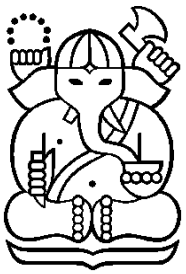
MENGEVALUASI PENGGUNAAN *MACHINE LEARNING* UNTUK PERAWATAN PREDIKTIF SARANA PERKERETAAPIAN

TESIS

Karya tulis sebagai salah satu syarat
untuk memperoleh gelar Magister dari
Institut Teknologi Bandung

Oleh
HAFID GALIH PRATAMA PUTRA
NIM: 23220095
(Program Magister Teknik Elektro)



INSTITUT TEKNOLOGI BANDUNG
November 2021

ABSTRAK

MENGEVALUASI PENGGUNAAN *MACHINE LEARNING* UNTUK PERAWATAN PREDIKTIF SARANA PERKERETAAPIAN

Oleh
Hafid Galih Pratama Putra
NIM: 23220095
(Program Magister Teknik Elektro)

Dalam era informasi ini, penggunaan komputer dan berbagai perangkat digital sudah tidak asing lagi dalam kehidupan sehari-hari. Perangkat tersebut dapat membantu dalam pekerjaan manusia dengan mengolah informasi dan data secara cepat. Dengan demikian, penerapan perangkat tersebut dalam industri mampu meningkatkan produktivitas. Di Indonesia beberapa sektor sedang berusaha untuk beradaptasi dan menerapkan digitalisasi dalam proses bisnisnya.

Salah satu sektor tersebut adalah industri transportasi, dimana sekarang sudah banyak sistem transportasi yang memanfaatkan teknologi terkini dalam proses bisnisnya. PT Kereta Api Indonesia dapat melakukan Transformasi digital Industri, dan diantara berbagai proses bisnis yang dapat dilakukan digitalisasi, perawatan aset atau *maintenance* merupakan proses krusial dalam operasi kereta api.

Dalam tesis ini akan dibahas potensial penggunaan perawatan prediktif dapat bermanfaat dalam memastikan lokomotif dan kereta pembangkit dalam kondisi baik, dengan memanfaatkan kecerdasan buatan berupa pembelajaran mesin yang dapat membantu otomatisasi pemeriksaan rutin dengan melakukan klasifikasi berdasarkan masukan nilai sensor.

Kata Kunci: Perawatan prediktif, kereta api, pembelajaran mesin.

ABSTRACT

EVALUATION OF MACHINE LEARNING USE FOR PREDICTIVE MAINTENANCE OF RAILWAY VEHICLES

By

Hafid Galih Pratama Putra
NIM: 23220095
(Master's Program in Electrical Engineering)

In this information age, the use of computers and various digital devices is familiar in everyday life. These devices can assist in human work by processing information and data quickly. Thus, the application of these devices in the industry can increase productivity. In Indonesia, several sectors are trying to adapt and implement digitalization in their business processes.

One of these sectors is the transportation industry, where now many transportation systems utilize the latest technology in their business processes. Here is the opportunity for PT Kereta Api Indonesia to carry out the Digital Transformation. Among the various business processes that can be digitized, asset maintenance or maintenance is essential in railroad operations.

This thesis will discuss the potential of predictive maintenance can be useful in ensuring that trains vehicles are in good condition by utilizing the artificial intelligence method of machine learning, which can help automate routine checks by classifying its condition from the on-board sensors.

Keywords: Predictive maintenance, trains, machine learning.

MENGEVALUASI PENGGUNAAN *MACHINE LEARNING* UNTUK PERAWATAN PREDIKTIF SARANA PERKERETAAPIAN

Oleh
HAFID GALIH PRATAMA PUTRA
NIM: 23220095
(Program Magister Teknik Elektro)

Institut Teknologi Bandung

Disetujui
Tim Pembimbing

Tanggal 29 November 2021

Pembimbing

_____

Prof. Dr.Ir. Suhono Harso Supangkat, M.Eng.

PEDOMAN PENGGUNAAN TESIS

Tesis Magister yang tidak dipublikasikan terdaftar dan tersedia di Perpustakaan Institut Teknologi Bandung, dan terbuka untuk umum dengan ketentuan bahwa hak cipta ada pada penulis dengan mengikuti aturan HaKI yang berlaku di Institut Teknologi Bandung. Referensi kepustakaan diperkenankan dicatat, tetapi pengutipan atau peringkasan hanya dapat dilakukan seizin penulis dan harus disertai dengan kaidah ilmiah untuk menyebutkan sumbernya.

Sitasi hasil penelitian Tesis ini dapat di tulis dalam bahasa Indonesia sebagai berikut:

Putra, Hafid G.P. (2021): *Mengevaluasi Penggunaan Machine Learning untuk Perawatan Prediktif Sarana Perkeretaapian*, Tesis Program Magister, Institut Teknologi Bandung.

dan dalam bahasa Inggris sebagai berikut:

Putra, Hafid G.P. (2021): *Evaluation of Machine Learning Use for Predictive Maintenance of Railway Vehicles*, Master's Thesis, Institut Teknologi Bandung.

Memperbanyak atau menerbitkan sebagian atau seluruh tesis haruslah seizin Dekan Sekolah Pascasarjana, Institut Teknologi Bandung.

Dipersembahkan kepada kedua orang tua yang ku hormati, sayangi, dan bukti bakti ku atas segala bantuan yang telah kalian berikan.

KATA PENGANTAR

Puji dan syukur penulis haturkan kepada Allah SWT atas rahmat dan karunia yang telah diberikan sehingga penulis dapat menyelesaikan tesis yang berjudul "*Mengevaluasi Penggunaan Machine Learning untuk Perawatan Prediktif Sarana Perkeretaapian"* sebagai prasyarat untuk memperoleh gelar Magister Teknik bagi mahasiswa program S2 di Program Studi Teknik Elektro Institut Teknologi Bandung.

Dalam proses pengerjaan tesis ini, penulis mendapatkan banyak bantuan dari berbagai pihak. Oleh karena itu, penulis ingin mengucapkan terima kasih kepada:

1. Ibu dan Ayah, atas semua dukungan, perhatian, serta kasih sayang untuk penulis dalam menyelesaikan studi di program Maagister ITB.
2. Ibu Prof. Reini Wirahadikusumah, Ph.D. Selaku Rektor Institut Teknologi Bandung saat penulisan tesis.
3. Bapak Prof. Dr.Ir. Suhono Harso Supangkat, M.Eng. Selaku Dosen Pembimbing tesis penulis yang telah memberikan kesempatan untuk mendalami topik ini, serta kritik dan saran maupun arahan yang bermanfaat dalam pelaksanaan penelitian ini.
4. Bapak Dr. Y. Bandung, ST., MT. Selaku Dosen Pengajar Metode Penelitian bagi penulis yang telah banyak memberikan pengetahuan dan saran maupun arahan yang bermanfaat dalam penulisan tesis ini.
5. PT. Kereta Api Indonesia, selaku pihak kerjasama dalam penelitian tesis ini yang menyediakan kesempatan untuk dapat berkontribusi dalam perkembangan teknologi industri negara Indonesia.
6. Teman Teknik Elektro Magister Institut Teknologi Bandung Angkatan 2020 Ganjil, yang senantiasa bersama penulis berjuang dan memberikan semangat yang berarti bagi satu sama lain.
7. Semua pihak yang telah membantu dalam penyelesaian tesis ini yang tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa karya tulis ini masih jauh dari kata sempurna. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan.

Akhir kata penulis mengucapkan terima kasih kepada semua pihak yang telah membaca karya tulis ini. Semoga bermanfaat.

Jakarta, November 2021

Penulis,

Hafid Galih Pratama Putra

NIM. 23220095

DAFTAR ISI

DAFTAR LAMPIRAN

DAFTAR GAMBAR DAN ILUSTRASI

DAFTAR TABEL

DAFTAR SINGKATAN DAN LAMBANG

SINGKATAN	Nama	Halaman awal digunakan
PT	Perseroan Terbatas	
KAI	Kereta Api Indonesia	1
ML	*Machine Learning*	3
PdM	*Predictive Maintenance*	3
RUL	*Remaining Useful Lifetime*	4
CBM	*Condition Based Maintenance*	6
MAE	*Mean Absolute Error*	6
DRM	*Design Research Methodology*	7
DL	*Deep Learning*	7
RC	*Research Clarification*	8
DS-I	Deskriptif Studi I	8
PS	Preskriptif Studi	9
DS-II	Deskriptif Studi II	10
DT	*Decision Tree*	12
ANN	*Artificial Neural Network*	15
LSTM-RNN	*Long Short Term Memory-Recurrent Neural Network*	16
SVM	*Support Vector Machine*	18
KNN	*K-Nearest Neighbor*	18
RF	*Random Forest*	22
LR	*Logistic Regression*	22
DBSCAN	*Density-Based Spatial Clustering of Applications with Noise*	29
RPM	Rotasi Per Menit	30
ECU	*Electronic Control Unit*	30
MTBM	*Mean Time Between Maintenance*	42
SMOTE	*Synthetic Minority Oversampling TEchnique*	44
LDA	*Linear Discriminant Analysis*	45
NB	*Naïve Bayes*	45
OLS	*Ordinary Least Square*	48
RF-reg	*Random Forest-Regressor*	48

BAB I PENDAHULUAN

Agar tidak terjadi kesalahpahaman mengenai istilah yang akan digunakan pada Tesis ini, akan dijelaskan pengertiannya terlebih dulu. Berdasarkan beberapa Peraturan Menteri Perhubungan Indonesia yang mendefinisikan berbagai istilah pada Perkeretaapian, sebagai berikut:

- Perkeretaapian adalah satu kesatuan sistem yang terdiri atas prasarana, sarana, dan sumber daya manusia, serta norma, kriteria, persyaratan, dan prosedur untuk penyelenggaraan transportasi kereta api [1].
- Prasarana perkeretaapian adalah jalur kereta api, stasiun kereta api, dan fasilitas operasi kereta api agar kereta api dapat dioperasikan [2].
- Sarana Perkeretaapian adalah kendaraan yang dapat bergerak di jalan reI [1].
- Kereta Api adalah sarana perkeretaapian dengan tenaga gerak, baik berjalan sendiri maupun dirangkaikan dengan sarana perkeretaapian lainnya, yang akan ataupun sedang bergerak di jalan rel yang terkait dengan perjalanan kereta api [1].
- Lokomotif adalah sarana perkeretaapian yang memiliki penggerak sendiri [1].
- Kereta yang Ditarik Lokomotif adalah kereta yang tidak mempunyai penggerak sendiri, terdiri atas: Kereta penumpang, Kereta makan, Kereta pembangkit, Kereta bagasi [3].
- Kereta adalah sarana perkeretaapian yang ditarik lokomotif atau mempunyai penggerak sendiri yang digunakan untuk mengangkut orang [4].
- Gerbong adalah sarana perkeretaapian yang ditarik lokomotif yang digunakan untuk mengangkut barang, terdiri atas: gerbong datar, gerbong terbuka, gerbong tertutup, dan gerbong tangka [5].
- Perawatan Sarana Perkeretaapian adalah kegiatan yang dilakukan untuk mempertahankan keandalan Sarana Perkeretaapian agar tetap laik operasi [6].

I.1 Latar Belakang

Dalam rangka mendukung perkambangan menuju revolusi industri 4.0 di Indonesia, maka mulai diperlukan untuk menerapkan digitalisasi pada berbagai sektor. Diantaranya adalah moda sistem transportasi Perkeretapian, yang merupakan salah satu sistem pendukung industri untuk mengirimkan kargo ataupun personel jarak jauh dengan cepat. PT Kereta Api Indonesia (Persero) sebagai salah satu penyedia layanan transportasi Kereta Api di Indonesia, perlu melakukan transformasi digitalisasi bisnis proses nya. Diantaranya adalah pada bagian operasi, yaitu kegiatan perawatan sarana.

Perawatan aset juga disebut dengan istilah *Maintenance,* merupakan bagian penting dalam kegiatan operasi kereta api untuk memastikan bahwa tidak ada masalah yang terjadi saat sedang bekerja. Ada banyak komponen yang harus dilakukan perawatan secara rutin, dan akan memerlukan sumber daya yang besar apabila diperiksa manual. Diantaranya adalah lokomotif dan kereta pembangkit, yang dapat berjumlah ratusan untuk keseluruhan armada PT. KAI.

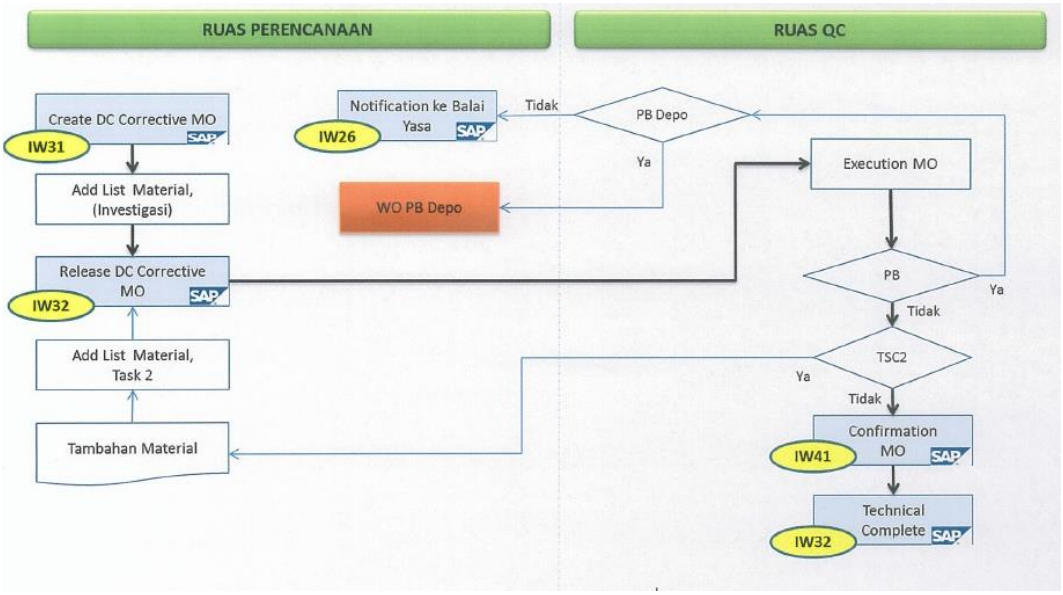
Untuk melakukan perawatan terhadap seluruh lokomotif dan kereta ini sulit tercapai. Pada tahun 2008, sekitar 7% rencana perawatan lokomotif tidak dapat direalisasikan. Selain itu juga terjadi beberapa insiden kerusakan pada tahun 2010. Pada saat itu ternyata strategi perawatan belum ditentukan oleh perusahaan, dan indikator untuk evaluasi hanya berupa *Availability* dan *Mean Kilometer Between Failure* saja. Skor yang didapat untuk kinerja perawatan masih banyak yang dapat ditingkatkan, terutama pada Kualitas dengan skor 37%, dan *Cost Effectiveness* dengan skor 53% [7]

Tabel I.1 Data ketersediaan lokomotif tahun 2019

JENIS LOK	SG Jan-Feb	SG Maret	SG April-Sep	SG Okt-Des	TSGO	SGO	TSO	SO DINAS KA	PERSENTASE	
									SO	PERAWATAN
BB 203	5	5	5	5	1	4	0	4	77%	23%
BB 204	1	0	0	0	0	0	0	0	0%	0%
BB 302	3	3	3	3	0	3	0	3	86%	14%
BB 303	20	20	18	18	4	15	1	14	78%	25%
CC 201	130	130	130	130	12	118	6	112	86%	14%
CC 202	47	47	47	47	5	42	2	40	86%	14%

CC 203	37	37	37	37	3	34	2	32	87%	13%
CC 204	37	37	37	37	3	34	2	32	88%	12%
CC 205	55	55	55	55	4	51	3	48	88%	12%
CC 206	150	150	150	150	5	145	3	143	95%	5%
Jumlah	**485**	**484**	**482**	**482**	**36**	**447**	**18**	**429**	89%	11%

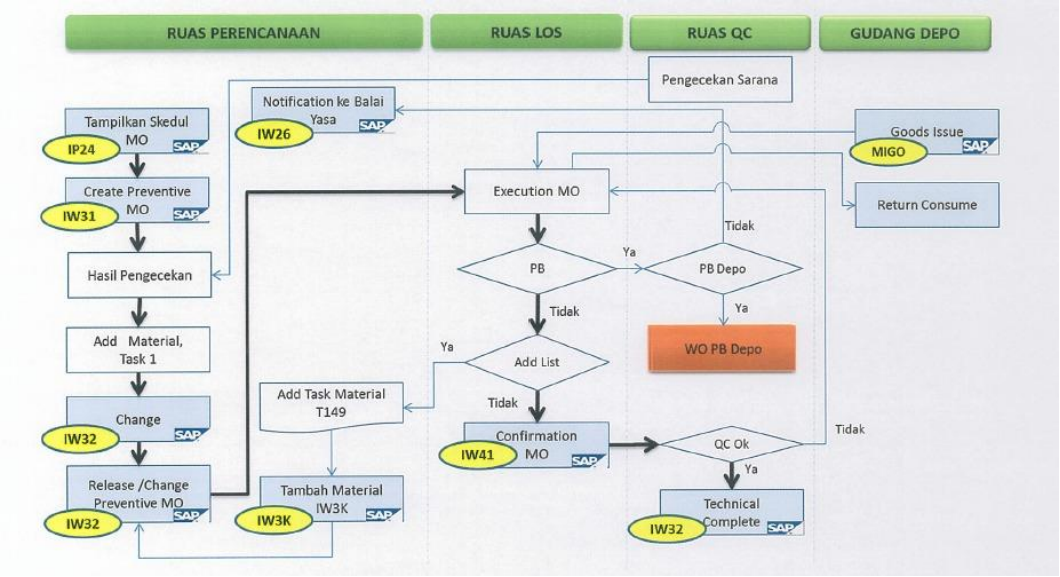
Pada tabel tersebut SG (siap guna) merupakan total sarana lokomotif yang tersedia, TSGO (Tidak siap guna operasi) mengacu pada sarana yang sedang dilakukan perbaikan di balai yasa, SGO (Siap guna operasi) merupakan hasil dari SG-TSGO, kemudian TSO (Tidak siap operasi) adalah Sarana yang di rawat di depo, dan SO (siap operasi) merupakan hasil SGO-TSO. Berdasarkan data tersebut, sekitar 11% dari lokomotif perlu dilakukan perawatan sehingga tidak dapat digunakan. Kebijakan batas kendali dari KAI membutuhkan setidaknya 85% dari armada untuk siap digunakan. Dengan demikian memang target tersebut sudah tercapai, tetapi ini merupakan rekap data selama satu tahun. Dalam operasinya pada rentang waktu mingguan atau bulanan angka tersebut dapat jatuh dibawah batas kendali tersebut, terutama pada jenis lokomotif tertentu yang sudah tua.



Gambar I.1 Bisnis Proses *Daily Check* pada Sarana KAI

Melihat dari bisnis proses perawatan pada KAI, terbagi menjadi dua yaitu Harian dan Bulanan. Aplikasi *enterprise resource management* seperti SAP sudah dipakai oleh PT KAI dalam merencanakan bisnis prosesnya, namun masih ada kegiatan yang datanya belum dalam bentuk digital sehingga tidak tercatat pada sistem.

Aktivitas ‘Execution MO’ dilakukan oleh mekanik secara manual mengacu pada kertas *checksheet* untuk menentukan kondisi keperluan perbaikan (PB). Jika dibutuhkan perbaikan kecil maka dapat dilakukan di depo, sementara jika perlu perbaikan yang cukup besar maka akan diberikan Notifikasi ke Balai Yasa untuk dikerjakan. Dari proses ini, apabila data komponen yang diperbaiki dan waktu perbaikan tersimpan datanya oleh sistem, maka dapat dilakukan prediksi *Remaining Useful Lifetime* (RUL) dengan ML jika jumlah datanya cukup banyak.



Gambar I.2 Bisnis Proses Perawatan Bulanan Sarana KAI

Dari bisnis proses perawatan bulanan pada KAI, beberapa bagian cukup serupa dengan sebelumnya. Hasil dari aktifitas ‘Pengecekan Sarana’ yang dilakukan harian akan menjadi masukan untuk perintah kerja perawatan. Aktivitas ‘Execution MO’ dilakukan oleh mekanik secara manual mengacu pada kertas *checksheet* untuk menentukan kondisi keperluan perbaikan (PB). Jika dibutuhkan perbaikan kecil maka dapat dilakukan di depo, sementara jika perlu perbaikan yang cukup besar maka akan diberikan Notifikasi ke Balai Yasa untuk dikerjakan. Jika tidak ada perbaikan lebih lanjut maka perintah kerja selesai, namun akan dilakukan *quality control* (QC) untuk memastikan tidak ada masalah pada sarana sebelum proses selesai. Dari proses ini, menggunakan informasi historis QC sebelumnya dan data berbagai sensor dapat dimanfaatkan untuk prediksi kondisi sarana menggunakan ML jika jumlah datanya cukup.

Disinilah diperlukannya otomasi untuk dapat mengidentifikasi kondisi lokomotif dan sarana lainnya dari kereta api untuk memastikan kelayakan operasinya. Jika bergantung pada inspeksi rutin periodik secara manual, belum tentu mesin memiliki masalah saat diperiksa, selain itu juga memakai sumber daya manusia dan waktu karena mengecek mesin yang kondisinya masih baik. Sementara itu dengan konsep perawatan prediktif, maka inspeksi hanya diperlukan saat mesin terdeteksi berdasarkan perkiraan akan memiliki masalah yang dapat mengganggu operasi, dan apabila terverifikasi maka kemudian dapat dilakukan perawatan. Dengan demikian akan mengurangi waktu saat mesin tidak dapat digunakan (meningkatkan *Availability*), dan dapat memaksimalkan keuntungan dengan mengoperasikan mesin serta mengurangi sumber daya yang digunakan (meningkatkan *Cost Effectiveness*).

Manfaat ini dibuktikan oleh French National Railways (SNCF) yang telah mulai menerapkan sistem diagnosis jarak jauh yang terpasang secara *on-board* pada sarana perkeretaapian. Sistem ini dapat digunakan untuk mendeteksi kesalahan, kode status, dan berbagai parameter sensor terkait. Data yang didapat kemudian diolah dan dianalisis untuk menjadi indikator yang dapat menentukan degradasi sistem, hasil berupa status kondisi sistem kemudian dapat digunakan untuk mempertimbangkan keperluan Perawatan. Strategi perawatan baru menggabungkan perawatan terjadwal (*Preventive*) dan *Condition Based Maintenance* (CBM), dengan tujuan meningkatkan keselamatan, ketersediaan, keandalan, dan mengurangi frekuensi perawatan terjadwal. Namun, beberapa tugas perawatan terjadwal, seperti pelumasan atau inspeksi visual, masih digunakan karena memerlukan campur tangan manusia. Hasil yang didapatkan dari pengamatan selama 5 tahun terakhir menunjukan: jumlah kerusakan berkurang setengahnya, sementara efisiensi alat *troubleshooting* telah diperkuat oleh alat analisis data. Peningkatan 30% pada pintu masuk/keluar di depot perawatan, yang menunjukkan keuntungan dua kereta per lokasi. Pemotongan 20% dalam jumlah kereta api yang dihentikan layanannya untuk perawatan, dan pengurangan 20% dalam biaya perawatan [8].

I.2 Masalah Penelitian

Berdasarkan latar belakang yang telah dikemukakan, dibuat pernyataan masalah yang akan dikerjakan dalam penelitian tesis ini yaitu: Mendesain program pembelajaran mesin untuk melakukan prediksi kebutuhan perawatan sarana berdasarkan rekaman data sensor yang dapat digunakan untuk menggantikan inspeksi rutin yang dilakukan manual sehingga dapat meningkatkan ketersediaan sarana.

Batasan dalam penelitian tesis ini adalah:

1. Dataset untuk melatih *Machine Learning* didapat dengan bekerjasama oleh pihak PT. KAI berdasarkan data sensor untuk kereta pembangkit yang beroperasi di Pulau Jawa tahun 2020 dan 2021.
2. Pengujian terbatas hanya pada komponen pelumas dan pendingin dari generator pada sarana kereta pembangkit.
3. Prediksi keperluan perawatan sarana ditunjukkan dalam bentuk keterangan hasil klasifikasi antara Outlier, Normal atau Maintenance.

I.3 Tujuan, Kontribusi, dan *Benefit* Penelitian

Tujuan Penelitian Tesis ini secara umum adalah:

1. Menganalisis proses perawatan yang diterapkan PT.KAI saat ini.
2. Membuat program pembelajaran mesin untuk sistem perawatan prediktif.
3. Mendesain berbagai alternatif opsi model pembelajaran mesin untuk perawatan prediktif sarana kereta api.
4. Mengevaluasi kinerja model pembelajaran mesin.

Kontribusi dari penelitian Tesis ini dibandingkan lainnya adalah objek penelitian berfokus pada kereta pembangkit, membandingkan berbagai algoritma dan opsi menggunakan data real dari sensor, dan menghasilkan prediksi kondisi yang ditunjukkan dalam bentuk keterangan hasil klasifikasi antara Outlier, Normal atau Maintenance serta perkiraan waktu komponen diperlukan perawatan berdasarkan input nilai sensor. Sementara itu dalam bidang keilmuan Rekayasa dan Manajemen Teknik Keamanan Informasi, kontribusi yang diberikan adalah manfaat penggunaan model ML untuk PdM yang dapat menjadi alternatif solusi manajemen resiko keamanan informasi untuk proses bisnis Perawatan sarana pada PT.KAI.

