PERBANDINGAN METODE SELEKSI FITUR *FILTER*, *WRAPPER*, DAN *EMBEDDED* PADA KLASIFIKASI DATA NIRS MANGGA MENGGUNAKAN RANDOM FOREST DAN SUPPORT VECTOR MACHINE (SVM)

Disusun Oleh:

N a m a : Dimas Ariyoga
NIM : 17523137

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2022

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PERBANDINGAN METODE SELEKSI FITUR *FILTER*, *WRAPPER*, DAN *EMBEDDED* PADA KLASIFIKASI DATA NIRS MANGGA MENGGUNAKAN RANDOM FOREST DAN SUPPORT VECTOR MACHINE (SVM)

TUGAS AKHIR

Disusun Oleh:

N a m a : Dimas Ariyoga
NIM : 17523137

Yogyakarta, 25 Mei 2022
Pembimbing,

(Arrie Kurniawardhani, S.Si, M.Kom.)

HALAMAN PENGESAHAN DOSEN PENGUJI

PERBANDINGAN METODE SELEKSI FITUR *FILTER*, *WRAPPER*, DAN *EMBEDDED* PADA KLASIFIKASI DATA NIRS MANGGA MENGGUNAKAN RANDOM FOREST DAN SUPPORT VECTOR MACHINE (SVM)

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjanadi Fakultas Teknologi Industri Universitas Islam Indonesia
Yogyakarta, 27 Mei 2022

Tim Penguji

Arrie Kurniawardhani, S.Si., M.Kom. _____

Anggota 1

Rahadian Kurniawan, S.Kom., M.Kom. _____

Anggota 2

Aridhanyati Arifin, S.T., M.Cs. _____

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama	:	Dimas Ariyoga
NIM	:	17523137

Tugas akhir dengan judul:

PERBANDINGAN METODE SELEKSI FITUR *FILTER*, *WRAPPER*, DAN *EMBEDDED* PADA KLASIFIKASI DATA NIRS MANGGA MENGGUNAKAN RANDOM FOREST DAN SUPPORT-VECTOR MACHINE (SVM)

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 21 April 2022

(Dimas Ariyoga)

HALAMAN PERSEMBAHAN

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

Alhamdulillahi Rabbil'alamin saya ucapkan puji dan syukur saya kepada Allah SWT atas segala rahmat, taufik, hidayah, dan inayah-Nya kepada saya sehingga saya dapat menyelesaikan tugas akhir di masa perkuliahan ini. Saya persembahkan karya ini kepada ayah, ibu, eyang, abang, dan adik saya tercinta atas segala limpahan doa, dukungan, perhatian, dan kasih sayang yang tulus selama ini. Semoga Allah SWT senantiasa memberikan kesehatan, keselamatan dan rezeki yang melimpah kepada semuanya.

HALAMAN MOTO

Neither does it lie in the sun's power to overtake the moon nor can the night outsrip the day. All glide along, each in its own orbit [QS. 36:40]

KATA PENGANTAR

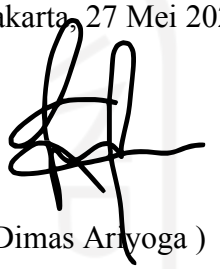
Alhamdulillahi rabbil'alamin, tidak pernah berhenti penulis ucapkan kehadirat Allah SWT, yang dengan rahmat dan hidayah-Nya penulis mampu menyelesaikan Tugas Akhir ini dengan baik. Tidak lupa selawat serta salam kepada Nabi dan Rasul-Nya, Nabi Muhammad SAW. Tugas Akhir ini disusun sebagai salah satu syarat untuk mendapatkan gelar kesarjanaan pada jurusan Informatika Universitas Islam Indonesia. Banyak sekali pihak yang telah membantu penulis dalam penyusunan Tugas Akhir ini, baik berupa materi ataupun berupa motivasi dan dukungan. Semua itu tentu terlalu banyak bagi penulis untuk membalasnya, namun pada kesempatan ini penulis hanya dapat mengucapkan terima kasih kepada:

