 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 |
| 0,000001 | 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 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Faulty Sample** | **Data Size** | **True Positives and True Negatives** | | | | **F-Score** | |
| **15** | **50** | **KMC** | | **DB** | | **KMC** | **DB** |
| Gamma | Ref Size | TP | TN | TP | TN |
| 0,0001 | 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 |
| 0,00005 | 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 |
| 0,00001 | 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 |
| 0,000005 | 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 |
| 0,000001 | 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 |

Dari pengujian yang dilakukan, didapatkan hasil sesuai yang tertera pada tabel V.1, yang berisikan jumlah true positive (TP) dan true negative (TN) menggunakan metode KMC dan DB pada berbagai nilai gamma dan ukuran referensi dengan jumlah data *faulty* sebanyak 5, 10, dan 20 secara berturutan. Jumlah TP dan TN yang tertera pada tabel-tabel tersebut hanya dari penggolongan data *buffer*, tidak termasuk data referensi. Pada keempat tabel, pengujian menggunakan gamma sebesar 0.2 selalu menghasilkan TP sempurna, namun TN selalu bernilai 0 kecuali tabel keempat. Hal ini sesuai dengan yang dijelaskan pada hasil pengujian ukuran referensi dan *buffer*, yaitu karena pada gamma yang terlalu besar, data *buffer* saling bertumpuk pada titik yang sama, baik itu data *normal* maupun data *faulty*, sehingga tidak dapat dipisahkan. Meskipun TP bernilai sempurna, namun pada nilai gamma tersebut, program tidak dapat membedakan data *faulty* dan data normal.

Pada beberapa pengujian menggunakan metode KMC yang ditandai dengan warna merah, didapatkan TP dan TN yang sangat rendah, bahkan bernilai 0 dan 0 secara berturutan. Hal tersebut dikarenakan KMC dengan sendirinya tidak mengetahui apakah kelompok data merupakan kelompok data *faulty* atau kelompok data normal, kemampun KMC terbatas pada mengelompokkan data, sehingga terdapat kondisi dimana pengelompokan KMC akurat, namun dengan label yang terbalik dari yang seharusnya. Kondisi tersebut ditemukan apabila digunakan nilai gamma yang kecil, yaitu pada gamma sebesar 0.02 dan 0.0085. Pada nilai gamma yang kecil, data normal akan diproyeksikan secara terpisah dari data *faulty* sehingga seharusnya lebih mudah bagi KMC untuk mengelompokkan data dibandingkan nilai gamma yang lain. Hasil yang didapatkan pun tidak jauh dari yang ekspektasi, pengujian KMC pada nilai gamma tersebut memiliki F-*score* yang relatif lebih tinggi dibandingkan nilai gamma lain, ditambah lagi mengingat pengujian berwarna merah sebenarnya menghasilkan pengelompokkan yang akurat, namun memiliki label yang keliru. Selain itu, dapat dilihat pula dengan meningkatnya jumlah data *faulty* pada data *buffer*, nilai F-*score* metode KMC relatif meningkat, berikut pengujian yang berwarna merah.

Peningkatan nilai gamma memiliki pengaruh positif bagi metode KMC, semakin besar nilai gamma yang digunakan, akurasi pengelompokan dengan KMC semakin meningkat. Hal tersebut disebabkan oleh semakin terpisahnya data *fault* dari data *normal* seiring meningkatnya nilai *gamma* yang digunakan. Dikarenakan KMC hanya dapat mengelompokkan data yang terpisah secara linier, sedangkan pada nilai gamma yang besar, data *fault* diproyeksikan didalam persebaran data normal, sehingga KMC kesulitan untuk memisahkan data *fault* dari data normal. Sifat KMC tersebut mengakibatkan metode KMC sulit untuk mendapatkan F-*score* bernilai sempurna, karena data *fault* dan data normal harus benar-benar terpisah. Sama seperti KMC, peningkatan nilai gamma juga diikuti oleh meningkatnya akurasi pengelompokan dari DBSCAN. Namun pada nilai gamma sebesar 0.0085, hasil pengelompokan DBSCAN relatif menurun dibandingkan apabila digunakan nilai gamma sebesar 0.05 atau 0.02. Hal tersebut dikarenakan pada nilai gamma yang terlalu kecil, data *faulty* diproyeksikan secara lebih tersebar, tidak lagi terpusat pada 1 daerah, menyebabkan F-*score* dari DBSCAN berkurang. Pada nilai gamma yang terlalu kecil, DBSCAN hanya mengelompokkan sebagian dari data *faulty* sebagai kelompok *faulty*, dan pada beberapa kasus, DBSCAN sama sekali tidak mengelompokkan data *faulty* sebagai kelompok *faulty*. Berbeda dengan metode KMC, DBSCAN dapat melakukan penggolongan data yang tidak terpisah secara linier, sehingga memiliki nilai F-*score* yang lebih besar.

Ukuran data referensi juga mempengaruhi bagaimana data diproyeksikan, dari keempat tabel dapat dilihat bahwa ukuran nilai referensi baru mempengaruhi nilai F-*score* pada nilai gamma yang kecil, dimana meningkatnya ukuran nilai referensi akan diikuti oleh meningkatnya persebaran data normal dan data *faulty*. Pada nilai gamma yang besar, perubahan persebaran data normal dan data *faulty* tidak signifikan untuk dapat merubah nilai F­-*score* dari kedua metode. Dengan meningkatnya ukuran data referensi, nilai F-*score* dari metode KMC ikut meningkat, dapat dilihat dengan jumlah TP dan TN pada metode KMC yang meningkat pada hitam atau yang menurun pada warna merah. Pada metode DBSCAN, ukuran data referensi baru memiliki pengaruh pada nilai gamma sebesar 0.0085. Dengan meningkatnya ukuran data referensi, nilai F-*score* pada nilai gamma tersebut justru semakin mengecil, dikarenakan proyeksi dari data *faulty* semakin menyebar seiring meningkatnya ukuran data referensi yang digunakan, sehingga DBSCAN hanya mengelompokkan sebagian data *faulty* atau bahkan tidak mengelompokkan data *faulty* sama sekali kedalam kelompok *faulty*.

Tidak ada perbedaan signifikan pada peningkatan jumlah data *faulty* terhadap akurasi pengelompokan oleh metode DBSCAN. Meskipun terlihat ada peningkatan nilai F-*score*, namun hal tersebut dikarenakan meningkatnya jumlah data *faulty* yang dapat dikelompokkan oleh DBSCAN. Pada KMC, pengaruh jumlah data *faulty* baru terlihat pada gamma yang kecil, dimana semakin besar jumlah data *faulty* pada *buffer*, F-*score* dari KMC semakin meningkat.

