ANALISIS KENYAMANAN SEPEDA MOTOR MENGGUNAKAN METODE MACHINE LEARNING

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Mesin

Disusun Oleh :

Nama : M. Fajrurrahman Al-Farouqy
No. Mahasiswa : 16525088
NIRM : 2016080681

JURUSAN TEKNIK MESIN
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA

2022

PERNYATAAN KEASLIAN

Dengan ini saya menyatakan bahwa karya ini benar-benar karya hasil kerja saya sendiri yang sepanjang pengetahuan saya, tidak terdapat karya maupun tulisan orang lain kecuali kutipan yang secara tertulis saya jelaskan setiap sumbernya. Apabila kemudian hari pernyataan saya tidak benar dan melanggar hak kekayaan intelektual, saya siap menerima hukuman atau sanksi sesuai hukuman yang berlaku.

Yogyakarta, 04 November 2022

Penulis

Muhammad Fajrurrahman Al-Farouqy

LEMBAR PENGESAHAN DOSEN PEMBIMBING

ANALISIS KENYAMANAN SEPEDA MOTOR MENGGUNAKAN METODE MACHINE LEARNING

TUGAS AKHIR

Disusun Oleh :

Nama : **M. Fajrurrahman Al-Farouqy**

No. Mahasiswa : **16525088**

NIRM : **2016080681**

Yogyakarta, 15 September 2022

Pembimbing I,	Pembimbing II,
Mohammad Faizun, S.T., M.Eng., Ph.D	**Ir. Donny Suryawan, S.T., M.Eng.**
NIP. 115250101	**NIP. 175250404**

LEMBAR PENGESAHAN DOSEN PENGUJI

ANALISIS KENYAMANAN SEPEDA MOTOR MENGGUNAKAN METODE MACHINE LEARNING

TUGAS AKHIR

Disusun Oleh :

Nama : **M. Fajrurrahman Al-Farouqy**

No. Mahasiswa : **16525088**

NIRM : **2016080681**

Tim Penguji

Ir. Donny Suryawan, S.T., M.Eng.,IPP

Ketua

Tanggal : Oktober 2022

Yustiasih Purwaningrum, S.T., M.T.

Anggota I

Tanggal : 31 Oktober 2022

Ir. Faisal Arif Nurgesang, S.T., M.Sc., IPP

Anggota II

Tanggal : 27 Oktober 2022

Mengetahui

Ketua Jurusan Teknik Mesin

UNIVERSITAS ISLAM INDONESIA ★ YOGYAKARTA ★ FAKULTAS TEKNOLOGI INDUSTRI

Dr. Ir. Muhammad Khafidh, S.T., M.T., IPP

NIP. 145250101

HALAMAN PERSEMBAHAN

Tugas akhir ini dipersembahkan untuk:

1. Allah SWT yang telah memberikan saya kesempatan untuk beribadah tholabul ilmi hingga jenjang perguruan tinggi.
2. Keluarga yang selalu memberikan dukungan, motivasi, dan doa.
3. Bapak Dr. Ir. Muhammad Khafidh, S.T., M.T., IPP sebagai ketua Prodi Teknik Mesin.
4. Bapak Mohammad Faizun, S.T., M.Eng., Ph.D sebagai dosen pembimbing tugas akhir.
5. Bapak Ir. Donny Suryawan, S.T., M.Eng., IPP sebagai dosen pembimbing tugas akhir.
6. Segenap mahasiswa Teknik Mesin Universitas Islam Indonesia.
7. Sahabat dan kerabat yang telah mendukung terlaksananya tugas akhir ini.

HALAMAN MOTTO

“Banyak orang bertitel tanpa kualitas, banyak orang berkualitas tanpa titel.”

(KH. Hasan Abdullah Sahal)

KATA PENGANTAR

Assalamualaikum Warahmatullahi Wabarakatuh

Alhamdulillahirabbilalamin, rasa syukur yang tak berujung terucap kehadirat *Allah Subhanahu Wa Ta'ala,* berkat izin-Nya laporan tugas akhir ini dapat dituntaskan. Shalawat beriring salam teruntuk baginda Muhammad *Shallallahu Alaihi Wasallam* atas terbukanya peradaban keilmuan yang telah dan akan selalu dapat menyelamatkan manusia dari belantara kebodohan. Dibuatnya laporan ini bertujuan untuk melengkapi syarat kelulusan dari program sarjana studi Teknik Mesin Universitas Islam Indonesia.

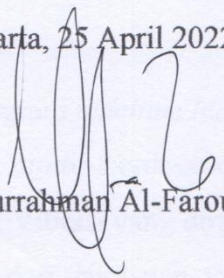
Tanpa dukungan berbagai pihak, niscaya laporan tugas akhir ini tidak dapat terselesaikan, dengan ini saya menuturkan terima kasih yang mendalam kepada :

1. Keluarga terkasih atas dukungan tanpa pamrih sepanjang perjalanan menempuh jenjang pendidikan sarjana ini.
2. Bapak Dr. Ir. Muhammad Khafidh, S.T., M.T., IPP selaku ketua program studi Teknik Mesin Universitas Islam Indonesia.
3. Bapak Mohammad Faizun, S.T., M.Eng., Ph.D dan Ir. Donny Suryawan, S.T., M.Eng., IPP sebagai dosen pembimbing tugas akhir.
4. Ibu Yustiasih Purwaningrum, S.T., M.T. sebagai dosen pembimbing akademik.
5. Segenap dosen program studi Teknik Mesin atas bimbingan dalam mengarungi samudera keilmuan.
6. Awi Hamzah Rambe sebagai partner dalam melaksanakan tugas akhir.
7. Segenap mahasiswa Teknik Mesin Universitas Islam Indonesia.
8. Sahabat dan kerabat yang telah mendukung terlaksananya tugas akhir ini.

Penulis telah mengerahkan segenap usaha untuk menyusun laporan ini menjadi sebaik mungkin, namun sebagai manusia yang tidak lekang dari kelalaian. Kritik dan saran sangat diharapkan demi proses perkembangan selanjutnya. Semoga laporan ini dapat memberikan manfaat kepada penulis dan pembaca.

Wassalamu'alaikum Warahmatullahi Wabarakatuh

Yogyakarta, 25 April 2022



M. Fajrurrahman Al-Farouqy

ABSTRAK

Penelitian ini dilaksanakan untuk dapat membuat program *machine learning* yang mampu melakukan klasifikasi kenyamanan sepeda motor berdasarkan besaran nilai vibrasi yang diperoleh dari alat perekam vibrasi. Vibrasi yang dirasakan oleh pengendara sepeda motor secara dominan berasal dari rambatan vibrasi yang terjadi saat adanya kontak antara roda sepeda motor yang berputar dan bersentuhan dengan permukaan jalan yang dilalui, kemudian suspensi sepeda motor berusaha untuk meredam getaran dari roda agar seminimal mungkin merambat ke komponen lainnya yang dapat menyebabkan ketidaknyamanan. Adapun tiga hal yang mempengaruhi besaran vibrasi adalah kontur jalan, performa suspensi, dan laju kendaraan. Terdapat tiga tahap dalam proses penelitian ini. Pertama perancangan sensor getar, baik dalam segi *software* dan *hardware*. Tahap kedua adalah proses perekaman data vibrasi sepeda motor. Tahap terakhir adalah mengolah data vibrasi menggunakan proses *machine learning* dengan metode *support v*ector classification dan *logistic regression*. Dari pengolahan data diketahui bahwa *support vector classification* dapat menghasilkan model prediksi dengan akurasi 100 persen dan *logistic regression* sebesar 90 persen.

Kata kunci : vibrasi, *machine learning*, *support vector classification*, *support vector regression*.

ABSTRACT

This research was carried out to be able to create a machine learning program that is able to classify motorcycle comfort based on the magnitude of the vibration value obtained from the vibration recording device. The vibration felt by the motorcycle rider predominantly comes from the vibration propagation that occurs when there is contact between the rotating motorcycle wheel and in contact with the road surface, then the motorcycle suspension tries to dampen the vibration from the wheel so that it propagates least possible transmission to other components that may cause discomfort. The three things that affect the amount of vibration are road contours, suspension performance, and vehicle speed. There are three stages in this research process. The first is the design of the vibration sensor, both in terms of software and hardware. The second stage is the process of recording motorcycle vibration data. The last stage is processing vibration data using machine learning processes with support vector classification and logistic regression methods. From data processing, it is known that support vector classification can produce prediction models with an accuracy of 100 percent and logistic regression of 90 percent.

Keywords: vibration, machine learning, support vector classification, support vector regression.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR NOTASI

G-force : Nilai percepatan terhadap gaya gravitasi

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Kenyamanan menjadi hal utama yang dirasakan oleh pengguna kendaraan saat berkendara. Kenyamanan secara umum merupakan suatu keadaan dimana tubuh merasa rileks dan terbebas dari rasa sakit maupun lelah. Kualitas kenyamanan dalam aplikasi kendaraan dapat dimaknai sebagai kemampuan kendaraan untuk membuat penumpang maupun pengendara sesedikit mungkin merasa lelah dalam perjalanan sejauh mungkin.

Namun kualitas kenyamanan bukanlah suatu hal yang mudah diukur, setiap orang dapat memberikan penilaian yang berbeda untuk suatu contoh kondisi kenyamanan yang identik, sehingga untuk dapat menentukan kualitas kenyamanan yang ideal bagi setiap individu hendaknya mereka melakukan pengujian kendaraan dan penilaian mandiri / *self-assessment* berdasarkan pengalaman berkendara masing-masing.

Saat ini kebanyakan alat ukur vibrasi atau *vibrometer* dibuat untuk pengukuran mesin atau alat yang bergerak statis, sehingga kurang praktis untuk digunakan pada benda yang bergerak dinamis seperti sepeda motor. Penyesuaian tingkat kenyamanan untuk para pengguna secara spesifik belum banyak diaplikasikan oleh para produsen sepeda motor.

Hal sangat diperlukan mengingat sepeda motor masih menjadi sarana utama transportasi di Indonesia. Menurut data jumlah kendaraan bermotor dari Badan Pusat Statistik yang diperoleh pada tahun 2020 menunjukkan bahwa jumlah kendaraan bermotor berjenis sepeda motor mencapai 115.188.762 unit, sedangkan jumlah kendaraan bermotor secara keseluruhan yang terdiri dari mobil penumpang, bus, truk, dan sepeda motor adalah 136.316.726 unit. Dari data tersebut dapat diketahui bahwa jumlah sepeda motor sebanyak 84,5% dari total populasi kendaraan bermotor.

Kontur atau kualitas jalan juga sangat mempengaruhi karakteristik dari sepeda motor yang digunakan, mengingat tidak semua jalan di Indonesia memiliki

kualitas yang baik. Kondisi jalan yang berbeda menuntut standar kenyamanan yang berbeda. Sehingga penyesuaian standar kenyamanan berdasarkan kebutuhan dan standar pengguna sangat diperlukan. Dalam hal ini penulis menawarkan mekanisme analisis standar kenyamanan pengendara berdasarkan vibrasi sepeda motor yang diproses menggunakan *machine learning*.

1.2 Rumusan Masalah

Berlandaskan latar belakang yang telah dijabarkan, maka disusun rumusan masalah sebagai berikut:

1. Bagaimana cara membuat alat perekam vibrasi portabel sederhana dan terjangkau?
2. Bagaimana cara mengolah data vibrasi dengan *Machine Learning*?
3. Bagaimana perbedaan antara proses klasifikasi dan regresi?

1.3 Batasan Masalah

Pada penelitian dan perancangan ini penulis menentukan batasan-batasan sebagai berikut :

1. Sepeda motor yang digunakan untuk penelitian adalah Honda Vario 150 eSP tahun produksi 2016.
2. Komposisi muatan pada proses pengumpulan data getaran sepeda motor terdiri dari pengemudi dan penumpang.
3. Besaran vibrasi diukur pada satu titik yang dianggap paling mewakili untuk kenyamanan pengendara, yakni pada pegangan penumpang belakang.
4. Aspek material dan mekanis seperti jenis dan kemiringan suspensi, ban yang digunakan, dan lain-lain sesuai standar pabrik.
5. Metode pengolahan data yang digunakan pada Python adalah *support vector machine* (SVM) regresi dan klasifikasi tahap mendasar.
6. Penentuan tingkat kenyamanan untuk target data berdasarkan *self-assessment* yang dilakukan oleh penulis.

1.4 Tujuan Penelitian atau Perancangan

Beracuan pada rumusan masalah diatas, maka terdapat beberapa tujuan penelitian sebagai berikut:

1. Membuat alat perekam vibrasi *portable*, dan sederhana.
2. Mampu mengolah data vibrasi menggunakan *Machine Learning*.
3. Menentukan proses pembuatan model yang tepat berdasarkan jenis data.

1.5 Manfaat Penelitian atau Perancangan

Diharapkan dari penelitian ini dapat menciptakan alat yang memungkinkan konsumen sepeda motor untuk mengumpulkan data respon pengendara atas kualitas produk sepeda motor dari aspek kenyamanan untuk kebutuhan pribadi.

1.6 Sistematika Penulisan

Komposisi penelitian ini terdiri dari dua eksperimen yang saling berkaitan. Eksperimen pertama dilakukan untuk mendapatkan nilai vibrasi sebagai variabel *input*, kemudian melakukan *self assesment* dari data tadi untuk membuat variabel *output*.

BAB I : PENDAHULUAN

Dalam bab ini dijabarkan latar belakang, rumusan masalah, batasan, tujuan, manfaat, dan sistematika penulisan.

BAB II : TINJAUAN PUSTAKA

Pada bab ini disajikan beberapa teori dan penelitian terdahulu yang dijadikan sandaran penelitian ini.

BAB III : METODE PENELITIAN

Bab ini berisi pembahasan mengenai metode yang digunakan dalam penelitian dari alur penelitian, peralatan maupun bahan yang digunakan, dan perancangan.

BAB IV : HASIL DAN PEMBAHASAN

Pada bab ini dibahas hasil penelitian serta pembuktian hasil *dataset* uji menggunakan *machine learning*.

BAB V : PENUTUP

Bab ini membahas kesimpulan yang diperoleh dari keseluruhan penelitian dan perancangan serta saran untuk memperbaiki penelitian atau perancangan selanjutnya.

BAB 2
TINJAUAN PUSTAKA

2.1 Kajian Pustaka

Penggunaan sepeda motor sebagai sarana utama untuk kebutuhan transportasi harian masih umum dijumpai. Selain memiliki ukuran yang relatif kompak sehingga memudahkan penggunaan dalam kemacetan, konsumsi bahan bakar yang ekonomis, dan sarana angkutan umum yang belum dapat sepenuhnya diandalkan menjadi alasan utama tingginya populasi sepeda motor di Indonesia. Permukaan jalan yang tidak rata menjadi salah satu faktor yang dapat mempengaruhi kenyamanan pengendara [1].

Komponen yang berfungsi untuk meredam energi vibrasi yang ditransfer ke pengendara adalah mekanisme suspensi. Beberapa penelitian berkaitan dengan kenyamanan berkendara sepeda motor telah banyak dilaksanakan. Diantaranya adalah yang telah dilakukan oleh Thompson dengan tinjauan aspek dinamika motor terhadap kemiringan suspensi [2], dan ada juga yang menggunakan pemodelan *multi-degree of freedom* (DOF) terhadap kendaraan beserta pengendaranya [3].

Kedua penelitian di atas menggunakan pemodelan matematika dan model mekanisme derajat kebebasan untuk menentukan tingkat kenyamanan dari kendaraan. Selain menggunakan pendekatan standar umum, metode ini menghasilkan konklusi dari data yang dikalkulasi atau biasa dikenal sebagai *hard computing*, bukan melalui aplikasi *machine learning*. Sehingga dirasa kurang tepat untuk dijadikan basis untuk penelitian kami.