1. Allah SWT atas segala nikmat dan rahmat-Nya selama ini
2. Orang tua, eyang, abang, dan adik yang tidak pernah lelah mendoakan, memberikan motivasi, nasehat serta dukungannya sepanjang menempuh pendidikan.
3. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D., sebagai Rektor Universitas Islam Indonesia.
4. Bapak Hendrik, S.T., M.Eng., sebagai Ketua Jurusan Informatika Universitas Islam Indonesia.
5. Bapak Dr. Raden Teguh Dirgahayu, S.T., M.Sc., sebagai Ketua Program Studi Informatika Program Sarjana Universitas Islam Indonesia.
6. Ibu Erika Ramadhani, S.T., M.Eng., sebagai Dosen Pembimbing Akademik (DPA) yang sangat baik dan selalu meluangkan waktu dalam membimbing saya selama belajar di Informatika Universitas Islam Indonesia.
7. Ibu Arrie Kurniawardhani, S.Si, M.Kom., sebagai Dosen Pembimbing Tugas Akhir yang selalu baik, ramah, peduli, dan sabar dalam memberikan arahan, dukungan, dan waktunya dalam membantu saya pada tugas akhir ini.
8. Bapak dan Ibu dosen Jurusan Informatika yang telah memberikan ilmu dan pengetahuan yang bermanfaat.
9. Sahabat-sahabat saya, Muhammad Sauqi Khatami dan Dimas Setyawan Ramadhansyah yang selalu mendukung saya dari awal kuliah di Universitas Islam Indonesia.
10. Sahabat-sahabat di luar jurusan, RAF, dan Delivery.
11. Teman-teman PIXEL 2017.

Semoga semua ilmu, do'a, dan dukungan yang diberikan kepada penulis dapat dibalas oleh Allah SWT. Saya menyadari bahwa tugas akhir ini masih jauh dari kata sempurna. Namun saya berharap semoga tugas akhir ini dapat diterima dan bermanfaat bagi para pembaca dan semua pihak.

Wassalamu'alaikum warahmatullahi wabarakatuh.

Yogyakarta, 27 Mei 2022



(Dimas Arryoga)

SARI

Pembangunan model klasifikasi seringkali dihadapkan pada masalah pengolahan data saat ingin menggunakan dataset yang berdimensi tinggi karena dapat memakan waktu dan memerlukan upaya komputasi yang berlebihan. Hal ini juga menyebabkan terjadinya sebuah fenomena pada data yaitu "*Curse of Dimensionality*". *Curse of Dimensionality* ini terjadi saat dimensi dari data sangat tinggi dan mengakibatkan nilai informasi penting yang didapatkan semakin menurun. Untuk mengatasi masalah tersebut, penelitian ini akan menggunakan teknik pengurangan dimensi yaitu teknik seleksi fitur. Penelitian ini akan menerapkan sembilan metode seleksi fitur dari tiga kategori berbeda yaitu *filter, wrapper,* dan *embedded* terhadap dataset spektrum NIRS dari buah mangga untuk kemudian dilakukan proses klasifikasi menggunakan *Random Forest* dan *Support Vector Machine* (SVM). Penelitian ini membangun empat skenario pada model klasifikasi yaitu *Random Forest* dengan 100, 150, dan 200 *trees* serta klasfikasi menggunakan SVM dengan RBF *kernel*. Hasil yang diperoleh dari masing-masing klasifikasi berbeda tergantung dari model klasifikasi, jumlah *tree* (*pada Random Forest*), metode seleksi fitur, dan jumlah fitur yang digunakan. Seluruh skenario klasifikasi yang menggunakan *Random Forest* dapat mencapai performa tertinggi dengan menggunakan fitur-fitur *Mutual Information*, perbedaan hanya terdapat pada jumlah fitur yang dibutuhkan. Pada klasifikasi *Random Forest* dengan 100 *tree*, performa terbaik diperoleh dari penggunaan 70 fitur dari *Mutual Information* yang menghasilkan 0.96 *accuracy,* 0.96 *recall*, dan 0.96 *precision* sedangkan yang terendah dihasilakan menggunakan 63 fitur dari ANOVA yang menghasilkan 0.88 *accuracy,* 0.88 *recall*, dan 0.88 *precision.* Pada klasifikasi *Random Forest* yang menggunakan 150 *tree,* hasil klasifikasi terbaik diperoleh dengan menggunakan 69 fitur *Mutual Information* yang menghasilkan 0.96 *accuracy,* 0.96 *recall*, dan 0.95 *precision*. Pada skenario klasifikasi ini, hasil terendah juga diperoleh dari penggunaan 63 fitur dari ANOVA yang mendapatkan 0.88 accuracy, 0.88 *recall,* dan 0.88 *precision.* Selanjutnya, klasifikasi *Random Forest* yang menggunakan 200 *trees* mendapatkan performa tertinggi dengan menggunakan 72 fitur dari metode *Mutual Information* dan menghasilkan 0.96 *accuracy,* 0.96 *recall*, dan 0.96 *precision*. Sedangkan performa terendah diperoleh dengan menggunakan 66 fitur hasil seleksi ANOVA yang menghasilkan 0.88 *accuracy*, 0.88 *recall,* dan 0.88 *precision.*