Penggunaan metode DBSCAN sekilas memang menghasilkan nilai F-*score* yang relatif lebih tinggi dibandingkan metode KMC, terutama pada nilai gamma yang besar. Sedangkan pada nilai gamma yang kecil, nilai F­-*score* metode DBSCAN akan jatuh, bahkan akan gagal sama sekali dalam mendeteksi data *faulty*, berbeda dengan metode KMC yang justru akan menghasilkan F-*score* apabila digunakan nilai gamma yang kecil. Perlu diingat bahwa SVM mengklasifikasikan data dengan membentuk garis pemisah antar kelompok data yang memiliki label berbeda. Hasil pengelompokan oleh metode *clustering* berupa label tiap sampel akan digunakan oleh SVM untuk mencari posisi dan orientasi garis pemisah yang optimum. Pada nilai gamma yang lebih tinggi, data *faulty* akan diproyeksikan didalam persebaran data referensi atau bahkan didalam persebaran data *normal*, sehingga meskipun DBSCAN dapat memberikan pengelompokan yang relatif akurat, namun SVM akan kesulitan untuk membuat garis pemisah untuk memisahkan data *faulty* dari data normal secara akurat. Pendekatan yang cocok bagi SVM adalah untuk menggunakan *gamma* yang cukup kecil sehingga data *faulty* akan diproyeksikan secara terpisah dari data referensi dan data normal. Pada data Tennessee Eastman Process, proyeksi data *faulty* pertama kali terpisah pada gamma sebesar 0.02 pada ukuran data referensi sebesar 200. Secara efektif, data *faulty* diproyeksikan secara terpisah pada pengujian menggunakan nilai *gamma* sebesar 0.0085, justru dimana F-*score* dari DBSCAN mulai menurun sedangkan F-*score* dari KMC meningkat atau ditandai dengan warna merah. Sebagai hasil dari pengujian ini, ditentukan bahwa KMC digunakan sebagai metode *clustering* pada program yang dirancang.

### Pengujian K-Means Clustering

Pengujian dilakukan untuk mendapatkan sebuah nilai *threshold* untuk memulai metode K-*means clustering* 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, dimana seorang pengguna mengevaluasi grafik WCSS untuk menentukan jumlah klaster 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 klaster. Sehingga pada tahap ini juga dilakukan pengujian implementasi metode K-means Clustering kedalam rangka kerja. Metode K-means Clustering pada program yang didesain harus dapat melakukan 3 tugas berikut:

a. Menghitung WCSS dari sampel

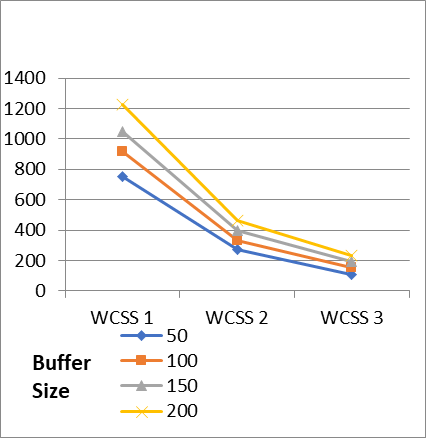
b. Mendeteksi apabila terdapat perbedaan signifikan antara nilai WCSS pada 1 klaster dibandingkan dengan 2 klaster

c. Melabelkan data secara tepat apabila data perbedaan WCSS signifikan

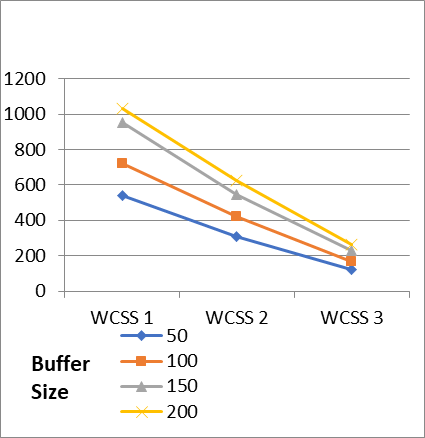
Pengujian dilakukan dengan memvariasikan ukuran referensi dan ukuran *buffer* pada 50 hingga 200 pada gamma sebesar 0.00005. Data referensi akan diisi dengan sampel normal, dan data *buffer* akan diisi dengan campuran sampel normal dan sampel *faulty*. Kedua data tersebut kemudian digabungkan dan dilakukan metode K-*means clustering* untuk dikelompokkan kedalam 2 kelompok, normal dan *faulty*. Apabila pada data *buffer*, jumlah sampel *faulty* terlalu sedikit, maka hasil dari K-*means clustering* akan terlihat seperti sekedar membagi data gabungan menjadi 2 kelompok sama rata. Hal tersebut berarti K-*means clustering* tidak memiliki akurasi tinggi pada komposisi data *buffer* yang digunakan. Oleh sebab itu, pengujian ini juga digunakan untuk mengetahui berapa jumlah data *faulty* pada data *buffer* agar K-means clustering dapat melakukan pelabelan secara akurat. Apabila akurasi pelabelan K-means clustering melewati batas tertentu, maka akan dicatat ketiga nilai WCSS serta jumlah sampel *faulty* yang digunakan. Dari pengujian tersebut didapatkan hasil seperti pada gambar V.8, V.9, V.10 dan V.11.

Tiap garis pada gambar V.8, V.9, V.10, dan V.11 mewakili pengujian menggunakan *buffer* dengan ukuran tertentu, dan tiap titik pada garis mewakili nilai WCSS apabila WCSS dihitung pada jumlah kluster yang bersangkutan. Referensi dengan ukuran 100 sampel memiliki nilai WCSS 1 minimum yang paling besar dibandingkan ukuran referensi lain. Nilai tersebut menandakan bahwa apabila data gabungan dikelompokkan hanya kedalam 1 kluster, terdapat persebaran yang besar pada kluster tersebut. Nilai WCSS 2 relatif rendah dibangingkan WCSS 1, menandakan bahwa persebaran intra-klaster apabila dikelompokkan kedalam 2 klaster jauh lebih kecil dibandingkan apabila hanya pada 1 klaster. ­0