Penelitian tentang analisa vibrasi sepeda motor juga telah dilakukan oleh Suhandoko, menggunakan Yamaha Jupiter Z tahun produksi 2004 sebagai spesimen uji, kemudian membuat rancangan simulasi komponen suspensi depan dan belakang menggunakan Solidworks 2010 untuk mendapatkan nilai kekakuan suspensi, juga melakukan pengujian untuk mengetahui nilai viskositas fluida *shock absorber*. Data dari penelitian tadi diolah menggunakan software MATLAB 6.5 untuk menentukan respon kekakuan dan redaman dari suspensi sepeda motor [4].

Dari penelitiannya, Suhandoko menghasilkan nilai respon dari rambatan vibrasi yang terjadi pada rangka (*body*) dan suspensi motor. Metode pengumpulan data ini cukup menyerupai penelitian yang kami lakukan, namun menggunakan variabel pengujian dan metode pengolahan data yang berbeda.

Sedangkan penelitian tentang efek vibrasi pada objek dan aplikasi pengolahan data menggunakan *machine learning* pernah dilakukan oleh Sukendi, Ikhwansyah, dan Suherman. Mereka melakukan pengujian dengan mengukur besaran getaran yang terjadi pada *bearing*, kemudian mengolah data hasil pengujian dengan *machine learning* (*support vector machine*) sehingga dapat digunakan untuk memprediksi kerusakan pada *bearing* [5].

Kami menggunakan metode pengumpulan data menyerupai yang ada pada penelitian yang dilakukan oleh Suhandoko, dengan beberapa penyesuaian pada metode berdasarkan data dan variabel yang kami inginkan. Kemudian data hasil pengujian kami olah menggunakan *machine learning*.

2.2 Dasar Teori

2.2.1 Vibrasi

Vibrasi atau getaran merupakan suatu gerak bolak-balik di sekitar kesetimbangan. Kesetimbangan dimaknai sebagai keadaan diam suatu benda saat tidak ada gaya yang bekerja pada benda tersebut. Vibrasi mungkin terjadi saat mesin atau alat dioperasikan dengan motor penggerak, sehingga pengaruhnya bersifat mekanis [6].

Getaran dapat diukur dalam tiga bentuk :

1. Frekuensi

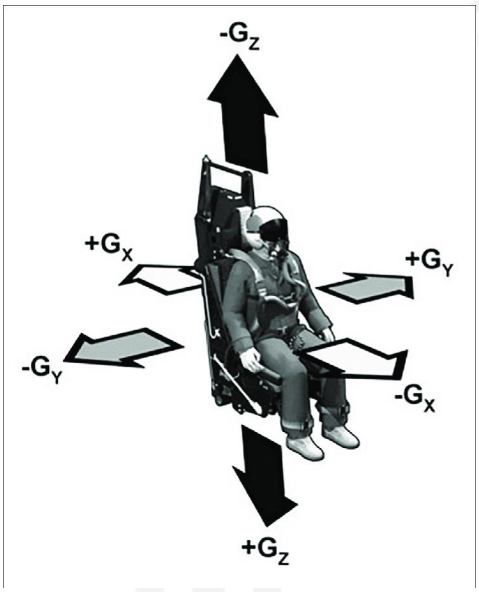
Frekuensi adalah jumlah gelombang yang terjadi dalam satuan unit periode waktu (detik) [7]. Frekuensi umum direpresentasikan oleh satuan Hertz (Hz), Satu Hertz melambangkan satu puncak panjang gelombang yang melewati titik tertentu per detik [8]. Dalam kata lain frekuensi mengukur jumlah gelombang yang terjadi atau seberapa cepat getaran dalam satu detik.

2. Amplitudo

Amplitudo mewakili besaran nilai dari ketinggian intensitas gelombang pada satuan periode waktu. Sederhananya jika frekuensi menunjukkan banyaknya gelombang dalam satu detik, amplitudo menunjukkan besarnya gelombang.

3. G-force

Gravitational force atau biasa disingkat G-*force* merupakan gaya yang terjadi akibat pergerakan atau percepatan suatu objek terhadap pengaruh gaya gravitasi, bisa dituliskan dalam satuan percepatan meter per detik persegi (m/s^2). Satu G setara dengan 9,8 m/s^2, dan dapat terjadi dalam sumbu pergerakan (*axis*) yang berbeda.



Gambar 2.1 G *force* (researchgate, 2022)

Sebagaimana ditampilkan pada gambar 2.1 G *force* yang terjadi pada sumbu X disebut *transverse* G, pada sumbu Y disebut *lateral* G, dan pada sumbu Z disebut *vertical* G.

2.2.2 Arduino

Arduino adalah *microcontroller* berbentuk *single-board* dan memiliki basis sistem *open-source* sehingga sangat mudah dan fleksibel untuk digunakan. Dengan prosesor Atmel AVR dan dapat memahami bahasa pemrograman C atau C++.

Arduino menjadi platform yang sangat populer untuk digunakan sebagai komponen pendukung prototype berbasis serial komunikasi elektronik dengan

kemudahan akses perangkat keras pendukung dan perangkat lunak dengan komunitas yang sangat besar.

Untuk penelitian ini kami menggunakan Arduino Uno yang memiliki komponen *microcontroller* ATMEGA328, 14 *pin input/output* (I/O) digital, dan 6 *pin input* analog.

2.2.3 Python

Python merupakan bahasa pemrograman tingkat tinggi (*high level language*) bersifat *multi-purpose* dan dapat dioperasikan melalui *interpreter*.

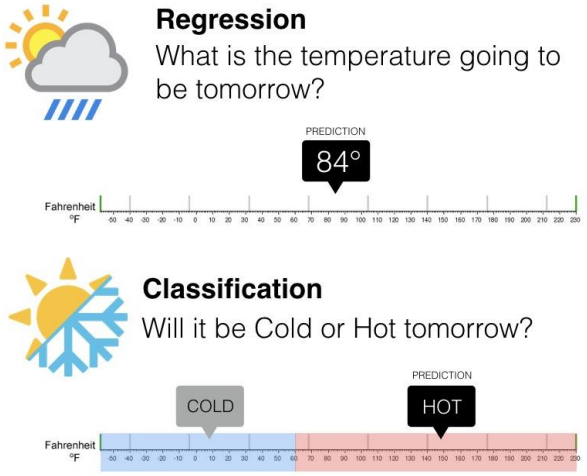
Dapat menggunakan bahasa C atau C++ yang merupakan bahasa pemrograman paling umum digunakan oleh pemula dan didukung *syntax* yang sederhana, sehingga banyak digunakan dalam pemrograman web, aplikasi *desktop*, pemrograman *backend* hingga komputasi saintifik dan *machine learning*.

Pengguna awam juga sangat dibantu oleh fungsi *package* yang memungkinkan pemanggilan fungsi tanpa menggunakan sejumlah kode yang rumit. Namun karena mekanisme *interpreter* memerlukan proses pemanggilan per urutan baris program, proses eksekusi program memakan waktu lebih lama dari bahasa pemrograman lain yang mengadopsi mekanisme *compiler* [9].

2.2.4 Machine Learning

Machine learning adalah sub bidang *artificial intelligence* (AI) atau bisa diartikan sebagai kecerdasan buatan, yang secara luas didefinisikan sebagai kemampuan mesin untuk meniru kemampuan kecerdasan manusia. Sistem kecerdasan buatan digunakan untuk melakukan tugas kompleks dengan cara yang mirip dengan cara manusia memecahkan masalah. *Machine learning* adalah bidang pembelajaran dan secara luas dibagi menjadi *supervised learning*, *unsupervised learning*, dan *reinforcement learning* [10].

Supervised machine learning terbagi menjadi dua, yakni *supervised machine classification*, dan *supervised machine regression*. *Support vector classification* atau SVC membuat model yang memprediksi berdasarkan label atau target output, sedangkan *support vector regression* memprediksi berdasarkan kecenderungan kemunculan data kontinu.



Gambar 2.2 SVC vs SVR (springboard, 2022)

Machine learning memungkinkan untuk membuat suatu pola korelasi dari sekumpulan data input dan data output dengan mekanisme algoritma. Inti dari kebanyakan algoritma *machine learning* adalah untuk membangun model optimasi dan mempelajari parameter dalam fungsi tujuan dari data yang diberikan [11]. Tidak jarang data yang perlu diolah berukuran sangat besar dan kompleks sehingga cukup sulit untuk dikelola dan diproses, karena itu data input untuk *machine learning* biasa disebut *big data*.

BAB 3
METODE PENELITIAN

3.1 Alur Penelitian

Alur penelitian berjalan seperti yang ditunjukkan gambar diagram alir 3.1 di bawah ini, setiap tahapan akan dijabarkan secara umum kemudian.

Gambar 3.1 Diagram Alir Penelitian

3.2 Studi Pustaka

Pada tahap ini kami melakukan studi tentang penelitian terkait yang pernah dilakukan sebelumnya sehingga dapat dijadikan rujukan untuk penelitian kami. Selain itu kami juga melakukan studi tentang metode perancangan perangkat lunak dan perangkat keras untuk pembuatan alat uji penelitian. Kami menggunakan

beberapa referensi seperti jurnal ilmiah, artikel, dokumen digital, dan media panduan pembelajaran.

3.3 Kriteria Desain

Setelah didapatkan rujukan penelitian kami menentukan beberapa kriteria desain. Bagian ini juga berfungsi sebagai batasan penentu produk akhir penelitian dari pembuatan suatu desain.

3.3.1 Perangkaian Alat Uji

Dalam tahap ini terdapat dua bagian; pertama perangkaian perangkat keras, kemudian perangkaian perangkat lunak. Karena untuk melakukan perangkaian program perangkat lunak diperlukan rangkaian perangkat keras, maka kami mendahulukan proses tersebut. Setelah itu proses perangkaian perangkat lunak dilakukan untuk menjalankan perangkat keras sebagai alat uji penelitian. Setelah alat uji memenuhi kriteria desain maka dilakukan tahap simulasi dan pengujian.

3.3.2 Pengujian

Kami melakukan pengujian di beberapa lokasi pengujian yang telah ditentukan untuk mendapatkan data penelitian. Proses pengujian setiap lokasi menggunakan tiga variasi kecepatan, yakni pelan (maksimal 10 kpj), sedang (diatas 10 kpj sampai 20 kpj), dan tinggi (30 kpj atau lebih). Data yang didapat kemudian diolah menggunakan Python untuk mendapatkan korelasi algoritma antara data *input* dan *output*.

Gambar 3.2 Lokasi Pengujian 1

Lokasi pengujian 1 merupakan jalur perlintasan kereta api yang memiliki 2 jalur rel, sebagaimana ditunjukkan pada gambar 3.2 jarak perlintasan sekitar 10 meter dan memiliki polisi tidur serta permukaan yang cukup bergelombang.

Gambar 3.3 Lokasi Pengujian 2

Gambar 3.3 menunjukkan lokasi pengujian 2 berjarak sekitar 20 meter dan dibuat dari *conblock* dengan permukaan yang menurun kualitasnya di bagian tengah hingga akhir.

Gambar 3.4 Lokasi Pengujian 3

Gambar 3.4 menunjukkan lokasi pengujian 3 yang merupakan jalan aspal datar tanpa lubang maupun gelombang dengan jarak sekitar 50 meter.

3.4 Kriteria Desain

Dalam pembuatan alat uji kami memiliki beberapa kriteria yang dijadikan acuan menyesuaikan fungsi yang diperlukan :

1. Alat uji berukuran kompak.
2. Alat uji mampu menyimpan data pengujian.
3. Sensor mampu mendeteksi vibrasi dengan satuan tertentu.
4. Komponen yang dipakai mudah didapat.

3.5 Alat dan Bahan

Alat dan bahan merupakan sarana utama pendukung terlaksananya penelitian ini, adapun alat dan bahan yang dipakai adalah sebagai berikut :

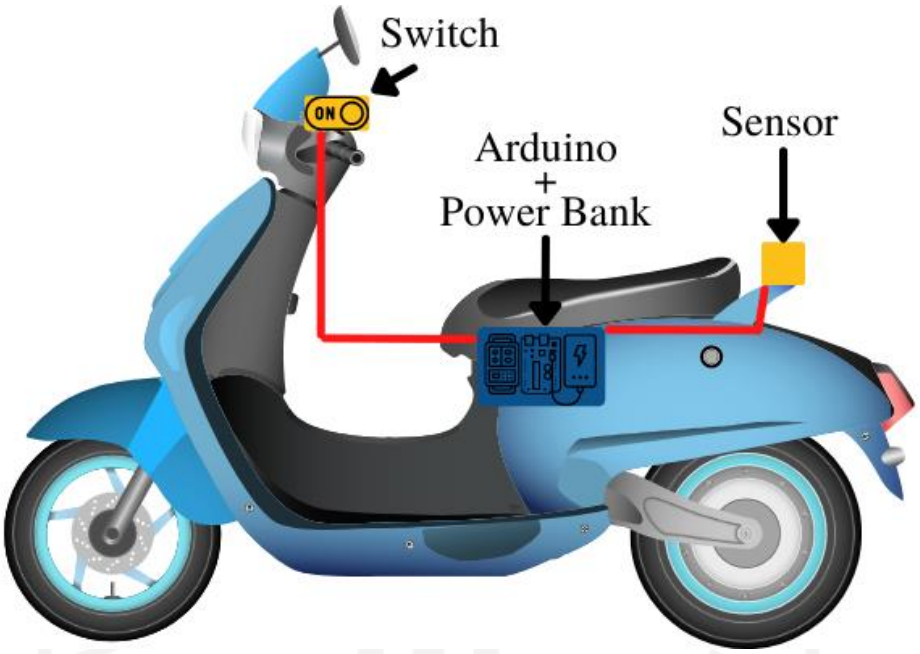
1. PC
2. *Software* Arduino IDE
3. *Software* Python
4. Solder
5. Timah
6. Arduino Uno
7. Data Logger XD-204
8. SD *Card*
9. Kabel Data
10. *Accelerometer* ADXL345
11. Kabel Jumper
12. 2 *compact connector terminal block*
13. *Power Bank*
14. Box Plastik 140 x 82 x 38 mm
15. Box Plastik 6.7 x 4.9 x 2.3 mm
16. *Case* akrilik Arduino
17. *Double Tape Foam*
18. *Masking Tape*
19. *Cable Tape*
20. *Toggle Switch* 3 Pin

3.6 Perancangan

Tahap perancangan terbagi menjadi dua bagian; pertama perancangan perangkat keras, kedua perancangan perangkat lunak yang berisi pembuatan program untuk menjalankan fungsi perangkat keras sebagai alat uji.

3.6.1 Perancangan Perangkat Keras

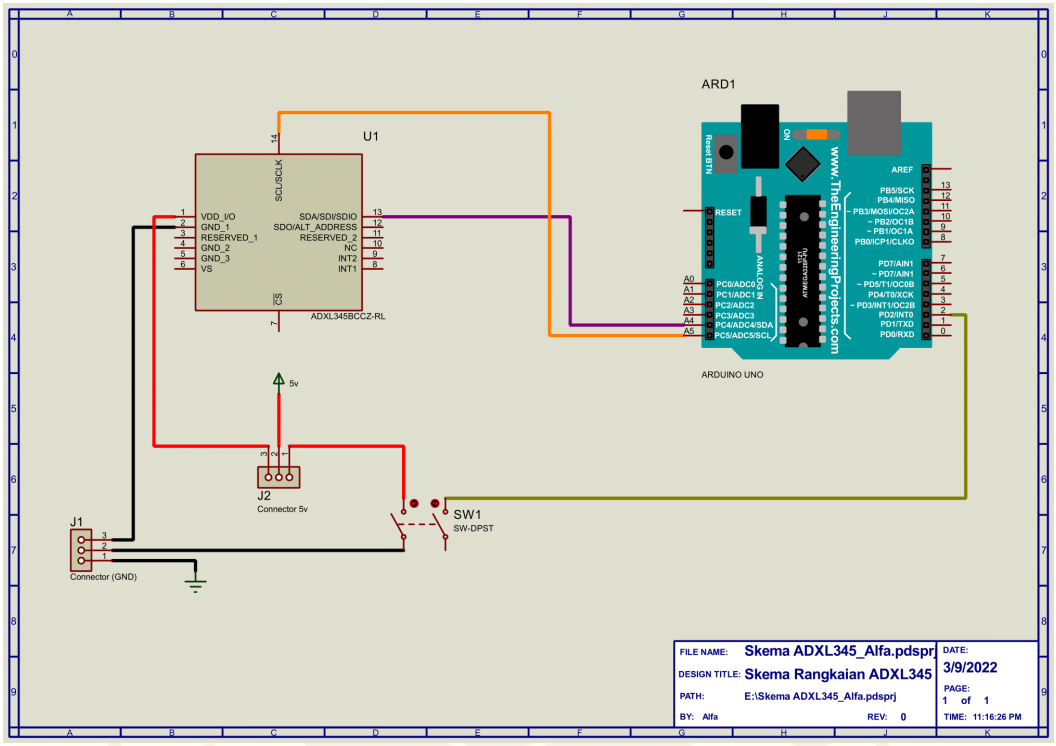
Tahap perancangan diawali dengan melakukan perakitan *hardware*, mulai dari mengukur panjang kabel yang diperlukan dari Arduino Uno yang ditempatkan di kompartemen penyimpanan ke titik penempatan sensor di tempat pegangan penumpang dan ke titik penempatan saklar di atas *speedometer* sebagaimana diilustrasikan pada gambar 3.5.