Kata kunci: *Embedded, Filter,* Klasifikasi, Random Forest, Seleksi Fitur, *Support Vector Machine*, *Wrapper*

GLOSARIUM

Glosarium memuat daftar kata tertentu yang digunakan dalam laporan dan membutuhkan penjelasan, misalnya kata serapan yang belum lazim digunakan. Urutkan sesuai abjad. Contoh penulisannya seperti di bawah ini:

Dataset	Kumpulan data yang digunakan untuk melatih model.
Library	Kumpulan kode dengan fungsi tertentu.
Seleksi Fitur	Proses pemilihan fitur terpenting berdasarkan atribut fitur.
Spektrum	Urutan atau rangkaian yang bersinambung.
NIRS	*Near-infrared Spectroscopy*.
Kultivar	Varietas tanaman yang dibudidayakan

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Klasifikasi merupakan suatu proses menemukan pola atau fungsi yang mendeskripsikan serta memisahkan kelas data yang satu dengan yang lainnya untuk menyatakan objek tersebut masuk pada kategori tertentu dari beberapa kategori yang telah ditentukan sebelumnya (Hendrian, 2018). Klasifikasi juga dapat didefinisikan sebagai sebuah proses pelatihan atau pembelajaran terhadap fungsi target yang memetakan setiap set atribut/fitur kepada label kelas yang tersedia, oleh sebab itu atribut atau fitur pada dataset menjadi suatu hal yang penting pada keakuratan hasil klasifikasi yang dilakukan (Utomo & Mesran, 2020).

Pada kenyataannya, atribut atau fitur yang biasanya digunakan untuk membangun model klasifikasi tidak selalu sederhana. Para peneliti sering kali harus berhadapan dengan dataset yang bersifat kompleks dengan fitur yang sangat banyak dan berdimensi tinggi (*high dimensional data*) (Putra, 2020). Data berdimensi tinggi seperti ini membuat analisis data semakin sulit, memakan waktu, dan memerlukan upaya komputasi yang berlebihan sehingga data sehingga terjadi sebuah fenomena pada data yaitu "*Curse of Dimensionality*". *Curse of Dimensionality* ini dapat terjadi ketika dimensi dari data terlalu tinggi sehingga dapat mengakibatkan nilai informasi penting yang didapatkan semakin menurun (Noordiansyah et al., 2016).

Data berdimensi tinggi atau *high dimensional data* pada dataset menimbulkan paling tidak 3 masalah pada model pembelajaran. Pertama, model pembelajaran akan sulit untuk memiliki kinerja yang optimal karena semakin banyak fitur yang digunakan, semakin kompleks pula suatu model *machine learning* harus membuat model permasalahan. Kedua, *high dimensional data* ini dapat menyebabkan terjadinya *overfitting* karena ada sangat banyak konfigurasi fitur walaupun kita hanya memiliki data yang terbatas. Ketiga, data dengan dimensi yang besar susah untuk diproses secara komputasi (*computationally expensive*), baik dari segi memori dan waktu (Putra, 2020).

Penelitian ini akan menggunakan dataset yang berisi spektrum data serapan hasil spektroskopi dari buah mangga menggunakan *Near-infrared spectroscopy* (NIRS) yang terdiri dari 1563 fitur dengan 186 baris data. Data ini digunakan karena sifat *high dimensional data* sendiri merupakan karakteristik dari data NIRS. Biasanya ada puluhan atau ratusan sampel

yang diukur dengan jumlah fitur yang mencapai ribuan. Ribuan fitur dari data NIRS ini selalu mengandung sejumlah besar fitur yang tidak relevan dan juga mencakup fitur redundan (Yuhua et al., 2013). Keberadaan fitur yang tidak relevan dan berlebihan seperti yang terdapat pada data NIRS ini akan sangat menurunkan efisiensi tugas pembelajaran dan kinerja klasifikasi.