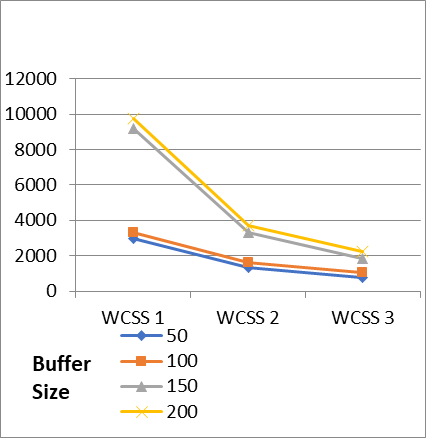
Variasi nilai minimum WCSS pada ketiga gambar utamanya disebabkan oleh perbedaan ukuran referensi serta perbedaan ukuran *buffer*, dimana ukuran referensi akan mempengaruhi proyeksi data dan ukuran *buffer* akan mempengaruhi pengelompokan oleh KMC. Nilai WCSS meningkat seiring membesarnya ukuran referensi yang digunakan, hal tersebut dikarenakan data referensi diproyeksikan dengan persebaran yang besar, sehingga menghasilkan nilai WCSS yang besar pula dan semakin meningkat apabila ukuran referensi meningkat.



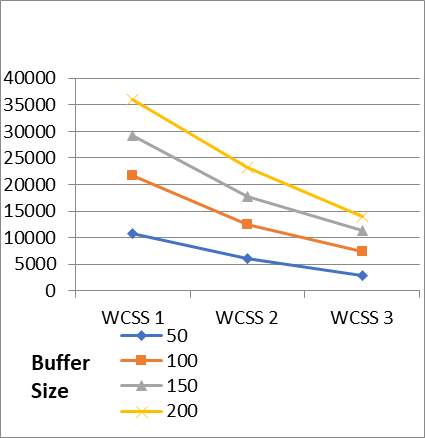
Gambar . Nilai WCSS Minimum Pada Ukuran Referensi = 100



Gambar . Nilai WCSS Minimum Pada Ukuran Referensi = 150



Gambar . Nilai WCSS Minimum Pada Ukuran Referensi = 200



Gambar . Nilai WCSS Minimum Pada Ukuran Referensi = 50

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

Gambar . Sampel *Faulty* Minimum

Pada *elbow method*, sebuah jumlah klaster disebut sebagai jumlah klaster optimal apabila grafik WCSS membentuk sebuah siku (*elbow*) pada jumlah klaster tersebut. Mempertimbangkan metode tersebut, penentuan *threshold* dilakukan dengan cara mencari rasio beda nilai WCSS pada tiap pengujian, dimana rasio didapatkan dengan . Rasio akan meningkat seiring dengan bertambahnya jumlah sampel *faulty* yang terdapat pada *buffer*, sehingga apabilanilai *threshold* terlalu tinggi, berarti program akan membiarkan beberapa sampel *faulty* masuk kedalam *buffer* tanpa deteksi, K-*means clustering* baru dijalankan apabila sampel *faulty* sudah cukup banyak sehingga rasio mencapai *threshold*. Rasio yang paling kecil untuk tiap ukuran referensi ditetapkan sebagai *threshold* pada ukuran referensi tersebut agar K-*means clustering* dapat segera dijalankan dengan jumlah sampel *faulty yang* sedikitmungkin. *Threshold* yang didapatkan yakni: *threshold*50= 1.62, *threshold*100= 3.18, *threshold*150 = 1.2, dan *threshold*200 ­­= 3.46.

**V.2.3. Penentuan Parameter**

Pengujian ini dilakukan untuk menentukan parameter yang akan digunakan sebagai parameter *default* dari program yang dibangun. Pada gambar 5.3 terlihat bahwa akurasi paling tinggi diraih oleh kelompok pengujian dengan parameter gamma sebesar 0.0085 dan ukuran referensi sebesar 50, namun ukuran referensi sebesar 50 sulit untuk dikelompokkan oleh KMC dikarenakan persebaran data yang terlalu besar, yang mana data normal diproyeksikan terpisah dari data referensi, sehingga kelompok parameter tersebut dikeluarkan dari pertimbangan. Pada pengujian sampel *faulty* minimum pada gambar 5.8, pengujian dengan parameter *gamma* sebesar 0.00005, ukuran referensi sebesar 150, dan ukuran *buffer* sebesar 50 membutuhkan sampel *faulty* yang paling kecil dibandingkan pengujian lain. Ditambah lagi, pada gambar 5.3, pengujian dengan parameter tersebut menghasilkan akurasi yang tinggi dibandingkan pengujian lain yakni sebesar 0.93. Oleh karena itu, program yang dibangun akan menggunakan kelompok parameter tersebut.

## Pembangunan Program

### Tahapan Program

Program dibangun menggunakan metode-metode yang telah diuji pada tahap sebelumnya, dan menggunakan parameteryang telah diuji sebagai parameter *default*. Program memiliki 4 tahapan utama, yakni pengolahan referensi, *feature extraction*, *data labeling*, dan *classification*.

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 *robust scaler* untuk mengubah skala dari data tersebut agar tidak ada *feature* yang mendominasi *feature* lainnya. Output yang didapatkan adalah data referensi yang sudah diubah skalanya dan persamaan *robust scaler* yang akan dipakai untuk mengubah skala pada data *buffer* nantinya. Data referensi yang sudah diubah skalanya kemudian digunakan untuk membentuk matriks proyeksi KFDA, setelah itu, tahap pengolahan referensi selesai dan tidak akan dijalankan kembali.

2. Tahap *Feature Extraction*

Setelah matriks transformasi KFDA dibentuk, data yang masuk melalui MQTT kemudian dimasukkan kedalam data *buffer* dan program akan menunggu hingga data *buffer* sudah penuh. Apabila data *buffer* sudah dipenuhi dengan sampel, data *buffer* akan diolah menggunakan *robust scaler* yang didapatkan dari tahap sebelumnya untuk mengubah skala dari data *buffer* dengan persamaan yang sama seperti yang digunakan untuk mengubah skala data referensi. Data *buffer* yang telah diubah skalanya kemudian diproyeksikan kedalam KFDA *subspace* menggunakan matriks *proyeksi* yang telah dibentuk pada tahap sebelumnya menggunakan data referensi. Output dari tahap ini berupa ekstraksi fitur data *buffer* menjadi hanya 2 fitur sehingga dapat di *plot* secara 2 dimensi. Saat sampel baru masuk melalui MQTT, program akan menambahkan sampel baru tersebut kedalam data *buffer* dan membuang sampel paling lama pada data *buffer*, sehingga ukuran data *buffer* tetap seperti yang telah ditetapkan. Tahap *feature extraction* akan dijalankansetiap sampel baru masuk kedalam data *buffer*.