Gambar 3.5 Penempatan Perangkat Keras Pada Objek Uji

Dari proses pengukuran didapatkan jarak antara saklar *on/off* dan bagasi adalah 1 meter, dan jarak dari bagasi ke pegangan penumpang 1,5 meter. Saklar memiliki 3 pin, sedangkan sensor ADXL345 memiliki 4 pin untuk dihubungkan ke Arduino Uno, sehingga total kabel *jumper* yang diperlukan adalah 9 meter.

Untuk memudahkan penempatan dan mengamankan komponen saat pemakaian maka digunakan *box* plastik ukuran 140 x 82 x 38 mm untuk wadah Arduino Uno yang sudah terlebih dahulu dipasang dengan *case* akrilik. *Case* akrilik dipasang ke *box* menggunakan *double tape foam*, kemudian Data Logger XD-204 dipasang di atas Arduino Uno dengan memasang kaki konektor XD-204 pada lubang soket Arduino uno. Box bawah perlu dilubangi untuk akses kabel data ke soket Arduino Uno, dan bagian atas dilubangi untuk akses SD *card* ke Data Logger.



Gambar 3.6 Diagram Rangkaian Perangkat Keras

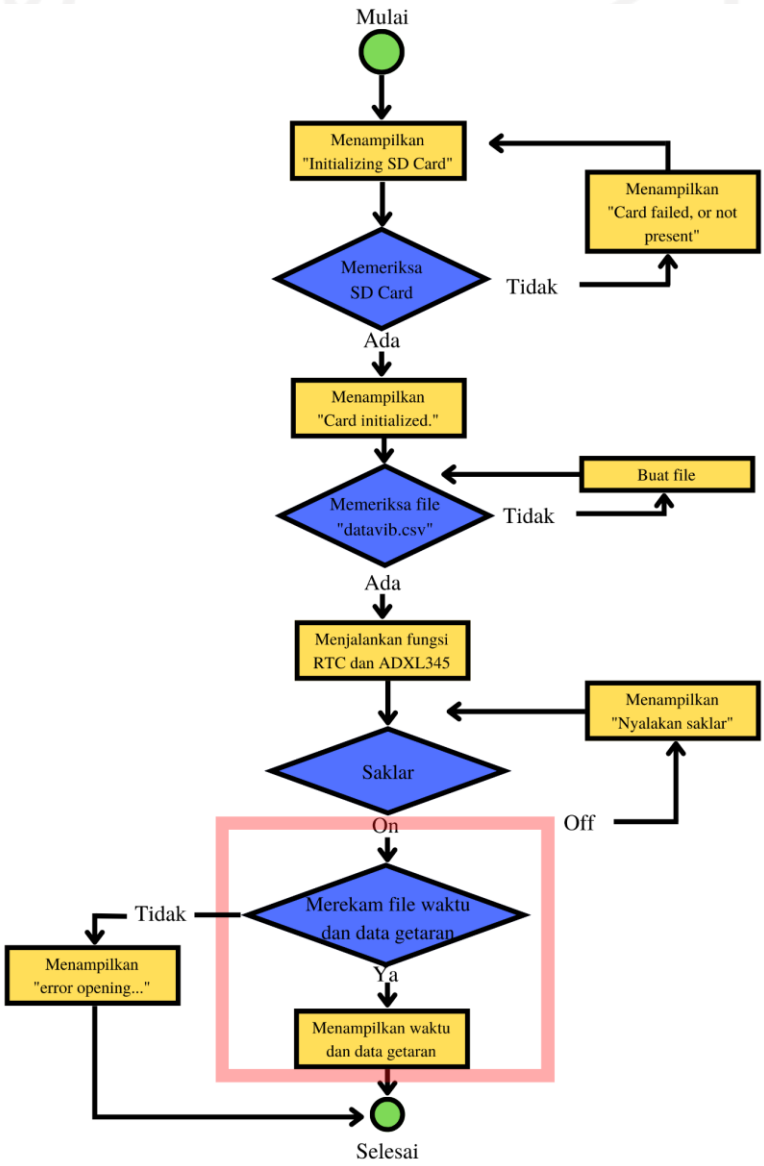
Sebagaimana ditampilkan pada Gambar 3.6 diatas, kami membuat masing masing cabang untuk *ground* (GND) dan 5V dengan *compact connector terminal block*. Untuk membuat *switc*h atau saklar yang bekerja *normally open*, kami menghubungkan pin paling kiri ke cabang 5v, pin tengah ke cabang *ground* , dan pin paling kanan ke soket 2 input digital Arduino Uno. Saklar ini berfungsi untuk memberikan instruksi mulai atau berhenti merekam data, bukan nyala atau mati sistem secara keseluruhan.

Pin *accelerometer* ADXL345 yang kami gunakan hanya 4 pin, yakni; pin 5V, GND, SCL, dan SDA. Pin 5V terhubung dengan cabang 5V, pin GND terhubung dengan cabang GND. Sedangkan untuk pin SDA terhubung dengan pin 4 *input* analog, dan pin SCL terhubung dengan pin 5 *input* analog.

3.6.2 Perancangan Perangkat Lunak

3.6.2.1 Program Arduino

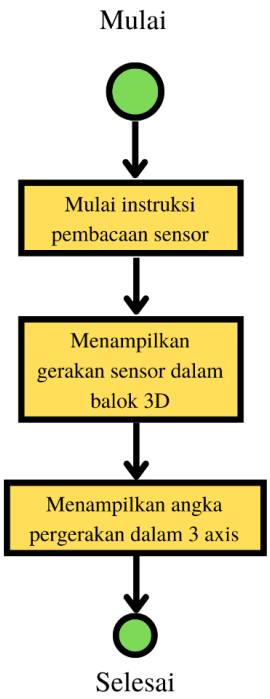
Pembuatan program Arduino diperlukan untuk menjalankan fungsi rangkaian Arduino dan sensor ADXL345 sebagai alat uji dan perekam data. Program ini akan merekam getaran yang terjadi pada motor saat dikendarai melewati jalan atau area yang dijadikan objek uji, getaran dibaca dan direkam dalam satuan g yang setara dengan 9,8 m/s^2. Untuk alur program Arduino berjalan sebagaimana ditampilkan pada gambar 3.7.



Gambar 3.7 Diagram alir program Arduino

3.6.2.2 Kalibrasi ADXL345

Untuk mendapatkan akurasi terbaik dari pembacaan sensor ADXL345 kami perlu melakukan kalibrasi dengan membuat program khusus dengan aplikasi *Processing*. Program ini berfungsi untuk membaca pergerakan *pitch* dan *roll* dari ADXL345 dan divisualisasikan dalam bentuk 3 dimensi. Proses kalibrasi dilakukan dengan menempatkan ADXL345 pada permukaan rata dan menyesuaikan nilai deviasi yang tertera pada program *Processing* dengan merubah nilai pada bagian program penyesuaian *axis* dalam program Arduino agar *axis* X dan Y agar terbaca 0, kemudian untuk *axis* Z dengan nilai 1 hal ini karena kondisi normal *axis* Z akan selalu menerima gaya tarik gravitasi. Alur program kalibrasi berjalan seperti yang ditampilkan gambar 3.8.



Gambar 3.8 Diagram alir program Kalibrasi ADXL345

BAB 4
HASIL DAN PEMBAHASAN

4.1 Hasil Alat Pengujian

Alat pengujian yang kami buat memiliki komposisi beberapa komponen yang dapat berfungsi sebagai pendeteksi getaran hingga dapat merekam data tersebut untuk diolah kemudian.

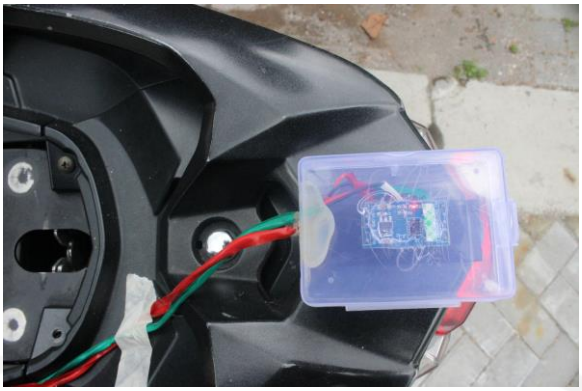
Gambar 4.1 Alat Pengujian

Gambar 4.1 menunjukkan rangkaian lengkap dari Alat Pengujian yang kami buat, adapun beberapa komponen utama adalah sebagai berikut :

A. *Accelerometer* ADXL345

Sensor ini bekerja dengan mengukur percepatan gerakan pada sumbu X, Y, dan Z. Umum digunakan sebagaimana fungsi aslinya yakni akselerometer yang mengukur percepatan gerak, atau *gyroscope* yang mengukur rotasi atau perputaran. Namun komponen ini juga dapat diadaptasikan sebagai sensor getar.

ADXL345 memiliki 10 pin yang terbagi pada kedua sisi pendek sejajar. Sisi pertama memiliki pin 5V, 3V3, GND, VS, dan CS. Sedangkan sisi lainnya memiliki pin SCL, SDA, SDO, INT 2, dan INT 1.



Gambar 4.2 Pemasangan ADXL345

Penempatan ADXL345 adalah pada besi pegangan penumpang, sebagaimana ditunjukkan gambar 4.2.

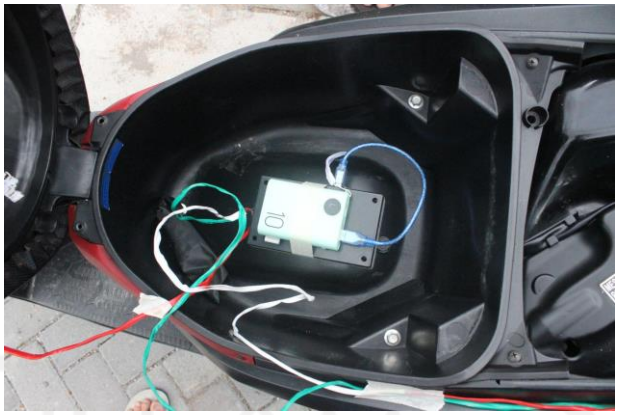
B. Arduino Uno

Sebagai komponen utama perangkat yang menerima semua instruksi program perangkat lunak dan memungkinkan berfungsinya berbagai komponen yang terhubung dengannya. Namun Arduino Uno tidak memiliki fitur perekaman data, sehingga diperlukan komponen tambahan untuk menjalankan fungsi tersebut.

C. Data logger XD-204

Karena Arduino Uno tidak memiliki fungsi untuk merekam data pengujian maka diperlukan Data Logger XD-204 untuk memenuhi fungsi perekaman data. Komponen ini dirangkai dengan memasukkan seluruh pin yang dimiliki dengan soket pin Arduino Uno.

XD-204 yang memiliki ruang baterai untuk menjalankan fungsi jam yang biasa dikenal sebagai RTC atau *real time clock*. Terdapat juga slot kartu SD dimana data rekaman disimpan.



Gambar 4.3 Box Arduino dan *Power Bank*

Gambar 4.3 menampilkan box yang berfungsi sebagai penyimpanan rakitan Arduino Uno dan Data Logger ditempatkan dalam bagasi motor. Dalam gambar di atas juga menampilkan sebuah *power bank* yang memiliki output 5V dan 2.1A sebagai sumber energi untuk menjalankan alat pengujian yang dihubungkan dengan Arduino Uno melalui USB *Jack*.

D. Saklar

Kami menggunakan sebuah saklar untuk menjalankan fungsi program perekaman data dalam kartu SD.

Gambar 4.4 Saklar

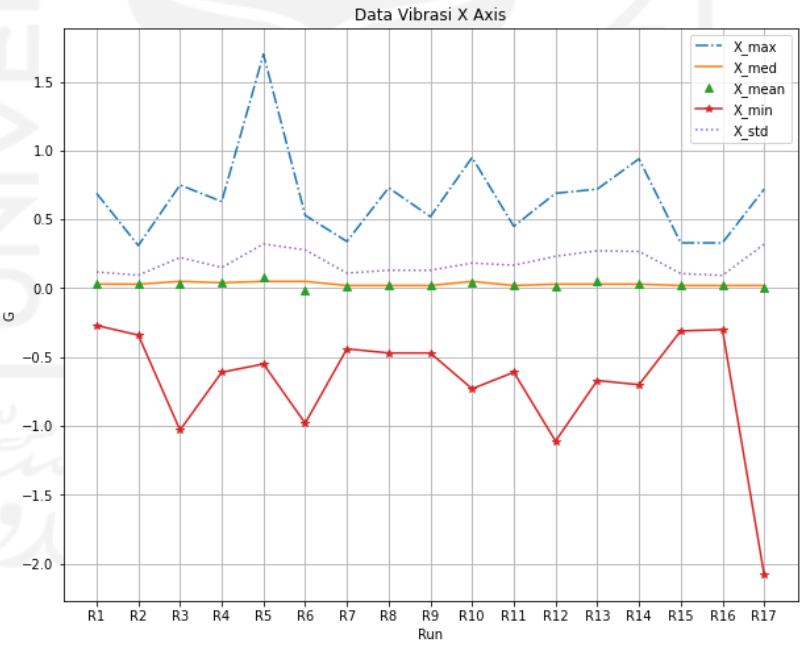
Agar penempatan saklar lebih mudah maka dimasukkan ke dalam sebuah box berukuran 67 x 49 x 23 mm yang ditempelkan di sebelah spion kiri sebagaimana ditampilkan pada gambar 4.4.

4.2 Data Hasil Pengujian

Berisi data hasil rekaman vibrasi tiga *axis* dari tiga lokasi berbeda yang telah melalui proses pemusatan data sehingga setiap *axis* menjadi beberapa variabel; *mean* atau nilai rerata, standar deviasi, *min* atau nilai terendah, *max* atau nilai tertinggi, dan median atau nilai tengah. Proses ini bertujuan untuk mengubah data menjadi sederhana dan mudah deskripsikan, data disajikan dalam satuan G, dimana 1 G setara dengan 9,8 g/ms^2. Setelah data dipusatkan kini terdapat 15 variabel, kemudian data ditampilkan dalam bentuk grafik garis dan dikelompokkan per lokasi pengujian.

Adapun *axis* yang digunakan adalah X, Y, dan Z. *Axis* X merupakan garis longitudinal atau sejajar dengan garis memanjang sepeda motor, sedangkan Y *axis* merupakan sumbu *lateral* atau garis menyamping, dan Z *axis* sebagai garis sumbu *vertical*.