Dengan hasil penelitian ini, diharapkan proses perawatan bisa menjadi lebih baik dan mengurangi insiden kerusakan akibat informasi yang tidak akurat karena beberapa komponen dinilai secara kualitatif tanpa alat ukur yang jelas saat perawatan dilakukan menggunakan cara manual seperti saat ini. Selain itu melihat dari sudut pandang ketersediaan (*Availability*) informasi perawatan, dapat ditingkatkan karena tidak lagi terbatas pada kertas, tetapi sudah dalam bentuk digital dan bisa diakses melalui internet. Berdasarkan aspek integritas (*Integrity*) informasi, dengan menggunakan verifikasi *hashing* dari file data masukan untuk prediksi perawatan dapat dipastikan bahwa tidak ada perubahan terhadap isi file.

Berdasarkan tujuan yang telah dikemukakan sebelumnya, dan dengan memahami kondisi saat ini, dapat dianalisis keuntungan (*benefit*) penelitian ini bagi PT.KAI adalah sebagai berikut.

Tabel I.2 Analisis *benefit* penelitian bagi PT.KAI

No	Kondisi Saat ini	Hasil Penelitian
1	Sebagian istilah pada bisnis proses Maintenance proses tidak sesuai teori	Mendefinisikan sebagian istilah pada bisnis proses Maintenance
2	Data sensor hanya di monitor dan disimpan saja.	Data sensor diolah untuk mendapatkan prediksi kondisi
3	Pengecekan kondisi manual oleh manusia berdasarkan waktu tertentu	Pengecekan kondisi otomatis oleh komputer berdasarkan kondisi mesin yang dioperasikan manusia
4	Belum ada penerapan ML maupun PdM untuk perawatan sarana.	Menerapkan ML dan PdM untuk perawatan sarana Kereta Pembangkit

I.4 Teknik Penelitian

Bagian ini menjelaskan teknik yang dilakukan dalam menjalankan langkah-langkah penelitian dan proses pembuatan program ML.

1. *Studi Literatur*

   Teknik ini berfungsi untuk dapat merumuskan tujuan dan permasalahan terhadap penelitian tesis ini. Dengan melakukan kajian yang meliputi dasar teori berkaitan topik tesis. Seperti konsep perawatan secara umum,

pembelajaran mesin, perangkat lunak tersedia yang dapat digunakan, serta algoritma yang sesuai untuk diterapkan. Data dan informasi yang diperoleh bersumber dari buku, jurnal ilmiah, dan internet.

2. *Studi Empiris*

   Teknik ini bertujuan untuk meningkatkan pemahaman terhadap penelitian tesis ini. Dengan mengetahui bagaimana proses perawatan dilakukan saat ini oleh PT. KAI berdasarkan proses bisnis dan laporan perawatan yang tersedia. Kemudian mencari penelitian terkait perawatan prediktif pada kereta api, dan menganalisis hasil yang didapatkan serta metode yang digunakan.

3. *Eksperimen*

   Teknik ini digunakan untuk menentukan hasil terbaik dari model pembelajaran mesin yang dibuat terhadap kasus perawatan prediktif untuk sarana kereta api. Dengan membandingkan beberapa algoritma, aturan Hyperparameter serta fitur yang akan memberikan hasil terbaik jika diterapkan pada kasus penelitian ini. Menggunakan Bahasa pemrograman Python versi 3 yang diterapkan pada aplikasi pengembangan program Google Colab serta *library* yang tersedia untuk umum. Parameter utama yang akan menjadi kriteria penilaian adalah Akurasi, *Mean Absolute Error* (MAE) dan *Runtime*.

BAB II TINJAUAN PUSTAKA

Pada bagian ini akan dijelaskan detail mengenai berbagai hasil studi literatur yang dilakukan untuk penelitian tesis berkaitan seperti penjelasan konsep perawatan, dan kecerdasan buatan, serta membandingkan dengan riset yang serupa, atau peluang untuk mengembangkan dari riset yang sudah ada sebelumnya.

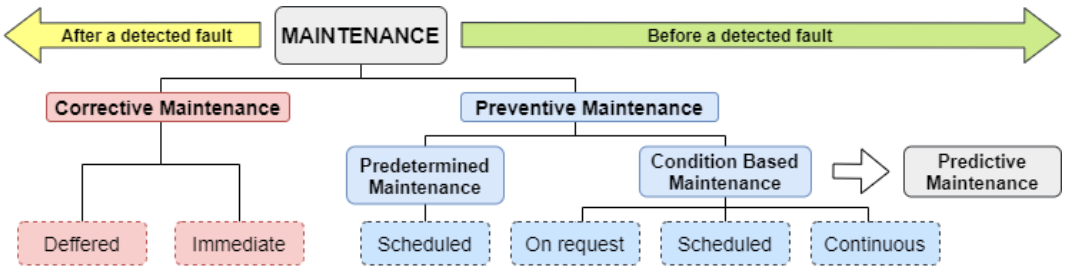
II.1 Konsep *Maintenance*

Yang dimaksud dengan istilah 'perawatan' adalah: kombinasi dari semua tindakan selama siklus hidup suatu barang yang dimaksudkan untuk mempertahankan atau memulihkannya ke keadaan di mana barang tersebut dapat melakukan fungsi yang diperlukan [9]. Untuk dapat menentukan suatu kondisi yang dapat dianggap baik akan bergantung pada berbagai macam faktor seperti kondisi operasi, tipe industri, dan tujuan bisnis. Meski demikian secara umum dapat ditentukan berdasarkan parameter berikut [10]:

- *Performance*, yaitu kemampuan suatu mesin untuk dapat menjalankan fungsinya.
- *Downtime*, merupakan periode saat mesin tidak bekerja yang harus diatur dalam level yang dapat ditoleransi.
- *Service Life*, adalah lama waktu yang diperlukan hingga mesin harus digantikan; penting untuk dipertimbangkan sehingga dapat memberikan keuntungan dari investasi terhadap mesin tersebut.
- *Efficiency*, yaitu rasio yang mengukur produktivitas kinerja mesin dengan sumber daya yang digunakan; harus selalu berada dalam level yang dapat diterima.
- *Safety*, merupakan ukuran yang dapat menjamin keamanan personil dari risiko bahaya yang mungkin terjadi.
- *Environmental impact*, yaitu ukuran yang menjamin dampak penggunaan mesin aman terhadap lingkungan dan perangkat sekitar.
- *Cost*, adalah parameter yang mengukur sumber daya yang digunakan pada mesin; diantaranya adalah biaya untuk perawatan harus berada dalam rentang yang dapat diterima.

Tujuan dari perawatan adalah untuk dapat memastikan bahwa mesin berada dalam kondisi yang baik dengan mempertimbangkan faktor tersebut. Dengan demikian maka dikembangkan berbagai strategi perawatan berbeda sebagai berikut.

II.1.1 Konsep *Predictive Maintenance*



Gambar II.1 Jenis strategi perawatan berdasarkan standar EN13306 [11]

Dalam melakukan perawatan, ada beberapa strategi atau sistem yang dapat diterapkan. Diantaranya adalah Pencegahan (*Preventive*), dan Pembenaran (*Corrective*). Strategi pencegahan digunakan untuk merawat suatu barang agar dapat bekerja dengan baik sebelum terjadi kerusakan, sementara strategi pembenaran digunakan setelah terjadi kerusakan. Dari kategori pencegahan terbagi lagi menjadi dua, perawatan rutin (*Predetermined Maintenance*) merupakan strategi yang menjadwalkan perlunya melakukan perawatan mesin secara periodic. Perawatan berbasis kondisi (*Condition based Maintenance*) adalah strategi baru yang dapat dilakukan sesuai keperluan ataupun terjadwal berdasarkan pengamatan kondisi mesin. Pemelihaan prediksi (*Predictive Maintenance*) adalah tahapan lanjut dari CBM yang dapat menggunakan bantuan sensor dan pembelajaran mesin. Jenis perawatan ini dibantu dengan menggunakan sensor untuk memantau berbagai parameter yang berpengaruh dalam mesin atau sistem, berdasarkan data yang didapat dan tren historis sebelumnya maka dapat dianalisis apakah diperlukan perawatan atau tidak. Data secara terus menerus dianalisis dan dimonitor untuk mengevaluasi kondisi sistem dan memberikan prediksi jika perlu dilakukan perawatan sebelum dapat terjadi kerusakan [9].

Melihat kembali pada gambar yang menjelaskan bisnis proses perawatan harian dan bulanan sarana pada KAI, dapat terlihat ada beberapa istilah yang tidak sesuai

artinya jika dibandingkan dengan penjelasan referensi [9]. Beberapa istilah yang menjadi permasalahan tersebut adalah:

- Daily Check, kegiatan pengecekan kondisi sarana yang dilakukan dengan periode Harian
- Perawatan Bulanan, kegiatan pengecekan kondisi sarana yang dilakukan dengan periode Bulanan
- Perbaikan (PB), kegiatan untuk membenarkan masalah sehingga penggunaan mesin menjadi normal kembali
- Corrective MO, perintah untuk melakukan kegiatan Daily Check
- Preventive MO, perintah untuk melakukan kegiatan Perawatan Bulanan

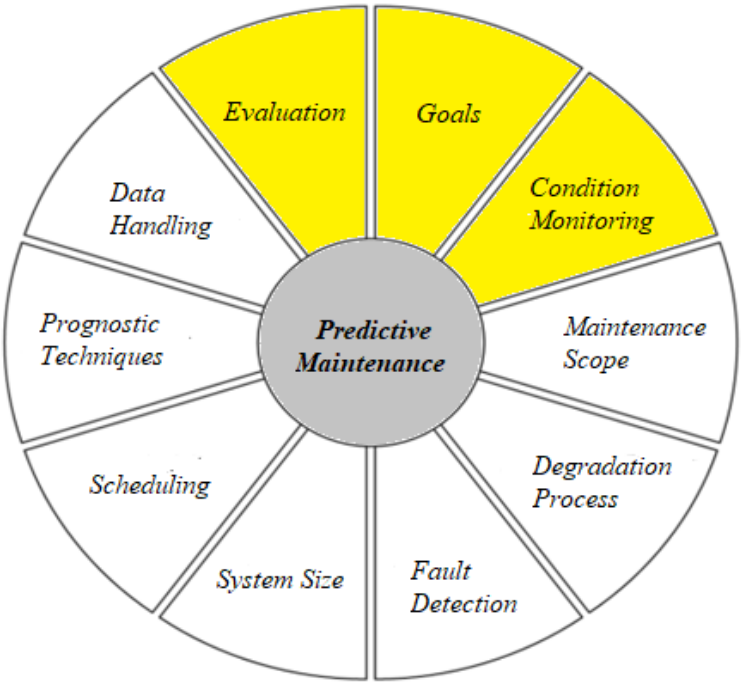
Ketidaksesuaian pengertian terkait istilah tersebut perlu diperbaiki dan diperjelas dengan membuat definisi formal atau standar yang dapat disahkan oleh pemerintah. Pada penelitian ini akan disesuaikan terhadap referensi [9], yang menghasilkan definisi dan perubahan istilah menjadi sebagai berikut:

- Perawatan Harian, kegiatan perawatan kondisi sarana yang dilakukan dengan periode Harian
- Perawatan Bulanan, kegiatan perawatan kondisi sarana yang dilakukan dengan periode Bulanan.
- Perawatan/Pemeliharaan, kombinasi dari semua kegiatan teknis, administratif dan tindakan manajerial selama siklus hidup item dimaksudkan untuk mempertahankan, atau mengembalikannya pada keadaan yang dapat melakukan fungsi yang diperlukan.
- Perbaikan (PB), tindakan fisik yang diambil untuk memulihkan fungsi yang diperlukan dari barang yang rusak.
- Inspeksi, pemeriksaan kesesuaian dengan mengukur, mengamati, atau menguji karakteristik yang relevan dari suatu barang.
- Pencegahan, kegiatan yang dilakukan untuk menilai atau mengurangi degradasi dan probabilitas kegagalan item

Dari survey [12] mengenai penelitian PdM, dapat diterapkan dengan berbasis ML tradisional ataupun menggunakan DL (*Deep Learning*). Kompleksitas model pembelajaran mesin, khususnya model DL terus meningkat setiap tahun. Salah satu

hal yang menyebabkan ini adalah meningkatnya jumlah lapisan yang digunakan di dalam *Neural Network*. Dengan meningkatnya kompleksitas serta jumlah data yang akan diolah, maka akan meningkatkan waktu eksekusi program. Tidak peduli apakah program melalui pelatihan menggunakan *cloud* ataupun secara lokal, maka akan memerlukan biaya yang besar. Jika menyewa penggunaan GPU terbaru pada *cloud* untuk melatih DL dapat mengeluarkan biaya beberapa dolar per jam, maupun menggunakan sumber daya komputasi lokal akan membutuhkan biaya tambahan untuk listrik dan pendingin udara [13].

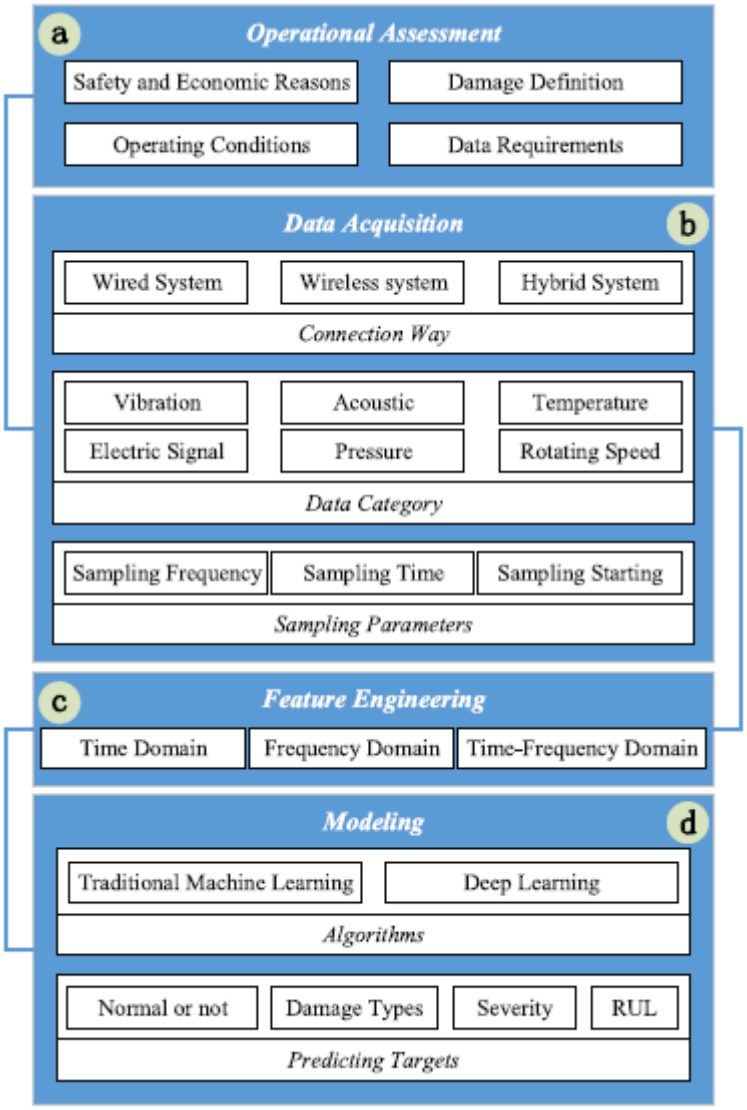
Saat ini PT.KAI baru mulai menerapkan sistem *data collection* pada beberapa kereta pembangkit, dengan data yang tersedia sekitar 1 tahun terakhir. Selain itu perawatan rutin masih dilakukan menggunakan referensi dari kertas *checksheet*, sehingga informasinya belum tersimpan dalam bentuk digital. Keterbatasan jumlah data ini menyebabkan penggunaan DL tidak terlalu dibutuhkan untuk saat ini. Selain itu infrastruktur dan arsitektur TI (Teknologi Informasi) yang belum mapan oleh PT.KAI akan meningkatkan kebutuhan biaya jika menerapkan DL. Karena proses transformasi digital ini dilakukan secara bertahap sesuai kemampuan organisasi, maka untuk tahap awal perlu menyediakan solusi dengan biaya murah yang dapat menjadi basis untuk perkembangan tahap selanjutnya. Berdasarkan pertimbangan tersebut, maka penelitian ini difokuskan pada pendekatan yang menggunakan ML tradisional. Dari survey [12] juga disampaikan beberapa jenis algoritma yang umum digunakan pada sistem perawatan prediktif, yaitu: *Artificial Neural Network* (ANN), *Decision Tree* (DT), *Support Vector Machine* (SVM), dan *k-Nearest Neighbor* (k-NN). Algoritma tersebut dapat dipertimbangkan penggunaannya, dengan membandingkan performanya untuk mengetahui algoritma yang cocok pada kasus penelitian ini.



Gambar II.2 Kategori dari kerangka untuk perawatan prediktif [14]

Hasil penelitian terhadap beberapa literatur mengenai PdM menghasilkan gambar yang menyatakan bahwa strategi perawatan prediktif memiliki 10 kategori besar kerangka (*framework)* yang dapat diterapkan dalam menyusun sistem. [14] Pada penelitian tesis ini, akan berkaitan dengan monitor kondisi (*Condition Monitoring*) kereta pembangkit dan evaluasi (*Evaluation*) kebutuhan perawatan berdasarkan prediksi Klasifikasi kondisi, serta tujuan (*Goals*) untuk meningkatkan ketersediaan (*Availability*) dari sarana yang siap beroperasi.

Dari studi yang dilakukan [15], disimpulkan bahwa penggunaan kecerdasan buatan dan teknik pemodelan yang dibuat cukup membantu untuk mencapai tujuan melakukan *Predictive Maintenance*. Dengan konsep ini, memungkinkan perawatan dilakukan dengan lebih efisien, karena pengawasan terhadap kondisi mesin atau sistem dilakukan secara kontinu tanpa menggangu kinerja. Berdasarkan penelitian lainnya, didapatkan hasil biaya perawatan yang lebih rendah apabila digunakan metode perawatan prediktif dengan bantuan sensor dibandingkan metode lainnya [16].



Gambar II.3 Proses implementasi PdM [17].

Dari hasil survey yang dilakukan [17], dibuat langkah proses implementasi *Predictive Maintenance* yang secara umum dilakukan oleh para peneliti dalam membuat sistem. Penelitian ini mengikuti langkah yang serupa dengan proses ini, terutama saat akan mendesain program pembelajaran mesin yang dapat diterapkan untuk perawatan prediktif sarana perkeretaapian.

II.1.2 Pemantauan Kondisi Komponen

Manfaat utama dari PdM adalah model ML yang dihasilkan untuk melakukan deteksi atau prediksi. Model ini menganalisis indikator yang diekstraksi dari sensor

untuk menentukan kondisi sistem saat ini (deteksi kesalahan dan diagnosis) atau memprediksi kondisi masa depan (prediksi RUL) [18].

Condition Monitoring menggunakan data dari mesin untuk menilai kondisinya saat ini dan untuk mendeteksi serta mendiagnosis kesalahan pada mesin. Data mesin adalah data seperti pengukuran suhu, tekanan, tegangan, kebisingan, atau getaran, yang dikumpulkan menggunakan sensor khusus. Algoritma *Condition Monitoring* memperoleh metrik dari data yang disebut indikator kondisi. Indikator kondisi adalah fitur apa pun dari data sistem yang perilakunya berubah dengan cara yang dapat diprediksi saat kinerja sistem menurun. Indikator kondisi dapat berupa nilai apa pun yang diturunkan dari kelompok data dengan status sistem yang sama, dan menetapkan status yang berbeda secara terpisah. Dengan demikian, algoritma pemantauan kondisi dapat melakukan deteksi atau diagnosis kesalahan dengan membandingkan data baru dengan indikator kondisi kesalahan yang sudah ada [18].

Condition Monitoring mencakup pembedaan antara keadaan rusak dan keadaan yang sehat (deteksi kerusakan) atau, ketika ada keadaan kerusakan, menentukan sumbernya (diagnosis kerusakan). Untuk mendesain model *Condition Monitoring*, dapat menggunakan indikator kondisi yang diekstrak dari data sistem untuk melatih model yang dapat menganalisis indikator yang diambil dari data uji untuk menentukan status sistem saat ini [19].

Berdasarkan referensi [19], beberapa contoh model untuk pemantauan kondisi meliputi :

- Nilai batas atau serangkaian batas pada nilai indikator kondisi yang menunjukkan kesalahan saat indikator melebihinya
- Distribusi probabilitas yang menggambarkan kemungkinan bahwa nilai tertentu dari indikator kondisi menunjukkan jenis masalah tertentu
- Pengklasifikasi yang membandingkan nilai indikator kondisi saat ini dengan nilai yang terkait dengan status kerusakan, dan mengembalikan kemungkinan adanya status kerusakan

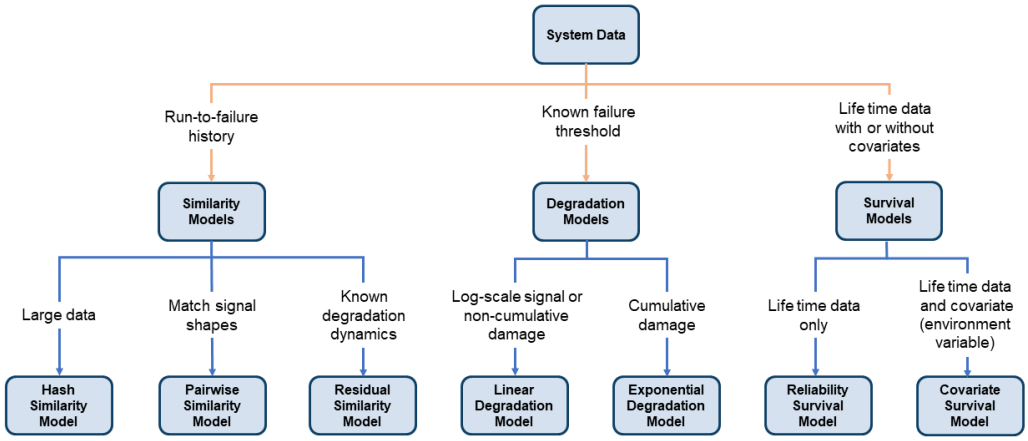
Dari data yang didapatkan dengan bekerjasama oleh pihak PT.KAI, didapatkan data banyak sensor pada generator kereta pembangkit. Namun data ini tidak memiliki label untuk membedakan saat kondisi Normal ataupun Rusak/Maintenance.

Sehingga diperlukan upaya untuk memisahkan kedua kondisi tersebut, sebelum dapat berlanjut untuk melakukan pemantauan kondisi.

II.1.3 Prediksi *Remaining Useful Lifetime*

Model estimasi RUL adalah cara melatih model menggunakan data historis dan menggunakannya untuk melakukan prediksi *Remaining Useful Lifetime*. Istilah *Lifetime* di sini mengacu pada masa pakai mesin yang didefinisikan dalam unit apapun yang digunakan untuk mengukur masa pakai sistem. Sisa waktu dapat berarti perubahan nilai dengan penggunaan, jarak yang ditempuh, jumlah siklus, atau kuantitas lain yang menggambarkan masa hidup [20]. Model estimasi RUL ini berguna jika memiliki data dan informasi historis seperti:

- Riwayat mesin yang mengalami kegagalan serupa dengan yang ingin dilakukan diagnosis
- Nilai batasan yang diketahui dari beberapa indikator kondisi yang menunjukkan adanya kegagalan
- Data tentang berapa banyak waktu atau berapa banyak penggunaan yang diperlukan untuk mesin serupa untuk mencapai kegagalan.



Gambar II.4 Pohon keputusan untuk memilih model prediksi RUL [20].

Similarity Models mendasarkan prediksi RUL dari mesin pada perilaku yang diketahui dalam data historis. Model tersebut membandingkan tren dalam data uji atau nilai indikator kondisi dengan informasi yang sama diekstraksi dari sistem serupa lainnya [20]. *Similarity Models* berguna ketika:

- Memiliki data *run-to-failure* dari sistem (komponen) serupa. Data *run-to-failure* adalah data yang dimulai selama operasi yang sehat dan berakhir ketika mesin dalam keadaan mendekati kegagalan atau saat terjadi perawatan.
- Data *run-to-failure* menunjukkan perilaku degradasi yang serupa. Artinya, data berubah dalam beberapa karakteristik yang sama saat kinerja sistem menurun.

Degradation Models mengekstrapolasi perilaku masa lalu untuk memprediksi kondisi masa depan. Jenis perhitungan RUL ini cocok dengan model linier atau eksponensial untuk profil degradasi indikator kondisi. Kemudian menggunakan profil degradasi komponen uji untuk menghitung secara statistik waktu yang tersisa sampai indikator mencapai beberapa ambang batas yang ditentukan. Model-model ini paling berguna ketika ada nilai indikator kondisi yang diketahui menunjukkan kegagalan [20].