3. Data Labeling

Tahap data labeling merupakan tahap dimana dilakukan pemberian label untuk setiap sampel pada data *buffer*, agar data *buffer* dapat digunakan untuk melatih SVM. Pada tahapan ini, data referensi dan data *buffer* digabung menjadi 1 gabungan data yang akan diolah menggunakan K-*means clustering* untuk memprediksi label dari tiap sampel. Sebelum *clustering* dilakukan, dilakukan pengujian terhadap rasio nilai WCSS data gabungan. Pengujian ini dilakukan dengan cara mencari nilai WCSS untuk 1 hingga 3 klaster, dan dihitung rasio dimana . K-*means clustering* akan dijalankan apabila nilai rasio diatas nilai *threshold* yang digunakan. Hal tersebut dilakukan untuk menjaga akurasi dari program, atau secara spesifik, untuk mencegah K-*means clustering* dari menggolongkan data gabungan yang dipenuhi dengan data normal menjadi 2 kelompok yang akan diartikan oleh SVM sebagai datanormal dan data *faulty*. Output dari K-*means clustering* 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 *faulty*. Data gabungan dan label tersebut digunakan untuk melatih SVM pada tahap berikutnya.

4. Classification

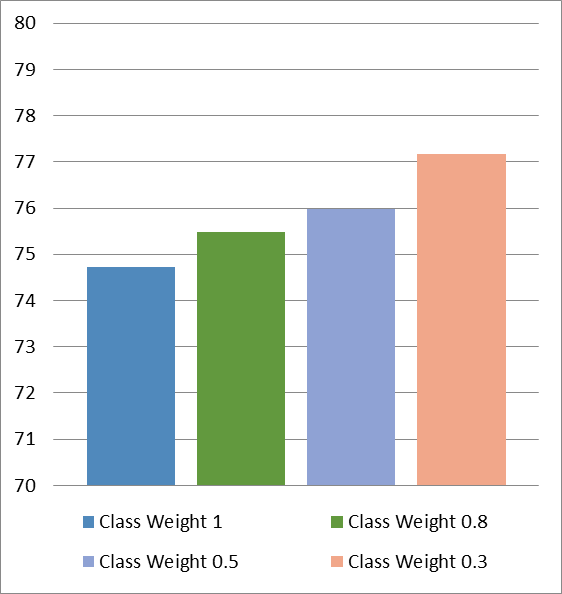
Pada tahap ini, SVM dilatih menggunakan data gabungan dan label yang didapatkan melalui tahap sebelumnya. SVM yang dilatih kemudian digunakan untuk memprediksi label dari setiap sampel, apakah sampel tersebut normal atau *faulty*. Sama seperti tahap sebelumnya, tahap *classification* juga akan dilakukan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

**V.4.2. Pengujian Program Menggunakan Data *Training***

Pengujian program dilakukan untuk menguji performa program yang telah dibangun untuk melakukan *fault detection* pada dataset. Dengan dilakukan pengujian, penulis dapat mengetahui parameter dan metode optimal yang tepat untuk digunakan pada program, sehingga *fault detection* dapat dilakukan dengan akurat.

Apabila program *fault detection* telah mendeteksi *fault* yang terdapat pada data *buffer*, berarti SVM yang dilatih berhasil membuat garis pemisah yang memisahkan data normal dan data *buffer*. Ketika program baru saja mulai mendeteksi *fault*, maka SVM yang dilatih akan membuat garis pemisah yang berubah-ubah bergantung pada posisi sampel berlabel *faulty* terhadap sampel berlabel normal, Lama kelamaan, garis pemisah SVM tidak lagi banyak berubah, dan kemudian menjadi relatif konstan, dan dikatakan bahwa SVM tersebut telah konvergen. Strategi konvergensi SVM ini patut untuk diuji karena pelatihan SVM dapat dihentikan setelah SVM konvergen, sehingga mengurangi beban komputasi program.SVM akan membagi 2 kelas dengan jarak yang sama, terhitung dari jarak antar *support vector,* sehingga apabila terdapat jarak yang besar antara *support vector*, maka SVM akan membagi jarak tersebut menjadi 2 dan mengelompokkan tiap sampel bergantung pada posisinya terhadap garis pemisah tersebut. Hal tersebut membuat strategi konvergensi SVM memiliki sebuah keterbatasan dimana, apabila SVM telah konvergen, maka sebuah data *faulty baru* apabila terproyeksi didalam daerah data normal, maka akan dianggap sebagai data normal oleh SVM. Terlebih lagi, apabila sebuah tipe *fault* terproyeksi didalam daerah data normal, maka SVM akan mengklasifikasikan sebagian atau seluruh sampel dari tipe *fault* tadi sebagai data *faulty*. Sehingga untuk dapat menggunakan strategi konvergensi SVM namun tetap mempertahankan akurasi, salah satu metode yang dapat digunakan adalah dengan mengatur *class weight* dari label data normal, sehingga garis pemisah SVM akan dibuat lebih dekat dengan data normal. Sebagai konsekuensi, daerah data *faulty* memiliki ruang lebih, sehingga apabila sebuah sample dengan jenis *fault* lainterproyeksi lebih dekat dengan data normal, sampel tersebut akan diklasifikan sebagai sampel *faulty*. Pada pengujian ini, akan diuji pengaruh penggunaan bobot kelas dan strategi konvergensi SVM terhadap akurasi *fault detection*. Pengukuran akurasi dilakukan dengan membandingkan prediksi label dari sampel pada data *buffer* 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 ukuran referensi 100 sampel, ukuran *buffer* sebesar 50 sampel, dan gamma sebesar 0.0085, serta pada pengujian ini, data yang masuk ke data *buffer* hanyalah data *faulty*. Hasil pengujian dapat dilihat pada gambar V.12.

Pada gambar V.12, terlihat bahwa dengan mengurangi bobo t kelas data normal menjadi semakin kecil, akurasi *fault detection* semakin meningkat. Hal tersebut dikarenakan meningkatnya sensitvitas SVM terhadap sampel data *faulty* yang menyebabkan algoritma SVM untuk lebih mengutamakan data yang dilabelkan *faulty* oleh metode *clustering* sehingga garis pemisah SVM akan diletakkan lebih jauh dari data *faulty* dan mengupayakan agar seluruh data *faulty* masuk kedalam daerah *faulty* pada garis pemisah. Gambar V.13 menunjukkan pembebaran secara terperinci terhadap akurasi program *fault detection* dalam mendeteksi tiap jenis *fault,* dimana hasil pada gambar V.13 didapatkan dari pengujian yang sama dari gambar V.12.