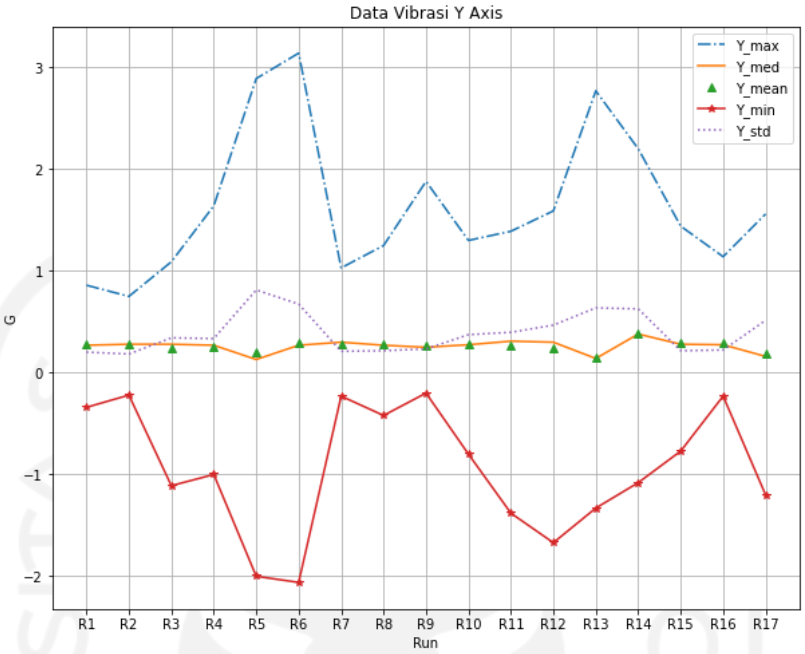
A. Lokasi 1



Gambar 4.5 Grafik Vibrasi X *Axis* Lokasi 1

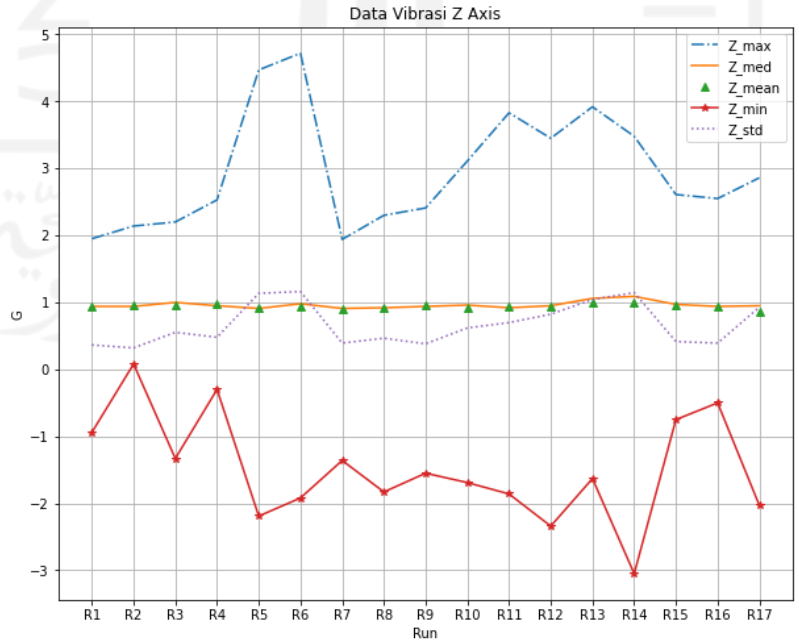
Pada lokasi 1 kami melakukan 17 kali perekaman dan dilakukan di jalan perlintasan kereta api, ditandai oleh label R dan angka setelahnya menunjukkan urutan perekaman. Gambar 4.5. menunjukkan hasil perekaman vibrasi yang terjadi

pada X *axis*. Angka tertinggi ditunjukkan oleh perekaman ke-5 sebesar 1,75 G, dan angka terendah ditunjukkan pada perekaman ke-17 sebesar -2,2 G.



Gambar 4.6 Grafik Vibrasi Y *Axis* Lokasi 1

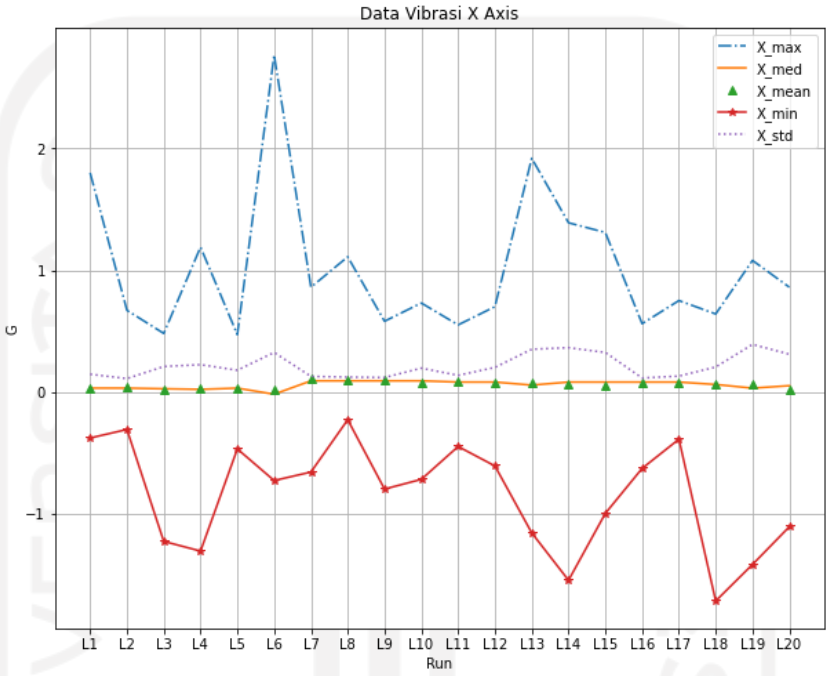
Sedangkan untuk data vibrasi Y *axis* sebagaimana ditampilkan gambar 4.6 angka tertinggi dan terendah ditunjukkan perekaman ke-6 dengan angka 3,1 G dan -2,1 G.



Gambar 4.7 Grafik Vibrasi Z *Axis* Lokasi 1

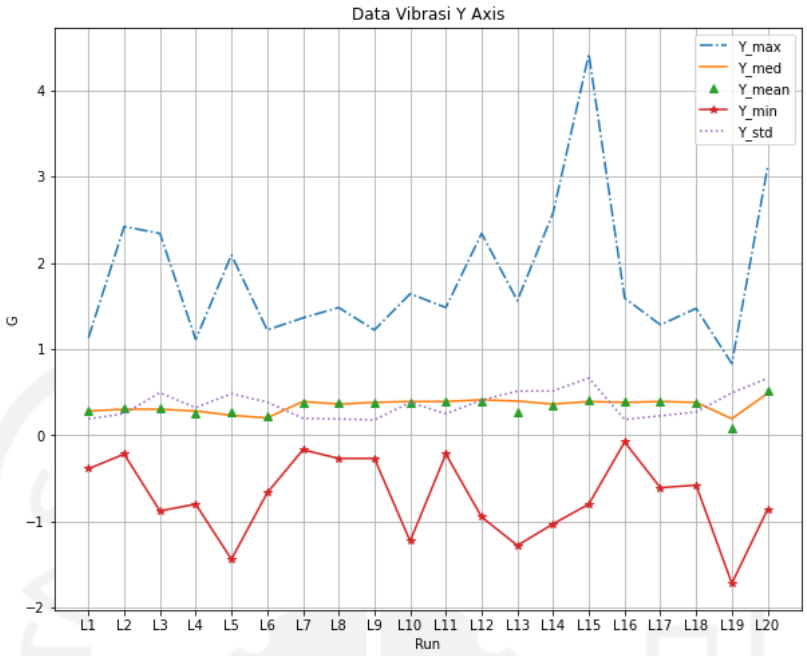
Gambar 4.7 menampilkan data Z *axis* atau sumbu vertikal, nilai tertinggi sebesar 4,8 G ditunjukkan pada perekaman ke-6, dan nilai terendah yakni -3 G dari perekaman ke-14.

B. Lokasi 2



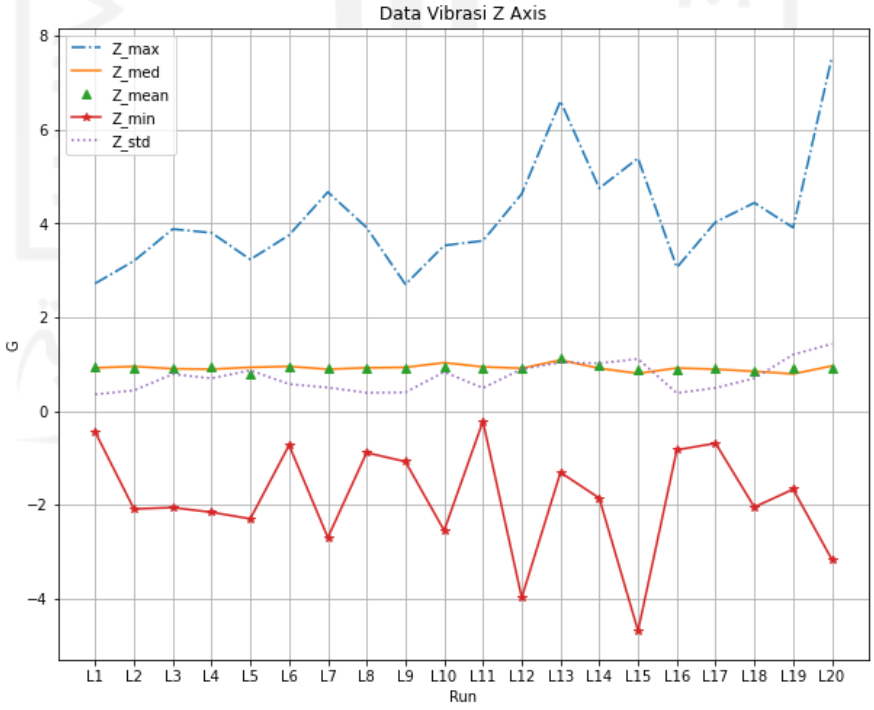
Gambar 4.8 Grafik Vibrasi X *Axis* Lokasi 2

Gambar 4.8 menunjukkan grafik perekaman vibrasi pada jalan dengan permukaan conblock yang dilakukan sebanyak 20 kali. Angka tertinggi ditunjukkan oleh perekaman ke-6 dengan 2,85 G, sedangkan angka terendah didapatkan pada perekaman ke-18 dengan -1,8 G.



Gambar 4.9 Grafik Vibrasi Y *Axis* Lokasi 2

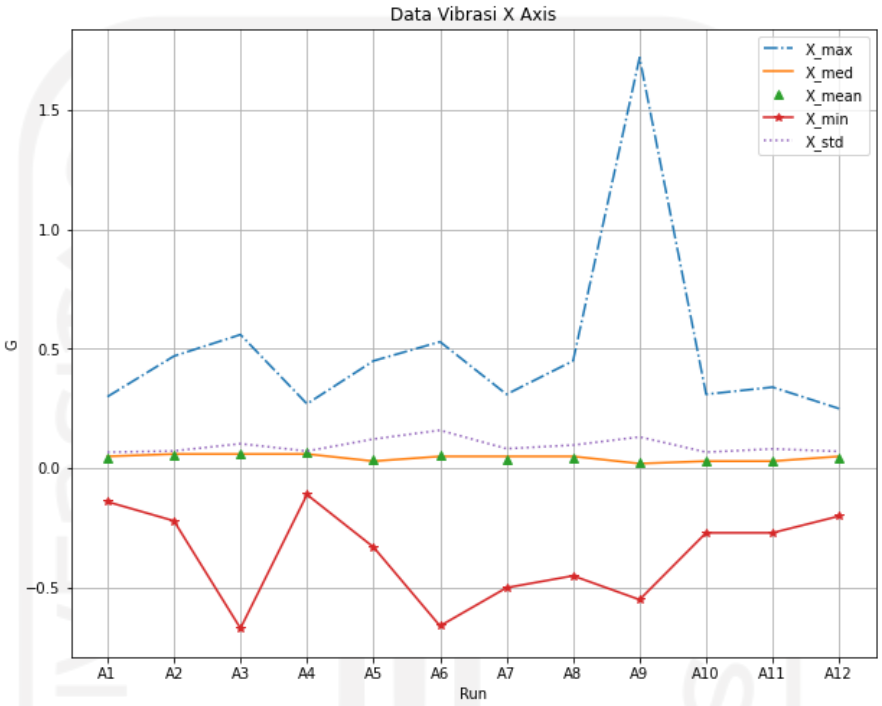
Gambar 4.9 menunjukkan grafik perekaman vibrasi Y *axis*. Angka tertinggi ditunjukkan oleh perekaman ke-15 dengan 4,5 G, sedangkan angka terendah didapatkan pada perekaman ke-19 dengan -1,8 G.



Gambar 4.10 Grafik Vibrasi Z *Axis* Lokasi 2

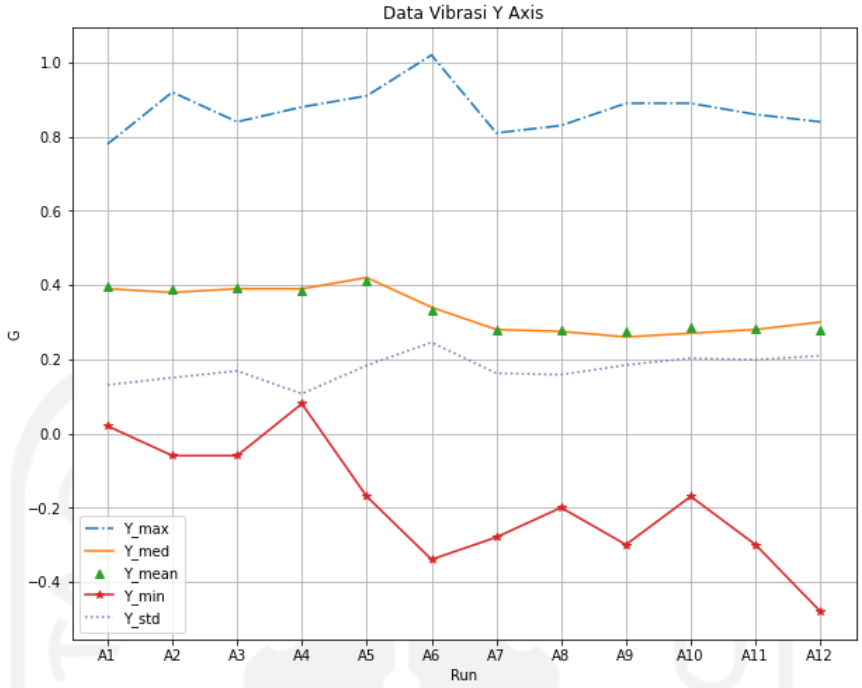
Sebagaimana ditampilkan oleh gambar 4.9, angka tertinggi ditunjukkan oleh perekaman ke-20 dengan 7,8 G, sedangkan angka terendah didapatkan pada perekaman ke-15 dengan -4,5 G.

C. Lokasi 3



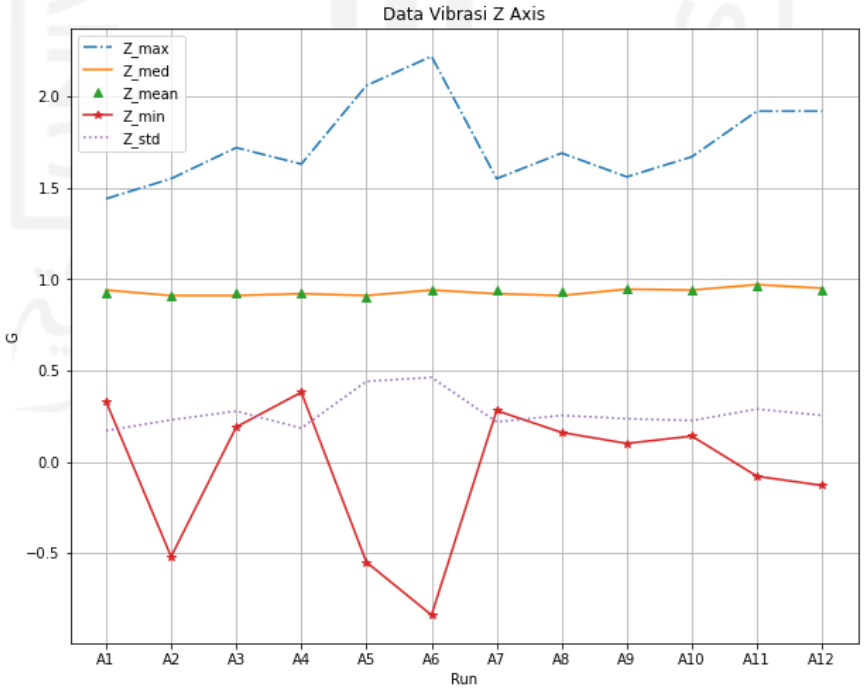
Gambar 4.11 Grafik Vibrasi X *Axis* Lokasi 3

Perekaman vibrasi di lokasi 3 dilakukan sebanyak 12 kali, gambar 4.11 menunjukkan nilai vibrasi tertinggi diperoleh dari pengujian ke-9 yakni sebesar 1,8 G, dan vibrasi terendah terjadi pada perekaman ke-3 sebesar -0,75 G.