Untuk mengatasi masalah tersebut, penelitian ini akan menggunakan teknik pengurangan dimensi yaitu teknik seleksi fitur. Seleksi fitur telah menjadi bidang penelitian aktif dalam pengenalan pola, statistik, dan *Data Mining* (Nugroho & Wibowo, 2017). Seleksi fitur sendiri dapat didefinisikan sebagai sebuah teknik yang digunakan untuk mengurangi dimensi pada atribut data. Hal ini bertujuan mengurangi kompleksitas dan mengurangi atribut-atribut yang belum tentu berguna dengan cara memilih sebagian kecil fitur yang paling cocok atau relevan dari data yang asli sesuai dengan kriteria evaluasi relevansi tertentu, yang biasanya mengarah pada kinerja pembelajaran yang lebih baik, komputasi yang lebih rendah, dan interpretasi model yang lebih baik (Tang et al., 2016).

Diterapkannya seleksi fitur pada penelitian ini diharapkan dapat meningkatkan performa dan akurasi dari model dalam mengklasifikasikan jenis kultivar buah mangga berdasarkan karakteristik data spektroskopi buah mangga (Samadi et al., 2020).

1.2 Rumusan Masalah

a. Bagaimana pengaruh implementasi masing-masing teknik seleksi fitur terhadap hasil klasifikasi jenis kultivar buah mangga?
b. Kategori seleksi fitur apa yang dapat menghasilkan performa pengujian paling tinggi dan paling rendah saat dikombinasikan dengan metode klasifikasi?

1.3 Batasan Masalah

a. Dataset yang digunakan merupakan data spektrum hasil spektroskopi NIR pada buah mangga dengan panjang gelombang 1000-2500 nm yang diambil dari penelitian sebelumnya oleh (Samadi et al., 2020).
b. Fitur-fitur yang akan digunakan pada penelitian ini dibatasi hingga 100 fitur

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah menerapkan dan menemukan kombinasi teknik seleksi fitur dan metode klasifikasi yang paling baik dan sesuai menggunakan spektrum data hasil spektroskopi NIR.

1.5 Manfaat Penelitian

a. Mengetahui efektivitas dari implementasi teknik seleksi fitur terhadap data spektroskopi NIR.
b. Penelitian ini dapat dijadikan bahan pertimbangan bagi penelitian selanjutnya khususnya penelitian terkait klasifikasi menggunakan data berdimensi tinggi dan klasifikasi menggunakan data NIRS.

1.6 Metodologi Penelitian

a. Perumusan Masalah

Tahapan ini dilakukan untuk mengidentifikasi masalah-masalah yang akan diangkat menjadi penelitian serta menentukan batasan, tujuan, dan manfaat dari penelitian.

b. Landasan Teori dan Kajian Pustaka

Pada tahap ini, dasar-dasar teori yang relevan dengan tema klasifikasi dan seleksi fitur akan dipelajari dan digunakan pada penelitian. Setelah itu, pustaka atau penelitian yang berkaitan dan relevan dengan tema penelitian akan dikaji dan digunakan sebagai referensi.

c. Pengumpulan dan Analisis Data

Dataset pada penelitian ini diambil dari *Mendeley Data* dengan judul "Dataset of NIR spectrum for intact mango fruits" oleh Agus Munawar. Selain itu, dilakukan juga analisis data untuk melihat pola dan karakteristik dari spektrum data hasil spektroskopi NIR menggunakan teknik *Exploratory Data Analysis* (EDA).

d. Seleksi Fitur

Pada tahap ini, fitur-fitur yang terdapat pada dataset NIRS akan diseleksi menggunakan tiga kategori dari seleksi fitur yaitu *filter, wrapper,* dan *embedded.*

e. Pembangunan Model

Pada penelitian ini, model klasifikasi akan dibangun menggunakan metode *Random Forest Classifier* dan *Support Vector Machine* (SVM).

f. Pengujian Model dan Penarikan Kesimpulan

Pada tahap ini, *Confusion Matrix* akan digunakan untuk menguji dan mengevaluasi model klasifikasi yang telah dibangun. Pada tahapan ini dilakukan juga penarikan kesimpulan dari evaluasi model yang telah dilakukan.