Gambar . Akurasi Program Pada Data Latih Menggunakan Variasi Bobot Kelas Normal

Pada gambar V.13, terlihat bahwa hampir semua mengujian memiliki kesulitan dalam mendeteksi data *faulty* pada *fault* tipe 5 dan *fault* tipe 7. Hal tersebut dikarenakan proyeksi sampel *fault* tipe 5 dan *fault* tipe 7 bertumpukan dengan data referensi, sehingga program *fault detection* mengalami kesulitan dalam memisahkan sampel *fault* tipe 5 dan *fault* 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 *faulty* dan menyebabkan akurasi SVM meningkat seiring mengecilnya bobot kelas dari data normal.

Gambar . Akurasi Program Pada Jenis *Fault* Menggunakan Variasi Bobot Kelas

*Fault* tipe 5 dan *fault* 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 *fault* 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 *steady* *state*. 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 *fault*, 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 *fault* tersendiri, program yang dirancang tidak dapat mendeteksi osilasi, hanya mampu mendeteksi *fault* berdasarkan proyeksi tiap sampel. Setelah data *fault 5* mencapai *steady state*, yakni ketika bacaan sensor tidak lagi berosilasi, data *fault 5* menghasilkan bacaan sensor yang mirip dengan data normal, yang menyebabkan data *steady state fault* tipe 5 diproyeksikan didalam persebaran data normal dan kemudian menyebabkan misklasifikasi. Misklasifikasi data *fault* tipe 7 memiliki penyebab yang sama dengan *fault* tipe 5, namun pada data *fault* tipe 7, *fault* disebabkan oleh masukan *step* berupa penurunan tekanan umpan C.

Bobot kelas data normal yang kecil memiliki berpengaruh terhadap deteksi data normal. Gambar V.14 merupakan akurasi program dalam melakukan klasifikasi terhadap data normal dari pengujian program menggunakan data *training*. Hasil tersebut didapatkan karena WCSS *threshold* yang digunakan terlalu kecil, sehingga K-*means clustering* akan melakukan pelabelan data tanpa menunggu nilai rasio WCSS mencapai *threshold* yang kemudian menyebabkan klastrering dan klasifikasi akan langsung dijalankan karena program selalu berasumsi bahwa terdapat data *faulty*. Dengan bobot data normal yang kecil, program *fault detection* akan mengalami kesulitan dalam mengenali sampel normal, dan justru akan memprediksi sampel normal sebagai sampel *faulty*. Hal ini merupakan akibat dari pelabelan sampel yang tidak akurat pada tahap K-*means clustering*. Pelabelan yang tidak akurat, atau secara lebih spesifik: pelabelan sampel normal sebagai sampel *faulty* oleh K-*means clustering,* hasil pelabelan akan digunakan oleh SVM untuk membuat garis pemisah yang memisahkan sampel berlabel normal dan sampel berlabel *faulty*. Pada bobot kelas data normal yang rendah, SVM akan lebih peka terhadap sampel berlabel *faulty* dan kurang memperdulikan data berlabel normal. Sebagai konsekuensi, SVM akan membuat garis pemisah yang melintasi klaster data normal tersebut demi data berlabel *faulty.* Sedangkan apabila data normal memiliki bobot kelas 1, dimana bobot kelas data berlabel normal akan sama dengan bobot kelas data berlabel *faulty*, alih-alih membuat garis pemisah yang menembus klaster data berlabel normal, SVM justru ak­­an membuat garis pemisah diujung klaster normal, karena SVM tidak lagi lebih sensitif terhadap data berlabel *faulty*, dan sebagai hasilnya program *fault detection* menghasilkan akurasi yang lebih tinggi untuk mendeteksi data normal.

Dari pengujian terhadap data *training*, dapat dilihat bahwa program menghasilkan akurasi yang relatif tinggi, dengan akurasi paling rendah sebesar 74.72% (bobot kelas 1), dan tertinggi sebesar 77.17% (bobot kelas 0.3). Program dapat mendeteksi *fault* apabila data *fault* diproyeksikan diluar persebaran data normal seperti data *fault* tipe 2 dan 6, namun untuk *fault* yang diproyeksikan didalam persebaran data normal seperti data *fault* tipe 5 dan 7, akurasi klasifikasi program menjadi rendah. Seperti yang disebutkan sebelumnya, program tidak dapat mendeteksi *fault* tipe 5 dan 7 karena mereka memiliki bacaan sensor yang mirip dengan data normal, sehingga diproyeksikan didalam persebaran data normal. Untuk data Tennessee Eastman Process, *fault* tipe 5 dan 7 dapat diproyeksikan secara terpisah dengan memperkecil nilai *gamma* pada kernel. Perubahan nilai *gamma* tersebut akan menyebabkan program memperkecil rentang kemiripan dari data referensi, sehingga *fault* tipe 5 dan 7 akan diproyeksikan terpisah dari persebaran data referensi, walau begitu, data normal juga akan dipisahkan dari persebaran data referensi dengan alasan yang sama seperti *fault* tipe 5 dan 7, menyebabkan program akan mendeteksi data normal sebagai data *fault*.

Gambar . Pengaruh Bobot Kelas Pada Akurasi Klasifikasi Data Normal

Pengujian dengan bobot kelas 0.3 dapat mencapai nilai akurasi yang lebih tinggi lagi apabila *threshold* rasio WCSS program sesuai dengan rasio WCSS ketika data *fault* masuk ke dalam program. Meskipun begitu, penentuan *threshold* rasio WCSS yang tepat sulit untuk ditentukan, karena apabila rasio WCSS terlalu rendah, maka *clustering* akan tetap dijalankan meskipun pada *buffer* yang dipenuhi dengan data normal saja. Namun apabila rasio WCSS terlalu tinggi, maka apabila terdapat tipe *fault* yang diproyeksikan didalam persebaran data normal seperti data *fault* tipe 5 dan 7, maka persebaran tersebut akan menghasilkan rasio WCSS yang relatif kecil sehingga *clustering* tidak akan dijalankan dan mengakibatkan *fault* 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 *threshold* berupa rasio WCSS tidak efektif untuk digunakan pada program yang dirancang, karena data *fault* dapat diproyeksikan diluar maupun didalam persebaran data normal.