Gambar 4.12 Grafik Vibrasi Y *Axis* Lokasi 3

Gambar 4.12 menyajikan nilai vibrasi tertinggi diperoleh dari pengujian ke-6 yakni sebesar 1,1 G, dan vibrasi terendah terjadi pada perekaman ke-12 sebesar -0,5 G.



Gambar 4.13 Grafik Vibrasi Z *Axis* Lokasi 3

Sebagaimana ditunjukkan gambar 4.13 vibrasi tertinggi diperoleh dari pengujian ke-6 yakni sebesar 2,25 G, dan vibrasi terendah terjadi pada perekaman ke-3 sebesar -0,8 G.

Untuk visualisasi kelompok *input* data vibrasi setiap lokasi dapat dilihat pada gambar 4.8 bahwa vibrasi sepeda motor cenderung memiliki karakter yang menyerupai satu sama lain berdasarkan jenis permukaan jalan yang dilalui.

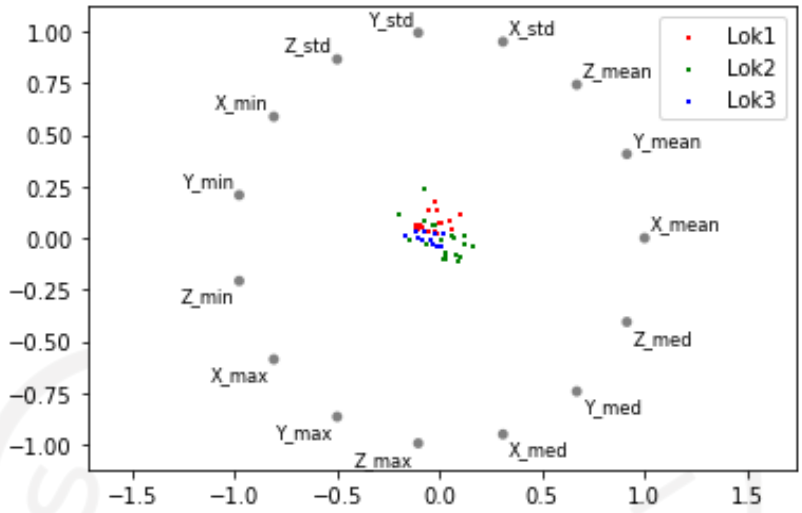
4.3 Analisis dan Pembahasan

Untuk dapat menentukan hasil prediksi *machine learning* diperlukan penentuan target *output*. Target *output* berbentuk tiga kelas kenyamanan yang terbagi menjadi; Rendah, Sedang, dan Tinggi. Proses penentuan target *output* berdasarkan *Self-assessment* oleh penulis. Untuk penentuan data hasil perekaman di lokasi 1, dan 2 tingkat kenyamanan berbanding terbalik dengan tingkat kecepatan, kecepatan rendah memiliki nilai kenyamanan tinggi, kecepatan sedang bernilai sedang, dan kecepatan tinggi bernilai Rendah. Namun untuk data hasil perekaman di lokasi 3 tingkat kenyamanan berbanding lurus dengan tingkat kecepatan, dimana nilai kenyamanan dari kecepatan rendah adalah rendah, kecepatan menengah adalah sedang, dan kecepatan tinggi bernilai tinggi.

Count	X_mean	Y_mean	Z_mean	X_std	Y_std	Z_std	X_min	Y_min	Z_min	X_max	Y_max	Z_max	X_med	Y_med	Z_med	Speed	Loc	Res
245	0.036204	0.280857	0.938000	0.118325	0.202595	0.364250	-0.27	-0.34	-0.94	0.69	0.86	1.95	0.03	0.27	0.94	Lo	Lok1	3
212	0.031509	0.278632	0.952406	0.094906	0.183519	0.320357	-0.34	-0.22	0.08	0.31	0.75	2.14	0.03	0.28	0.94	Lo	Lok1	3
151	0.033642	0.239272	0.951391	0.223566	0.342865	0.552731	-1.03	-1.11	-1.33	0.75	1.09	2.20	0.05	0.28	1.00	Med	Lok1	2
162	0.043333	0.261420	0.977037	0.150684	0.335875	0.479205	-0.61	-1.00	-0.30	0.63	1.64	2.53	0.04	0.27	0.95	Med	Lok1	2
85	0.076824	0.205294	0.912824	0.321974	0.813121	1.134543	-0.55	-2.00	-2.19	1.70	2.89	4.47	0.05	0.13	0.91	Hi	Lok1	1

Gambar 4.14 Kolom Nilai Target

Penempatan target *output* sendiri berada dalam satu kolom data set baru yang diberi nama ‘Res’ yang berada di urutan paling kanan *dataframe* sebagaimana ditampilkan pada gambar 4.14.

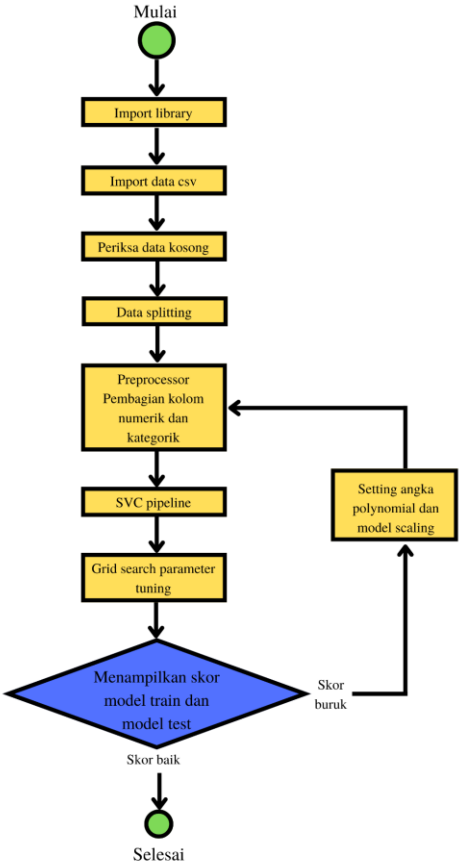


Gambar 4.15 Grafik Persebaran Data Vibrasi

Sebagaimana dapat dilihat pada gambar 4.15 bahwa pembuatan model data vibrasi tidak cukup hanya menggunakan visualisasi grafik, melainkan diperlukan metode *machine learning* untuk mengolahnya. Program *machine learning* untuk mengolah data vibrasi menggunakan dua jenis metode *support vector machine* yang berbeda; pertama *classification*, dan *regression*. Untuk menjalankan program di Python diperlukan instalasi beberapa *library*, daftar *library* terdapat di lampiran 3.

A. Support Vector Classification

Sesuai namanya, model *support vector machine* ini menghasilkan *output* berdasarkan prediksi kelas label atau kelompok data. Data yang telah diproses dari 3 lokasi perekaman digabungkan ke dalam satu data dalam format csv. Seperti yang ditampilkan oleh gambar 4.16, setelah import *library* yang diperlukan, data vibrasi dipanggil ke dalam program dan ditampilkan dalam bentuk *dataframe*. Untuk memastikan keterisian kolom data, perlu menjalankan program *plot missing value*. Setelah memastikan seluruh kolom data terisi, data dipisah menjadi dua kelompok. Data *train* atau pengujian, dan data *test*. Secara keseluruhan data berjumlah 49, rasio pemisahan data atau data *splitting* adalah 80 banding 20. Dimana 80 persen untuk dijadikan data *train*, dan sisanya untuk data *test*. Sehingga data *train* berjumlah 39, dan data *test* 10.



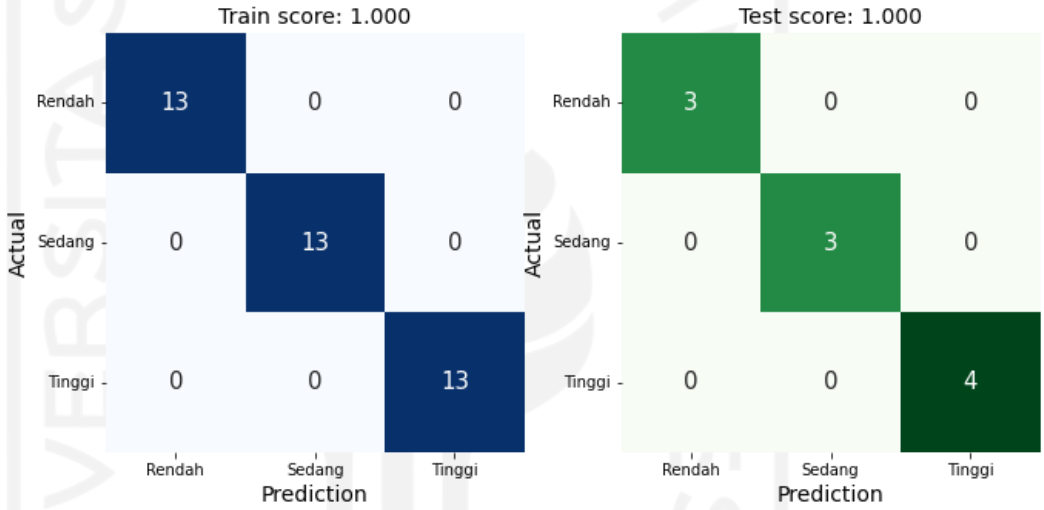
Gambar 4.16 Diagram Alir *Support Vector Classification*

Dalam *data splitting* juga terjadi pemisahan kolom ‘Res’ sebagai label prediksi *output*. Kemudian program *preprocessor* memisahkan antara kolom yang berisi data numerik dan kategorik. Untuk memudahkan proses pengolahan, data dalam kolom kategorik diadaptasikan menjadi angka atau kode menggunakan fitur *encoder onehot*.

Setelah tahap *preprocessor*, data diolah menggunakan pipeline SVC. Untuk memaksimalkan hasil digunakan beberapa pengaturan pada proses *gridsearch*, yaitu penggunaan svm *params*, nilai *random state* 42 dan *cross validation* sebanyak 3. Setelah semua proses tersebut dijalankan akan tampil hasil pengolahan model *train* dan model *test machine learning*. Model pertama

menunjukkan akurasi sebesar 100% untuk model *train*, dan 20% untuk model *test*. Nilai C sebesar 10, dan gamma sebesar 0,1.

Untuk meningkatkan akurasi, maka diberikan fitur *scaling* pada tahap *preprocessor*. Pada kolom numerik digunakan *scaling* jenis *robust*. Kemudian setelah program dijalankan, nilai model *train* sebesar 100%, dan model *test* naik menjadi 100%. Nilai C sebesar 100, dan gamma 0,01. Nilai C yang sangat tinggi dan gamma yang rendah disebabkan variasi data yang sangat tinggi untuk label data yang tersedia. Nilai tersebut dirasa memuaskan sehingga program dicukupkan.



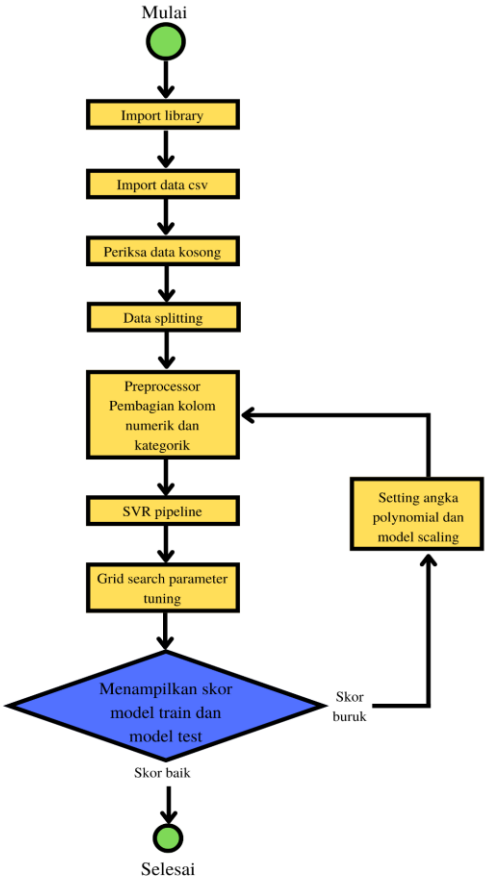
Gambar 4.17 *Confusion Matrix* Klasifikasi

Gambar 4.17 menunjukkan visualisasi akurasi kedua model, untuk model *test* hanya terdapat satu kesalahan prediksi dari keseluruhan data *test*. Dikarenakan jumlah data *test* yang tidak begitu besar, satu kesalahan menurunkan nilai akurasi sebesar 10%.

B. Support Vector Regression

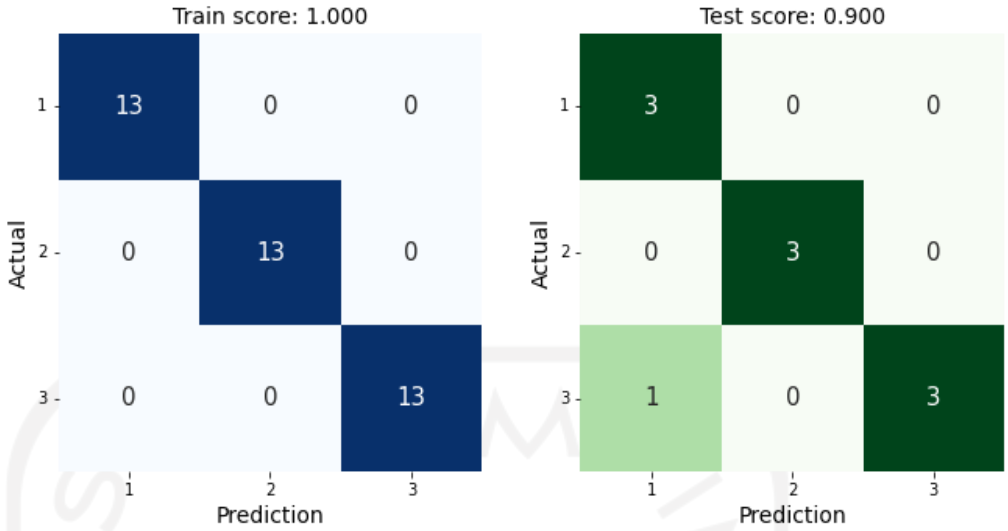
Program *machine learning* ini menghasilkan *output* berdasarkan prediksi nilai berkelanjutan atau tren. Karena jenis data *ouput* berupa tiga label klasifikasi, maka digunakan metode regresi jenis *logistic*. Sebagaimana ditampilkan oleh gambar 4.18, alur program *logistic regression* tidak jauh berbeda dengan klasifikasi. Perbedaan terdapat pada penentuan jenis *pipeline*, yakni jenis *Logistic*

Regression. Kemudian pada pengaturan *randomized search* menggunakan iterasi sejumlah 50, dan *cross validation* sebanyak 5. Setelah program dijalankan, model *train* menampilkan akurasi sebesar 97%, dan 80% untuk model *test*. Nilai C sebesar 17,7.