1.7 Sistematika Penulisan

Sistematika penulisan laporan dibuat untuk mempermudah penyusunan laporan tugas akhir ini dan membantu dalam memahami isi dari penelitian. Sistematika penelitian ini adalah sebagai berikut:

Bab I Pendahuluan, Bab ini berisi latar belakang mengenai permasalahan yang mendasari penelitian. Bagian ini memuat latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penyusunan.

Bab II Landasan Teori, Bab ini berisi uraian dari teori-teori yang menjadi referensi dalam melakukan penelitian. Bab ini juga mengkaji penelitian-penelitian sebelumnya yang relevan dengan laporan penelitian ini.

Bab III Metodologi Penelitian, Bab ini berisi tahapan-tahapan dalam melakukan penelitian. Bab ini terdiri dari perencanaan, pengumpulan data, implementasi model, pengujian, dan analisis.

Bab IV Hasil dan Pembahasan, Bab ini menjelaskan tentang hasil implementasi seleksi fitur pada hasil klasifikasi jenis mangga yang dilakukan berdasarkan rancangan yang telah disusun sebelumnya. Pada bagian ini juga dilakukan pengujian untuk mendapatkan sebuah hasil yang diharapkan.

Bab V Kesimpulan dan Saran, Bab ini berisi kesimpulan mengenai hasil yang didapatkan dari penelitian yang telah dilakukan. Bagian ini juga berisi saran untuk penelitian-penelitian selanjutnya.

BAB II
LANDASAN TEORI

2.1 Buah Mangga

Buah mangga atau dengan nama spesies Mangifera indica L merupakan salah satu jenis tumbuhan yang umum dan komersial di Asia Tenggara seperti Filipina, Indonesia, Malaysia, dan Thailand (Sembiring et al., 2020). Buah mangga diketahui telah dibudidayakan sejak 4000 tahun silam (Candole, 1984). Buah mangga memiliki potensi untuk dikembangkan karena tingkat keragaman genetiknya yang tinggi (Luqyana Z. T. M & Husni, 2019). Buah mangga merupakan salah satu buah yang memiliki karakteristik yang khas dengan kandungan nutrisi yang tinggi. Buah mangga juga memiliki beragam jenis kultivar seperti Cengkir, Kweni, Kent, dan Palmer. Berdasarkan KBBI, kultivar adalah kelompok tanaman yang dibudidayakan petani dan mempunyai sifat-sifat yang dapat dibedakan dari varietas lainnya secara khas, berdasarkan bentuk, rasa, warna, ketahanan pada penyakit, atau sifat lainnya. Secara umum, daging buah mangga yang telah matang akan berwarna kekuningan dengan rasa manis namun masih terdapat rasa asam (Kusumiyati et al., 2020).

2.2 *Near Infrared Reflectance Spectroscopy* (NIRS)

Near Infrared Reflectance Spectroscopy (NIRS) adalah sebuah teknik yang memanfaatkan pancaran sinar *near infrared* untuk memecahkan struktur kimia dari buah-buahan atau bahan organik lainnya. Spektroskopi atau *spectroscopy* sendiri merupakan sebuah disiplin ilmiah tentang interaksi radiasi elektromagnetik dengan atom atau molekul yang umumnya digunakan untuk mengidentifikasi suatu kandungan atau substansi dari objek melalui spektrum yang dipancarkan atau yang diserap (Bani-Fwaz, 2020). Teknik NIRS berlandaskan bahwa suatu objek biologi mempunyai ciri-ciri sifat optik dan elektromagnetik yang khas pada setiap objeknya sehingga dapat dianalisis menjadi data mengenai unsur kimia suatu objek (Marzatillah, 2021).