## Pengujian Program Menggunakan Data Validasi

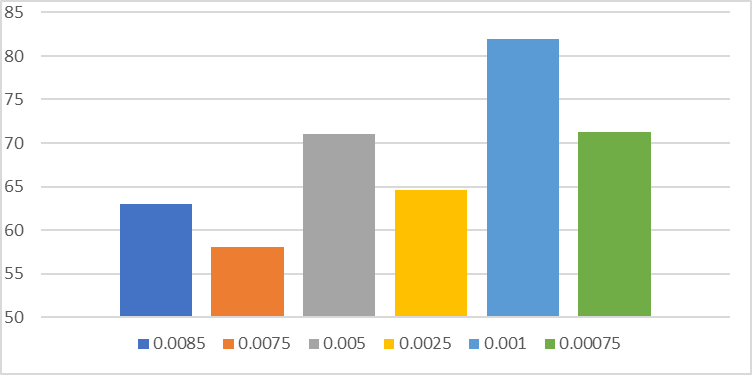
### V.4.1. Hasil Pengujian Program Menggunakan Data Validasi

Meskipun terdapat beberapa kekurangan, program yang dirancang berhasil melakukan *fault detection* pada data Tennessee Eastman Process dengan akurasi tertinggi sebesar 77.17%, mengingat program yang dirancang tidak memerlukan *model training* secara eksplisit, melainkan *training* dilakukan dengan menerima 150 data pertama yang masuk yang kemudian dijadikan sebagai referensi data normal. Tahap validasi dilakukan untuk menguji apakah program yang dirancang dengan parameter yang telah ditentukan sebelumnya dapat melakukan *fault detection* pada data selain Teneessee Eastman Process yaitu pada data dari penelitian Brooks. Pada tahap ini, program akan dijalankan untuk melakukan *fault detection* pada data dari penelitian Brooks dilihat performanya, menggunakan parameter-parameter persis seperti yang digunakan pada pengujian menggunakan Tennessee Eastman Process.

Pada pengujian menggunakan bobot kelas normal senilai 1, program memiliki akurasi sebesar 63.03%. Dari pengujian tersebut, diamati bahwa data *buffer* diproyeksikan dengan persebaran yang berbeda dengan data referensi, meskipun data *buffer* berisi data normal, sehingga menyebabkan misklasifikasi dimana data normal diklasifikasikan sebagai data *faulty*. Pada beberapa kejadian, data *fault* dan data normal diproyeksikan secara terpisah, namun karena data normal diproyeksikan terpisah dari data referensi, data normal tersebut tetap diklasifikasikan sebagai data *faulty*, mengikuti pelabelan oleh KMC, namun sering kali data *fault* dan data normal diproyeksikan secara bertumpukan dan terpisah dari data referensi. Hal tersebut mengindikasikan bahwa program menganggap bahwa data normal sama sekali tidak mirip dengan data referensi, sehingga diproyeksikan di daerah yang sama dengan data *faulty.* Pada pengujian ini, WCSS *threshold* memiliki andil dalam mengendalikan kapan *clustering* harus dijalankan, berbeda dengan pada pengujian menggunakan data *training* dimana WCSS *threshold* terlalu kecil. Pada beberapa kejadian, *threshold* WCSS menahan program dari melakukan *clustering* pada data yang semuanya merupakan data normal karena nilai rasio WCSS belum mencapai *threshold* dan program terhindar dari misklasifikasi. Namun karena proyeksi data normal maupun data *faulty* terpisah dari data referensi, terdapat beberapa kejadian dimana *threshold* WCSS justru menyebabkan misklasifikasi karena rasio WCSS tidak mencapai *threshold* padahal terdapat data *faulty*. Pengujian validasi menggunakan data dari penelitian brooks menunjukkan bahwa program dengan parameter yang telah ditentukan tidak dapat melakukan *fault detection* dengan akurat, yang perlu diingat bahwa parameter yang digunakan di pengujian ini merupakan parameter yang menghasilkan hasil terbaik apabila menggunakan data Tennessee Eastman Process. Pengujian ini membuktikan bahwa parameter dari program yang dirancang tidak bersifat universal, parameter harus disesuaikan dengan karakteristik data dari tiap pabrik. Pada bagian berikutnya, penulis akan menguji parameter yang tepat untuk melakukan *fault detection* secara akurat menggunakan data dari penelitian *brooks*. Pengujian tersebut bertujuan untuk menguji apakah apabila parameter telah disesuaikan, program *fault detection* dapat menghasilkan akurasi yang lebih baik.

### V.4.2. Penyesuaian Parameter Program Untuk Data Validasi

Akurasi yang rendah ini utamanya disebabkan karena kebanyakan data normal diproyeksikan secara terpisah dari data referensi, dan terdapat banyak kejadian dimana data *faulty* diproyeksikan secara bertumpukan dengan data normal. Parameter yang mempengaruhi kejadian tersebut adalah nilai *gamma* dan ukuran data referensi. Nilai gamma dapat mempengaruhi proyeksi dengan memperkecil pengaruh dari tiap data referensi sehingga proyeksi dari data referensi akan lebih menyebar. Dengan mengecilnya pengaruh data referensi, maka akan mengecil pula kejadian dimana data normal dan data referensi diproyeksikan secara bertumpukan. Pengujian ini akan membandingkan beberapa pengaruh nilai gamma pada akurasi *fault detection* untuk data penelitian *brooks*.



Gambar . Akurasi Program Pada Data Validasi Menggunakan Ukuran Referensi = 150 Dengan Variasi Nilai Gamma