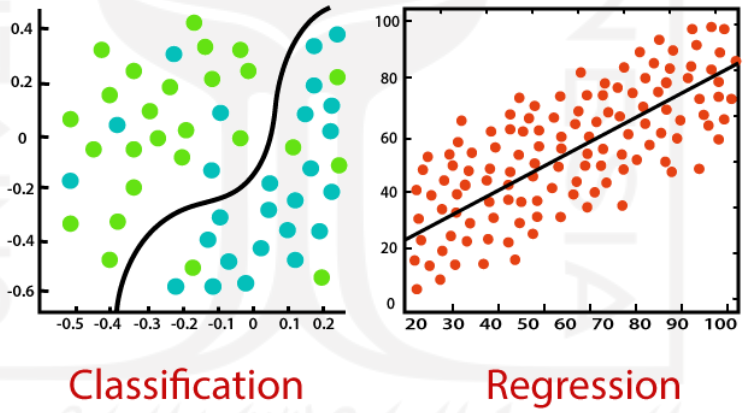
Gambar 4.18 Diagram Alir Regresi

Seperti pada proses *Classification*, kemudian digunakan fitur *scaling robust* pada tahap *preprocessor*. Setelah program dijalankan, model *train* menunjukkan 100% akurasi, dan model *test* 90%. Nilai C 3,9.



Gambar 4.19 *Confusion* Matrix Regresi

Gambar 4.19 menunjukkan visualisasi perbandingan model *train* dan model *test* dari metode regresi.



Gambar 4.20 Perbandingan Klasifikasi dan Regresi (javatpoint, 2022)

Program klasifikasi dan regresi menunjukkan hasil akurasi model yang sedikit berbeda sebagaimana ditunjukkan gambar 4.20. Metode klasifikasi membuat model yang memprediksi dengan mengelompokkan data berdasarkan label, sedangkan regresi menggunakan data sebagai acuan untuk memprediksi tren atau nilai berkelanjutan.

BAB 5
PENUTUP

5.1 Kesimpulan

1. Telah dibuat alat perekam vibrasi portabel, dan sederhana menggunakan komponen *gyroscope* yang diadaptasikan untuk merekam vibrasi pada sepeda motor.
2. Telah dibuat dua jenis program *machine learning* untuk memproses data vibrasi, yakni *support vector classification* dan *support vector logistic regression*.
3. Kedua program *machine learning* menghasilkan akurasi yang identik, namun memiliki nilai konstanta C atau parameter penalti yang berbeda. Nilai C yang jauh lebih tinggi pada hasil klasifikasi menunjukkan kecilnya potensi misklasifikasi *data training* sehingga menjadikan model ini lebih bisa diandalkan.

5.2 Saran atau Penelitian Selanjutnya

Berdasarkan penelitian yang telah dilakukan, ada beberapa hal yang layak dipertimbangkan untuk penelitian selanjutnya:

1. Menggunakan jenis sumber energi / baterai yang lebih kompak.
2. Ditambahkan layar atau monitor yang dapat menampilkan proses perekaman data vibrasi.
3. Diperlukan keterlibatan beberapa individu dengan disiplin ilmu spesifik; yakni teknik mesin untuk meningkatkan analisa rambatan vibrasi, dan *data scientist* untuk pengolahan data / *machine learning*.
4. Mengembangkan aplikasi hasil penelitian yang telah dilaksanakan.

DAFTAR PUSTAKA

[1] Chika, O., dan Harus, L. G., "Analisa Kenyamanan Kendaraan Roda Dua dengan Pemodelan Pengendara sebagai Sistem Multi D.O.F," *Jurnal Teknik Pomits*, Vol. 3, No. 2, 2014.

[2] William T. Thompson,"Theory Vibrations and Application," Erlangga, Jakarta, 1986

[3] Riyanto, Andik, "Simulasi Ride Comfort kendaraan roda dua dengan sistem multi dof," Teknik Mesin ITS, Surabaya, 2008.

[4] Suhandoko, "Analisis Getaran Pada Sistem Suspensi Kendaraan Roda Dua (Yamaha Jupiter Z 2004) Menggunakan Simulasi Software Matlab 6.5," Teknik Mesin, Fakultas Teknik, Universitas Muhammadiyah, Surakarta, 2014.

[5] Sukendi, I., Ikhwansyah, dan Suherman, "Analisa Karakteristik Getaran dan Machine Learning Untuk Deteksi Dini Kerusakan Bearing," *Widya Teknika*, Vol. 23 No. 2, 2015.

[6] Ickwanda, "Simulasi Getaran Pada Piringan Tunggal Akibat Perubahan Putaran," *Tugas Akhir*, Teknik Mesin, Fakultas Teknik, Universitas Muhammadiyah Sumatera Utara, Medan, 2019.

[7] Cooper, J. O., Heron, T.E., & Heward, W. L, "Applied Behavior Analysis," Second Edition, Pearson Prentice, New York, 2007.

[8] Hazen, R. M., & Trefil, J, "Achieving Scientific Literacy," Science Matters, Anchor Books, New York, 1990.

[9] Edi, S. N., "Karakterisasi Aspek Permukaan Menggunakan Fitur Berbasis Fourier Transform," Tugas Akhir, Teknik Mesin, Fakultas Teknologi Industri, Universitas Islam Indonesia, Yogyakarta, 2021.

[10] Simran, G, "Brief Study on Machine Learning," Nepal College of Information and Technology, Latipur, 2020.

[11] P. Kshirsagar and D. S. Akojwar, "Hybrid Heuristic Optimization for Benchmark Datasets," International Journal of Computer Application, Vol. 146-No, 7, 2016.

LAMPIRAN 1

SKETSA PROGRAM ARDUINO

```
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include "RTClib.h"

const int chipSelect = 10;

int jam, menit, detik;
String waktu, tanggal;
RTC_DS1307 RTC;
char
daysOfTheWeek[7][21]={"Sunday","Monday","Tuesday","
Wednesday","Thursday","Friday","Saturday"};
File myFile;

int ADXL345 = 0x53; // The ADXL345 sensor I2C address
float X_out, Y_out, Z_out;  // Outputs

int sw = 2;

void setup() {
  // put your setup code here, to run once:

  Wire.begin();
  pinMode(sw, INPUT);
  Serial.begin(9600); // Initiate serial communication for
printing the results on the Serial monitor
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native
USB port only
  }

  Serial.print("Initializing SD card...");

  // see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {
```

```
  Serial.println("Card failed, or not present");
  // don't do anything more:
  return;
 }
 Serial.println("card initialized.");
 if (SD.exists("datavib.csv")) {
  Serial.println("datavib.txt exists.");
 } else {
  Serial.println("datavib.txt doesn't exist.");
 }
 Serial.println("Creating example.txt...");
 myFile = SD.open("datavib.csv", FILE_WRITE);
 myFile.close();

 if (SD.exists("datavib.csv")) {
  Serial.println("datavib.txt exists.");
 } else {
  Serial.println("datavib.txt doesn't exist.");
 }

 #ifndef ESP8266
 while (!Serial); // wait for serial port to connect. Needed
for native USB
#endif

 if (! rtc.begin()) {
  Serial.println("Couldn't find RTC");
  Serial.flush();
  abort();
 }

 if (! rtc.isrunning()) {
  Serial.println("RTC is NOT running, let's set the time!");
  // When time needs to be set on a new device, or after a
power loss, the
  // following line sets the RTC to the date & time this
sketch was compiled
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  // This line sets the RTC with an explicit date & time, for
example to set
  // January 21, 2014 at 3am you would call:
```

```
  // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
 }

 // When time needs to be re-set on a previously configured device, the
 // following line sets the RTC to the date & time this sketch was compiled
 // rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
 // This line sets the RTC with an explicit date & time, for example to set
 // January 21, 2014 at 3am you would call:
 //rtc.adjust(DateTime(2021, 10, 28, 10, 06, 0));



 // Set ADXL345 in measuring mode
 Wire.beginTransmission(ADXL345); // Start communicating with the device
 Wire.write(0x2D); // Access/ talk to POWER_CTL Register - 0x2D
 // Enable measurement
 Wire.write(8); // (8dec -> 0000 1000 binary) Bit D3 High for measuring enable
 Wire.endTransmission();
 delay(10);

 //8 G resolution SET
 Wire.beginTransmission(ADXL345); // Start communicating with the device
 Wire.write(0x31); // Access/ talk to Register 0x31—DATA_FORMAT
 // Enable measurement
 Wire.write(2); // (2dec -> 0000 0010 binary) Bit D1 High for +/-8g enable
 Wire.endTransmission();
 delay(10);


 // This code goes in the SETUP section
// Off-set Calibration
```

```
  //X-AXIS
  Wire.beginTransmission(ADXL345);
  Wire.write(0x1E);  // X-AXIS OFFSET REGISTER
  Wire.write(-6);
  Wire.endTransmission();
  delay(10);
  //Y-AXIS
  Wire.beginTransmission(ADXL345);
  Wire.write(0x1F); // Y-AXIS OFFSET REGISTER
  Wire.write(-4);
  Wire.endTransmission();
  delay(10);

  //Z-AXIS
  Wire.beginTransmission(ADXL345);
  Wire.write(0x20); // Z-AXIS OFFSET REGISTER
  Wire.write(3);
  Wire.endTransmission();
  delay(10);

}

void loop() {
  // put your main code here, to run repeatedly:
  int kond = digitalRead (sw);
  //Serial.println (kond);
  //delay (1000);
  if (kond == 1) {

  DateTime now = rtc.now();
  String dataString = "";
  String tahun = String(now.year(), DEC);
  String('/');
  String bulan = String(now.month(), DEC);
  String('/');
  String hari = String(now.day(), DEC);
  String("(");
  String(daysOfTheWeek[now.dayOfTheWeek()]);
  String(")");
  String jam = String(now.hour(), DEC);
  String(':');
```

```
  String menit = String(now.minute(), DEC);
  String(':');
  String detik = String(now.second(), DEC);
  String waktu = String("(" + tahun + "/" + bulan + "/" + hari
+ ")" + "," + jam + ":" + menit + ":" + detik);


  //Serial.println(waktu);
  //Serial.println(now.second(),DEC);



   // === Read accelerometer data === //
  Wire.beginTransmission(ADXL345);
  Wire.write(0x32); // Start with register 0x32
(ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(ADXL345, 6, true); // Read 6 registers
total, each axis value is stored in 2 registers
  X_out = ( Wire.read()| Wire.read() << 8); // X-axis value
  X_out = X_out/64; //For a range of +-2g, we need to divide
the raw values by 64, according to the datasheet
  Y_out = ( Wire.read()| Wire.read() << 8); // Y-axis value
  Y_out = Y_out/64;
  Z_out = ( Wire.read()| Wire.read() << 8); // Z-axis value
  Z_out = Z_out/64;



   // read three axis and append to the string:
   // make a string for assembling the data to log:String
dataString = "";



   // open the file. note that only one file can be open at a
time,
   // so you have to close this one before opening another.
   myFile = SD.open("datavib.csv", FILE_WRITE);

   // if the file is available, write to it:
   if (myFile) {
```

```
    myFile.println(waktu + "," + X_out + "," + Y_out + "," +
Z_out);
    myFile.close();
    // print to the serial port too:
    Serial.println(waktu + ", Getaran: " + X_out + "," +
Y_out + "," + Z_out);
   }
   // if the file isn't open, pop up an error:
   else {
    Serial.println("error opening...");
   }

   delay(10);
 }
 else {
  Serial.println ("nyalakan saklar");
 }
}
```

LAMPIRAN 2

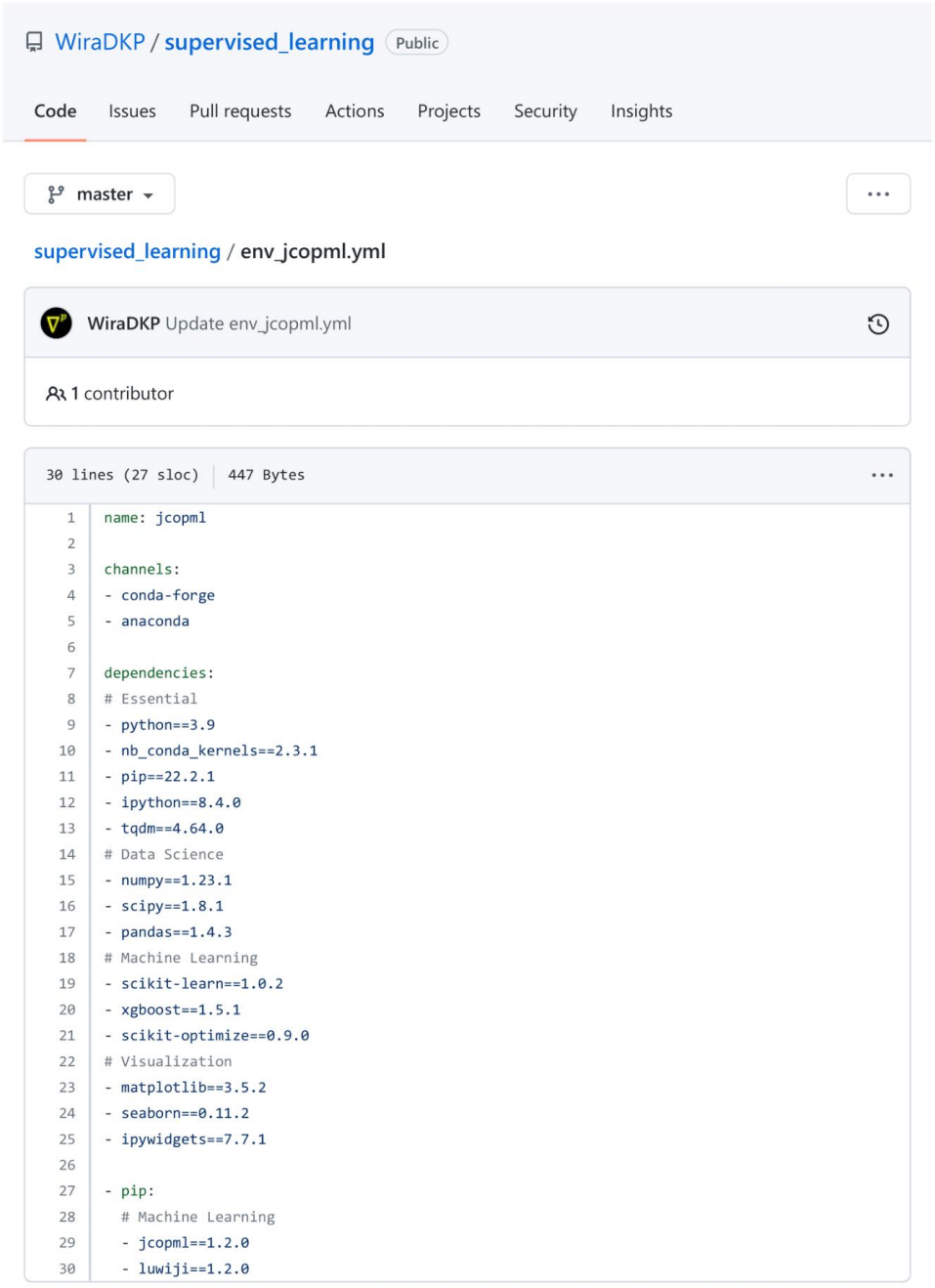
SKETSA PROGRAM KALIBRASI AXIS

```
#include <Wire.h>  // Wire library - used for I2C communication
int ADXL345 = 0x53; // The ADXL345 sensor I2C address
float X_out, Y_out, Z_out;  // Outputs
float roll,pitch,rollF,pitchF=0;
void setup() {
  Serial.begin(9600); // Initiate serial communication for printing the results on the Serial monitor

  Wire.begin(); // Initiate the Wire library
  // Set ADXL345 in measuring mode
  Wire.beginTransmission(ADXL345); // Start communicating with the device
  Wire.write(0x2D); // Access/ talk to POWER_CTL Register - 0x2D
  // Enable measurement
  Wire.write(8); // Bit D3 High for measuring enable (8dec -> 0000 1000 binary)
  Wire.endTransmission();
  delay(10);
  //Off-set Calibration
  //X-axis
  Wire.beginTransmission(ADXL345);
  Wire.write(0x1E);
  Wire.write(1);
  Wire.endTransmission();
  delay(10);
  //Y-axis
  Wire.beginTransmission(ADXL345);
  Wire.write(0x1F);
  Wire.write(-2);
  Wire.endTransmission();
  delay(10);
  //Z-axis
  Wire.beginTransmission(ADXL345);
  Wire.write(0x20);
```

```
  Wire.write(-9);
  Wire.endTransmission();
  delay(10);
}
void loop() {
  // === Read acceleromter data === //
  Wire.beginTransmission(ADXL345);
  Wire.write(0x32); // Start with register 0x32
(ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(ADXL345, 6, true); // Read 6 registers
total, each axis value is stored in 2 registers
  X_out = ( Wire.read() | Wire.read() << 8); // X-axis value
  X_out = X_out / 256; //For a range of +-2g, we need to divide
the raw values by 256, according to the datasheet
  Y_out = ( Wire.read() | Wire.read() << 8); // Y-axis value
  Y_out = Y_out / 256;
  Z_out = ( Wire.read() | Wire.read() << 8); // Z-axis value
  Z_out = Z_out / 256;
  // Calculate Roll and Pitch (rotation around X-axis,
rotation around Y-axis)
  roll = atan(Y_out / sqrt(pow(X_out, 2) + pow(Z_out, 2))) *
180 / PI;
  pitch = atan(-1 * X_out / sqrt(pow(Y_out, 2) + pow(Z_out,
2))) * 180 / PI;
  // Low-pass filter
  rollF = 0.94 * rollF + 0.06 * roll;
  pitchF = 0.94 * pitchF + 0.06 * pitch;
  Serial.print(rollF);
  Serial.print("/");
  Serial.println(pitchF);
}
```