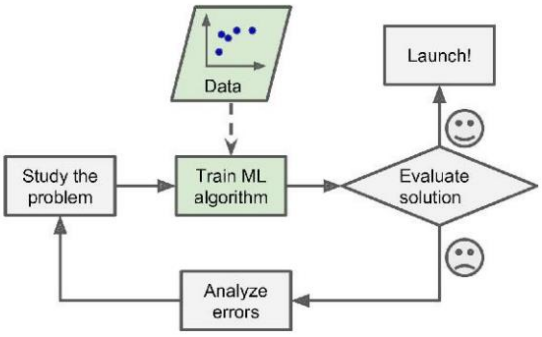
Survival Models adalah metode statistik yang digunakan untuk memodelkan data *time-to-event* [20]. Ini berguna ketika tidak memiliki riwayat *run-to-failure* yang lengkap, tetapi memiliki:

- Hanya data tentang masa pakai komponen serupa. Misalnya, mungkin diketahui berapa mil setiap mesin berjalan sebelum membutuhkan perawatan, atau berapa jam operasi setiap mesin berjalan sebelum gagal.
- Baik rentang hidup maupun beberapa data variabel lain (kovariat) yang berkorelasi dengan RUL. Kovariat, juga disebut variabel lingkungan atau variabel penjelas, terdiri dari informasi seperti penyedia komponen, lokasi di mana komponen digunakan, atau batch manufaktur.

Dari data sensor pada generator kereta pembangkit yang didapatkan dengan bekerjasama oleh pihak PT.KAI, tidak diketahui data *run-to-failure* karena belum ada label yang menandakan bahwa komponen tersebut gagal, tidak diketahui juga waktu spesifik setiap kereta dilakukan perawatan sehingga tidak bisa menggunakan *Similarity Models*. Sementara jika akan menggunakan *Degradation Models* belum diketahui nilai batasan kegagalan berdasarkan data sensor yang ada. *Survival Models* dapat digunakan setelah memiliki data sensor yang diberikan label Maintenance untuk mengetahui rentang hidup komponen.

II.2 *Machine Learning*

Revolusi Industri 4.0 dikarakterisasi dengan lebih banyaknya internet seluler yang tersebar di mana-mana, juga tersedianya sensor yang semakin kecil dan akurat menjadi lebih murah dan terjangkau, serta munculnya kecerdasan buatan dan *machine learning* [21]. Dalam bahasa sehari-hari, istilah "kecerdasan buatan" sering digunakan untuk mendeskripsikan mesin (atau komputer) yang meniru fungsi "kognitif" yang diasosiasikan dengan pikiran manusia, seperti kemampuan untuk "belajar" dan "pemecahan masalah" [22]. Dengan demikian, pembelajaran mesin merupakan suatu bagian dari kecerdasan buatan. Pembelajaran mesin adalah pemrograman komputer untuk mengoptimalkan kinerja suatu kriteria menggunakan data contoh atau pengalaman sebelumnya. Dengan memiliki model yang ditentukan berisi beberapa parameter. Maka dapat dilakukan pembelajaran mesin yaitu melaksanakan suatu program komputer untuk mengoptimalkan parameter dari model dengan menggunakan data pelatihan atau pengalaman masa lalu. Namun pembelajaran mesin bukan hanya itu saja, tetapi juga merupakan bagian kecerdasan buatan. Untuk menjadi cerdas, saat sistem berada di lingkungan berbeda, maka harus memiliki kemampuan untuk belajar sehingga dapat beradaptasi. Jika sistem dapat belajar dan beradaptasi dengan perubahan tersebut, maka perancang sistem tidak perlu meramalkan dan menulis program untuk semua kemungkinan situasi [23].



Gambar II.5 Diagram alir pendekatan pembelajaran mesin [24].

Berdasarkan diagram alir tersebut, pembelajaran mesin adalah suatu siklus yang terus berulang hingga dapat menghasilkan nilai yang memuaskan untuk dapat diluncurkan. Data yang akan digunakan perlu diatur formatnya menyesuaikan

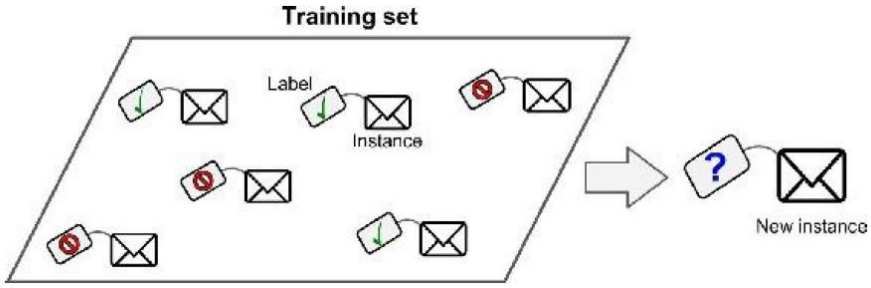
algoritma yang digunakan. Hal ini membuat algoritma ML menjadi hal utama yang perlu ditentukan dalam membuat program. Algoritma tersebut dapat diklasifikasi berdasarkan beberapa kriteria yaitu keterlibatan pengawasan manusia, kemampuan sistem dalam pembelajaran aliran data bertahap, dan proses prediksi hasil. Penggunaan kriteria tersebut tidak eksklusif dan dapat digunakan secara bersamaan [24]. Pada bagian berikut ini akan dijelaskan mengenai beberapa jenis algoritma yang diterapkan dalam penelitian Tesis ini.

II.2.1 Keterlibatan Pengawasan Manusia

Berdasarkan kriteria keterlibatan pengawasan manusia dalam proses pembelajaran mesin, terbagi menjadi kategori sebagai berikut [24] :

1. Diawasi (*Supervised*)

   Pada pembelajaran dengan metode ini, data pelatihan akan diberikan informasi mengenai solusi yang diinginkan, disebut dengan istilah label. Label inilah yang menjadi keterlibatan manusia dalam sistem pembelajaran mesin, karena ditentukan oleh manusia sebelum diolah oleh algoritma.

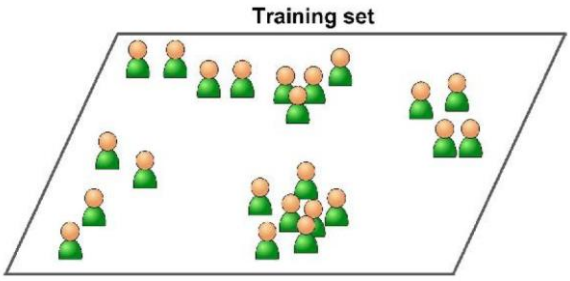


Gambar II.6 Set data pelatihan dengan label

   Sistem yang menerapkan metode ini akan melakukan pembelajaran dengan menganalisis set data pelatihan dan mencari pola yang dapat menentukan hasil yang sesuai. Jika diberikan suatu data uji baru tanpa label, maka sistem akan membandingkan nya dengan pola dari dataset untuk menentukan hasil.

2. Tanpa Pengawasan (*Unsupervised*)

   Pada pembelajaran dengan metode ini, data pelatihan tidak diberikan informasi label sehingga sistem belajar tanpa keterlibatan manusia dalam mengolah set data pelatihan.



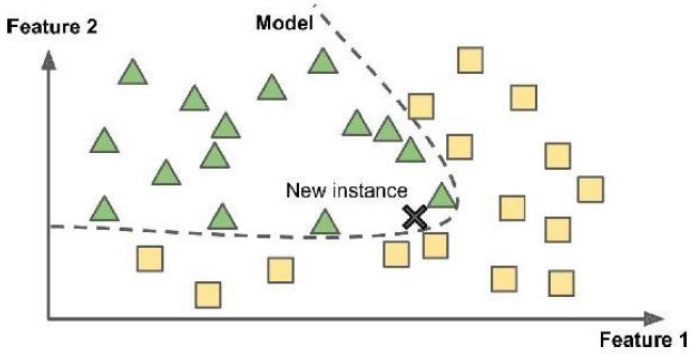
Gambar II.7 Set data pelatihan tanpa label [24]

Sistem yang menerapkan metode ini maka algoritma yang digunakan langsung menganalisis hubungan antar elemen yang ada dari data pelatihan dan menentukan sendiri hasilnya. Jika diberikan suatu data uji baru tanpa label, maka sistem akan membandingkan karakteristik nya lalu menentukan hasil berdasarkan kedekatan nya dengan kategori yang didapat dari data pelatihan.

II.2.2 Proses Prediksi Hasil

Berdasarkan metode yang digunakan dalam proses prediksi hasil, sistem berbasis pembelajaran mesin dibagi menjadi dua kategori sebagai berikut [24] :

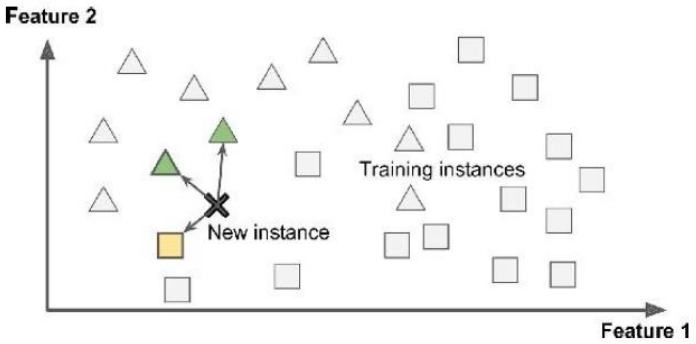
1. *Model-Based*



Gambar II.8 Pengelompokan hasil dengan pembelajaran *Model-based*

Metode ini bekerja dengan melakukan penarikan kesimpulan berdasarkan model hasil pembelajaran dari set data pelatihan. Saat melakukan proses pembelajaran, sistem akan mengolah set data pelatihan terlebih dulu untuk menemukan hubungan antar elemen dan menyimpulkan model yang paling cocok untuk dapat menjadi representasi set data. Ketika sistem diterapkan maka prediksi hasil dari data terbaru akan sesuai berdasarkan model yang telah ditentukan.

2. *Instance-Based*



Gambar II.9 Pengelompokan hasil dengan pembelajaran *Instance-based*

Metode ini bekerja dengan melakukan penarikan kesimpulan berdasarkan karakteristik yang berdekatan antar data baru dengan data yang sudah ada. Proses prediksi hasil akan mengambil kesimpulan bahwa data baru akan dikategorikan berdasarkan data lama yang paling dekat karakteristiknya.

II.3 Pembahasan Penelitian Terkait

Salah satu studi yang paling mendekati dengan topik ini dilakukan oleh Antony Barnes pada tahun 2014 [25]. Membahas mengenai meningkatkan performa lokomotif dengan melakukan *Condition Based Maintenance* (CBM). Namun dalam studi tersebut belum mencapai *predictive maintenance* maupun menggunakan *machine learning*. Metode yang dijelaskan sudah menggunakan berbagai perangkat sensor mengambil data kondisi lokomotif, namun masih dianalisis secara manual. Penelitian tersebut ditujukan untuk dapat memberikan pendeteksian dan peringatan dini atas kesalahan yang berkaitan dengan komponen yang memiliki kategori masalah berikut:

- Kesalahan Bantalan Elemen Bergulir (*Rolling Element Bearing faults*)
- Kerusakan roda gigi *(Gear mesh faults)*
- Masalah daya dan kelistrikan (*Load and electrical problems)*
- Masalah pelumasan (*Lubrication issues)*
- Masalah pembakaran (*Combustion Problems)*

Dari kesimpulan penelitian ini menyatakan bahwa metode dan strategi program perawatan CBM ini telah beroperasi selama 5 tahun dan dianggap sukses oleh pihak operator kereta api. Ini membuktikan bahwa metode ini layak untuk dapat

dikembangkan lebih lanjut menjadi sistem perawatan prediktif yang akan dilakukan pada tesis ini.

Studi lainnya yang dilakukan di Singapura pada tahun 2018. [26] Berusaha menggunakan ANN sebagai cara untuk melakukan perawatan prediktif pada sistem kereta, namun belum sampai tahap diujinya model yang ditunjukkan. Beberapa parameter yang dipilih untuk menjadi basis prediksi adalah *Oil Level*, *Screw compressor input current and input voltage*, *Lubricant temperature*, dan *Output compressed air pressure*. Penelitian baru memberikan proposal untuk memasang sensor yang mengumpulkan data tersebut, dan kesimpulan hanya mengatakan untuk melakukan finalisasi pemasangan sensor. Dari uraian tersebut, keberhasilan dari implementasi PdM masih belum diberikan.

Hasil penelitian terbaru pada tahun 2020 oleh Wang, et al. [27] berusaha menggabungkan sistem perawatan berdasarkan *data-driven* dengan *model-based* memanfaatkan LSTM-RNN untuk perangkat daya pada kereta cepat. Penelitian dilakukan hingga pengujian model yang membuktikan bahwa pendekatan yang diajukan memungkinkan. Meski demikian, metode penelitian yang disampaikan masih menggunakan data yang berdasarkan simulasi Monte-Carlo saja, walaupun menggunakan parameter berdasarkan peralatan *Traction Power Supply System* (TPSS) yang ada pada kereta cepat.

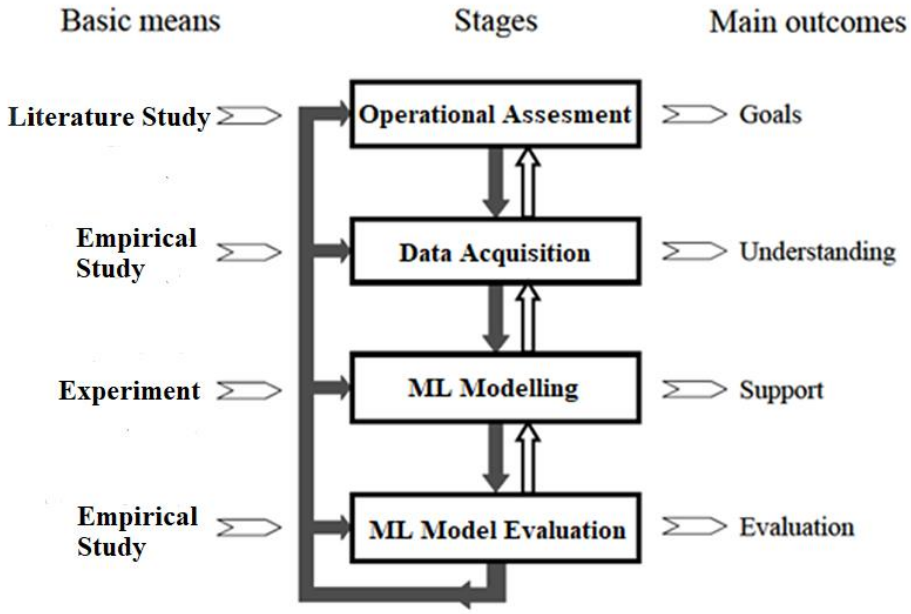
Penelitian lain dilakukan di Indonesia, pada tahun 2020 oleh Nafisah. [28] Mencoba memprediksi kejadian patah rel menggunakan beberapa algoritma untuk membuat model pembelajaran mesin, algoritma yang dicoba adalah *Logistic Regression* (LR), SVM, Decision Tree, *Random Forest* (RF), dan KNN. Hasil terbaik dicapai saat menggunakan RF dengan akurasi 67%. Walaupun memiliki perbedaan objek prediksi dengan tesis ini, namun metodologi nya dapat dipertimbangkan.

Penelitian [29] bertujuan untuk mengimplementasikan PdM untuk kendaraan *rolling stock* menggunakan pendekatan berbasis model. Pendekatan dilakukan dengan membuat model matematika untuk ML yang diperoleh dari skematik kendaraan pengujian PV7 EVO. Meskipun masih hasil penelitian awal, hasilnya

dapat mengidentifikasi kondisi di mana perawatan harus dilakukan dengan memperhatikan perpindahan as roda dengan jarak yang ditempuh.

BAB III LANGKAH PENELITIAN

Pada bagian ini akan dijelaskan detail mengenai tahapan proses penelitian yang direncanakan untuk tesis yang dilakukan. Penelitian berbasis metodologi penelitian desain (*Design Research Methodology*) atau disebut DRM [30] dengan penyesuaian berdasarkan tahapan implementasi PdM [12], [17].



Gambar III.1 Prosedur sistematis yang akan digunakan

Inti pada metodologi penelitian desain adalah melakukan proses iterasi yang terdiri dari berbagai tahap [30]. Tahap *Operational Assessment*, keluaran nya adalah untuk membuat beberapa tujuan yang ingin dicapai pada tahap iterasi kali ini. Kemudian dilanjutkan ke tahap *Data Acquisition*, dengan hasil mendapatkan pemahaman tentang masalah yang diteliti dan mencari faktor yang mempengaruhi hasil penelitian. Tahap ketiga adalah *ML Modelling*, bertujuan untuk mendapatkan data atau informasi yang mampu mendukung hasil pada proses sebelumnya. Terakhir dilakukan *ML Model Evaluation*, bertujuan untuk mengevaluasi dan menginvestigasi dampak hasil yang didapat sebelumnya terhadap tujuan kita. Proses ini akan terus berulang, berlanjut sampai pemahaman yang memadai atau menemukan solusi untuk masalah penelitian tercapai.

III.1 *Operational Assessment*

Menyerupai tahap *Research Clarification* (RC) pada DRM, akan dilakukan pencarian untuk menemukan beberapa bukti atau setidaknya indikasi yang mendukung asumsi penelitian agar dapat dirumuskan secara realistis dan tujuan penelitian yang bermanfaat. Berdasarkan temuan tersebut, gambaran awal tentang situasi yang ada dikembangkan, serta deskripsi situasi yang diinginkan, kemudian membuat asumsi yang mendasari setiap deskripsi situasi ideal. Kemudian merumuskan beberapa kriteria atau parameter yang bisa dijadikan ukuran terhadap hasil penelitian [30]. Pada tahap ini, teknik yang digunakan adalah studi literatur. Berbagai macam informasi yang dibutuhkan dalam penelitian akan dicari dari sumber yang terpercaya. Diantaranya pencarian berbagai macam spesifikasi dari perangkat lunak yang cocok untuk dapat digunakan pada penelitian ini, mempelajari berbagai jenis algoritma pembelajaran mesin serta aplikasinya pada bidang PdM, dan yang paling utama adalah melakukan kajian terhadap kegiatan perawatan pada PT. KAI melalui dokumen yang disediakan.

Keluaran dari tahap ini secara umum adalah berbagai hal yang telah dijelaskan pada Bab I Pendahuluan, terutama pernyataan tujuan penelitian dan latar belakang perlunya dilakukan penelitian. Secara khusus tahap ini akan menentukan tujuan yang perlu dicapai saat proses iterasi tertentu, adapun tujuan yang telah dapat dirumuskan saat ini adalah sebagai berikut:

Iterasi	Tujuan
0	• Mendefinisikan tujuan penelitian • Menentukan kontribusi & *benefit* penelitian • Melakukan kajian perawatan PT. KAI • Mempelajari data perawatan yang tersedia pada KAI • Menentukan target Akurasi model • Menentukan target batasan *Runtime* model • Membuat program pemrosesan dataset
1	• Membuat program pemodelan Klasifikasi

- Mengoptimalkan kinerja model
- Melatih model Klasifikasi dengan data training

2
- Menguji akurasi model Klasifikasi dengan data uji
- Menguji runtime model Klasifikasi dengan membangkitkan data acak
- Menentukan Model Klasifikasi terbaik

III.2 *Data Acquisition*

Mengikuti tahap Deskriptif Studi I (DS-I) pada DRM, setelah mendapatkan gambaran yang jelas mengenai tujuan dan fokus, perlu meninjau literatur untuk faktor lain yang lebih mempengaruhi untuk menguraikan deskripsi awal dari situasi yang ada. Tujuannya adalah untuk membuat deskripsi cukup rinci untuk menentukan faktor mana yang harus ditangani untuk meningkatkan agar desain menjadi efektif dan efisien. Namun, jika tidak menemukan bukti yang cukup dalam literatur untuk menentukan dengan jelas faktor-faktor penting ini, dapat mengamati secara langsung pada lokasi untuk mendapatkan pemahaman yang lebih baik tentang situasi yang ada, sebelum melanjutkan ke tahap berikutnya [30]. Cara yang digunakan adalah studi empiris dengan menyelidiki topik penelitian serupa, yang telah dijelaskan pada bagian sebelumnya, dan juga mengumpulkan sumber data apa pun yang memiliki kemungkinan menjadi sumber untuk melatih ML, seperti: hasil inspeksi rutin dari sarana kereta pembangkit dan lokomotif, rekap pemantauan kondisi di lokomotif, serta perangkat koleksi data yang dipasang pada generator kereta pembangkit.

III.2.1 Sumber Dataset

Untuk penelitian ini, sumber data yang digunakan adalah dari perangkat GRAMS untuk *data collection* yang terpasang pada beberapa kereta pembangkit. Kereta pembangkit merupakan bagian dari rangkaian kereta yang biasa digabungkan dengan kereta penumpang, sebagai penyedia listrik yang didistribusikan ke berbagai lokasi di kereta seperti TV, lampu, serta stop kontak untuk fasilitas penumpang. Umumnya kereta pembangkit dipasang pada kereta antar-kota, dan bukan merupakan bagian untuk penggerak rangkaian kereta.

Tabel III.1 Daftar kereta pembangkit yang digunakan datanya

No	Kereta Pembangkit		DIPO
	Jenis	Nomor	
1	MP3 (Makan dan Pembangkit Ekonomi AC)	01605	PWT (Purwokerto)
2	MP3 (Makan dan Pembangkit Ekonomi AC)	01703	JAKK (Jakarta)
3	P (Pembangkit)	01601	CN (Cirebon)
4	P (Pembangkit)	01801	BD (Bandung)
5	P (Pembangkit)	01808	SDT (Sidotopo)
6	P (Pembangkit)	01810	SLO (Solo)
7	P (Pembangkit)	01811	JAKK (Jakarta)
8	P (Pembangkit)	01812	JAKK (Jakarta)
9	P (Pembangkit)	01818	SLO (Solo)
10	P (Pembangkit)	01820	SLO (Solo)
11	P (Pembangkit)	01821	SLO (Solo)
12	P (Pembangkit)	01822	BD (Bandung)
13	P (Pembangkit)	01823	BW (Banyuwangi)
14	P (Pembangkit)	01825	BW (Banyuwangi)
15	P (Pembangkit)	01903	JAKK (Jakarta)
16	P (Pembangkit)	01908	SBI (Surabaya)
17	P (Pembangkit)	01914	SBI (Surabaya)

Sumber data diperoleh dari beberapa kereta pembangkit yang dipasangkan perangkat untuk mengambil informasi sensor secara periodik dan disimpan pada *cloud platform*. Ada total 45 unit kereta pembangkit yang telah terpasang perangkat ini, dan diberikan data 17 unit kereta yang tertulis pada Tabel III.1 untuk digunakan pada penelitian. Sebagian besar kereta yang sudah terpasang perangkat, dirawat oleh Dipo yang berada di Pulau Jawa sehingga untuk penelitian ini dibatasi kepada kereta tersebut.

№	Grouping	(Deepsea Control Mode)	(L1)	(L2)	(L3)	(Frequency)	(kVA Total)	(kVA_L1)	(kVA_L2)	(kVA_L3)	(KVAr)	(Oil Pressure)	(Coolant Temperature)	(Charger Alternator)	(Power Factor Average)	(Power Factor_L1)	(Power Factor_L2)	(Power Factor_L3)	(L1_N)	(L2_N)	(L3_N)	(Source Ext Voltage)	(ECU Temperature)	(RPM)
1.1	2020-04-01 04:02:06	2.00	26.00	26.00	26.00	50.00	17.38	5.88	5.74	5.70	4294964.46	636.00	81.00	27.90	0.89	0.86	0.89	0.92	222.70	222.80	222.80	27.38	36.00	1500.00
1.2	2020-04-01 05:00:15	2.00	38.00	34.00	31.00	50.10	25.40	8.42	7.76	6.82	2.40	632.00	82.00	27.90	0.93	0.92	0.94	0.92	222.90	222.50	222.80	27.22	38.00	1500.00
1.3	2020-04-01 06:00:15	2.00	87.00	84.00	82.00	50.00	58.10	19.42	18.88	18.40	24.16	624.00	82.00	27.90	0.89	0.89	0.90	0.87	222.60	222.60	222.30	27.36	40.00	1499.00
1.4	2020-04-01 07:00:15	2.00	103.00	101.00	99.00	50.00	68.77	22.94	22.52	22.08	29.04	620.00	82.00	27.90	0.89	0.89	0.90	0.88	222.50	222.70	222.00	27.33	43.00	1499.00
1.5	2020-04-01 08:00:16	2.00	93.00	89.00	88.00	50.00	62.03	20.62	19.74	19.56	24.32	620.00	82.00	27.90	0.90	0.90	0.91	0.89	221.70	222.20	221.80	27.36	43.00	1500.00

Gambar III.2 Data sensor untuk kereta pembangkit

Dari data yang didapatkan tersebut, sebagai contoh pada kereta jenis P-01801 memiliki berbagai informasi yang dapat digunakan. Diantaranya adalah pemantauan sensor dengan entri data pertama dimulai pada 2020-04-01 pukul 04:02:06 hingga berakhir pada 2020-10-31 pukul 23:00:14 atau sekitar 213 hari dengan total entri sebanyak 1954 baris. Data ini akan bertambah banyak seiring dengan meningkatnya waktu operasi perangkat pada sarana yang terpasang. Berikut adalah penjelasan Parameter yang dikumpulkan oleh perangkat ini yaitu:

1) ***Grouping*** **:** Tanggal dan waktu saat data dikumpulkan (datetime string)
2) ***Deepsea Control Mode*** **:** Mode operasi perangkat pengumpul data (integer)
3) **L1, L2, dan L3 :** Nilai arus pada generator di titik pengukuran (float)
4) ***Frequency*** **:** Nilai frekuensi sumber AC (float)
5) **kVA Total :** Jumlah Daya dari titik pengukuran di Generator (float)
6) **kVA_L1, kVA_L2,** dan **kVA_L3 :** Besar Daya di titik pengukuran Generator (float)
7) **kVAr :** Besar Daya reaktif di Generator (float)
8) ***Oil Pressure*** **:** Nilai tekanan oli pelumas dalam kPa (float)
9) ***Coolant Temperature*** **:** Nilai Pendingin mesin dalam Celsius (float)
10) ***Charger Alternator*** **:** Output Daya charger baterai dalam kVA (float)
11) ***Power Factor Average*** **:** Nilai rata-rata rasio daya kerja dari titik pengukuran dalam kW (float)

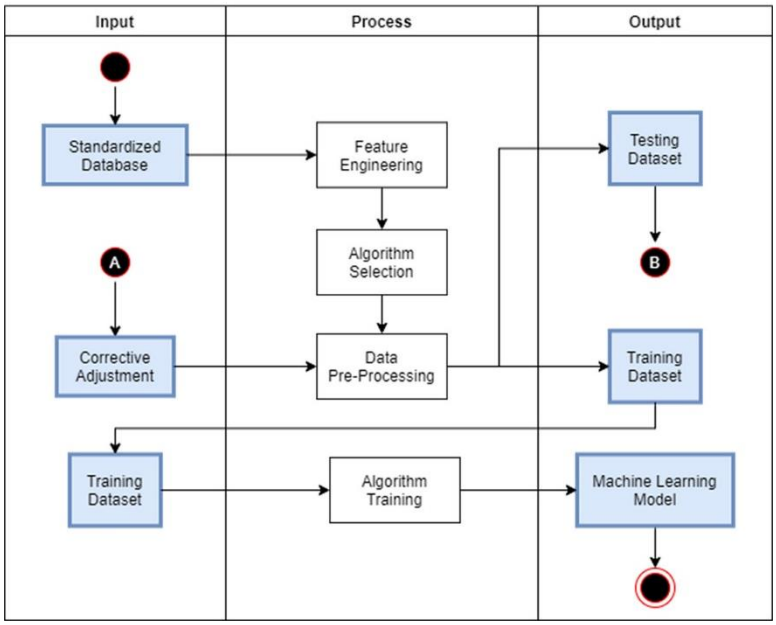
12) ***Power Factor_L1, Power Factor_L2,*** dan ***Power Factor_L3*** **:** Nilai rasio daya kerja di titik pengukuran dalam kW (float)
13) **L1_N, L2_N,** dan **L3_N :** Nilai tegangan di titik pengukuran dalam V (float)
14) ***Source Ext Voltage*** **:** Nilai tegangan sumber eksternal dari PLN (float)
15) ***ECU Temperature*** **:** Nilai temperatur unit kontrol elektronik dalam Celsius (float)
16) **RPM :** Nilai kecepatan rotasi mesin per menit (integer)

Selain itu juga ada pencatatan kondisi awal dan kondisi akhir saat kereta beroperasi, yang hanya mencatat waktu, lokasi, kecepatan, *ECU temperature* dan *coolant temperature*. Untuk data ini dilakukan monitor sejak 2020-03-07 pukul 16:40:11 hingga 2021-03-08 pukul 09:00:26 atau sekitar 366 hari dengan total sebanyak 643 baris entri data. Untuk penelitian ini, lebih diutamakan pada jenis data pemantauan sensor yang diberikan pada gambar karena informasinya yang lebih lengkap.