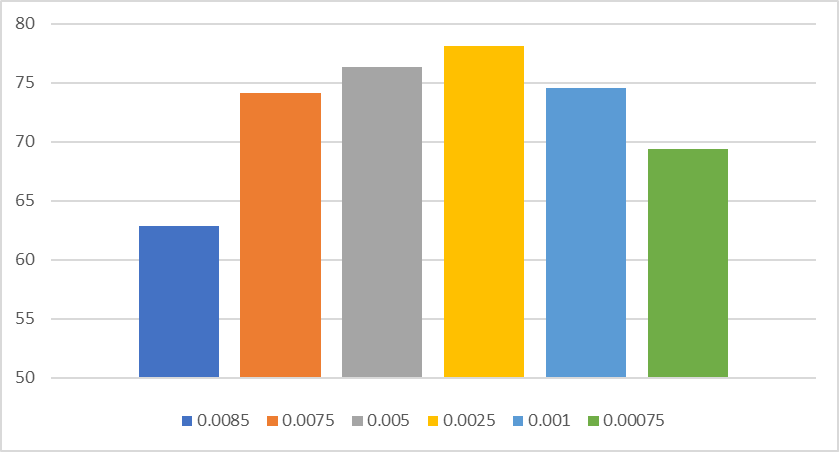
Gambar V.15memberikan informasi mengenai akurasi *fault detection* yang dihasilkan oleh program menggunakan nilai-nilai gamma tertentu. Akurasi paling tinggi sebesar 81.97 dihasilkan oleh pengujian menggukan nilai *gamma* sebesar 0.001 dan akurasi paling rendah sebesar 58.05 dihasilkan oleh pengujian menggunakan nilai gamma sebesar 0.0075. Dapat dilihat bahwa pada rentang nilai gamma yang diuji terdapat 2 kelompok pola utama yang terus meningkat seiring semakin mengecilnya nilai *gamma*, kelompok pola tersebut adalah pengujian dengan nilai gamma sebesar 0.0085, 0.005, 0.001 dan 0.0075, 0.0025, 0.00075. Selain itu, hanya 1 pengujian nilai *gamma* yang menghasilkan akurasi yang lebih rendah dibandingkan nilai akurasi awal (pengujian menggunakan nilai gamma sebesar 0.0085). Meski munculnya pola tersebut merupakan sesuatu yang tidak dapat diprediksi sebelumnya oleh penulis ataupun pengguna program, namun hal tersebut menunjukkan bahwa meningkatkan akurasi program terhadap sebuah data pabrik dengan menyesuaikan nilai gamma merupakan tugas yang mudah, namun sulit untuk menemukan nilai gamma yang menghasilkan akurasi paling tinggi.

Hal lain yang dapat menyebabkan akurasi *fault detection* yang rendah pada pengujian ini adalah karena pada data yang digunakan terdapat variasi data normal. Secara spesifik, 150 data awal yang digunakan oleh program sebagai data referensi tidak secara penuh mewakili data normal, sebagai akibatnya, banyak data normal diproyeksikan diluar persebaran data referensi yang kemudian membuat data tersebut diklasifikasikan sebagai data *faulty*. Sensor yang menghasilkan bacaan dengan variasi data normal khususnya adalah sensor AI2, yang memiliki bacaan %Fe yang sangat bervariasi serta sensor AI3. Pada kedua sensor tadi, program dapat diuntungkan apabila menerima data referensi yang lebih banyak untuk mendapatkan data referensi yang mewakili variasi nilai data normal. Selain itu, pada sensor FC4, FC5, FC6, FC7, FC8, dan FI1 terdapat penurunan pengukuran secara drastis namun tetap merupakan data normal. Penurunan tersebut terjadi tidak lama setelah batas data *training* yang digunakan pada penelitian *brooks*. Program dapat diuntungkan dengan menerima data tersebut sebagai data referensi, sayangnya pada waktu yang sama, terdapat beberapa sensor *fault* pada sensor AI1 dan AI2 (yang tidak berhubungan dengan penurunan bacaan 6 sensor tadi), sehingga data tersebut tidak dapat digunakan sebagai data referensi.

Pengujian berikut menggunakan data referensi yang sama dengan data *training* pada penelitian brooks, yakni sejumlah 301 data sejak 29 September 2015 pukul 02.00 hingga 1 Oktober 2015 pukul 04.00 dan nilai gamma sebesar 0.005. Rentang data tersebut digunakan karena memiliki variasi data normal yang lebih besar dibandingkan apabila hanya digunakan 150 data. Pengujian dilakukan menggunakan rentang nilai *gamma* yang sama seperti pengujian sebelumnya. Hasil pengujian dapat dilihat pada Gambar V.16.

Berbeda dibandingkan apabila menggunakan 150 sampel sebagai data referensi, apabila menggunakan 300 sampel, data referensi diproyeksikan secara menyebar dalam sebuah garis diagonal, namun tidak banyak data normal yang diproyeksikan jauh dari persebaran data referensi tersebut, sehingga mengurangi misklasifikasi dari data normal. Data *faulty* selalu diproyeksikan terpisah dari data referensi, jarang terjadi misklasifikasi dari data *faulty* menjadi data normal. Misklasifikasi biasanya terjadi sebagai akibat dari data sekelompok data normal yang tersebar, namun terproyeksikan terpisah dari data referensi, sehingga kemudian diklasifikasikan sebagai data *faulty*. Meskipun begitu, terdapat sekelompok data normal yang masih diproyeksikan bertumpukan dengan data *faulty* sehingga terjadi misklasifikasi. Dari pengujian yang dilakukan didapatkan bahwa akurasi paling rendah sebesar 62,91% dihasilkan oleh pengujian menggunakan gamma sebesar 0.0085, yakni nilai gamma yang direkomendasikan oleh pengujian menggunakan data Tennessee Eastman Process. Nilai akurasi kemudian meningkat seiring mengecilnya nilai gamma dan memuncak pada gamma sebesar 0.0025 dengan akurasi sebesesar 78.14%, dan kemudian menurun seiring mengecilnya nilai gamma.

Dapat dilihat meskipun ukuran referensi sebesar 300 memiliki variasi data yang lebih besar daripada ukuran referensi sebesar 150, namun akurasi tertinggi tidak dihasilkan oleh pengujian dengan ukuran referensi sebesar 300. Pada rentang nilai gamma yang diuji, data referensi memiliki persebaran yang relatif lebih besar dibandingkan apabila menggunakan ukuran data referensi sebesar 150, hal tersebut dikarenakan data referensi memiliki variansi yang besar, sehingga akan diproyeksikan secara terpisah apabila menggunakan gamma yang kecil. Meskipun begitu, pengujian menggunakan ukuran referensi sebesar 301 menghasilkan nilai akurasi yang lebih tinggi pada beberapa rentang nilai gamma 0.0075 hingga 0.0025, dan menghasilkan nilai akurasi yang tidak kalah jauh pada nilai gamma 0.0085 dan 0.00075. Selain itu, pengujian juga menghasilkan peningkatan dan penurunan nilai akurasi yang relatif lebih stabil dibandingkan pengujian dengan ukuran data referensi sebesar 150, sehingga meskipun tidak menghasilkan nilai akurasi yang paling tinggi, penyesuaian nilai gamma lebih mudah dilakukan pada pengujian menggunakan ukuran referensi sebesar 301.