LAMPIRAN 3

LIBRARY SVM



WiraDKP / supervised_learning Public

Code Issues Pull requests Actions Projects Security Insights

master

supervised_learning / env_jcopml.yml

WiraDKP Update env_jcopml.yml

1 contributor

30 lines (27 sloc) | 447 Bytes

```yaml
name: jcopml

channels:
- conda-forge
- anaconda

dependencies:
# Essential
- python==3.9
- nb_conda_kernels==2.3.1
- pip==22.2.1
- ipython==8.4.0
- tqdm==4.64.0
# Data Science
- numpy==1.23.1
- scipy==1.8.1
- pandas==1.4.3
# Machine Learning
- scikit-learn==1.0.2
- xgboost==1.5.1
- scikit-optimize==0.9.0
# Visualization
- matplotlib==3.5.2
- seaborn==0.11.2
- ipywidgets==7.7.1

- pip:
  # Machine Learning
  - jcopml==1.2.0
  - luwiji==1.2.0
```

LAMPIRAN 4

PROGRAM KLASIFIKASI

Import library

In [1]:

```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedShuffleSplit,
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.svm import SVC

from jcopml.pipeline import num_pipe, cat_pipe
from jcopml.utils import save_model, load_model
from jcopml.plot import plot_missing_value, plot_classification_report, plot_confusion_matr
from jcopml.feature_importance import mean_score_decrease
from jcopml.tuning import grid_search_params as gsp
```

Import data

In [2]:

```
df = pd.read_csv("ALL.csv", index_col="Code")
df.head()
```

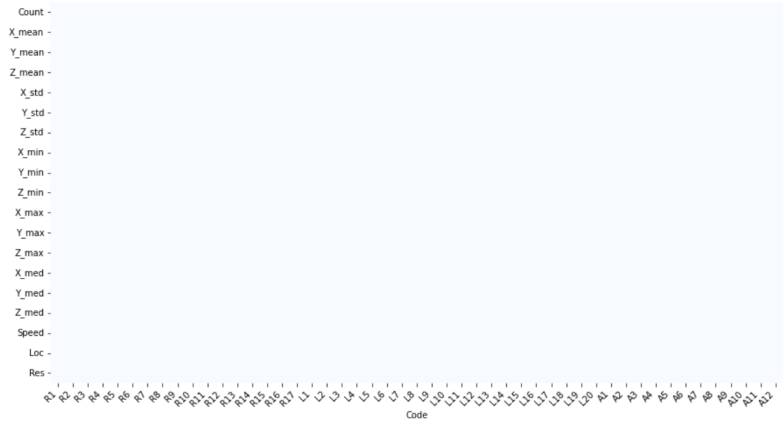
Out[2]:

Code	Count	X_mean	Y_mean	Z_mean	X_std	Y_std	Z_std	X_min	Y_min	Z_mi
R1	245	0.036204	0.280857	0.938000	0.118325	0.202595	0.364250	-0.27	-0.34	-0.9
R2	212	0.031509	0.278632	0.952406	0.094906	0.183519	0.320357	-0.34	-0.22	0.(
R3	151	0.033642	0.239272	0.951391	0.223566	0.342865	0.552731	-1.03	-1.11	-1.3
R4	162	0.043333	0.261420	0.977037	0.150684	0.335875	0.479205	-0.61	-1.00	-0.3
R5	85	0.076824	0.205294	0.912824	0.321974	0.813121	1.134543	-0.55	-2.00	-2.1

Cek data kosong

In [3]:

```
plot_missing_value(df)
```



Data splitting

In [4]:

```
X = df.drop(columns="Res")
y = df.Res

X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, stratify=y, rando
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[4]:

```
((39, 18), (10, 18), (39,), (10,))
```

In [5]:

```
X_train.columns
```

Out[5]:

```
Index(['Count', 'X_mean', 'Y_mean', 'Z_mean', 'X_std', 'Y_std', 'Z_std',
       'X_min', 'Y_min', 'Z_min', 'X_max', 'Y_max', 'Z_max', 'X_med', 'Y_me
d',
       'Z_med', 'Speed', 'Loc'],
      dtype='object')
```

Buat program SVC

In [9]:

```
preprocessor = ColumnTransformer([
    ('numeric', num_pipe(), ["Count", "X_mean", "Y_mean", "Z_mean", "X_std", "Y_std", "Z_st
                                                "X_min", "Y_min", "Z_min", "X_max", "Y_max",
                                                "X_med", "Y_med", "Z_med"]),
    ('categoric', cat_pipe(encoder='onehot'), ["Speed", "Loc"]),
])


pipeline = Pipeline([
    ('prep', preprocessor),
    ('algo', SVC(max_iter=500))
])



model = GridSearchCV(pipeline, gsp.svm_params, cv= 3, n_jobs=-1, verbose=1)
model.fit(X_train, y_train)

print(model.best_params_)
print(model.score(X_train, y_train), model.best_score_, model.score(X_test, y_test))
```

```
Fitting 3 folds for each of 49 candidates, totalling 147 fits
{'algo__C': 10.0, 'algo__gamma': 0.1}
1.0 0.3333333333333333 0.2
```

Scaling

In [10]:

```
preprocessor = ColumnTransformer([
    ('numeric', num_pipe(scaling= 'robust'), ["Count", "X_mean", "Y_mean", "Z_mean", "X_std
                                              "X_min", "Y_min", "Z_min", "X_max", "Y_max",
                                              "X_med", "Y_med", "Z_med"]),
    ('categoric', cat_pipe(encoder='onehot'), ["Speed", "Loc"]),
])


pipeline = Pipeline([
    ('prep', preprocessor),
    ('algo', SVC(max_iter=500))
])



model = GridSearchCV(pipeline, gsp.svm_params, cv= 3, n_jobs=-1, verbose=1)
model.fit(X_train, y_train)

print(model.best_params_)
print(model.score(X_train, y_train), model.best_score_, model.score(X_test, y_test))
```

```
Fitting 3 folds for each of 49 candidates, totalling 147 fits
{'algo__C': 100.0, 'algo__gamma': 0.01}
1.0 0.8461538461538461 1.0
```

In [13]:

Out[13]:

Code	Count	X_mean	Y_mean	Z_mean	X_std	Y_std	Z_std	X_min	Y_min	Z_mi
R1	245	0.036204	0.280857	0.938000	0.118325	0.202595	0.364250	-0.27	-0.34	-0.9
R2	212	0.031509	0.278632	0.952406	0.094906	0.183519	0.320357	-0.34	-0.22	0.0
R3	151	0.033642	0.239272	0.951391	0.223566	0.342865	0.552731	-1.03	-1.11	-1.3
R4	162	0.043333	0.261420	0.977037	0.150684	0.335875	0.479205	-0.61	-1.00	-0.3
R5	85	0.076824	0.205294	0.912824	0.321974	0.813121	1.134543	-0.55	-2.00	-2.1

Visualisasi hasil

Hasil Train vs Test

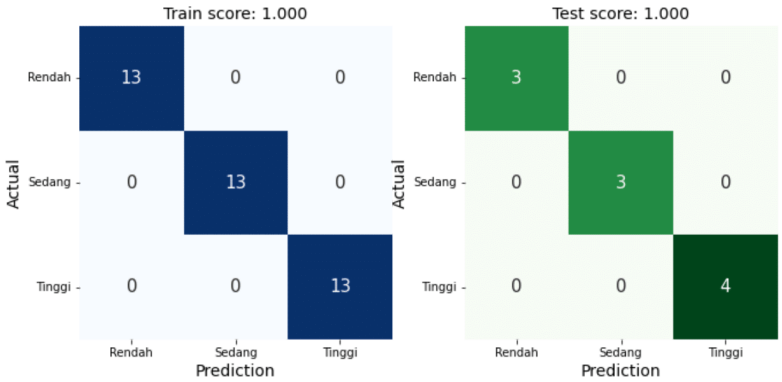
In [13]:

```
plot_classification_report(X_train, y_train, X_test, y_test, model, report= True)
```

```
Train report
              precision    recall  f1-score   support

      Rendah       1.00      1.00      1.00        13
      Sedang       1.00      1.00      1.00        13
      Tinggi       1.00      1.00      1.00        13

    accuracy                           1.00        39
   macro avg       1.00      1.00      1.00        39
weighted avg       1.00      1.00      1.00        39


Test report
              precision    recall  f1-score   support

      Rendah       1.00      1.00      1.00         3
      Sedang       1.00      1.00      1.00         3
      Tinggi       1.00      1.00      1.00         4

    accuracy                           1.00        10
   macro avg       1.00      1.00      1.00        10
weighted avg       1.00      1.00      1.00        10
```

In [14]:

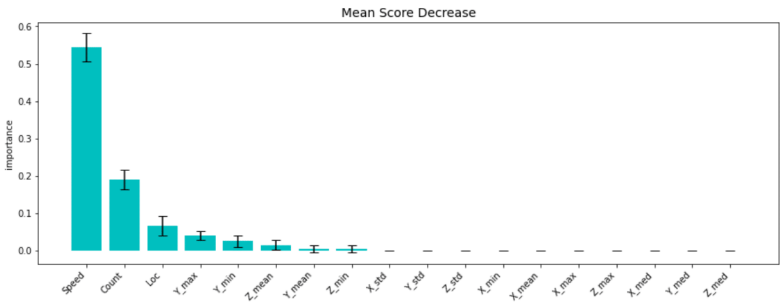
```
from jcopml.plot import plot_confusion_matrix
plot_confusion_matrix(X_train, y_train, X_test, y_test, model)
```



Urutan pengaruh variabel

In [15]:

```
df_imp = mean_score_decrease(X_train, y_train, model, plot=True, topk=18)
```



In [20]:

```
df1 = pd.read_csv("Predict.csv", index_col="Code")
df1.drop(columns="Res", inplace=True)
df1.head()
```

Out[20]:

Code	Count	X_mean	Y_mean	Z_mean	X_std	Y_std	Z_std	X_min	Y_min	Z_mi
R1	245	0.036204	0.280857	0.938000	0.118325	0.202595	0.364250	-0.27	-0.34	-0.9
R2	212	0.031509	0.278632	0.952406	0.094906	0.183519	0.320357	-0.34	-0.22	0.0
R3	151	0.033642	0.239272	0.951391	0.223566	0.342865	0.552731	-1.03	-1.11	-1.3
R4	162	0.043333	0.261420	0.977037	0.150684	0.335875	0.479205	-0.61	-1.00	-0.3
R5	85	0.076824	0.205294	0.912824	0.321974	0.813121	1.134543	-0.55	-2.00	-2.1

In [21]:

```
X_pred = df1
```

LAMPIRAN 5

PROGRAM REGRESI

In [1]:

```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LogisticRegression

from jcopml.pipeline import num_pipe, cat_pipe
from jcopml.utils import save_model, load_model
from jcopml.plot import (plot_missing_value,plot_classification_report, plot_confusion_matr
plot_roc_curve, plot_pr_curve, plot_association_matrix)
from jcopml.feature_importance import mean_score_decrease
from jcopml.tuning import random_search_params as rsp
```

In [2]:

```
df = pd.read_csv("ALL.csv", index_col="Code")
df.tail()
```

Out[2]:

Code	Count	X_mean	Y_mean	Z_mean	X_std	Y_std	Z_std	X_min	Y_min	Z_mi
A8	666	0.042508	0.277688	0.932132	0.097371	0.158339	0.253786	-0.45	-0.20	0.1
A9	532	0.023571	0.276560	0.946297	0.131494	0.184333	0.235614	-0.55	-0.30	0.1
A10	546	0.033755	0.287949	0.937692	0.067815	0.202716	0.225260	-0.27	-0.17	0.1
A11	344	0.032442	0.281512	0.963866	0.081254	0.198640	0.288050	-0.27	-0.30	-0.(
A12	359	0.045320	0.278468	0.938357	0.070978	0.209342	0.253897	-0.20	-0.48	-0.1

In [3]:

```
df.Res.value_counts()
```

Out[3]:

```
Tinggi    17
Sedang    16
Rendah    16
Name: Res, dtype: int64
```

In [4]:

```
X = df.drop(columns="Res")
y = df.Res

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[4]:

```
((39, 18), (10, 18), (39,), (10,))
```

In [5]:

```
X_train.head()
```

Out[5]:

Code	Count	X_mean	Y_mean	Z_mean	X_std	Y_std	Z_std	X_min	Y_min	Z_mi
L20	110	0.017727	0.522636	0.910818	0.308800	0.659861	1.429980	-1.11	-0.86	-3.1
A5	271	0.038672	0.413358	0.897934	0.122543	0.183283	0.439547	-0.33	-0.17	-0.5
L6	121	0.017190	0.225041	0.940331	0.323556	0.382827	0.572678	-0.73	-0.67	-0.7
R5	85	0.076824	0.205294	0.912824	0.321974	0.813121	1.134543	-0.55	-2.00	-2.1
R8	164	0.020793	0.287073	0.929085	0.130990	0.216373	0.465321	-0.47	-0.42	-1.8

In [6]:

```
X_train.columns
```

Out[6]:

```
Index(['Count', 'X_mean', 'Y_mean', 'Z_mean', 'X_std', 'Y_std', 'Z_std',
       'X_min', 'Y_min', 'Z_min', 'X_max', 'Y_max', 'Z_max', 'X_med', 'Y_me
d',
       'Z_med', 'Speed', 'Loc'],
      dtype='object')
```

In [7]:

```
preprocessor = ColumnTransformer([
    ('numeric', num_pipe(), ["Count", "X_mean", "Y_mean", "Z_mean", "X_std", "Y_std", "Z_st
    ('categoric', cat_pipe(encoder='onehot'), ["Speed", "Loc"]),
])



pipeline = Pipeline([
    ('prep', preprocessor),
    ('algo', LogisticRegression(solver='lbfgs', n_jobs=-1, random_state=42))
])


model = RandomizedSearchCV(pipeline, rsp.logreg_params, cv=5, n_iter=50, n_jobs=-1, verbose
model.fit(X_train, y_train)

print(model.best_params_)
print(model.score(X_train, y_train), model.best_score_, model.score(X_test, y_test))
```