Keunggulan data ini jika dibandingkan dengan data lainnya, yaitu memiliki sensor yang lebih lengkap dan tersedia dalam bentuk digital, sehingga dengan semakin bertambahnya arsip data seiring waktu menjadi pertimbangan utama data ini dipilih untuk menjadi basis yang akan digunakan pada perancangan PdM. Permasalahan dari data ini adalah penggunaan perangkat *data collection* masih dalam tahap percobaan, sehingga data sensor terkadang tidak terbaca atau memiliki nilai yang jauh berbeda dari seharusnya. Selain itu tidak ada label yang diketahui untuk menjadi acuan kondisi dari kereta pembangkit, sehingga data perlu dianalisis lebih lanjut dan membuat asumsi kondisi sebagai label.

III.3 *ML Modelling*

Setara dengan tahap Preskriptif Studi (PS) pada DRM, bertujuan untuk mengoreksi dan menguraikan awal deskripsi situasi yang diinginkan dengan menggunakan peningkatan pemahaman tentang situasi yang ada saat ini. Dibuat berbagai kemungkinan skenario dengan memvariasikan faktor yang ditargetkan untuk mencapai hasil yang dapat mendukung situasi yang diinginkan [30]. Tahap ini mungkin diulangi sampai dapat mencapai hasil yang diinginkan, diharapkan hasilnya mampu lebih baik jika dibandingkan dari penelitian sejenis lainnya.



Gambar III.3 Alur kerja proses desain model ML

1) *Standardized Database*: Sebelum proses dimulai, data mentah yang dimiliki dengan format excel (.xlsx) berisi beberapa *sheet* dengan konten yang berbeda, sehingga perlu mengambil informasi yang diperlukan saja. Data yang dipilih ini digabungkan menjadi satu file sebagai *dataset* utama dan diubah menjadi format *commas separated values* (.csv). Dilakukan pemeriksaan nilai *hashing* memanfaatkan hashlib untuk memastikan integritas file sebelum di upload ke github untuk dapat diakses lebih mudah. Kemudian dalam program akan dibandingkan nilai *hashing* dari file yang di upload untuk memastikan tidak ada perubahan yang terjadi, dan apabila dilakukan revisi dapat dipastikan bahwa menggunakan dataset yang benar.

2) *Feature Engineering*: Langkah ini bertujuan untuk menentukan parameter yang digunakan dalam PdM. Dari 16 parameter berbeda pada tiap kolom, belum tentu semua cocok digunakan dalam ML. Selain itu, data mentah yang didapatkan tidak memiliki ketentuan kondisi untuk bisa dinyatakan normal atau tidak, sehingga perlu diberikan label untuk mengelompokkannya. Ada dua alternatif pendekatan yang dapat dilakukan: menggunakan algoritma *clustering* atau menggunakan pendekatan statistik. Data akan dibagi menjadi tiga kategori: Outlier, Normal, dan Maintenance. Selain itu juga bisa ditambahkan label secara manual untuk setiap parameter

jika diketahui, seperti memberi label kelipatan 100 jam kerja sebagai perawatan berdasarkan *checksheet*.

3) *Algorithm Selection*: Pada tahap ini akan dipilih algoritma yang akan digunakan dalam membuat model ML. Algoritma disediakan oleh *library* pemrograman yang secara public tersedia untuk digunakan, seperti TensorFlow atau Scikit-learn [24]. Pilihan algoritma yang akan diuji kemungkinan adalah Logistic Regression (LR), k-NN, RF, Decision Tree, dan SVM untuk mengklasifikasikan data. Selain itu algoritma clustering seperti Density-Based Spatial Clustering of Applications with Noise (DBSCAN) dan Kmeans juga dipertimbangkan untuk memberi label pada dataset.

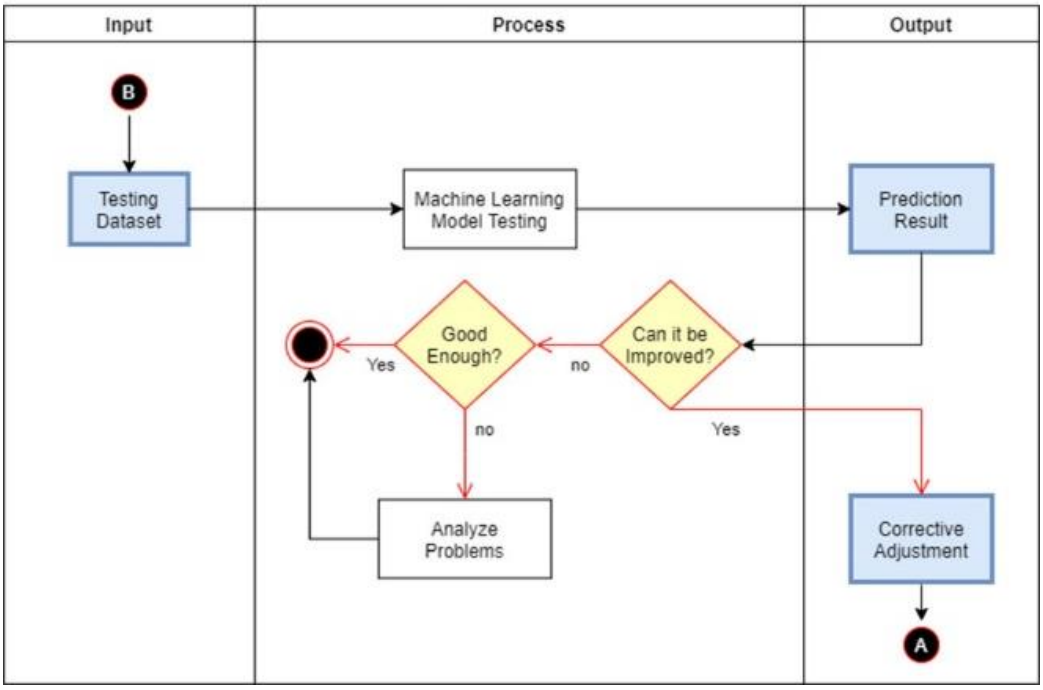
4) *Data Pre-Processing*: Rincian langkah ini akan berbeda tergantung pada algoritma yang digunakan dan akan menyesuaikan dengan situasi. Karena kumpulan data saat ini mungkin perlu informasi yang akan ditambahkan, diubah, atau dihapus. Setelah itu, data akan dibagi menjadi dua: data berlabel digunakan sebagai *training dataset* yang akan menjadi masukan untuk langkah selanjutnya. Sementara itu, *validation dataset* yang labelnya diketahui namun disembunyikan digunakan untuk menguji model.

5) *Algorithm Training*: Pada langkah ini, *training dataset* akan diproses oleh algoritma yang dipilih, menghasilkan model ML. Untuk mengetahui performanya menggunakan *confusion matrix* sehingga dapat diketahui Akurasinya, serta Runtime diukur dari saat model dilatih hingga mengeluarkan hasil prediksinya.

III.4 *ML Model Evaluation*

Menyerupai tahap Deskriptif Studi II (DS-II) pada DRM, yaitu untuk menyelidiki dampak dari informasi dukungan dan kemampuan desain untuk memenuhi tujuan. Dilakukan dua studi empiris untuk mendapatkan pemahaman, studi pertama dilakukan untuk mengevaluasi penerapan informasi yang mendukung untuk membuat desain menjadi lebih baik. Studi kedua digunakan untuk mengevaluasi kegunaan, yaitu, keberhasilan desain, berdasarkan kriteria yang dikembangkan

sebelumnya [30]. Tahap evaluasi akan mengecek hasil data yang didapat dari tahap sebelumnya, kemudian dicari kontrol yang tepat untuk mencapai hasil yang diinginkan. Hasil evaluasi dibandingkan dari keseluruhan proses iterasi untuk mendapatkan informasi nilai mana yang terbaik. Sehingga pada proses berikutnya dapat diterapkan kontrol yang sesuai untuk membuatnya jadi lebih baik.



Gambar III.4 Alur kerja proses evaluasi model ML

1) *ML Model Testing*: Pada langkah ini, model ML akan diuji menggunakan dua dataset. Pertama menggunakan *testing dataset* yang labelnya diketahui namun disembunyikan, untuk mengetahui performa Akurasi dari algoritma *Rule-Based*. Sementara untuk algoritma ML, diuji menggunakan *Stratified K-Fold Cross Validation*. Sementara itu, *Runtime* diukur dengan memberikan nilai acak seragam sebagai masukan, kemudian menghitung waktu dari awal program tes dieksekusi sampai hasil prediksi muncul menggunakan model yang telah dilatih pada dataset *training*.
2) Evaluasi Kemungkinan Peningkatan: Langkah ini menentukan apakah model dapat ditingkatkan untuk mencapai hasil yang lebih baik pada iterasi berikutnya. Salah satu cara untuk melakukannya adalah dengan menyesuaikan *Hyperparameter* yang berbeda. Contohnya adalah nilai k dalam k-NN, batas error di DT, fungsi kernel pada SVM, dan sebagainya.

Kemungkinan lain adalah menyesuaikan dataset pelatihan atau menggabungkan beberapa algoritma [24]. Bagaimanapun caranya, penyesuaian dibuat kemudian kembali ke tahap pemodelan ML. Jika tidak ada perbaikan yang diamati setelah mencoba pendekatan yang berbeda, maka akan berlanjut ke langkah berikutnya.

3) Evaluasi Kriteria Keberhasilan: Hasil pengujian dibandingkan dengan Kriteria Sukses dan hasil dari penelitian yang serupa. Selain itu, hasil dari iterasi lain juga akan dibandingkan dan ditentukan mana yang lebih baik. Kriteria sukses untuk Akurasi minimal bernilai 90%, dengan Runtime maksimal sekitar 15 menit. Nilai-nilai ini hanyalah asumsi untuk hasil yang dapat diterima dan dapat berubah sebagai situasi berkembang. Jika hasil saat ini memuaskan keberhasilan kriteria, maka tahap ini selesai. Sementara itu, jika hasilnya masih kurang memuaskan, maka dilanjutkan ke tahap berikutnya.
4) *Analyze Problems*: Mencapai titik ini berarti kemungkinan besar tidak menemukan atau tidak memiliki cara untuk mempengaruhi faktor yang menyebabkan hasil untuk menjadi lebih baik. Permasalahan ini mungkin akan menyebabkan untuk kembali ke tahap sebelumnya dalam Metodologi agar menemukan pendekatan yang berbeda untuk solusi lain. Pada akhirnya, jika masalah tetap ada sampai periode penelitian ini berakhir, maka akan dibahas kemungkinan dan rekomendasi untuk pekerjaan yang akan datang.

BAB IV PEMROSESAN DATASET

Pada Bab ini akan dibahas tahapan dalam menyiapkan dataset untuk digunakan dalam pelatihan dan pengujian model klasifikasi. Tujuan yang ingin dicapai pada tahap ini adalah untuk berhasil membuat dataset yang memiliki label untuk diolah model klasifikasi.

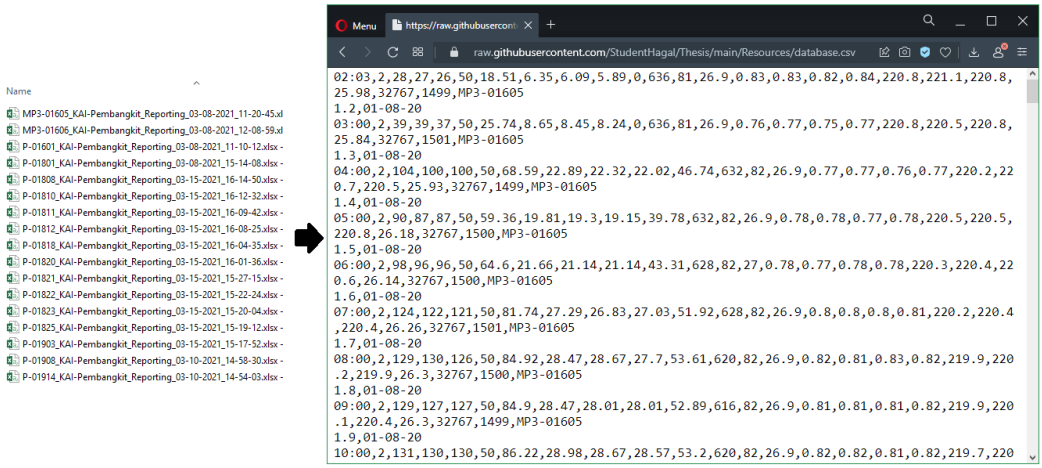
```
# Check the versions of libraries
# !!! WARNING !!!
# Important because model result may be different for other version

print('Python: {}'.format(sys.version))
print('scipy: {}'.format(scipy.__version__))
print('numpy: {}'.format(np.__version__))
print('matplotlib: {}'.format(matplotlib.__version__))
print('pandas: {}'.format(pd.__version__))
print('sklearn: {}'.format(sklearn.__version__))
print('joblib: {}'.format(joblib.__version__))
```

```
Python: 3.7.12 (default, Sep 10 2021, 00:21:48)
[GCC 7.5.0]
scipy: 1.4.1
numpy: 1.19.5
matplotlib: 3.2.2
pandas: 1.1.5
sklearn: 1.0.1
joblib: 1.1.0
time: 7.95 ms (started: 2021-11-19 12:12:03 +00:00)
```

Gambar IV.1 Kode untuk menampilkan versi *library* yang digunakan

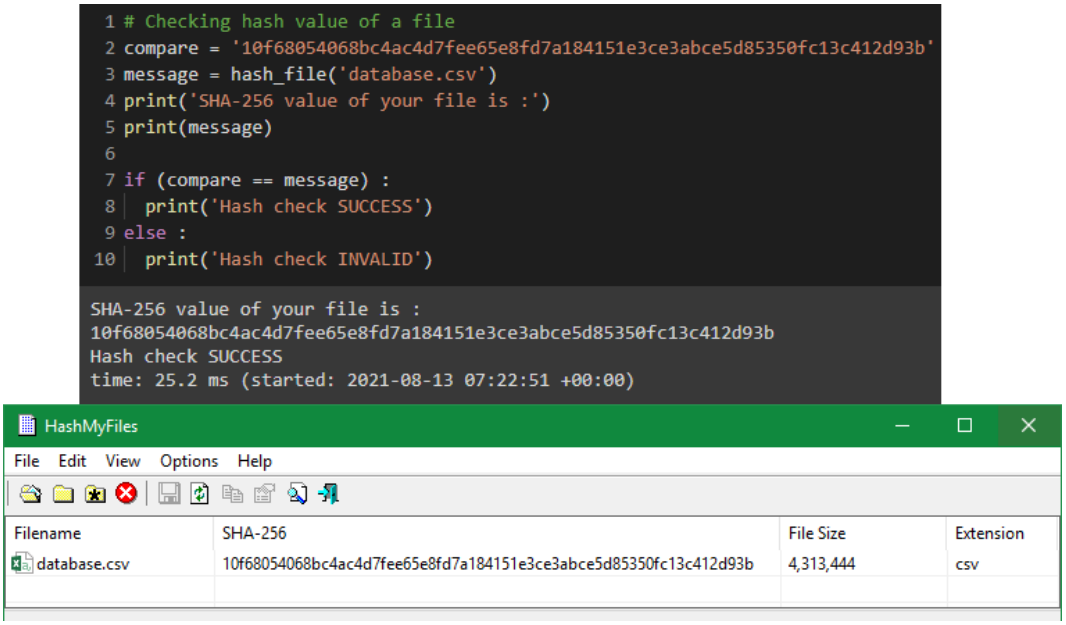
Sebelum mulai memproses data, kegiatan awal yang dilakukan adalah untuk membuat program di Google Colab menggunakan Bahasa pemrograman Python, dengan detail versi keseluruhan library yang digunakan ditampilkan pada gambar. Pengecekan versi *library* yang digunakan penting, karena jika menggunakan versi berbeda ada kemungkinan akan memberikan hasil yang tidak sesuai dengan penelitian ini.

IV.1 Membuat database standar



Gambar IV.2 Pemrosesan menjadi database standar

Untuk dapat lebih mudah mengolah data, maka dari 17 file didapatkan untuk jenis kereta yang berbeda data pemantauan sensor diambil lalu disatukan pada file dengan format csv, kemudian diupload ke layanan penyimpanan online untuk lebih mudah diakses. Standar yang dimaksud disini adalah bahwa memastikan pada setiap kolom merupakan entri data dari parameter yang sama, juga menyesuaikan tipe data agar seragam pada tiap kolom. Pada database tidak ada tag nama pada setiap kolom, ini akan ditambahkan saat pemrograman di tahap berikutnya.



Gambar IV.3 Pengecekan nilai hash file

Setelah itu juga ada pengecekan nilai Hash untuk memastikan integrasi file saat diunggah, diketahui dari pengecekan file lokal di laptop bahwa nilai SHA-256 file disimpan pada variabel compare. Kemudian nilai ini akan dibandingkan pada program dengan memanfaatkan hashlib yang akan menghitung nilai Hash dari file yang diunggah. Jika ada perbedaan, maka harus diunggah ulang hingga berhasil memiliki nilai yang sama.

IV.2 Merekayasa Fitur

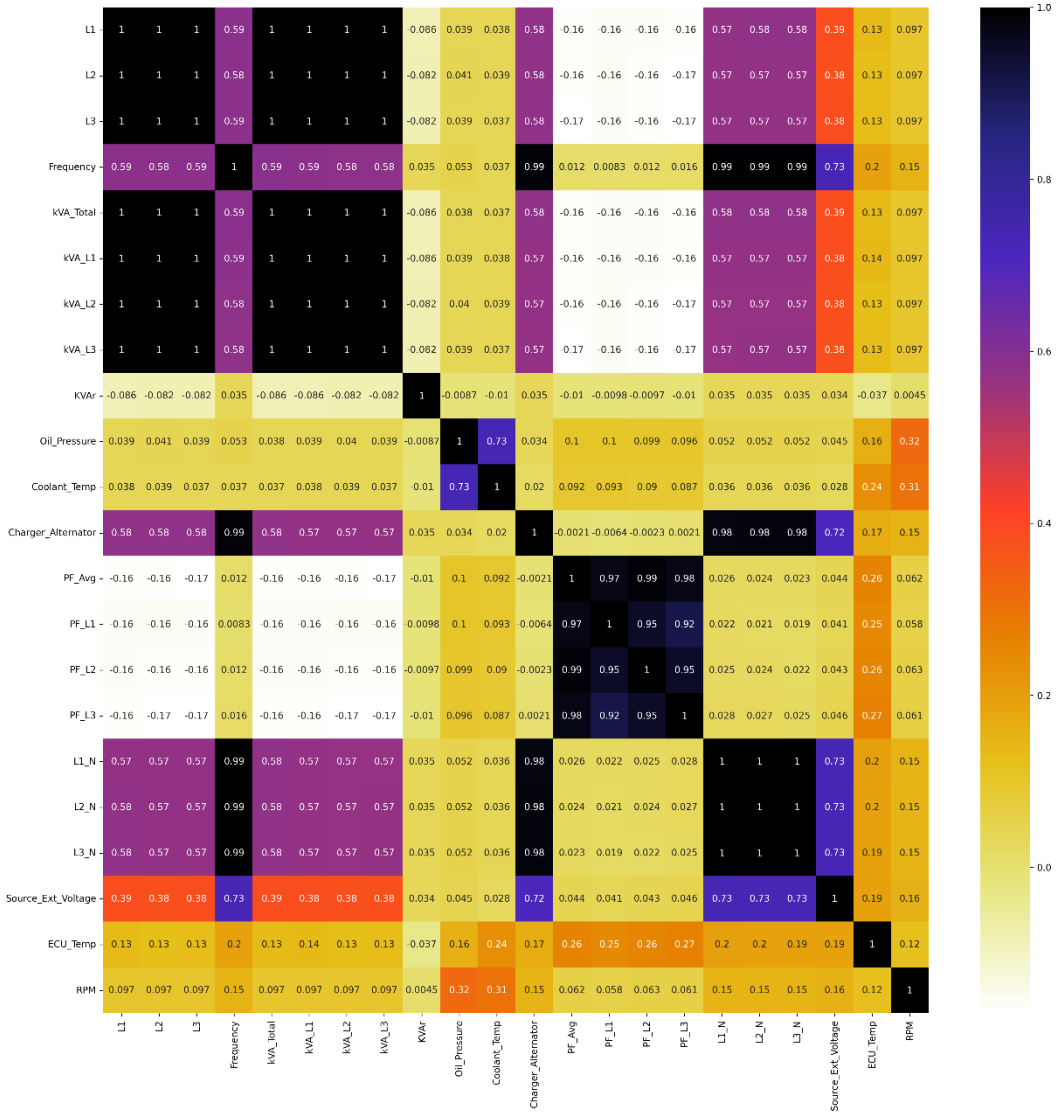
Pada tahap ini, penting sekali untuk memperhatikan urutan pemberian nama setiap kolom, karena jika salah maka nilainya tidak sesuai dengan data mentah yang didapatkan. Kesalahan ini menyebabkan hasil ML tidak merepresentasikan kondisi sesungguhnya, sehingga perlu dipastikan bahwa setiap nama pada setiap kolom sesuai dengan sensor yang tercatat dari dokumen KAI. Setelah memberikan nama kolom sesuai, kemudian dilakukan analisis data awal dengan melihat nilai deskripsi statistik.

Tabel IV.1 Hasil deskripsi statistik sensor pada kereta pembangkit

Sensors	mean	std	min	25%	50%	75%	max
L1	104.6612	79.09262	0	39	104	159	555
L2	103.4818	78.6856	0	38	103	157	562
L3	103.8361	78.72584	0	38	103	157	557
Frequency	41.84928	18.3292	0	50	50	50	51.9
kVA_Total	69.517	52.48381	0	26.0375	69.02	105.42	376.51
kVA_L1	23.09815	17.57388	0	8.44	22.985	35.16	125.34
kVA_L2	22.854	17.49758	0	8.28	22.725	34.7	127.24
kVA_L3	22.91934	17.48234	0	8.26	22.94	34.68	125.7
KVAr	25677.71	330882.7	0	9.68	33.54	53.88	4294967
Oil_Pressure	1533.996	8092.571	0	592	612	624	65535
Coolant_Temp	663.1184	4389.881	0	76	81	82	65531
Charger_Alternator	23.12604	10.26394	0	27.1	27.7	27.9	29.9
PF_Avg	5.95315	40.23436	0	0.78	0.87	0.89	327.65
PF_L1	5.577243	39.60832	0	0.78	0.87	0.89	327.65

PF_L2	5.935014	41.01353	0	0.78	0.86	0.89	327.65
PF_L3	6.34712	42.58512	0	0.78	0.86	0.88	327.65
L1_N	186.3499	80.73168	0	219.6	221.1	222.2	227.8
L2_N	186.4015	80.86799	0	219.7	221.3	222.4	227.9
L3_N	186.174	80.89059	0	219.675	221.1	222.3	229.1
Source_Ext_ Voltage	23.34032	9.010295	0	26.33	27.11	27.3	30.72
ECU_Temp	6278.679	12862.11	0	36	41	48	32767
RPM	1345.972	2602.134	0	1499	1499	1500	65531

Ada beberapa teknik yang dapat digunakan untuk menentukan fitur, disini akan menggunakan metode statistik untuk memilih fitur yang digunakan. Pertama, dipilih parameter yang memiliki nilai cukup bervariasi terutama pada rentang *interquartile* yaitu pada percentil 25% hingga 75% untuk digunakan. Hal ini dilakukan karena apabila nilai tidak memiliki variasi yang cukup, maka dapat dikatakan tidak terlalu berpengaruh terhadap kondisi komponen. Ini mengeliminasi sensor RPM, Source_Ext_Voltage, Charger_Alternator, dan Frequency.