Gambar . Akurasi Program Pada Data Validasi Menggunakan Ukuran Referensi = 300 Dengan Variasi Nilai Gamma

# KESIMPULAN DAN SARAN

## Kesimpulan

Penelitian yang telah dilakukan bertujuan untuk merancang dan menguji performa dari program untuk melakukan *fault detection* dengan fase *training* secara *on-the-fly*, dimana tiap sampel masuk kedalam program satu per satu, meniru kegiatan kumpulan sensor sebuah pabrik kimia yang memiliki waktu cuplikan tertentu. Beberapa sampel pertama yang masuk akan digolongkan sebagai data referensi, yakni acuan data normal untuk program. Program yang dirancang menggunakan teknik *semi-supervised learning* dimana data yang masuk akan diolah oleh metode *clustering* untuk memberi label dari tiap data dan dilanjutkan dengan *classification* 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 *feature* dari data program menjadi 2, sehingga data dapat diproyeksikan kedalam grafik 2 dimensi. Setelah itu, sampel-sampel baru akan dimasukkan kedalam *buffer* untuk diolah. Terdapat 3 tahap utama ketika sampel baru masuk kedalam program, *pre-processing, clustering*, *classification*. Setelah data diolah dengan *robust scaler*,sampel baru akan diproyeksikan kedalam KFDA *feature space* menggunakan matriks KFDA. Sampel baru dan sampel lainnya didalam *buffer* kemudian diolah dengan metode *clustering* berupa *k-means clustering* untuk mendapatkan label dari tiap sampel. Metode *classification* berupa SVM kemudian digunakan untuk membuat garis pemisah yang optimum antara data *referensi* dan data normal dengan data *faulty*.

Program yang dirancang mendapatkan akurasi sebesar 77,17% pada kumpulan data Tennessee Eastman Process dan mendapatkan akurasi hingga 81,97% pada kumpulan data pada penelitian Brooks. Dari penelitian yang dilakukan juga disimpulkan bahwa untuk setiap pabrik, parameter dari program harus disesuaikan terlebih dahulu dengan karakteristik data dari pabrik. Pada penelitian ini, parameter yang disesuaikan adalah nilai gamma dan ukuran data referensi.

## Saran

Pada penelitian ini, dihasilkan program yang dapat melakukan *fault detection* 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 *fault detection* yang optimum. Studi lebih lanjut juga diperlukan untuk menguji metode *feature extraction*, *clustering*, dan *classification* yang optimum untuk diimplementasikan pada program yang dirancang. 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," *Chemometrics and Intelligent Laboratory Systems,* vol. 50, no. 2, p. 243–252, 2000. |
| [2] | Y. Yin, J. Shi, F. Liu and Y. Liu, "Robust fault detection of singular Markov jump systems with partially unknown information," *Information Sciences,* vol. 537, p. 368–379, 2020. |
| [3] | V. N. Vapnik, The Nature of Statistical Learning Theory, New York: Springer, 2000. |
| [4] | J. Liu, Y.-F. Li and E. Zio, "A SVM framework for fault detection of the braking system in a high speed train," *Mechanical Systems and Signal Processing,* vol. 87, pp. 401-409, 2017. |
| [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," *Computers in Industry,* vol. 62, pp. 42-50, 2011. |
| [6] | Z. M. Hira and D. F. Gillies, "A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data," *Advances in Bioinformatics,* vol. 1, pp. 1-13, 2015. |
| [7] | Z.-B. Zhu and Z.-H. Song, "A novel fault diagnosis system using pattern classification on kernel FDA subspace," *Expert Systems with Application,* vol. 38, pp. 6895-6905, 2011. |
| [8] | P. Konar and P. Chattopadhyay, "Bearing fault detection of induction motor using wavelet and Support VectorMachines (SVMs)," *Applied Soft Computing,* vol. 11, pp. 4203-4211, 2011. |
| [9] | J. Yang, Z. Jin, J.-y. Yang, D. Zhang and A. F. Frangi, "Essence of kernel Fisher discriminant: KPCA plus LDA," *Pattern Recognition,* vol. 37, pp. 2097-2100, 2004. |
| [10] | S. Mika, G. Ratsch, J. Weston and B. Scholkopf, "Fisher Discriminant Analysis with Kernels," in *Neural Networks for Signal Processing IX*, 1999. |
| [11] | G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural Computation,* vol. 12, 2000. |
| [12] | S. Yin, X. Gao, H. R. Karimi and X. Zhu, "Study on Support Vector Machine-Based Fault Detection in Tennessee Eastman Process," *Abstract and Applied Analysis,* 2014. |
| [13] | S. Gunn, "Support Vector Machines for Classification and Regression," 1998. |
| [14] | C.-W. Hsu and C.-J. Lin, "A Comparison of Methods for Multi-class Support Vector Machines," National Taiwan University. |
| [15] | 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 acylic graph support vector machine," *Advances in Mechanical Engineering,* pp. 1-6, 2016. |
| [16] | A. Ng, "CS229 Lecture notes - Supervised learning," 2012. |
| [17] | L. v. d. Maaten, E. Postma and J. v. d. Herik, "Dimensionality Reduction: A Comparative Review," Tilburg University, 2009. |
| [18] | R. Iserman, "Model-based fault-detection and diagnosis – status and applications," *Annual Reviews in Control,* vol. 29, pp. 71-85, 2005. |
| [19] | Y.-J. Yoo, "Data-driven fault detection process using correlation based clustering," *Computers in Industry,* vol. 122, 2020. |
| [20] | Q. Yang, "MODEL-BASED AND DATA DRIVEN FAULT DIAGNOSIS METHODS WITH APPLICATIONS TO PROCESS MONITORING," 2004. |
| [21] | 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, Redbooks, 2012. |
| [22] | IBM Eurotech, "MQTT V3.1 Protocol Specification," 2010. [Online]. Available: https://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html. [Accessed July 2020]. |
| [23] | F. Konstantina, "Supervision of Data Transfer With Python," Heraklion, 2016. |
| [24] | M. Kpizingui, "Demystifying the MQTT maze: clients, servers, connection, publish, subscribe and its applications.," [Online]. Available: https://morioh.com/p/93ba2353480e. [Accessed July 2020]. |
| [25] | hefinioanrhys, "What is the support vector machine (SVM) algorithm?," October 2019. [Online]. Available: https://www.r-bloggers.com/support-vector-machines-with-the-mlr-package/. [Accessed July 2020]. |
| [26] | L. H. Chiang, E. L. Russell and R. D. Braatz, "Fault diagnosis in chemical processes using Fisher discriminant," *Chemometrics and Intelligent Laboratory Systems,* vol. 50, no. 2, p. 243–252, 2000. |

# LAMPIRAN