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
{'algo__C': 17.71884735480683, 'algo__fit_intercept': False}
0.9743589743589743 0.95 0.8
```

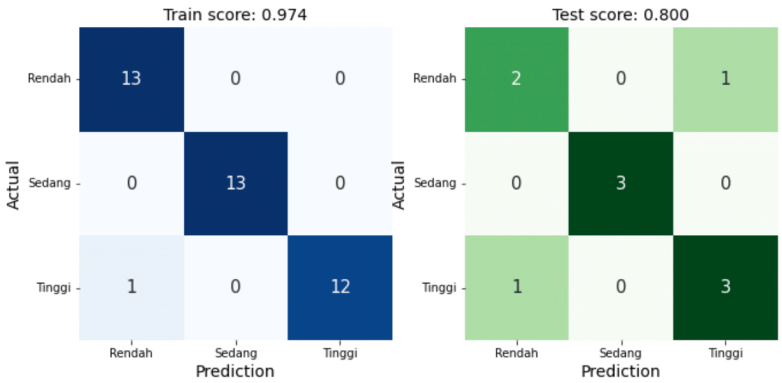
In [8]:

```
plot_classification_report(X_train, y_train, X_test, y_test, model, report= True)
```

```
Train report
              precision    recall  f1-score   support

      Rendah       0.93      1.00      0.96        13
      Sedang       1.00      1.00      1.00        13
      Tinggi       1.00      0.92      0.96        13

    accuracy                           0.97        39
   macro avg       0.98      0.97      0.97        39
weighted avg       0.98      0.97      0.97        39


Test report
              precision    recall  f1-score   support

      Rendah       0.67      0.67      0.67         3
      Sedang       1.00      1.00      1.00         3
      Tinggi       0.75      0.75      0.75         4

    accuracy                           0.80        10
   macro avg       0.81      0.81      0.81        10
weighted avg       0.80      0.80      0.80        10
```

In [9]:

```
plot_confusion_matrix(X_train, y_train, X_test, y_test, model)
```



In [10]:

```
preprocessor = ColumnTransformer([
    ('numeric', num_pipe(scaling='robust'), ["Count", "X_mean", "Y_mean", "Z_mean", "X_std"
    ('categoric', cat_pipe(encoder='onehot'), ["Speed", "Loc"]),
])




pipeline = Pipeline([
    ('prep', preprocessor),
    ('algo', LogisticRegression(solver='lbfgs', n_jobs=-1, random_state=42))
])


model = RandomizedSearchCV(pipeline, rsp.logreg_params, cv=5, n_iter=50, n_jobs=-1, verbose
model.fit(X_train, y_train)

print(model.best_params_)
print(model.score(X_train, y_train), model.best_score_, model.score(X_test, y_test))
```

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
{'algo__C': 3.907967156822884, 'algo__fit_intercept': True}
1.0 0.8964285714285714 0.9
```

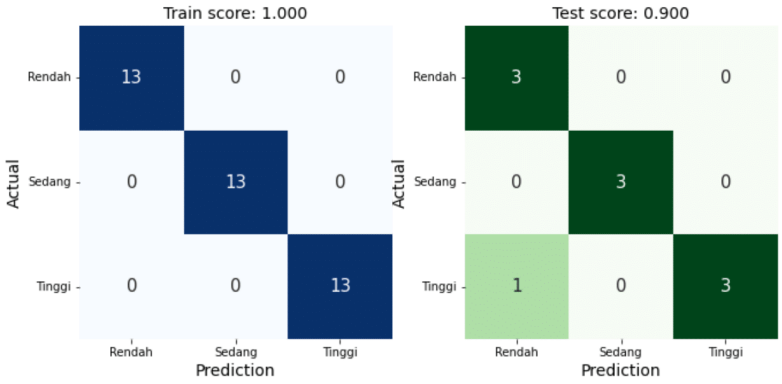
In [11]:

```
plot_classification_report(X_train, y_train, X_test, y_test, model, report= True)
```

```
Train report
              precision    recall  f1-score   support

      Rendah       1.00      1.00      1.00        13
      Sedang       1.00      1.00      1.00        13
      Tinggi       1.00      1.00      1.00        13

    accuracy                           1.00        39
   macro avg       1.00      1.00      1.00        39
weighted avg       1.00      1.00      1.00        39


Test report
              precision    recall  f1-score   support

      Rendah       0.75      1.00      0.86         3
      Sedang       1.00      1.00      1.00         3
      Tinggi       1.00      0.75      0.86         4

    accuracy                           0.90        10
   macro avg       0.92      0.92      0.90        10
weighted avg       0.93      0.90      0.90        10
```

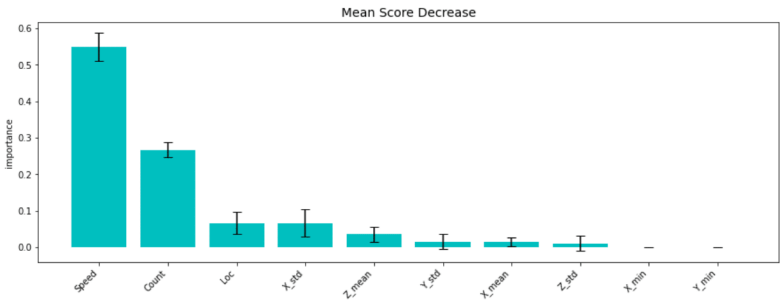
In [12]:

```
plot_confusion_matrix(X_train, y_train, X_test, y_test, model)
```



In [13]:

```
df_imp = mean_score_decrease(X_train, y_train, model, plot=True, topk=10)
```



LAMPIRAN 6

DATAFRAME

Code	Count	X_mean	Y_mean	Z_mean	X_std	Y_std	Z_std	X_min	Y_min	Z_min	X_max	Y_max	Z_max	X_med	Y_med	Z_med	Speed	Loc	Res
R1	245	0.036204	0.280857	0.938	0.118325	0.202595	0.36425	-0.27	-0.34	-0.94	0.69	0.86	1.95	0.03	0.27	0.94	Lo	Lok1	Tinggi
R2	212	0.031509	0.278632	0.952406	0.094906	0.183519	0.320357	-0.34	-0.22	0.08	0.31	0.75	2.14	0.03	0.28	0.94	Lo	Lok1	Tinggi
R3	151	0.033642	0.239272	0.951391	0.223566	0.342865	0.552731	-1.03	-1.11	-1.33	0.75	1.09	2.2	0.05	0.28	1	Med	Lok1	Sedang
R4	162	0.043333	0.26142	0.977037	0.150684	0.335875	0.479205	-0.61	-1	-0.3	0.63	1.64	2.53	0.04	0.27	0.95	Med	Lok1	Sedang
R5	85	0.076824	0.205294	0.912824	0.321974	0.813121	1.134543	-0.55	-2	-2.19	1.7	2.89	4.47	0.05	0.13	0.91	Hi	Lok1	Rendah
R6	80	-0.009875	0.295875	0.946125	0.278665	0.674694	1.162364	-0.98	-2.06	-1.92	0.53	3.14	4.72	0.05	0.27	0.98	Hi	Lok1	Rendah
R7	181	0.018232	0.28337	0.900939	0.110011	0.210645	0.392685	-0.44	-0.23	-1.36	0.34	1.03	1.94	0.02	0.3	0.91	Lo	Lok1	Tinggi
R8	164	0.020793	0.287073	0.929085	0.13099	0.216373	0.465321	-0.47	-0.42	-1.83	0.73	1.25	2.3	0.02	0.27	0.92	Lo	Lok1	Tinggi
R9	160	0.0195	0.27325	0.958625	0.130528	0.233657	0.381125	-0.47	-0.2	-1.55	0.52	1.88	2.41	0.02	0.25	0.94	Lo	Lok1	Tinggi
R10	120	0.041	0.275667	0.918417	0.183264	0.374087	0.618522	-0.73	-0.8	-1.69	0.95	1.3	3.11	0.05	0.275	0.96	Lo	Lok1	Tinggi
R11	133	0.019699	0.270376	0.939098	0.167535	0.397321	0.698155	-0.61	-1.38	-1.86	0.45	1.39	3.83	0.02	0.31	0.92	Med	Lok1	Sedang
R12	205	0.012439	0.247366	0.932488	0.232189	0.467928	0.823865	-1.11	-1.67	-2.34	0.69	1.59	3.45	0.03	0.3	0.95	Med	Lok1	Sedang
R13	76	0.056053	0.148026	1.006842	0.272608	0.637194	1.042293	-0.67	-1.33	-1.63	0.72	2.77	3.92	0.03	0.14	1.06	Med	Lok1	Sedang
R14	85	0.033176	0.384118	0.989882	0.267316	0.627236	1.146606	-0.7	-1.08	-3.05	0.94	2.2	3.48	0.03	0.38	1.09	Med	Lok1	Sedang
R15	173	0.024682	0.288902	0.950289	0.10736	0.216176	0.415946	-0.31	-0.77	-0.75	0.33	1.44	2.61	0.02	0.28	0.97	Hi	Lok1	Rendah
R16	182	0.022473	0.298242	0.944396	0.093436	0.223583	0.392153	-0.3	-0.23	-0.5	0.33	1.14	2.55	0.02	0.275	0.94	Hi	Lok1	Rendah
R17	81	0.008765	0.189383	0.862469	0.320283	0.515105	0.928411	-2.08	-1.2	-2.03	0.72	1.56	2.86	0.02	0.16	0.95	Hi	Lok1	Rendah
L1	304	0.028454	0.286776	0.960559	0.144288	0.187021	0.354234	-0.38	-0.39	-0.45	1.8	1.13	2.72	0.03	0.28	0.92	Lo	Lok2	Tinggi
L2	287	0.038571	0.308711	0.929373	0.10862	0.249649	0.438659	-0.31	-0.22	-2.09	0.67	2.42	3.2	0.03	0.3	0.95	Lo	Lok2	Tinggi
L3	172	0.016512	0.318721	0.913372	0.20785	0.492443	0.784715	-1.23	-0.88	-2.06	0.48	2.34	3.88	0.025	0.3	0.9	Med	Lok2	Sedang
L4	199	0.025126	0.259598	0.947186	0.223145	0.31873	0.695925	-1.31	-0.8	-2.16	1.19	1.11	3.8	0.02	0.28	0.89	Med	Lok2	Sedang
L5	140	0.027571	0.273357	0.795857	0.17718	0.480181	0.866251	-0.47	-1.44	-2.3	0.47	2.09	3.23	0.03	0.23	0.93	Hi	Lok2	Rendah
L6	121	0.01719	0.225041	0.940331	0.323556	0.382827	0.572678	-0.73	-0.67	-0.73	2.77	1.22	3.75	-0.02	0.2	0.95	Hi	Lok2	Rendah
L7	274	0.105	0.383577	0.908321	0.126611	0.195357	0.499334	-0.66	-0.17	-2.7	0.86	1.36	4.67	0.09	0.39	0.89	Lo	Lok2	Tinggi
L8	305	0.097082	0.37682	0.925115	0.120603	0.187422	0.38845	-0.23	-0.27	-0.89	1.11	1.48	3.91	0.09	0.36	0.92	Lo	Lok2	Tinggi
L9	296	0.0975	0.383649	0.921149	0.117158	0.176488	0.396167	-0.8	-0.27	-1.08	0.58	1.22	2.7	0.09	0.38	0.93	Lo	Lok2	Tinggi
L10	133	0.073985	0.379173	0.930902	0.194593	0.380944	0.830326	-0.72	-1.23	-2.55	0.73	1.64	3.53	0.09	0.39	1.03	Lo	Lok2	Tinggi
L11	201	0.081841	0.388955	0.92791	0.134982	0.248663	0.489564	-0.45	-0.22	-0.22	0.55	1.48	3.63	0.08	0.39	0.94	Lo	Lok2	Tinggi
L12	189	0.078413	0.392222	0.899577	0.200435	0.406803	0.893589	-0.61	-0.95	-3.98	0.7	2.34	4.63	0.08	0.41	0.91	Med	Lok2	Sedang
L13	90	0.077111	0.272667	1.139444	0.349723	0.511218	1.034159	-1.16	-1.28	-1.31	1.92	1.56	6.61	0.055	0.395	1.085	Med	Lok2	Sedang
L14	104	0.066058	0.339519	0.968462	0.362476	0.512508	1.017145	-1.55	-1.03	-1.86	1.39	2.58	4.75	0.08	0.36	0.91	Med	Lok2	Sedang
L15	85	0.052235	0.406	0.872353	0.323376	0.663681	1.109861	-1	-0.8	-4.69	1.31	4.42	5.39	0.08	0.39	0.8	Med	Lok2	Sedang
L16	304	0.078882	0.388355	0.897171	0.113199	0.181724	0.382173	-0.63	-0.08	-0.83	0.56	1.59	3.06	0.08	0.38	0.915	Hi	Lok2	Rendah
L17	229	0.074236	0.394934	0.912795	0.129025	0.224438	0.491301	-0.39	-0.61	-0.69	0.75	1.28	4.03	0.08	0.39	0.89	Hi	Lok2	Rendah
L18	168	0.058214	0.384821	0.855536	0.203703	0.268776	0.696652	-1.72	-0.58	-2.05	0.64	1.47	4.44	0.06	0.38	0.845	Hi	Lok2	Rendah
L19	74	0.059324	0.086081	0.907973	0.389722	0.490498	1.202663	-1.42	-1.72	-1.67	1.08	0.83	3.91	0.03	0.19	0.79	Hi	Lok2	Rendah
L20	110	0.017727	0.522636	0.910818	0.3088	0.659861	1.42998	-1.11	-0.86	-3.17	0.86	3.11	7.55	0.05	0.485	0.96	Hi	Lok2	Rendah
A1	598	0.04393	0.395535	0.922843	0.067463	0.130702	0.169866	-0.14	0.02	0.33	0.3	0.78	1.44	0.05	0.39	0.94	Lo	Lok3	Rendah
A2	603	0.055755	0.390381	0.908955	0.07275	0.15018	0.228985	-0.22	-0.06	-0.52	0.47	0.92	1.55	0.06	0.38	0.91	Lo	Lok3	Rendah
A3	685	0.062234	0.392482	0.928613	0.103102	0.168413	0.276878	-0.67	-0.06	0.19	0.56	0.84	1.72	0.06	0.39	0.91	Med	Lok3	Sedang
A4	716	0.070126	0.387067	0.925251	0.072042	0.106825	0.183845	-0.11	0.08	0.38	0.27	0.88	1.63	0.06	0.39	0.92	Med	Lok3	Sedang
A5	271	0.038672	0.413358	0.897934	0.122543	0.183283	0.439547	-0.33	-0.17	-0.55	0.45	0.91	2.06	0.03	0.42	0.91	Hi	Lok3	Tinggi
A6	474	0.053945	0.330316	0.944304	0.159555	0.245255	0.461452	-0.66	-0.34	-0.84	0.53	1.02	2.22	0.05	0.34	0.94	Hi	Lok3	Tinggi
A7	772	0.039236	0.278187	0.941464	0.082253	0.162209	0.21803	-0.5	-0.28	0.28	0.31	0.81	1.55	0.05	0.28	0.92	Lo	Lok3	Rendah
A8	666	0.042508	0.277688	0.932132	0.097371	0.158339	0.253786	-0.45	-0.2	0.16	0.45	0.83	1.69	0.05	0.275	0.91	Lo	Lok3	Rendah
A9	532	0.023571	0.27656	0.946297	0.131494	0.184333	0.235614	-0.55	-0.3	0.1	1.72	0.89	1.56	0.02	0.26	0.945	Med	Lok3	Sedang
A10	546	0.033755	0.287949	0.937692	0.067815	0.202716	0.22526	-0.27	-0.17	0.14	0.31	0.89	1.67	0.03	0.27	0.94	Med	Lok3	Sedang
A11	344	0.032442	0.281512	0.963866	0.081254	0.19864	0.28805	-0.27	-0.3	-0.08	0.34	0.86	1.92	0.03	0.28	0.97	Hi	Lok3	Tinggi
A12	359	0.04532	0.278468	0.938357	0.070978	0.209342	0.253897	-0.2	-0.48	-0.13	0.25	0.84	1.92	0.05	0.3	0.95	Hi	Lok3	Tinggi