Gambar IV.4 Heatmap korelasi pearson parameter sensor

Selanjutnya dilakukan perhitungan koefisien korelasi pearson, yaitu suatu ukuran korelasi linier antara dua set data. Perhitungan koefisien korelasi Pearson pada sampel berdasarkan rumus berikut:

$$r_{xy} = \frac{\Sigma_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\Sigma_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\Sigma_{i=1}^{n}(y_i - \bar{y})^2}}$$

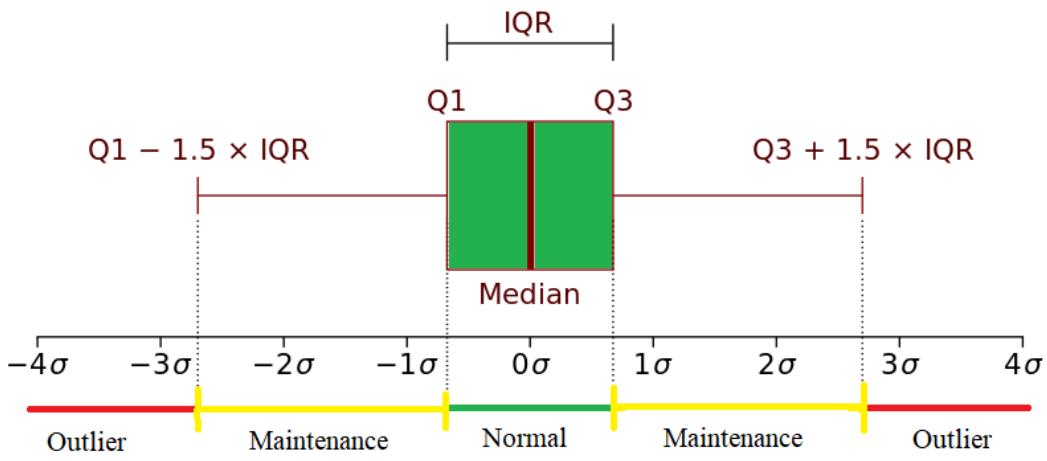
Dengan n ukuran sampel, x_i dan y_i merupakan sampel nilai pada index i, dan $\bar{x}$ dan $\bar{y}$ adalah mean dari sampel. Diambil parameter dengan hasil korelasi dibawah 0,75 yang menunjukan bahwa sensor tersebut memiliki korelasi rendah dengan parameter lainnya. Berdasarkan hipotesis dari penelitian [31], bahwa subset fitur

yang baik berisi fitur yang sangat berkorelasi dengan klasifikasi, namun tidak berkorelasi satu sama lain.

Dari hasil ini, didapatkan 6 parameter sensor tersisa, yaitu: L1, KVAr, Oil_Pressure, Coolant_Temp, PF_Avg, ECU_Temp. Untuk membatasi cakupan, digunakan 4 fitur berdasarkan sensor Oil_Pressure, Coolant_Temp, PF_Avg, ECU_Temp yang berkaitan dengan sistem pelumas dan pendingin dari generator kereta pembangkit. Dari data mentah yang didapatkan, tidak ada label untuk dapat mengetahui kondisi keperluan perawatan, sehingga perlu memberikan label. Diasumsikan ada 3 label, yaitu Outlier, Normal, dan Maintenance. Dipertimbangkan 2 alternatif pendekatan yang dapat dilakukan untuk pemberian label yaitu memanfaatkan algoritma clustering, atau pendekatan statistik.

IV.2.1 Pemberian label dengan pendekatan statistik

Untuk opsi pertama akan dicoba menggunakan metode *Rule-based* dengan pendekatan statistik, karena pada *checksheet* perawatan dari KAI tidak ada ketentuan batasan nilai pada parameter ini, sehingga diasumsikan menggunakan metode *interquartile range* (IQR) berikut [32].



Gambar IV.5 Ilustrasi kondisi klasifikasi label dengan Box Plot

- Stat_Normal :

$$(Q1 < \mu < Q3)$$

Untuk label klasifikasi Outlier ditentukan bahwa jika nilai pada sensor berada pada nilai IQR, yaitu pada nilai diantara kuartil 1 hingga kuartil 3. Kemudian

Cek nilai fitur lain, jika data tidak termasuk pada kondisi ‘Outlier’ atau ‘Maintenance’ maka akan diberi label ‘Normal’.

- Stat_Outlier :

$$(\mu < Q1 - IQR * 1.5) \text{or } (\mu > Q3 + IQR * 1.5)$$

Outlier adalah pengamatan yang terletak pada jarak abnormal dari nilai-nilai lain dalam sampel acak dari suatu populasi [32]. Kriteria untuk menentukan label klasifikasi Outlier dengan metode *Tukey's fences* didefinisikan dengan nilai k=1,5 untuk menunjukan Outlier, sehingga sampel yang bernilai lebih kecil dari batas bawah (1,5 kali IQR lebih kecil dari Kuartil 1) atau lebih besar dari batas atas (1,5 kali IQR lebih besar dari Kuartil 3) termasuk data Outlier. Kemudian berdasarkan pengecekan nilai fitur lain, jika ALL sensor bernilai Stat_Outlier maka diberi label sebagai ‘Outlier’

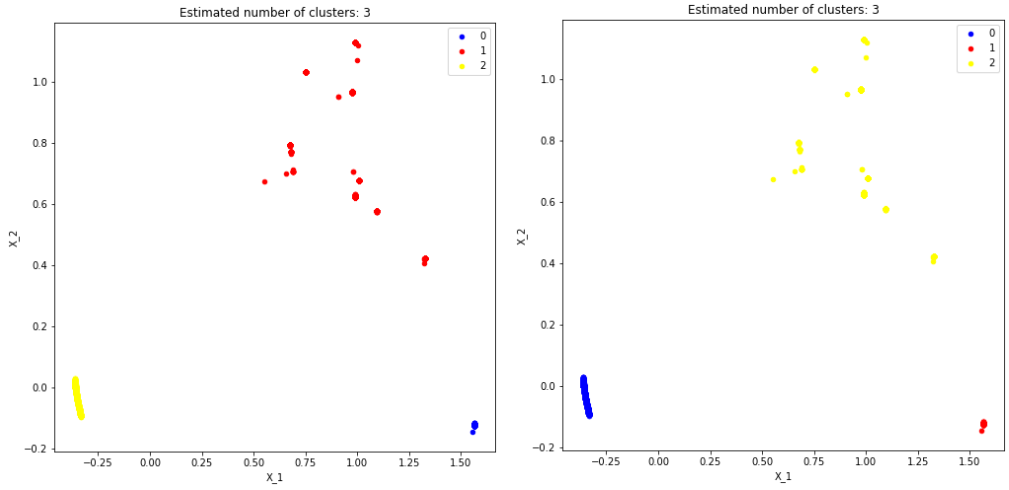
- Stat_Maintenance :

$$(Q1 - IQR * 1.5 \leq \mu \leq Q1) \text{ or } (Q3 \geq \mu \geq Q3 + IQR * 1.5)$$

Untuk label klasifikasi Maintenance, berada diantara nilai batas bawah dengan Kuartil 1, serta batas atas dengan Kuartil 3. Kemudian periksa nilai fitur lain, jika ANY sensor bernilai Stat_Maintenance, maka label sebagai ‘Maintenance’.

IV.2.2 Pemberian label dengan algoritma clustering

Opsi yang lain adalah penggunaan algoritma *unsupervised* ML berupa clustering untuk memberikan label pada dataset. Ada 2 algoritma yang digunakan, yaitu DBSCAN dan Kmeans. DBSCAN dipertimbangkan karena penggunaan nya yang cocok untuk dapat mendeteksi noise berupa titik sampel data yang jauh dari kumpulan sampel lainnya, sehingga noise dapat terdeteksi untuk diproses lebih lanjut. Sementara itu Kmeans dipertimbangkan karena kita dapat melakukan inisialisasi centroid sesuai jumlah cluster yang diinginkan, dalam kasus ini berjumlah 3 sesuai dengan label yang direncanakan. Dikarenakan ada 4 fitur untuk pertimbangan, perlu dilakukan reduksi dimensionalitas menggunakan *Principal Component Analysis* (PCA) menjadi 2 dimensi untuk dapat melihat hasil plot cluster.



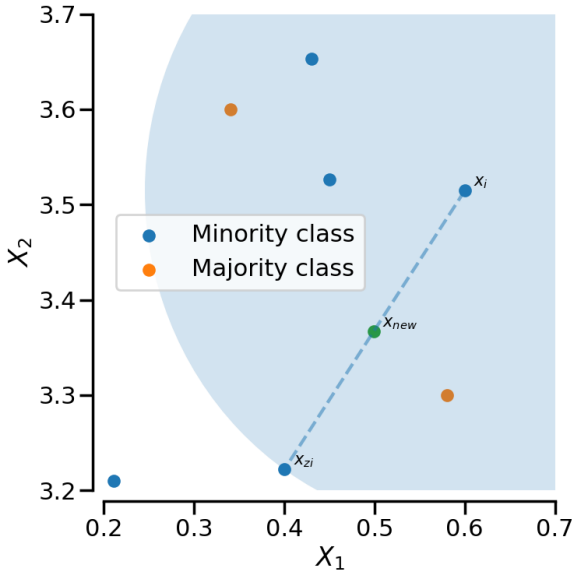
(a) (b)

Gambar IV.6 Hasil cluster DBSCAN (a) dan hasil cluster Kmeans (b)

Untuk mendapat hasil ini, pengaturan pada *hyperparameter* untuk DBSCAN adalah eps=0.5 dan min_samples=100, sementara Kmeans yaitu n_clusters=3 dan random_state=0. Semua data pada dataset diberikan label untuk pendekatan ini, dengan rasio jumlah data pada setiap cluster DBSCAN yaitu 5475:1288:27594 dengan format biru:merah:kuning. Sementara pada cluster Kmeans yaitu 27594:5475:1288 dengan format yang sama. Dapat terlihat bahwa kedua algoritma menghasilkan jumlah cluster yang sama, walaupun lokasi cluster (ditunjukan dengan warna) berbeda. Hasil cluster ini berdasarkan posisi data relatif terhadap satu sama lain, sehingga belum jelas terkaitannya dengan kondisi pada kereta pembangkit.

IV.3 Membuat Data Sintesis Menggunakan SMOTE

Setelah data diberikan label sesuai kondisi, dapat dibuat data sintesis menggunakan teknik *oversampling* dengan SMOTE untuk meningkatkan jumlah data yang dapat digunakan dengan interpolasi. Penerapan dengan *library* imbalanced-learn berdasarkan scikit-learn.



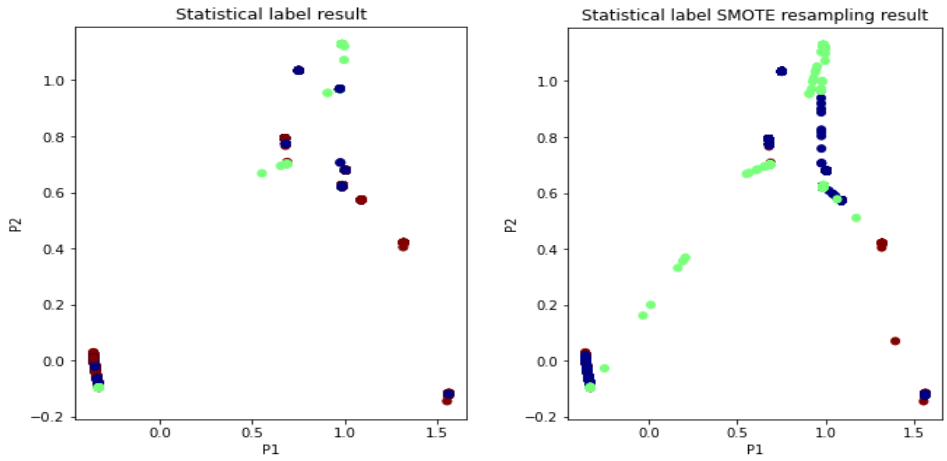
Gambar IV.7 Ilustrasi cara kerja SMOTE

Berdasarkan [33], SMOTE bekerja dengan mengambil sampel yang sudah ada X_i, kemudian sampel baru X_{new} akan dihasilkan dengan mempertimbangkan jumlah k tetangga terdekatnya (sesuai dengan kNN). Misalnya, 3 tetangga terdekat termasuk dalam lingkaran biru seperti yang diilustrasikan pada gambar. Kemudian, salah satu dari tetangga terdekat X_{zi} ini dipilih dan sampel dibuat berdasarkan persamaan sebagai berikut:

$$X_{new} = X_i + \lambda \times (X_{zi} - X_i)$$

di mana λ adalah angka acak dalam rentang [0, 1]. Interpolasi ini akan membuat sampel pada garis antara X_i dan X_{zi} seperti yang diilustrasikan.

IV.3.1 Hasil SMOTE label pendekatan Statistik

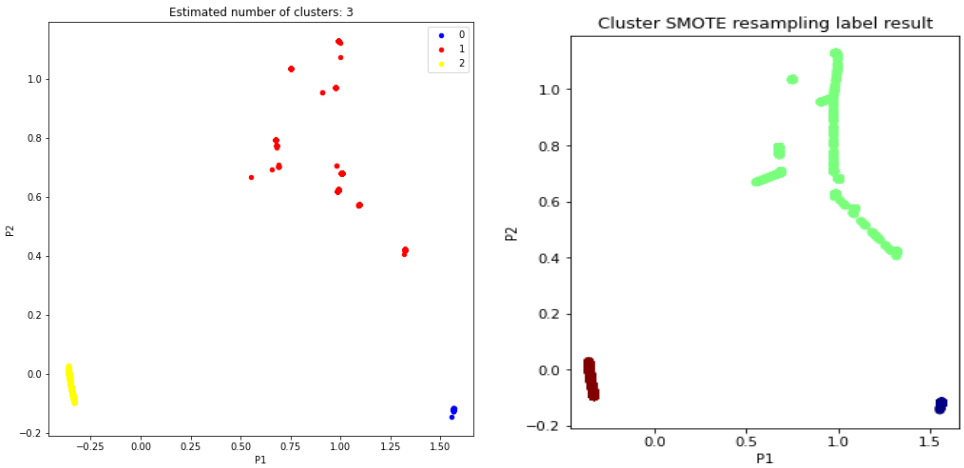


(a) (b)

Gambar IV.8 Hasil label statistik (a) dan hasil *oversampling* SMOTE (b)

Hasil yang didapat pada gambar adalah saat menggunakan SMOTE untuk membangkitkan data sintesis sebanyak 30000 bagi setiap label, dengan 90000 total sampel untuk keseluruhan dataset. Jumlah data yang akan dibangkitkan ini dapat diubah sesuai keperluan namun tidak bisa menjadi lebih sedikit dari jumlah data awal. Pada hasil label dengan pendekatan statistik, terlihat bahwa posisi sampel saling campur dengan data yang memiliki label lain. Hasil *oversampling* dengan SMOTE terlihat menambah sampel baru pada lokasi antar label yang berkaitan, sehingga terlihat hampir menghubungkan data yang berada di posisi lain.

IV.3.2 Hasil SMOTE pendekatan Clustering



(a) (b)

Gambar IV.9 Hasil cluster DBSCAN (a) dan hasil *oversampling* SMOTE (b)

Hasil yang didapat pada gambar adalah saat menggunakan SMOTE untuk membangkitkan data sintesis sebanyak 30000 bagi setiap label, dengan 90000 total sampel untuk keseluruhan dataset. Jumlah data yang akan dibangkitkan ini dapat diubah sesuai keperluan namun tidak bisa menjadi lebih sedikit dari jumlah data awal. Pada hasil label dengan algoritma clustering, terlihat bahwa posisi sampel terpisah dengan data yang memiliki label berbeda. Hasil *oversampling* dengan SMOTE juga tampak demikian dan hanya memperbanyak jumlah data yang berada pada cluster masing-masing tanpa menghubungkan antar cluster.

BAB V PEMODELAN ALGORITMA KLASIFIKASI

V.1 Metode *Rule-Based*

Metode klasifikasi konvensional dengan *Rule-based* bekerja dengan membuat keputusan untuk memberikan label data bergantung dari berbagai aturan yang biasanya dalam bentuk "*if.else*". Aturan-aturan ini umumnya ditentukan oleh manusia berdasarkan pengamatan dari data yang dimiliki. Inilah yang membedakan metode *rule-based* dengan *Machine-Learning* yang dapat mempelajari kondisi tanpa perlu membuat peraturan langsung pada kode program.

Pada penelitian ini, ada 2 set aturan yang berbeda dibuat sesuai dengan dataset yang digunakan. Untuk dataset dengan label statistik, cara kerja pengklasifikasi yang dibuat berdasarkan nilai statistic deskriptif seperti yang dijelaskan pada bab sebelumnya, dengan peraturan dijelaskan dalam pseudo code sebagai berikut:

```
#Rule-Based_Statistical

Input: x = array nilai dari 4 fitur

Output: Label kondisi Outlier/Maintenance/Normal

x = [fitur1, fitur2, fitur3, fitur4]
Q1 = 1st Quantile x dari Training dataset
Q3 = 3rd Quantile x dari Training dataset
IQR = Q3-Q1
Stat_Outlier = (x< Q1 – IQR*1.5) or  (x> Q3 + IQR*1.5)
Stat_Maintenance = ( Q1 - IQR*1.5≤x≤Q1) or (Q3≥x≥ Q3 + IQR*1.5)

if (fitur1 and fitur2 and fitur3 and fitur4)  == Stat_Outlier
        then Berikan label Outlier pada data
else if (fitur1 or fitur2 or fitur3 or fitur4)  == Stat_Maintenance
        then Berikan label Maintenance pada data
else
        then Berikan label Normal pada data
endif
```

Sementara untuk dataset dengan label clustering, karena kedua algoritma cluster menghasilkan jumlah cluster yang sama, maka hanya digunakan salah satu hasil saja berdasarkan cluster DBSCAN. Aturan pengklasifikasi yang dibuat berdasarkan komponen P1 dan P2 hasil PCA nilai fitur, dari hasil plot dapat ditentukan nilai yang dapat memisahkan setiap cluster. Untuk lebih detail, peraturan dijelaskan dalam pseudo code sebagai berikut:

#Rule-Based_Clustering

Input: komponen P1 dan P2 dari hasil PCA 4 fitur

Output: Label cluster 0/1/2

if P1 > 0.25 **and** P2 < 0.2
    then Berikan label 0 pada data
else if P1 > 0.25 AND P2 > 0.2
    then Berikan label 1 pada data
else
    then Berikan label 2 pada data
endif

Perbedaan antara keduanya yang perlu diperhatikan adalah pada Rule-Based_Statistical menggunakan dataset training untuk mengetahui nilai kuantil dari fitur. Sementara pada Rule-Based_Clustering kondisi batasan nilai komponen P1 dan P2 diatur konstan berdasarkan pengamatan pada cluster dataset original, sehingga tidak memerlukan dataset *training* untuk menentukan batasan. Selain itu juga pada masukan yang dibutuhkan, Rule-Based_Statistical langsung mengolah data nilai-nilai fitur, sementara Rule-Based_Clustering perlu dilakukan *pre-processing* dengan PCA untuk mendapat komponen P1 dan P2 dari nilai fitur.

V.2 Metode *Machine Learning*

Dengan pertimbangan berdasarkan survey [12] dan [17] akan dilakukan eksperiman untuk membandingkan beberapa jenis algoritma klasifikasi yang umum digunakan pada sistem PdM, yaitu: DT, SVM, LR, RF dan kNN. LR berbasis probabilitas cocok jika sampel semakin banyak. kNN berbasis jarak cocok untuk sampel sedikit pada penerapan awal. DT hanya butuh sedikit pre-proses data sehingga cocok untuk penerapan awal. SVM efektif pada dimensionalitas tinggi cocok jika menggunakan fitur yang semakin banyak. RF membangkitkan banyak pohon untuk pengambilan keputusan diharapkan dapat meningkatkan nilai Akurasi. Selain itu juga ditambahkan Naïve-Bayes (NB) karena berbasis probabilitas cocok jika sampel semakin banyak serta kemampuan skalabilitasnya dengan waktu pemrosesan yang cepat. *Linear discriminant analysis* (LDA) untuk kemampuan reduksi dimensionalitasnya sehingga akan cocok jika diterapkan dengan fitur yang semakin banyak. Semua algoritma disini menggunakan *library* yang tersedia pada scikit-learn.

V.3 Melakukan Pre-Processing dataset Klasifikasi

```
# Split-out training and test dataset
array = dataset.values
X = array[:,0:4]
y = array[:,4]
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.20,
                                                    random_state=1, stratify=y)

print('training data size = ', X_train.shape)
print('testing data size = ', X_test.shape)
print('data split completed')
```

Gambar V.1 Blok kode pembagian dataset

Semua dataset berlabel memiliki 2 variasi opsi yaitu: original dan sintesis dengan SMOTE. Keduanya akan digunakan untuk membandingkan algoritma mana yang lebih baik. Selanjutnya dataset dengan label dibagi menjadi *training dataset* & *testing dataset* dengan rasio 9:1. Pembagian dilakukan secara acak berdasarkan kondisi *random_state*=1 untuk memastikan hasilnya dapat direproduksi, parameter *stratify* berguna agar pemecahan data memiliki rasio yang sama pada setiap kelas label. Selain itu karena dataset clustering perlu menggunakan reduksi dimensionalitas untuk hasil cluster, maka dilakukan tahapan *Scaling*, Normalisasi dan PCA kemudian hasil komponen P1 dan P2 ditambahkan ke dataset.

V.4 Hasil *Training* Model Klasifikasi

Setiap algoritma dilatih menggunakan *training dataset* untuk menghasilkan model ML yang dapat memprediksi label kondisi perawatan. Kemudian model ML di export pada suatu file, sehingga dapat digunakan pada lingkungan sistem yang berbeda. Namun Rule-Based model tidak dapat diekspor langsung, karena set aturannya yang langsung dibuat pada kode, sehingga perlu melakukan *copy-paste* dari sumber kemudian menyesuaikan variabelnya yang berkaitan.

Tabel V.1 Data hasil pelatihan model klasifikasi

Opsi	Metode	Kriteria	Algoritma							
			LR	kNN	DT	SVM	RF	LDA	NB	Rule-Based
1	Original, Statistical Label	Acc	0.346	0.986	1	0.942	1	0.353	0.357	1
		Runtime (s)	3.8	2.516	0.859	61.214	2.56	0.749	0.562	1.5
2		Acc	0.682	0.994	1	0.971	1	0.424	0.427	0.39

	SMOTE, Statistical Label	Runtime (s)	4.835	14.455	1.227	257.8	6.427	1.123	0.927	3.7
3	Original, Cluster Label	Acc	1	0.999	1	1	1	0.949	0.998	1
		Runtime (s)	13.963	2.301	0.442	60.774	1.806	0.362	0.418	0.284
4	SMOTE, Cluster Label	Acc	1	0.999	1	1	1	0.9488	0.998	1
		Runtime (s)	37.69	4.221	0.1481	286.29	3.801	0.2013	0.103	0.431
Rata-Rata Akurasi			0.757	0.995	1	0.978	1	0.669	0.695	0.848
Rata-Rata Runtime			15.072	5.873	0.669	166.52	3.649	0.609	0.503	1.479

Dari perbandingan rata-rata akurasi untuk keseluruhan opsi dan algoritma, DT dan RF menunjukan performa terbaik dengan skor Akurasi rata-rata 1 yang menyatakan semua label berhasil diprediksi dengan benar. Hasil akurasi terburuk yaitu pada model LDA. Sementara untuk Runtime, NB mendapatkan hasil terbaik dengan SVM memiliki hasil terburuk.

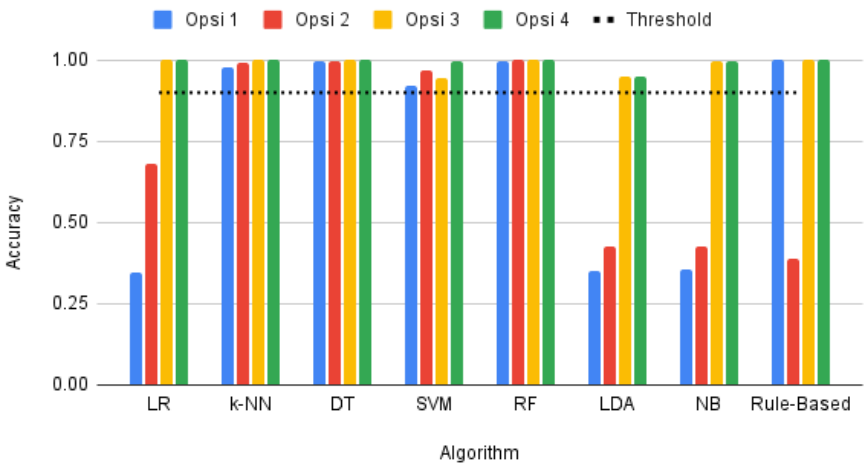
BAB VI EVALUASI MODEL KLASIFIKASI

VI.1 Menguji Model Klasifikasi

Ada dua pengujian yang akan dilakukan, yaitu menguji Akurasi menggunakan dataset original dan sintesis dengan target sukses minimal 90%. Sementara pengujian Runtime akan dilakukan dengan memberikan jumlah data yang bervariasi, kemudian melihat trend yang dihasilkan dengan batas waktu maksimal 15 menit.

VI.1.1 Menguji Akurasi

Pada tahap ini akan digunakan *testing dataset* untuk menguji performa model *Rule-based*, sementara untuk model ML akan diuji menggunakan *stratified k-fold.* Dengan demikian dapat diketahui nilai Accuracy untuk dibandingkan dengan hasil training. Jika hasilnya konsisten, atau tidak terlalu menyimpang jauh dengan nilai Akurasi dari hasil *training model*, maka model ML telah tervalidasi bahwa model dapat digunakan untuk prediksi data selain *dataset training*.



Gambar VI.1 Hasil Akurasi model Klasifikasi dengan dataset uji

Opsi mengacu pada Tabel V.1 pada bab sebelumnya, yaitu jenis dataset yang digunakan. Dari perbandingan rata-rata akurasi untuk keseluruhan opsi dan algoritma, DT dan RF menunjukan performa terbaik dengan skor Akurasi rata-rata mendekati 1 yang menyatakan hampir semua label berhasil diprediksi dengan benar. Hasil akurasi terburuk yaitu pada model LDA. Hasil ini menyerupai nilai

yang didapat pada *training* model, dengan demikian dapat tervalidasi bahwa hasil model tepat.

VI.1.2 Menguji Runtime

Dikarenakan Runtime meningkat dengan jumlah data yang diproses, maka pengujian akan dilakukan dengan memberikan jumlah data yang bervariasi, kemudian melihat trend yang dihasilkan dengan batas waktu maksimal 15 menit (900 detik). Karena pengujian ini tidak mempedulikan nilai akurasi model, maka dapat membangkitkan sampel acak sebanyak yang dibutuhkan dari distribusi *uniform* pada rentang data original. kemudian diproses oleh model yang telah dilatih menggunakan dataset original, lalu mengukur waktu yang diperlukan untuk mengeluarkan hasil.

Tabel VI.1 Data hasil pengujian Runtime model klasifikasi dalam detik

Metode Pemberian Label	Jumlah Data	Runtime Model (detik)							
		LR	kNN	DT	SVM	RF	LDA	NB	Rule-Based
Statistik	3×10^3	0.002	0.123	0.002	1.470	0.049	0.003	0.003	0.051
	3×10^4	0.005	1.254	0.005	13.767	0.252	0.005	0.008	0.096
	3×10^5	0.023	11.601	0.024	132.77	2.408	0.020	0.053	0.121
	3×10^6	0.156	117.638	0.211	1331.78	23.867	0.180	0.362	0.570
Clustering	3×10^3	0.004	0.128	0.003	0.796	0.033	0.003	0.004	0.038
	3×10^4	0.005	1.270	0.005	7.549	0.198	0.004	0.015	0.053
	3×10^5	0.016	11.254	0.012	73.690	1.862	0.015	0.042	0.137
	3×10^6	0.129	114.381	0.101	737.325	18.213	0.133	0.316	1.030
Runtime rata-rata		0.034	25.765	0.036	229.915	4.688	0.036	0.080	0.210

Dari perbandingan rata-rata runtime model dari kedua metode pemberian label, LR menunjukan performa terbaik dengan waktu rata-rata 0.034 detik. Hasil akurasi terburuk yaitu pada model SVM dengan waktu rata-rata sekitar 300 detik. Perlu diperhatikan bahwa batas waktu 900 detik hanya terlampaui oleh model SVM untuk mengolah sebanyak 3×10^6 data dari label statistik.

VI.2 Mengevaluasi Hasil Pengujian

VI.2.1 Mengevaluasi Kemungkinan Peningkatan Kinerja Model ML

Hasil saat ini mayoritas masih menggunakan *hyperparameter* default yang diatur oleh *library* scikit-learn [34], kecuali pada LR mengubah max_iter=1000 dan SVM dengan gamma=’auto’. Selain itu masih ada peluang untuk meningkatkan performa menjadi lebih baik, dengan melakukan *hyperparameter tuning* menggunakan teknik seperti *Random search* atau *Grid search*. Namun berdasarkan hasil saat ini, sudah terlihat bahwa ada model yang sudah berhasil memenuhi target kriteria sukses, sehingga dapat menggunakan model tersebut untuk penerapan awal PdM. Tidak perlu menggunakan semua model yang diuji untuk diterapkan, tetapi memilih beberapa model terbaik saja.

VI.2.2 Mengevaluasi Keberhasilan Model Klasifikasi

Menentukan model terbaik yang didapatkan dari keseluruhan opsi dalam pengujian evaluasi akan dilakukan dengan membandingkan kriteria suksesnya. Data ini akan ditabulasi dan diberi nilai untuk menentukan mana yang lebih cocok untuk sistem perawatan prediktif. Penilaian dilakukan dengan memberikan poin dari 1 (terendah) hingga 8 (tertinggi) berdasarkan performa, kemudian dilihat total skor dan akan dipilih yang memiliki total terbanyak.

Tabel VI.2 *Decision matrix* algoritma model ML terbaik

Kriteria	LR	KNN	DT	SVM	RF	LDA	NB	Rule-Based
Akurasi	4	6	8	5	8	1	2	4
Runtime	8	2	7	1	3	7	5	4
Akurasi/Runtime	7	2	8	1	3	6	5	4
Total	19	10	23	7	14	14	12	12

Nilai tersebut berdasarkan hasil yang didapatkan dari bagian VI.1.1 untuk Akurasi dan bagian VI.1.2 untuk Runtime. Dari pengujian Akurasi dapat terlihat bahwa kNN, DT, RF dan SVM sudah berhasil memenuhi target kriteria sukses diatas 90% pada keseluruhan opsi. Sementara untuk Runtime, batas waktu 15 menit hanya terlampaui oleh model SVM untuk mengolah sebanyak 3×10^6 data dari label statistik. Pada *decision matrix* nilai akurasi DT dan RF seimbang karena memiliki nilai Akurasi yang serupa pada pengujian, Begitupun untuk skor Runtime DT dan LDA

seimbang karena memiliki nilai rata-rata yang serupa pada pengujian. Secara keseluruhan kriteria, DT merupakan model Klasifikasi terbaik yang cocok untuk digunakan pada sistem PdM kasus ini.

VI.3 Menganalisis Masalah

Hasil pengujian akurasi untuk model *Rule-based_Statistical* pada opsi 2 memiliki nilai Akurasi rendah, meskipun pada opsi lain berhasil mendapatkan hasil yang baik. Hal ini mungkin disebabkan oleh cara kerja model *Rule-based_Statistical* yang menggunakan dataset *training* untuk menentukan batasan nilai statistik yang digunakan untuk Klasifikasi. Karena pada opsi 2 dilakukan resampling menggunakan SMOTE, sehingga nilai IQR akan berubah dari data original. Solusi yang dapat digunakan adalah mengatur nilai konstan untuk variabel IQR dan turunannya berdasarkan analisis data original. Namun dengan demikian akan mengurangi kemampuan untuk adaptasi apabila data baru tersedia seiring dengan berjalannya waktu, dan perlu dilakukan penyesuaian lagi berdasarkan analisis oleh manusia. Begitupun untuk model *Rule-based_Clustering*, meskipun hasilnya untuk saat ini telah memberikan hasil yang baik, tetapi karena diatur nilai konstan sebagai kondisi Klasifikasi maka apabila data baru tersedia seiring dengan berjalannya waktu, perlu dilakukan penyesuaian lagi berdasarkan analisis oleh manusia.

Seluruh model yang menggunakan opsi 3 dan 4 yang berdasarkan algoritma clustering untuk pemberian label menghasilkan Akurasi yang bagus dalam memprediksi Klasifikasi dari cluster, namun hasil cluster ini tidak diketahui keterkaitannya dengan kondisi kereta pembangkit. Perlu dianalisis lebih lanjut dengan mempelajari data cluster, dibutuhkan pengetahuan mengenai domain perawatan kereta pembangkit untuk dapat menjelaskan relasi antara cluster dengan kondisi komponen.

Untuk memastikan bahwa model yang telah dilatih dapat memprediksi kondisi dengan benar, perlu melakukan pengujian langsung pada kereta pembangkit yang jelas memiliki kondisi ‘Normal’ ataupun ‘Maintenance’. Namun karena keterbatasan kondisi dikarenakan pandemik COVID-19, langkah ini belum dijalankan pada penelitian ini.

BAB VII Penutup

VII.1 Kesimpulan

Berikut ini adalah kesimpulan yang didapatkan dari penelitian yang telah dilakukan dalam pengerjaan tesis.

1. Proses bisnis perawatan sarana dari KAI untuk saat ini telah dibuat alur kerjanya dengan bantuan program *Enterprise Resource Management* berupa SAP. Meski demikian, dalam pelaksanaan untuk mengecek kondisi sarana masih dilakukan manual berdasarkan *checksheet* dari kertas dan hasilnya belum disimpan dalam bentuk digital. Lokasi perbaikan ringan yang berbasis di depo masing-masing tempat kereta api beroperasi menyebabkan data yang tersedia tersebar dan sulit diakses secara menyeluruh, sementara itu balai yasa hanya melayani perawatan besar (*overhaul*) sehingga tidak memiliki data inspeksi rutin.

2. Penelitian ini menghasilkan model ML untuk mendeteksi kemungkinan keperluan perawatan berdasarkan 4 fitur dari nilai sensor (Oil_Pressure, Coolant_Temp, PF_Avg, ECU_Temp) pada generator kereta pembangkit yang merepresentasikan kondisi komponen pelumas dan pendingin. Proses evaluasi model tersebut meliputi langkah pemrosesan dataset, pelatihan model, hingga pengujian hasil dalam program sederhana yang dapat menentukan Klasifikasi kondisi dari masukan nilai fitur tersebut.

3. Ada 4 opsi dataset untuk memprediksi Klasifikasi dalam penerapan sistem PdM yang telah diuji pada penelitian ini. Opsi 1 menggunakan pendekatan statistik untuk memberi label, dengan 3 label (Outlier, Normal, Maintenance). Opsi 2 juga menggunakan pendekatan statistik untuk memberi label, tetapi jumlah data diperbanyak secara sintesis dengan SMOTE. Opsi 3 menggunakan algoritma clustering untuk memberikan label pada dataset, yang menghasilkan 3 cluster. Sementara Opsi 4 juga menggunakan algoritma clustering untuk memberi label, tetapi jumlah data diperbanyak secara sintesis dengan SMOTE.

4. Model DT memberikan hasil Akurasi yang lebih baik serta waktu pemrosesan yang lebih cepat pada keseluruhan opsi dibandingkan cara konvensional dengan

Rule-based. Begitupula jika dibandingkan dengan model Klasifikasi ML lainnya, DT memiliki keunggulan pada rasio Akurasi/Runtime dengan hasil 0,999/0,036 sehingga cocok digunakan untuk prediksi Klasifikasi pada kasus ini.

VII.2 Saran

Berikut ini adalah saran yang diberikan untuk pengembangan lanjutan dari penelitian ini.

1. Meningkatkan jumlah data sensor yang digunakan, saat ini data original hanya berjumlah sekitar 33832 meskipun dapat ditingkatkan dengan SMOTE. Dari hasil eksplorasi data tersebut, terlihat bahwa perangkat pengumpul data menyimpan nilai pada interval 1 jam. Periode ini perlu dipersingkat untuk mendapatkan data lebih banyak pada satu kereta pembangkit.

2. Dibutuhkan data yang sudah memiliki label untuk mendukung data sensor. Contohnya seperti hasil perawatan Harian dan Bulanan, terutama saat terjadi perbaikan pada komponen terkait. Sebaiknya sudah tersedia dalam bentuk digital untuk lebih mudah diolah menjadi dataset yang dapat dipercaya untuk digunakan pada ML dalam sistem PdM.

3. Penambahan fitur yang digunakan sebagai basis prediksi untuk dapat meningkatkan ketepatan dan melakukan monitor kondisi komponen lainnya pada sarana KAI.

4. Pengembangan aplikasi dalam penelitian ini harus dilakukan secara lebih lanjut untuk dapat memberikan manfaat nyata, yakni meningkatkan ketersediaan sarana KAI. Contohnya adalah membuat antarmuka untuk pengguna, ataupun *Application Programming Interface* (API) untuk dapat mengambil data sensor secara otomatis.

5. Perlu memastikan bahwa dataset yang telah digunakan dalam model terus diperbarui seiring dengan bertambahnya data yang terkumpul, kemudian melakukan pelatihan ulang model, sehingga ketepatan model tidak menurun dari masa ke masa.

6. Melakukan uji banding antara hasil prediksi model ML dengan kondisi nyata saat pengamatan langsung dilapangan. Terutama dengan mengumpulkan data pada komponen kereta pembangkit saat kondisi jelas ‘Normal’ atau butuh ‘Maintenance’ sehingga label merepresentasikan keadaan nyata. Kemudian memperbaiki model berdasarkan hasil yang didapatkan, sehingga model ML menjadi lebih baik lagi dalam melakukan prediksinya.

7. Mengolah data yang telah memiliki label untuk memprediksi RUL. Ada beberapa metode untuk memprediksi RUL yang berbeda bergantung dengan data yang dimiliki, salah satunya adalah dengan *Survival Model* dapat menggunakan data waktu antara label ‘Maintenance’ kemudian menghitung rata-rata untuk mendapat *Mean Time Between Maintenance* (MTBM). Untuk mengetahui RUL, maka dapat dihitung dengan nilai MTBM dikurang waktu saat ini kereta beroperasi semenjak perawatan terakhir.

DAFTAR PUSTAKA

[1] Menteri Perhubungan Republik Indonesia, *PM 153 Tentang Standar Spesifikasi Teknis Lokomotif Sarana Perkeretaapian,* Jakarta: Menteri Perhubungan Republik Indonesia, 2016.

[2] Menteri Perhubungan Republik Indonesia, *PM 30 Tentang Tata Cara Pengujian dan Pemberian Sertifikat Prasarana Perkeretaapian,* Jakarta: Menteri Perhubungan Republik Indonesia, 2011.

[3] Menteri Perhubungan Republik Indonesia, *PM 15 Tentang Standar, Tata Cara Pengujian dan Sertifikasi Kelaikan,* Jakarta: Menteri Perhubungan Republik Indonesia, 2011.

[4] Menteri Perhubungan Republik Indonesia, *KM 41 Tentang Standar Spesifikasi Teknis Kereta Yang Ditarik Lokomotif,* Jakarta: Menteri Perhubungan Republik Indonesia, 2010.

[5] Menteri Perhubungan Republik Indonesia, *KM 43 Tahun 2010 Tentang Standar Spesifikasi Teknis Gerbong,* Jakarta: Menteri Perhubungan Republik Indonesia, 2010.

[6] Menteri Perhubungan Republik Indonesia, *PM 18 Tentang Standar Tempat dan Peralatan Perawatan Sarana Perkeretaapian,* Jakarta: Menteri Perhubungan Republik Indonesia, 2019.

[7] W. Septiani, D. Suhardini and E. Sari, "Pengukuran Kinerja Perawatan Lokomotif PT. Kereta Api Indonesia (Persero) Berdasarkan Model Maintenance Scorecard," *J@ti Undip : Jurnal Teknik Industri,* vol. 7, no. 3, pp. 191-198, 2013.

[8] D. Burrough, "Digital transformation improves SNCF’s maintenance systems," International Railway Journal, 25 06 2020. [Online]. Available: https://www.railjournal.com/in_depth/digital-transformation-improves-sncfs-maintenance-systems. [Accessed 29 09 2021].

[9] The British Standards Institution, *Maintenance - Maintenance terminology,* BSI Standards, 2018.

[10] D. Jeffrey, Principles of Machine Operation and Maintenance, Oxford: Butterworth Heinemann Ltd, 1991.

[11] B. Schmidt and L. Wang, "Cloud-enhanced predictive maintenance," *International Journal Advanced Manufacturing Technology,* 2016.

[12] Y. Ran, X. Zhou, P. Lin, Y. Wen and R. Deng, "A Survey of Predictive Maintenance: Systems,," *IEEE Communications Surveys & Tutorials*, 2019.

[13] D. Justus, J. Brennan, S. Bonner and A. S. McGough, "Predicting the Computational Cost of Deep Learning Models," in *IEEE International Conference on Big Data*, 2018.

[14] C. Krupitzer, T. Wagenhals, M. Zufle, V. Lesch, D. Schafer, A. Mozaffarin, J. Edinger, C. Becker and S. Kounev, "A Survey on Predictive Maintenance for Industry 4.0," 2020. [Online]. Available: https://www.semanticscholar.org/paper/A-Survey-on-Predictive-Maintenance-for-Industry-4.0-Krupitzer-Wagenhals/ef0cec144f3132f47f3a3e39b96aa8067bee90af. [Accessed 1 December 2020].

[15] M. C. Garcia, M. A. Sanz-Bobi and J. d. Pico, "SIMAP: Intelligent System for Predictive Maintenance Application to the health condition monitoring of a windturbine gearbox," *Computers in Industry*, pp. 552-568, 2006.

[16] K. A. Kaiser and N. Z. Gebraeel, "Predictive Maintenance Management Using Sensor-Based Degradation Models," *IEEE Transactions On Systems, Man, And Cybernetics*, vol. 39, no. 4, p. 840, 2009.

[17] W. Zhang, D. Yang and H. Wang, "Data-Driven Methods for Predictive Maintenance of Industrial Equipment: A Survey," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2213-2227, 2019.

[18] MathWorks, "Designing Algorithms for Condition Monitoring and Predictive Maintenance," [Online]. Available: https://www.mathworks.com/help/predmaint/gs/designing-algorithms-for-condition-monitoring-and-predictive-maintenance.html?s_eid=PSM_15028. [Accessed 11 10 2021].

[19] MathWorks, "Decision Models for Fault Detection and Diagnosis," [Online]. Available: https://www.mathworks.com/help/predmaint/ug/decision-models-for-fault-detection-and-diagnosis.html. [Accessed 12 10 2021].

[20] MathWorks, "RUL Estimation Using RUL Estimator Models," [Online]. Available: https://www.mathworks.com/help/predmaint/ug/rul-estimation-using-rul-estimator-models.html. [Accessed 11 10 2021].

[21] K. Schwab, The Fourth Industrial Revolution, New York: Crown Publishing Group, 2016.

[22] S. J. Russel and P. Norvig, Artificial Intelligence: A Modern Approach (3rd Ed), New Jersey: Prentice Hall, 2009.

[23] E. Alpaydin, Introduction to Machine Learning (Second ed.), London: The MIT Press, 2010.

[24] A. Geron, Hands-On Machine Learning with Scikit-Learn & TensorFlow, California: O'Reilly Media, Inc, 2017.

[25] A. Barnes, "Increased Locomotive performance using Condition Based Maintenance," in *6th IET Conference on Railway Condition Monitoring (RCM 2014)*, Birmingham, 2014.

[26] T. Y. En, T. J. Jie, M. S. Ki, N. T. Hui and M. Ashrof, "Predictive Maintenance of a Train System Using a Multilayer Perceptron Artificial Neural Network," in *2018 International Conference on Intelligent Rail Transportation (ICIRT)*, Singapore, 2018.

[27] Q. Wang, S. Bu and Z. He, "Achieving Predictive and Proactive Maintenance for High-Speed Railway Power Equipment with LSTM-RNN," *IEEE Transactions On Industrial Informatics*, 2020.

[28] N. N. Hakim, Pengembangan Model Machine Learning untuk Analisis Prediksi Kemungkinan Kejadian Patah Rel Kereta Api di Indonesia, Bandung: ITB, 2020.

[29] R. Nappi, G. Cutrera, A. Vigliotti and G. Franzè, "A Predictive-based Maintenance Approach for Rolling Stocks Vehicles," in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Vienna, 2020.

[30] A. Chakrabarti and L. T. Blessing, DRM, a Design Research Methodology, London: Springer, 2009.

[31] B. Senliol, G. Gulgezen, L. Yu and Z. Cataltepe, "Fast Correlation Based Filter (FCBF) with a different search strategy," in *2008 23rd International Symposium on Computer and Information Sciences*, 2008.

[32] F. M. Dekking, C. Kraaikamp, H. P. Lopuhaä and L. E. Meester, A Modern Introduction to Probability and Statistics, London: Springer London, 2005.

[33] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, no. 1, pp. 321-357, 2002.

[34] Pedregosa and et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

LAMPIRAN

1. Perhitungan Koefisien Pearson Correlation Sensor Parameter

Sensor	L1	L2	L3	Frequency	kVA_Total	kVA_L1	kVA_L2	kVA_L3	KVAr	Oil_Pressure	Coolant_Temp	Charger_Alternator	PF_Avg	PF_L1	PF_L2	PF_L3	L1_N	L2_N	L3_N	Source_Ext_Voltage	ECU_Temp	RPM
L1	1.00	1.00	1.00	0.59	1.00	1.00	1.00	1.00	-0.09	0.04	0.04	0.58	-0.16	-0.16	-0.16	-0.16	0.57	0.58	0.58	0.39	0.13	0.10
L2	1.00	1.00	1.00	0.58	1.00	1.00	1.00	1.00	-0.08	0.04	0.04	0.58	-0.16	-0.16	-0.16	-0.17	0.57	0.57	0.57	0.38	0.13	0.10
L3	1.00	1.00	1.00	0.59	1.00	1.00	1.00	1.00	-0.08	0.04	0.04	0.58	-0.17	-0.16	-0.16	-0.17	0.57	0.57	0.57	0.38	0.13	0.10
Frequency	0.59	0.58	0.59	1.00	0.59	0.59	0.58	0.58	0.03	0.05	0.04	0.99	0.01	0.01	0.01	0.02	0.99	0.99	0.99	0.73	0.20	0.15
kVA_Total	1.00	1.00	1.00	0.59	1.00	1.00	1.00	1.00	-0.09	0.04	0.04	0.58	-0.16	-0.16	-0.16	-0.16	0.58	0.58	0.58	0.39	0.13	0.10
kVA_L1	1.00	1.00	1.00	0.59	1.00	1.00	1.00	1.00	-0.09	0.04	0.04	0.57	-0.16	-0.16	-0.16	-0.16	0.57	0.57	0.57	0.38	0.14	0.10
kVA_L2	1.00	1.00	1.00	0.58	1.00	1.00	1.00	1.00	-0.08	0.04	0.04	0.57	-0.16	-0.16	-0.16	-0.17	0.57	0.57	0.57	0.38	0.13	0.10
kVA_L3	1.00	1.00	1.00	0.58	1.00	1.00	1.00	1.00	-0.08	0.04	0.04	0.57	-0.17	-0.16	-0.16	-0.17	0.57	0.57	0.57	0.38	0.13	0.10
KVAr	-0.09	-0.08	-0.08	0.03	-0.09	-0.09	-0.08	-0.08	1.00	-0.01	-0.01	0.04	-0.01	-0.01	-0.01	-0.01	0.03	0.03	0.03	0.03	-0.04	0.00
Oil_Pressure	0.04	0.04	0.04	0.05	0.04	0.04	0.04	0.04	-0.01	1.00	0.73	0.03	0.10	0.10	0.10	0.10	0.05	0.05	0.05	0.05	0.16	0.32
Coolant_Temp	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	-0.01	0.73	1.00	0.02	0.09	0.09	0.09	0.09	0.04	0.04	0.04	0.03	0.24	0.31
Charger_Alt	0.58	0.58	0.58	0.99	0.58	0.57	0.57	0.57	0.04	0.03	0.02	1.00	0.00	-0.01	0.00	0.00	0.98	0.98	0.98	0.72	0.17	0.15

ernator																						
PF_Avg	-0.16	-0.16	-0.17	0.01	-0.16	-0.16	-0.16	-0.17	-0.01	0.10	0.09	0.00	1.00	0.97	0.99	0.98	0.03	0.02	0.02	0.04	0.26	0.06
PF_L1	-0.16	-0.16	-0.16	0.01	-0.16	-0.16	-0.16	-0.16	-0.01	0.10	0.09	-0.01	0.97	1.00	0.95	0.92	0.02	0.02	0.02	0.04	0.25	0.06
PF_L2	-0.16	-0.16	-0.16	0.01	-0.16	-0.16	-0.16	-0.16	-0.01	0.10	0.09	0.00	0.99	0.95	1.00	0.95	0.03	0.02	0.02	0.04	0.26	0.06
PF_L3	-0.16	-0.17	-0.17	0.02	-0.16	-0.16	-0.17	-0.17	-0.01	0.10	0.09	0.00	0.98	0.92	0.95	1.00	0.03	0.03	0.03	0.05	0.27	0.06
L1_N	0.57	0.57	0.57	0.99	0.58	0.57	0.57	0.57	0.03	0.05	0.04	0.98	0.03	0.02	0.03	0.03	1.00	1.00	1.00	0.73	0.20	0.15
L2_N	0.58	0.57	0.57	0.99	0.58	0.57	0.57	0.57	0.03	0.05	0.04	0.98	0.02	0.02	0.02	0.03	1.00	1.00	1.00	0.73	0.20	0.15
L3_N	0.58	0.57	0.57	0.99	0.58	0.57	0.57	0.57	0.03	0.05	0.04	0.98	0.02	0.02	0.02	0.03	1.00	1.00	1.00	0.73	0.19	0.15
Source_Ext_Voltage	0.39	0.38	0.38	0.73	0.39	0.38	0.38	0.38	0.03	0.05	0.03	0.72	0.04	0.04	0.04	0.05	0.73	0.73	0.73	1.00	0.19	0.16
ECU_Temp	0.13	0.13	0.13	0.20	0.13	0.14	0.13	0.13	-0.04	0.16	0.24	0.17	0.26	0.25	0.26	0.27	0.20	0.20	0.19	0.19	1.00	0.12
RPM	0.10	0.10	0.10	0.15	0.10	0.10	0.10	0.10	0.00	0.32	0.31	0.15	0.06	0.06	0.06	0.06	0.15	0.15	0.15	0.16	0.12	1.00

2. Kode persiapan dataset

```
# -*- coding: utf-8 -*-
"""Dataset_Processing.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1l0uZPBHJ61SIFB1CMqOwj2H5H86-GZTI

# 0) Preparation

## Load Requirements
"""

# Commented out IPython magic to ensure Python compatibility.
# Libraries
import sys
import pandas as pd
import hashlib
import matplotlib
import numpy as np
import sklearn
import scipy
import joblib
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from collections import Counter
from imblearn.over_sampling import SMOTE
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import normalize
from sklearn.decomposition import PCA
from sklearn.metrics import r2_score
from sklearn.cluster import DBSCAN
from sklearn.cluster import KMeans

#Extension
!pip install ipython-autotime
# %load_ext autotime

print('Load completed')

"""## Check Lib version"""
```

```
# Check the versions of libraries
# !!! WARNING !!!
# Important because model result may be different for other version

print('Python: {}'.format(sys.version))
print('scipy: {}'.format(scipy.__version__))
print('numpy: {}'.format(np.__version__))
print('matplotlib: {}'.format(matplotlib.__version__))
print('pandas: {}'.format(pd.__version__))
print('sklearn: {}'.format(sklearn.__version__))
print('joblib: {}'.format(joblib.__version__))

"""## Global Variables declaration"""

# Global Variable

#known hash value of file
compare = '10f68054068bc4ac4d7fee65e8fd7a184151e3ce3abce5d85350fc13c412d93b'
#Stored file path
raw_file = "/content/drive/MyDrive/Colab Notebooks/ML Resources/Raw Data/Format
csv/database.csv"
#environment path
model_dir = '/content/drive/MyDrive/Colab Notebooks/ML Resources/models/'
process_dir = '/content/drive/MyDrive/Colab Notebooks/ML Resources/Processed Data/'
#Column names for dataset
col_names = ['No', 'Datetime', 'Control_Mode', 'L1', 'L2', 'L3', 'Frequency', 'kVA_Total',
        'kVA_L1', 'kVA_L2', 'kVA_L3', 'KVAr', 'Oil_Pressure', 'Coolant_Temp',
        'Charger_Alternator', 'PF_Avg', 'PF_L1', 'PF_L2',
        'PF_L3', 'L1_N', 'L2_N', 'L3_N', 'Source_Ext_Voltage', 'ECU_Temp',
        'RPM', 'Train_code']
sensors = ['L1', 'L2', 'L3', 'Frequency', 'kVA_Total',
        'kVA_L1', 'kVA_L2', 'kVA_L3', 'KVAr', 'Oil_Pressure', 'Coolant_Temp',
        'Charger_Alternator', 'PF_Avg', 'PF_L1', 'PF_L2',
        'PF_L3', 'L1_N', 'L2_N', 'L3_N', 'Source_Ext_Voltage', 'ECU_Temp',
        'RPM']
features = ['Oil_Pressure', 'Coolant_Temp', 'PF_Avg', 'ECU_Temp']
df = pd.read_csv(raw_file, names=col_names)

#columns with numeric dtype for calculating z-score
numeric_col = ['Control_Mode', 'L1', 'L2', 'L3', 'Frequency', 'kVA_Total',
        'kVA_L1', 'kVA_L2', 'kVA_L3', 'KVAr', 'Oil_Pressure', 'Coolant_Temp',
        'Charger_Alternator', 'PF_Avg', 'PF_L1', 'PF_L2',
        'PF_L3', 'L1_N', 'L2_N', 'L3_N', 'Source_Ext_Voltage', 'ECU_Temp',
        'RPM']

"""## Define Function"""

#Function to show cluster result
def show_clusters(dataset, labels):
  df = pd.DataFrame(dict(x=X_transform['P1'], y=X_transform['P2'], label=labels))
  colors = {-1:'black', 0:'blue', 1:'red', 2:'yellow' ,3:'green', 4:'orange'}
  fig, ax = plt.subplots(figsize=(8,8))
  grouped = df.groupby('label')
```

```
  for key, group in grouped:
    group.plot(ax=ax, kind='scatter', x='x', y='y', label=key, color=colors[key])
  plt.xlabel('P1')
  plt.ylabel('P2')
  plt.title('Estimated number of clusters: %d' %len(set(labels)))
  plt.show()

# Function that returns the SHA-2 hash of the file
def hash_file(filepath):

    # make a hash object with SHA-2
    h = hashlib.sha256()

    # open file for reading in binary mode
    with open(filepath,'rb') as file:
        # loop till the end of the file
        chunk = 0
        while chunk != b'':
            # read only 1024 bytes at a time
            chunk = file.read(1024)
            h.update(chunk)

    # return the hex representation of digest
    return h.hexdigest()

# with the following function we can select highly correlated features
# it will remove the first feature that is correlated with anything other feature

def correlation(dataset, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff
value
                colname = corr_matrix.columns[i]  # getting the name of column
                col_corr.add(colname)
    return col_corr

"""# 1) Processing Dataset

### Hash Validation
"""

# Checking hash value of a file
compare = '10f68054068bc4ac4d7fee65e8fd7a184151e3ce3abce5d85350fc13c412d93b'
message = hash_file(raw_file)
print('SHA-256 value of your file is :')
print(message)

if (compare == message) :
  print('Hash check SUCCESS')
else :
```

```
  print('Hash check INVALID')

"""## Feature Selection"""

# Summarize raw dataset
pd.set_option('display.max_columns', None)
# size of (row, column) data
print('Total data size : ', df.shape)
# show sample entry of data
df.head(5)

# statistical value of data
df.describe()

#Using Pearson Correlation
plt.figure(figsize=(20,20), dpi=300)
cor = df[sensors].corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.CMRmap_r)
plt.show()

#export correlation table
cor.to_csv(process_dir+'pearson_corr.csv')

# class distribution
df.groupby('Train_code').size()

#Data cleaning
# fill nan
df = df.dropna()
#drop duplicates
df = df.drop_duplicates()
df

# convert Datetime Dtype
df['Datetime'] = pd.to_datetime(df.Datetime)

df.info()

# Feature selection

#Columns which are highly correlated
unused_col = correlation(df, 0.75)
unused_col
#Non-sensor columns in dataset
unused_col.add('No')
unused_col.add('Control_Mode')
#Columns which IQR value (25% - 75%) doesn't have many variations from
df.describe().
unused_col.add('RPM')
unused_col.add('Source_Ext_Voltage')
unused_col.add('Charger_Alternator')
unused_col.add('Frequency')
```

```
# delete columns with condition
df.drop(labels=unused_col, axis=1, inplace=True)
df

# statistical value of data
df.describe()

# visualisasi dalam grafik
df.hist()
pyplot.show()

"""## Data Processing"""

#Threshold value variable 25% min, 75% max
oil_q25, oil_q75 = df.Oil_Pressure.quantile([0.25 , 0.75])
coolant_q25, coolant_q75 = df.Coolant_Temp.quantile([0.25 , 0.75])
pf_q25, pf_q75 = df.PF_Avg.quantile([0.25 , 0.75])
ecu_q25, ecu_q75 = df.ECU_Temp.quantile([0.25 , 0.75])
# calculate the outlier cutoff per-feature
#feature 1
oil_cut_off = (oil_q75 - oil_q25)*1.5
oil_lower, oil_upper = oil_q25 - oil_cut_off, oil_q75 + oil_cut_off
#feature 2
coolant_cut_off = (coolant_q75 - coolant_q25)*1.5
coolant_lower, coolant_upper = coolant_q25 - coolant_cut_off, coolant_q75 +
coolant_cut_off
#feature 3
pf_cut_off = (pf_q75 - pf_q25)*1.5
pf_lower, pf_upper = pf_q25 - pf_cut_off, pf_q75 + pf_cut_off
#feature 4
ecu_cut_off = (ecu_q75 - ecu_q25)*1.5
ecu_lower, ecu_upper = ecu_q25 - ecu_cut_off, ecu_q75 + ecu_cut_off

q25 = df.quantile(q=0.25)
q75 = df.quantile(q=0.75)
iqr = (q75 - q25)
cut_off = iqr*1.5
lower, upper  = (q25 - cut_off), (q75 + cut_off)

#Print quantile value for label threshold
print('1st Quantile values (25%) :')
print(q25)
print('3rd Quantile values (75%) :')
print(q75)
print('IQR values (75%-25%) :')
print(iqr)
print('Lower Threshold values :')
print(lower)
print('Upper Threshold values :')
print(upper)

#amount of Outlier data for each feature
Outlier_data = df[(df[features] < lower) | (df[features] > upper)].count()
```

```
print('Outlier data Count :')
print(Outlier_data)
#amount of Maintenance data for each feature
Maintenance_data = df[((df[features] > lower) & (df[features] < q25)) | ((df[features] >
q75) & (df[features] < upper))]
print('Maintenance data Count :')
print(Maintenance_data.count())
#amount of Normal data for each feature
Normal_data = df[(df[features] > q25) & (df[features] < q75)]
print('Normal data Count :')
print(Normal_data.count())

# Scaling and Normalize data
features_norm = ['Oil_Pressure_Norm',  'Coolant_Temp_Norm', 'PF_Avg_Norm',
          'ECU_Temp_Norm']
X_scaled = StandardScaler().fit_transform(df[features])        #Bring all the attributes to a
comparable level (Z-score normalization)
X_normalized = normalize(X_scaled)                             #Centering data to Origin
X_normalized = pd.DataFrame(X_normalized, columns=features_norm)     #Converting
the numpy array into a pandas DataFrame
X_normalized

X_normalized.plot.kde()  #Density estimation
X_normalized.hist()      # Visualize data using histograms

"""### Data Labelling"""

#Stat_Outlier = (value < q25 - (q75-q25) * 1.5) or (value > q75 + (q75-q25) * 1.5)
Outlier_Oil = (df['Oil_Pressure'] < oil_lower) | (df['Oil_Pressure'] > oil_upper)
Outlier_Coolant =(df['Coolant_Temp'] < coolant_lower) | (df['Coolant_Temp'] >
coolant_upper)
Outlier_ECU = (df['ECU_Temp'] < ecu_lower) | (df['ECU_Temp'] > ecu_upper)
Outlier_PFA = (df['PF_Avg'] < pf_lower) | (df['PF_Avg'] > pf_upper)

#Stat_Maintenance = (q25 - (q75-q25) * 1.5 <= value <= q25) or (q75 <= value <= q75 +
(q75-q25) * 1.5)
Maintenance_Oil = (df['Oil_Pressure'].between(oil_lower, oil_q25)) |
(df['Oil_Pressure'].between(oil_q75, oil_upper))
Maintenance_Coolant = (df['Coolant_Temp'].between(coolant_lower, coolant_q25)) |
(df['Coolant_Temp'].between(coolant_q75, coolant_upper))
Maintenance_ECU = (df['ECU_Temp'].between(ecu_lower, ecu_q25)) |
(df['ECU_Temp'].between(ecu_q75, ecu_upper))
Maintenance_PFA = (df['PF_Avg'].between(pf_lower, pf_q25) |
(df['PF_Avg'].between(pf_q75, pf_upper)))

#Stat_Normal = q25 < value < q75
Normal_Oil = df['Oil_Pressure'].between(oil_q25, oil_q75, inclusive=False)
Normal_Coolant = df['Coolant_Temp'].between(coolant_q25, coolant_q75,
inclusive=False)
Normal_ECU = df['ECU_Temp'].between(ecu_q25, ecu_q75, inclusive=False)
Normal_PFA = df['PF_Avg'].between(pf_q25, pf_q75, inclusive=False)

#Label Condition
```

```
Outlier =  Outlier_Oil & Outlier_Coolant & Outlier_ECU & Outlier_PFA
Maintenance = Maintenance_Oil | Maintenance_Coolant |  Maintenance_ECU |
Maintenance_PFA

#Check conditions for each row in dataframe, preserve original DataFrame from raw data
#If ALL Features fulfill Outlier condition then Label 'Outlier'
#If ANY Features fulfill Maintenance condition then Label 'Maintenance'
#else Label 'Normal'
df_dataset1 = df[features].copy()
df_dataset1['Label'] = 'Normal'
df_dataset1.loc[Outlier, 'Label'] = 'Outlier'
df_dataset1.loc[Maintenance, 'Label'] = 'Maintenance'

#add label to original data
df_dataset1 = pd.concat([df, df_dataset1.Label], axis=1, join='outer')

#Dataset 1 (With 3 Label: Outlier, Maintenance, Normal)
# class distribution
print('Dataset 1 label:')
print(df_dataset1.groupby('Label').size())

# Dataset 2 (With 2 Label: Maintenance, Normal) can be obtained from previous dataset
# No need to create separate file.
#Dataset 2
df_dataset2 = df_dataset1[df_dataset1.Label != 'Outlier']
print('Dataset 2 no Outlier :')
print(df_dataset2.groupby('Label').size())

df_dataset1

"""### Clustering Data"""

# reduce dimension for easier visualization using Principal Component Analysis (PCA)

# set n_components value based on the numbers of features, then select two with highest
variation
pca = PCA(n_components = 4)
pca.fit(X_normalized)

# Scree plot
PC_values = np.arange(pca.n_components_) + 1
plt.bar(x=PC_values, height=pca.explained_variance_ratio_, color='blue')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Variance Explained')
plt.show()
print('PCA Explained Variance ratio in order : ')
print(pca.explained_variance_ratio_)

# Scatter plot data
PCA_comp = ['P1', 'P2', 'P3', 'P4'] #set Px value based on the numbers of features
X_transform = pca.fit_transform(X_normalized)
X_transform = pd.DataFrame(X_transform, columns=PCA_comp)
```

```
print(X_transform.head(10))

# Select two PCA components with highest variation (Dimensionality reduction)
df_cluster = pd.DataFrame(dict(x=X_transform['P1'], y=X_transform['P2']))
fig, ax = plt.subplots(figsize=(8,8))
df_cluster.plot(ax=ax, kind='scatter', x='x', y='y')
plt.xlabel('P1')
plt.ylabel('P2')
plt.show()
print(df_cluster.head(10))

#Reset index to match new data
df_dataset1 = df_dataset1.set_index('Train_code')   #set new index to change the
numbering order
df_dataset1 = df_dataset1.reset_index()            #reset index to restart numbering from 0
to n-1

#Merge Transformed features value into dataset
df_dataset1 = pd.concat([df_dataset1, X_transform], axis=1)
df_dataset1

# Fit Clustering algorithm

# Build DBSCAN Model
model_dbscan = DBSCAN(eps=0.5, min_samples=100)
model_dbscan.fit(df_cluster)
labels_dbscan = model_dbscan.labels_
# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels_dbscan)) - (1 if -1 in labels_dbscan else 0)
n_noise_ = list(labels_dbscan).count(-1)
print('Estimated number of dbscan clusters: %d' %n_clusters_)
print('Estimated number of noise points: %d' %n_noise_)

# cluster result from DBSCAN
show_clusters(df_cluster, labels_dbscan)

# Build K means model
model_Kmeans = KMeans(n_clusters=3, random_state=0)
model_Kmeans.fit(df_cluster)
labels_Kmeans = model_Kmeans.labels_
print('Estimated number of kmeans clusters: ', len(set(labels_Kmeans)))
# cluster result from K-Means
show_clusters(df_cluster, labels_Kmeans)

# Compare with Statistical label result from dataset 1

# Convert string label (From statistical labeling) to integer
df_dataset1.insert(7, 'Label_id', value=np.nan)
df_dataset1.loc[df_dataset1['Label'] == 'Normal', ['Label_id']] = 0       #Normal = 0
df_dataset1.loc[df_dataset1['Label'] == 'Outlier', ['Label_id']] = 1      #Outlier = 1
df_dataset1.loc[df_dataset1['Label'] == 'Maintenance', ['Label_id']] = 2  #Maintenance =
2
```

```
# setup the plot
fig, ax = plt.subplots(1,1, figsize=(6,6))
# define the data
x = X_transform['P1']
y = X_transform['P2']
N = 3 # Number of labels
tag = df_dataset1['Label_id'] # Tag each point with a corresponding label

# define the colormap
cmap = plt.cm.jet
# extract all colors from the .jet map
cmaplist = [cmap(i) for i in range(cmap.N)]
# create the new map
cmap = cmap.from_list('Custom cmap', cmaplist, cmap.N)

# define the bins and normalize
bounds = np.linspace(0,N,N+1)
norm = matplotlib.colors.BoundaryNorm(bounds, cmap.N)

# make the scatter
scat = ax.scatter(x,y,c=tag,cmap=cmap, norm=norm)
# create the colorbar
cb = plt.colorbar(scat, spacing='proportional',ticks=bounds)
cb.set_label('Label Color')
plt.xlabel('P1')
plt.ylabel('P2')
ax.set_title('Statistical label result')
plt.show()

# add predicted cluster into dataset

#DBSCAN
df_dataset1.insert(8, 'clusters_dbscan', value=labels_dbscan)
print(df_dataset1.groupby('clusters_dbscan').size())

#Kmeans
df_dataset1.insert(8, 'clusters_kmeans', value=labels_Kmeans)
print(df_dataset1.groupby('clusters_kmeans').size())

df_dataset1

df_dataset1.describe()

#Export dataset into external file

df_dataset1.to_csv(process_dir+'final_dataset.csv')

"""### Synthetic Data
Using SMOTE, resample all data for each label.

"""

# resampling & balancing dataset from statistical label with SMOTE
```

```
resampled_col = ['Oil_Pressure', 'Coolant_Temp', 'PF_Avg', 'ECU_Temp',
                 'P1',   'P2',      'P3',      'P4']

#Synthetic data generator
resampling = {'Outlier': 30000, 'Maintenance': 30000, 'Normal': 30000}
resampler_smote = SMOTE(random_state=1, sampling_strategy = resampling)
X1 = df_dataset1[resampled_col]
y1 = df_dataset1['Label']

print('Original dataset shape %s' % Counter(y1))
X1_test, Y1_test = resampler_smote.fit_resample(X1, y1)
df_X1_test = pd.DataFrame(X1_test, columns=resampled_col)
df_Y1_test = pd.DataFrame(Y1_test, columns=['Label'])
print('Resampled dataset shape %s' % Counter(Y1_test))
df_SMOTE_statistical = pd.concat([df_X1_test, df_Y1_test], axis=1)
print(df_SMOTE_statistical)

#export resampled df to csv
df_SMOTE_statistical.to_csv(process_dir+'SMOTE_statistical.csv')

# Show label result from statistical label after using SMOTE

# Convert string label (From statistical labeling) to integer
df_SMOTE_statistical.insert(7, 'Label_id', value=np.nan)
df_SMOTE_statistical.loc[df_SMOTE_statistical['Label'] == 'Normal', ['Label_id']] = 0
#Normal = 0
df_SMOTE_statistical.loc[df_SMOTE_statistical['Label'] == 'Outlier', ['Label_id']] = 1
#Outlier = 1
df_SMOTE_statistical.loc[df_SMOTE_statistical['Label'] == 'Maintenance', ['Label_id']]
= 2  #Maintenance = 2

# setup the plot
fig, ax = plt.subplots(1,1, figsize=(6,6))
# define the data
x = df_SMOTE_statistical['P1']
y = df_SMOTE_statistical['P2']
N = 3 # Number of labels
tag = df_SMOTE_statistical['Label_id'] # Tag each point with a corresponding label

# define the colormap
cmap = plt.cm.jet
# extract all colors from the .jet map
cmaplist = [cmap(i) for i in range(cmap.N)]
# create the new map
cmap = cmap.from_list('Custom cmap', cmaplist, cmap.N)

# define the bins and normalize
bounds = np.linspace(0,N,N+1)
norm = matplotlib.colors.BoundaryNorm(bounds, cmap.N)

# make the scatter
scat = ax.scatter(x,y,c=tag,cmap=cmap, norm=norm)
```

```
# create the colorbar
cb = plt.colorbar(scat, spacing='proportional',ticks=bounds)
cb.set_label('Label Color')
plt.xlabel('P1')
plt.ylabel('P2')
ax.set_title('Statistical label SMOTE resampling result')
plt.show()

# resampling & balancing dataset from clustering label with SMOTE
resampled_col = ['Oil_Pressure', 'Coolant_Temp', 'PF_Avg', 'ECU_Temp',
                 'P1',   'P2',      'P3',      'P4']

#Synthetic data generator
resampling = {0: 30000, 1: 30000, 2: 30000}
resampler_smote = SMOTE(random_state=1, sampling_strategy = resampling)
X2 = df_dataset1[resampled_col]
y2 = df_dataset1['clusters_dbscan']

print('Original dataset shape %s' % Counter(y2))
X2_test, Y2_test = resampler_smote.fit_resample(X2, y2)
df_X2_test = pd.DataFrame(X2_test, columns=resampled_col)
df_Y2_test = pd.DataFrame(Y2_test, columns=['clusters_dbscan'])
print('Resampled dataset shape %s' % Counter(Y2_test))
df_SMOTE_cluster = pd.concat([df_X2_test, df_Y2_test], axis=1)
print(df_SMOTE_cluster)

#export resampled df to csv
df_SMOTE_cluster.to_csv(process_dir+'SMOTE_cluster.csv')

# Show cluster result from DBSCAN after using SMOTE

# setup the plot
fig, ax = plt.subplots(1,1, figsize=(6,6))
# define the data
x = df_SMOTE_cluster['P1']
y = df_SMOTE_cluster['P2']
N = 3 # Number of labels
tag = df_SMOTE_cluster['clusters_dbscan'] # Tag each point with a corresponding label

# define the colormap
cmap = plt.cm.jet
# extract all colors from the .jet map
cmaplist = [cmap(i) for i in range(cmap.N)]
# create the new map
cmap = cmap.from_list('Custom cmap', cmaplist, cmap.N)

# define the bins and normalize
bounds = np.linspace(0,N,N+1)
norm = matplotlib.colors.BoundaryNorm(bounds, cmap.N)

# make the scatter
scat = ax.scatter(x,y,c=tag,cmap=cmap, norm=norm)
# create the colorbar
```

```
cb = plt.colorbar(scat, spacing='proportional',ticks=bounds)
cb.set_label('Label Color')
plt.xlabel('P1')
plt.ylabel('P2')
ax.set_title('Cluster SMOTE resampling label result')
plt.show()
```

3. Kode perbandingan metode klasifikasi

```
# -*- coding: utf-8 -*-
"""Classifier Comparison.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1ETRIM9DmfKLaSDZv_bCeI2-gBbvI2hj8

# Load Requirements
"""

# Commented out IPython magic to ensure Python compatibility.
# Libraries
import sys
import pandas as pd
import hashlib
import matplotlib
import numpy as np
import sklearn
import scipy
import joblib
from matplotlib import pyplot
from collections import Counter
from imblearn.over_sampling import SMOTE
from pandas import read_csv
from pandas.plotting import scatter_matrix
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import accuracy_score
from sklearn.metrics import balanced_accuracy_score
from sklearn.metrics import plot_confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.cluster import DBSCAN
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import normalize
from sklearn.decomposition import PCA
from timeit import default_timer as timer

#Extension
!pip install ipython-autotime
```

```
# %load_ext autotime

print('Load completed')

"""# Check Lib Version"""

# Check the versions of libraries
# !!! WARNING !!!
# Important because model result may be different for other version

print('Python: {}'.format(sys.version))
print('scipy: {}'.format(scipy.__version__))
print('numpy: {}'.format(np.__version__))
print('matplotlib: {}'.format(matplotlib.__version__))
print('pandas: {}'.format(pd.__version__))
print('sklearn: {}'.format(sklearn.__version__))
print('joblib: {}'.format(joblib.__version__))

"""# Global Variables Declaration"""

# Global Variable

#Stored file path
raw_file = "/content/drive/MyDrive/Colab Notebooks/ML Resources/Raw Data/Format
csv/database.csv"
#environment path
model_dir = '/content/drive/MyDrive/Colab Notebooks/ML Resources/models/'
process_dir = '/content/drive/MyDrive/Colab Notebooks/ML Resources/Processed Data/'
#Column names for dataset
features = ['Oil_Pressure', 'Coolant_Temp', 'PF_Avg', 'ECU_Temp']
resampled_col = ['Oil_Pressure', 'Coolant_Temp', 'PF_Avg', 'ECU_Temp',
                 'P1',   'P2',       'P3',       'P4']
#Synthetic data generator
balancing_smote = SMOTE(random_state=1)

"""# Hash Check Function"""

# Function that returns the SHA-2 hash of the file
def hash_file(filepath):

  # make a hash object with SHA-2
  h = hashlib.sha256()

  # open file for reading in binary mode
  with open(filepath,'rb') as file:
     # loop till the end of the file
     chunk = 0
     while chunk != b'':
        # read only 1024 bytes at a time
        chunk = file.read(1024)
        h.update(chunk)

  # return the hex representation of digest
```

```
  return h.hexdigest()

"""# Prepare Dataset

## Hash Check
"""

# Checking hash value of a file
message = hash_file(process_dir+'final_dataset.csv')
print('SHA-256 value of your file is :')
print(message)

"""## Process Dataset

### Dataset Statistical Label
"""

# Dataset for 3 label from statistical labelling (Outlier, Normal, Maintenance)

#change dataset source to 'SMOTE_statistical.csv' for resampled data, OR
'final_dataset.csv' for original
df_statistical = pd.read_csv(process_dir+'final_dataset.csv', index_col=0)
df_statistical

#split data for training and testing
X1 = df_statistical[resampled_col]
y1 = df_statistical['Label']
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.1,
random_state=1, stratify = y1)

#Training dataset 1 summary
# data size
print('Training dataset 1 size : ', X1_train.shape)
print(' ')
print('Testing dataset 1 size : ', X1_test.shape)
print(' ')
# statistical value of data
print(X1_train.describe())

"""### Dataset Clustering"""

# Dataset3 for 3 label from DBSCAN clustering (0,1,2)

#change dataset source to 'SMOTE_cluster.csv' for resampled data, OR 'final_dataset.csv'
for original
df_clustering = pd.read_csv(process_dir+'final_dataset.csv', index_col=0)
df_clustering

#split data for training and testing
X3 = df_clustering[resampled_col]
y3 = df_clustering['clusters_dbscan']
X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y3, test_size=0.1,
random_state=1, stratify = y3)
```

```
#Training dataset 3 summary
# data size
print('Training dataset 3 size : ', X3_train.shape)
print(' ')
print('Testing dataset 3 size : ', X3_test.shape)
print(' ')
# statistical value of data
print(df_clustering.describe())

"""# Classifier Result

## Rule-based

### Dataset Statistical Label

Training Dataset
"""

# calculate the outlier cutoff per-feature (value change depends on dataset used)

#Threshold value variable 25% min, 75% max
oil_q25, oil_q75 = X1_train.Oil_Pressure.quantile([0.25 , 0.75])
coolant_q25, coolant_q75 = X1_train.Coolant_Temp.quantile([0.25 , 0.75])
pf_q25, pf_q75 = X1_train.PF_Avg.quantile([0.25 , 0.75])
ecu_q25, ecu_q75 = X1_train.ECU_Temp.quantile([0.25 , 0.75])
#feature 1
oil_cut_off = (oil_q75 - oil_q25)*1.5
oil_lower, oil_upper = oil_q25 - oil_cut_off, oil_q75 + oil_cut_off
#feature 2
coolant_cut_off = (coolant_q75 - coolant_q25)*1.5
coolant_lower, coolant_upper = coolant_q25 - coolant_cut_off, coolant_q75 +
coolant_cut_off
#feature 3
pf_cut_off = (pf_q75 - pf_q25)*1.5
pf_lower, pf_upper = pf_q25 - pf_cut_off, pf_q75 + pf_cut_off
#feature 4
ecu_cut_off = (ecu_q75 - ecu_q25)*1.5
ecu_lower, ecu_upper = ecu_q25 - ecu_cut_off, ecu_q75 + ecu_cut_off

#Print quantile value for label threshold
print('Threshold values (25%, 75%) :')
print('Oil_Pressure  = ', [oil_q25, oil_q75])
print('Coolant_Temp  = ', [coolant_q25, coolant_q75])
print('PF_Avg        = ', [pf_q25, pf_q75])
print('ECU_Temp      = ', [ecu_q25, ecu_q75])
print('')
print('Lower & Upper Threshold values :')
print('Oil_Pressure  = ', [oil_lower, oil_upper])
print('Coolant_Temp  = ', [coolant_lower,coolant_upper])
print('PF_Avg        = ', [pf_lower, pf_upper])
print('ECU_Temp      = ', [ecu_lower, ecu_upper])
```

```
#Change df_dataset if using other source

#Stat_Outlier = (value < q25 - (q75-q25) * 1.5) or (value > q75 + (q75-q25) * 1.5)
Outlier_Oil = (df_statistical['Oil_Pressure'] < oil_lower) | (df_statistical['Oil_Pressure'] > oil_upper)
Outlier_Coolant =(df_statistical['Coolant_Temp'] < coolant_lower) | (df_statistical['Coolant_Temp'] > coolant_upper)
Outlier_ECU = (df_statistical['ECU_Temp'] < ecu_lower) | (df_statistical['ECU_Temp'] > ecu_upper)
Outlier_PFA = (df_statistical['PF_Avg'] < pf_lower) | (df_statistical['PF_Avg'] > pf_upper)

#Stat_Maintenance = (q25 - (q75-q25) * 1.5 <= value <= q25) or (q75 <= value <= q75 + (q75-q25) * 1.5)
Maintenance_Oil = (df_statistical['Oil_Pressure'].between(oil_lower, oil_q25)) | (df_statistical['Oil_Pressure'].between(oil_q75, oil_upper))
Maintenance_Coolant = (df_statistical['Coolant_Temp'].between(coolant_lower, coolant_q25)) | (df_statistical['Coolant_Temp'].between(coolant_q75, coolant_upper))
Maintenance_ECU = (df_statistical['ECU_Temp'].between(ecu_lower, ecu_q25)) | (df_statistical['ECU_Temp'].between(ecu_q75, ecu_upper))
Maintenance_PFA = (df_statistical['PF_Avg'].between(pf_lower, pf_q25) | (df_statistical['PF_Avg'].between(pf_q75, pf_upper)))

#Stat_Normal = q25 < value < q75
Normal_Oil = df_statistical['Oil_Pressure'].between(oil_q25, oil_q75, inclusive=False)
Normal_Coolant = df_statistical['Coolant_Temp'].between(coolant_q25, coolant_q75, inclusive=False)
Normal_ECU = df_statistical['ECU_Temp'].between(ecu_q25, ecu_q75, inclusive=False)
Normal_PFA = df_statistical['PF_Avg'].between(pf_q25, pf_q75, inclusive=False)

#Label Condition
Outlier =  Outlier_Oil & Outlier_Coolant & Outlier_ECU & Outlier_PFA
Maintenance = Maintenance_Oil | Maintenance_Coolant |  Maintenance_ECU | Maintenance_PFA

#Check conditions for each row in dataframe

df_RBClassifier = pd.DataFrame(X1_train)
df_RBClassifier['Label'] = 'Normal'
df_RBClassifier.loc[Outlier, 'Label'] = 'Outlier'
df_RBClassifier.loc[Maintenance, 'Label'] = 'Maintenance'

# class distribution
print(df_RBClassifier.groupby('Label').size())

#Check performance of Rule-based Classifier
y_true = y1_train
y_pred = df_RBClassifier['Label']
ConfusionMatrixDisplay.from_predictions(y_true, y_pred)
print(classification_report(y_true, y_pred))

"""Testing Dataset"""
```

```
#Check conditions for each row in dataframe

df_test_rbc = pd.DataFrame(X1_test)
df_test_rbc['Label'] = 'Normal'
df_test_rbc.loc[Outlier, 'Label'] = 'Outlier'
df_test_rbc.loc[Maintenance, 'Label'] = 'Maintenance'

# class distribution
print(df_test_rbc.groupby('Label').size())

#Check performance of Rule-based Classifier
y_true = y1_test
y_pred = df_test_rbc['Label']
ConfusionMatrixDisplay.from_predictions(y_true, y_pred)
print(classification_report(y_true, y_pred))

"""### Dataset Clustering

Training Dataset
"""

#Label Condition based on P1 and P2 Component from PCA result and clusters_dbscan

#cluster_0 = P1 > 0.25 AND P2 < 0.2
cluster_0 = (X3_train['P1'] > 0.25) & (X3_train['P2'] < 0.2)
#cluster_1 = P1 > 0.25 AND P2 > 0.2
cluster_1 = (X3_train['P1'] > 0.25) & (X3_train['P2'] > 0.2)
#cluster_2 = P1 <= 0.25 AND P2 <= 0.2
cluster_2 = (X3_train['P1'] <= 0.25) & (X3_train['P2'] <= 0.2)

#Add label to dataset
df_RBClassifier = pd.DataFrame(X3_train)
df_RBClassifier['Label_cluster'] = np.nan
df_RBClassifier.loc[cluster_0, 'Label_cluster'] = 0
df_RBClassifier.loc[cluster_1, 'Label_cluster'] = 1
df_RBClassifier.loc[cluster_2, 'Label_cluster'] = 2

# Label distribution result
print(df_RBClassifier.groupby('Label_cluster').size())

#Check performance of Rule-based Classifier
y_true = y3_train
y_pred = df_RBClassifier['Label_cluster']
ConfusionMatrixDisplay.from_predictions(y_true, y_pred)
print(classification_report(y_true, y_pred))

"""Testing Dataset"""

#Label Condition based on P1 and P2 Component from PCA result and clusters_dbscan

#cluster_0 = P1 > 0.25 AND P2 < 0.2
cluster_0 = (X3_test['P1'] > 0.25) & (X3_test['P2'] < 0.2)
#cluster_1 = P1 > 0.25 AND P2 > 0.2
```

```
cluster_1 = (X3_test['P1'] > 0.25) & (X3_test['P2'] > 0.2)
#cluster_2 = P1 <= 0.25 AND P2 <= 0.2
cluster_2 = (X3_test['P1'] <= 0.25) & (X3_test['P2'] <= 0.2)

#Add label to dataset
df_test_rbc = pd.DataFrame(X3_test)
df_test_rbc['Label_cluster'] = np.nan
df_test_rbc.loc[cluster_0, 'Label_cluster'] = 0
df_test_rbc.loc[cluster_1, 'Label_cluster'] = 1
df_test_rbc.loc[cluster_2, 'Label_cluster'] = 2

# Label distribution result
print(df_test_rbc.groupby('Label_cluster').size())

#Check performance of Rule-based Classifier
y_true = y3_test
y_pred = df_test_rbc['Label_cluster']
ConfusionMatrixDisplay.from_predictions(y_true, y_pred)
print(classification_report(y_true, y_pred))

"""## ML Algorithm"""

# ML Algorithm considered

model_LR = LogisticRegression(max_iter=1000)
model_KNN = KNeighborsClassifier()
model_DT = DecisionTreeClassifier()
model_SVM = SVC(gamma='auto')
model_RF = RandomForestClassifier()
model_LDA = LinearDiscriminantAnalysis()
model_NB = GaussianNB()
########################################################################
# save into a list for easy training & cross validation
models = []
models.append(('LR', model_LR))
models.append(('KNN', model_KNN))
models.append(('DT', model_DT))
models.append(('SVM', model_SVM))
models.append(('RF', model_RF))
models.append(('LDA', model_LDA))
models.append(('NB', model_NB))

"""### Dataset Statistical Label

Training Results
"""

#Loop for training ML model
fitted_models1 = []
training_results1 = []
names = []

for name, model in models:
```

```
  start = timer() #Start measure Runtime
  fitting = model.fit(df_statistical[features], df_statistical['Label']) #Fit model with dataset
  fitted_models1.append(fitting)
  #Show metrics to evaluate performance
  pred = model.predict(df_statistical[features])
  acc = balanced_accuracy_score(df_statistical['Label'], pred)
  training_results1.append(acc)
  names.append(name)
  print('%s: %f' % (name, acc))
  ConfusionMatrixDisplay.from_predictions(df_statistical['Label'], pred)
  pyplot.title('%s' %name)
  joblib.dump(model,model_dir+'Opsi1_%s'%name) #export to external files
  end = timer() #End measure Runtime
  print('Runtime (s) :', (end - start))

# Compare result
pyplot.bar(x=names, height=training_results1)
pyplot.title('Algorithm Training Accuracy Comparison 1')
pyplot.show()

"""Testing (CV) Results"""

# Loop for Cross validation of each model in turn
cv_results1 = []

#change x(features), y(labels) value in cross_val_score if using different dataset
for name, model in models:
 start = timer()
 kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
 cv_scores = cross_val_score(model, df_statistical[features], df_statistical['Label'],
cv=kfold,
                             scoring='balanced_accuracy')
 cv_results1.append(cv_scores)
 print('%s: %f (%f)' % (name, cv_scores.mean(), cv_scores.std()))
 end = timer()
 print('Avg Runtime (s) :', (end - start)/10)

# Compare result
pyplot.boxplot(cv_results1, labels=names)
pyplot.title('Algorithm CV Accuracy Comparison 1')
pyplot.show()

"""### Dataset Clustering

Training Results
"""

#Loop for training ML model
fitted_models3 = []
training_results3 = []
names = []

for name, model in models:
```

```
  start = timer() #Start measure Runtime
  fitting = model.fit(df_clustering[features], df_clustering['clusters_dbscan']) #Fit model
with dataset
  fitted_models3.append(fitting)
  #Show metrics to evaluate performance
  pred = model.predict(df_clustering[features])
  acc = balanced_accuracy_score(df_clustering['clusters_dbscan'], pred)
  training_results3.append(acc)
  names.append(name)
  print('%s: %f' % (name, acc))
  ConfusionMatrixDisplay.from_predictions(df_clustering['clusters_dbscan'], pred)
  pyplot.title('%s' %name)
  joblib.dump(model,model_dir+'Opsi3_%s'%name) #export to external files
  end = timer() #End measure Runtime
  print('Runtime (s) :', (end - start))

# Compare result
pyplot.bar(x=names, height=training_results3)
pyplot.title('Algorithm Training Accuracy Comparison 3')
pyplot.show()

"""Testing (CV) Results"""

# Loop for Cross validation of each model in turn
cv_results3 = []

#change x(features), y(labels) value in cross_val_score if using different dataset
for name, model in models:
 start = timer()
 kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
 cv_scores = cross_val_score(model, df_clustering[features],
df_clustering['clusters_dbscan'], cv=kfold,
                             scoring='balanced_accuracy')
 cv_results3.append(cv_scores)
 print('%s: %f (%f)' % (name, cv_scores.mean(), cv_scores.std()))
 end = timer()
 print('Avg Runtime (s) :', (end - start)/10)

# Compare result
pyplot.boxplot(cv_results3, labels=names)
pyplot.title('Algorithm Accuracy Comparison 3')
pyplot.show()
```

4. Kode pengujian Runtime model Klasifikasi

```
# -*- coding: utf-8 -*-
"""Runtime Evaluation Test.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1Y5nNDeFBFpGgMBE6QC7_YD0c6DNxKS5L

# 0) Preparation

## Load Requirements
"""

# Commented out IPython magic to ensure Python compatibility.
# Libraries
import sys
import numpy as np
import pandas as pd
import sklearn
import joblib
import hashlib
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from timeit import default_timer as timer
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import normalize
from sklearn.decomposition import PCA


#Extension
!pip install ipython-autotime
# %load_ext autotime

print('Load completed')

# Check the versions of libraries
# !!! WARNING !!!
# Important because model result may be different for other version

print('Python: {}'.format(sys.version))
print('numpy: {}'.format(np.__version__))
print('pandas: {}'.format(pd.__version__))
print('sklearn: {}'.format(sklearn.__version__))
```

```
print('joblib: {}'.format(joblib.__version__))

"""## Global Variables"""

#Global variables declaration

#Stored file path
raw_file = "/content/drive/MyDrive/Colab Notebooks/ML Resources/Raw Data/Format
csv/database.csv"
#environment path
model_dir = '/content/drive/MyDrive/Colab Notebooks/ML Resources/models/'
process_dir = '/content/drive/MyDrive/Colab Notebooks/ML Resources/Processed Data/'
#Column names for dataset
col_names = ['No', 'Datetime', 'Control_Mode', 'L1', 'L2', 'L3', 'Frequency', 'kVA_Total',
          'kVA_L1', 'kVA_L2', 'kVA_L3', 'KVAr', 'Oil_Pressure', 'Coolant_Temp',
          'Charger_Alternator', 'PF_Avg', 'PF_L1', 'PF_L2',
          'PF_L3', 'L1_N', 'L2_N', 'L3_N', 'Source_Ext_Voltage', 'ECU_Temp',
          'RPM', 'Train_code']
features = ['Oil_Pressure', 'Coolant_Temp', 'PF_Avg', 'ECU_Temp']
#Unused columns in dataset
unused_col = ['No', 'Datetime', 'Control_Mode', 'L1', 'L2', 'L3', 'Frequency',
           'kVA_Total', 'kVA_L1', 'kVA_L2', 'kVA_L3', 'KVAr', 'Charger_Alternator',
           'RPM', 'PF_L1', 'PF_L2', 'PF_L3', 'L1_N','L2_N', 'L3_N',
           'Source_Ext_Voltage', 'Train_code']

"""## Function Declaration"""

# Function that returns the SHA-2 hash of the file
def hash_file(filepath):

  # make a hash object with SHA-2
  h = hashlib.sha256()

  # open file for reading in binary mode
  with open(filepath,'rb') as file:
     # loop till the end of the file
     chunk = 0
     while chunk != b'':
        # read only 1024 bytes at a time
        chunk = file.read(1024)
        h.update(chunk)

  # return the hex representation of digest
  return h.hexdigest()

"""# 1) Prediction Test

## Generate Random Data
"""

#Runtime testing dataset
#Generate n samples for each features uniformly at range of min(0) to max(upper+x)
```

```
s_oil = np.random.uniform(low=0.0, high=700.0, size=3000000)
s_coolant = np.random.uniform(low=0.0, high=100.0, size=3000000)
s_pf = np.random.uniform(low=0.0, high=1.5, size=3000000)
s_ecu = np.random.uniform(low=0.0, high=75.0, size=3000000)

df_random = pd.DataFrame({'Oil_Pressure':s_oil, 'Coolant_Temp':s_coolant,
                          'PF_Avg':s_pf, 'ECU_Temp':s_ecu})
df_random

"""## Runtime test

### ML model
"""

#Import and load Model
Opsi1_LR = joblib.load(model_dir+'Opsi1_LR')
Opsi1_KNN = joblib.load(model_dir+'Opsi1_KNN')
Opsi1_DT = joblib.load(model_dir+'Opsi1_DT')
Opsi1_SVM = joblib.load(model_dir+'Opsi1_SVM')
Opsi1_RF = joblib.load(model_dir+'Opsi1_RF')
Opsi1_LDA = joblib.load(model_dir+'Opsi1_LDA')
Opsi1_NB = joblib.load(model_dir+'Opsi1_NB')
Opsi3_LR = joblib.load(model_dir+'Opsi3_LR')
Opsi3_KNN = joblib.load(model_dir+'Opsi3_KNN')
Opsi3_DT = joblib.load(model_dir+'Opsi3_DT')
Opsi3_SVM = joblib.load(model_dir+'Opsi3_SVM')
Opsi3_RF = joblib.load(model_dir+'Opsi3_RF')
Opsi3_LDA = joblib.load(model_dir+'Opsi3_LDA')
Opsi3_NB = joblib.load(model_dir+'Opsi3_NB')

# save into a list for easy training & cross validation
models = []
models.append(('LR_Opsi1', Opsi1_LR))
models.append(('KNN_Opsi1', Opsi1_KNN))
models.append(('DT_Opsi1', Opsi1_DT))
models.append(('SVM_Opsi1', Opsi1_SVM))
models.append(('RF_Opsi1', Opsi1_RF))
models.append(('LDA_Opsi1', Opsi1_LDA))
models.append(('NB_Opsi1', Opsi1_NB))
models.append(('LR_Opsi3', Opsi3_LR))
models.append(('KNN_Opsi3', Opsi3_KNN))
models.append(('DT_Opsi3', Opsi3_DT))
models.append(('SVM_Opsi3', Opsi3_SVM))
models.append(('RF_Opsi3', Opsi3_RF))
models.append(('LDA_Opsi3', Opsi3_LDA))
models.append(('NB_Opsi3', Opsi3_NB))

#Loop for evaluation of ML model
model_predictions = []
names = []
runtimes = []

for name, model in models:
```

```
  start = timer() #Start measure Runtime
  #Show metrics to evaluate performance
  pred = model.predict(df_random)  #predict result from model
  model_predictions.append(pred)
  names.append(name)
  print('%s Predict Completed!' % (name))
  end = timer() #End measure Runtime
  duration = end-start
  runtimes.append(duration)
  print('Runtime (s) : %f' % (duration))

"""### Rule-based"""

#Rule-Based_Statistical

#Stat_Outlier = (value < q25 - (q75-q25) * 1.5) or (value > q75 + (q75-q25) * 1.5)
Outlier_Oil = (df_random['Oil_Pressure'] < 544.000) | (df_random['Oil_Pressure'] >
672.000)
Outlier_Coolant =(df_random['Coolant_Temp'] < 67.000) | (df_random['Coolant_Temp']
> 91.000)
Outlier_ECU = (df_random['ECU_Temp'] < 18.000) | (df_random['ECU_Temp'] >
66.000)
Outlier_PFA = (df_random['PF_Avg'] < 0.615) | (df_random['PF_Avg'] > 1.055)

#Stat_Maintenance = (q25 - (q75-q25) * 1.5 <= value <= q25) or (q75 <= value <= q75 +
(q75-q25) * 1.5)
Maintenance_Oil = (df_random['Oil_Pressure'].between(544.000, 592.00)) |
(df_random['Oil_Pressure'].between(624.00, 672.000))
Maintenance_Coolant = (df_random['Coolant_Temp'].between(67.000, 76.00)) |
(df_random['Coolant_Temp'].between(82.00, 91.000))
Maintenance_ECU = (df_random['ECU_Temp'].between(18.000, 36.00)) |
(df_random['ECU_Temp'].between(48.00, 66.000))
Maintenance_PFA = (df_random['PF_Avg'].between(0.615, 0.78) |
(df_random['PF_Avg'].between(0.89, 1.055)))

#Stat_Normal = q25 < value < q75
Normal_Oil = df_random['Oil_Pressure'].between(592.00, 624.00, inclusive=False)
Normal_Coolant = df_random['Coolant_Temp'].between(76.00, 82.00, inclusive=False)
Normal_ECU = df_random['ECU_Temp'].between(36.00, 48.00, inclusive=False)
Normal_PFA = df_random['PF_Avg'].between(0.78, 0.89, inclusive=False)

#Label Condition
Outlier =  Outlier_Oil & Outlier_Coolant & Outlier_ECU & Outlier_PFA
Maintenance = Maintenance_Oil | Maintenance_Coolant |  Maintenance_ECU |
Maintenance_PFA

#Check conditions for each row in dataframe
df_RB_Statistical = df_random.copy()
df_RB_Statistical['Label'] = 'Normal'
df_RB_Statistical.loc[Outlier, 'Label'] = 'Outlier'
df_RB_Statistical.loc[Maintenance, 'Label'] = 'Maintenance'

# class distribution
```

```
print(df_RB_Statistical.groupby('Label').size())
print('Total sample : %d' % len(df_RB_Statistical))

#Rule-based_Clustering

#Label Condition based on P1 and P2 Component from PCA result and clusters_dbscan
#need to pre-process features data with PCA

# Scaling and Normalize data
features_norm = ['Oil_Pressure_Norm',  'Coolant_Temp_Norm', 'PF_Avg_Norm',
          'ECU_Temp_Norm']
X_scaled = StandardScaler().fit_transform(df_random)        #Bring all the attributes to a
comparable level (Z-score normalization)
X_normalized = normalize(X_scaled)                          #Centering data to Origin
X_normalized = pd.DataFrame(X_normalized, columns=features_norm)     #Converting
the numpy array into a pandas DataFrame

# fit transform data
pca = PCA(n_components = 4)
PCA_comp = ['P1', 'P2', 'P3', 'P4'] #set Px value based on the numbers of features
X_transform = pca.fit_transform(X_normalized)
X_transform = pd.DataFrame(X_transform, columns=PCA_comp)

# Select two PCA components with highest variation (Dimensionality reduction)
df_cluster = pd.DataFrame(dict(x=X_transform['P1'], y=X_transform['P2']))

#cluster_0 = P1 > 0.25 AND P2 < 0.2
cluster_0 = (df_cluster['x'] > 0.25) & (df_cluster['y'] < 0.2)
#cluster_1 = P1 > 0.25 AND P2 > 0.2
cluster_1 = (df_cluster['x'] > 0.25) & (df_cluster['y'] > 0.2)
#cluster_2 = P1 <= 0.25 AND P2 <= 0.2
cluster_2 = (df_cluster['x'] <= 0.25) & (df_cluster['y'] <= 0.2)

#Add label to dataset
df_RB_Clustering = df_RB_Statistical.copy()
df_RB_Clustering['Label_cluster'] = np.nan
df_RB_Clustering.loc[cluster_0, 'Label_cluster'] = 0
df_RB_Clustering.loc[cluster_1, 'Label_cluster'] = 1
df_RB_Clustering.loc[cluster_2, 'Label_cluster'] = 2

# Label distribution result
print(df_RB_Clustering.groupby('Label_cluster').size())
print('Total sample : %d' % len(df_RB_Clustering))

"""## Results"""

df_eval = pd.DataFrame({'Model':names, 'Runtime':runtimes})
df_eval

df_pred = pd.DataFrame(model_predictions, index=names)
df_pred = df_pred.transpose()

# class distribution
```

```
for col in df_pred.columns :
  print(df_pred.groupby(col).size())

# Rule-Based statistical distribution result
print(df_RB_Statistical.groupby('Label').size())
print('Total sample : %d' % len(df_RB_Statistical))
# Rule-Based clustering distribution result
print(df_RB_Clustering.groupby('Label_cluster').size())
print('Total sample : %d' % len(df_RB_Clustering))
```