NIRS sendiri telah disempurnakan menjadi sebuah teknik non-destruktif (tanpa merusak bahan) yang dapat menganalisis suatu bahan pertanian tanpa peralatan kimia, tanpa menghasilkan polusi, dan bekerja dalam waktu yang sangat cepat. NIRS memakai gelombang elektromagnetik pada selang panjang gelombang 780 nm - 2500 nm atau sama dengan selang gelombang per 12.800 cm^{-1} hingga 4.000 cm^{-1} (Wiradinata, 2019).

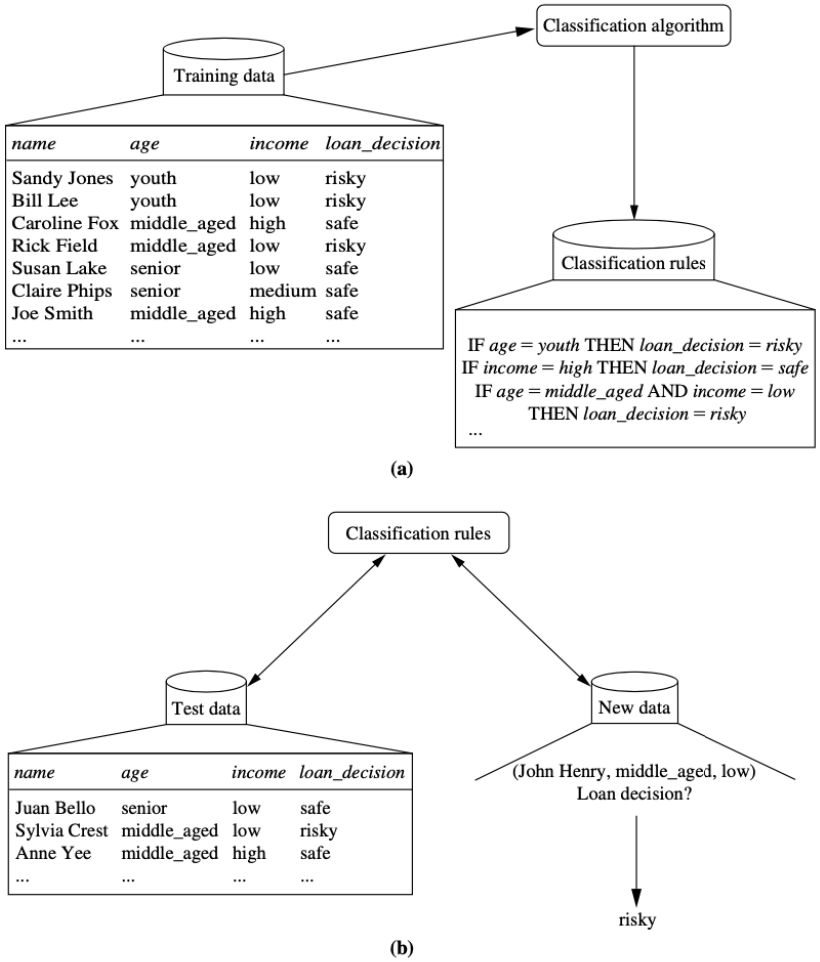
Setiap bahan organik memiliki spektrum NIR yang berbeda dan unik, spektrum tersebut diperoleh dari efek penyebaran, penyerapan (*absorbance)* dan pantulan (*reflectance)* dari gelombang NIR oleh bahan yang dapat mendeteksi berbagai komponen kimia dalam satu spektrum (Ayu, 2017).

2.3 Teknik Klasifikasi

Klasifikasi adalah sebuah teknik pengelompokan fitur ke dalam kelas-kelas yang sesuai. Model klasifikasi ditemukan berdasarkan proses menemukan pola atau fungsi yang menggambarkan dan memisahkan kelas-kelas data satu sama lain untuk menyatakan sebuah objek berada dalam kategori tertentu dari beberapa kategori yang telah ditentukan sebelumnya. (Hendrian, 2018). Klasifikasi dapat dilakukan secara manual maupun dengan bantuan teknologi. Klasifikasi yang dilakukan secara manual adalah klasifikasi yang dilakukan oleh manusia tanpa bantuan algoritma komputer. Sedangkan klasifikasi yang dilakukan dengan bantuan algoritma *machine learning* memiliki beberapa algoritma antara lain *Naïve Bayes, Support Vector Machine, Decision Tree, Random Forest, Fuzzy,* dan *Artificial Neural Networks* (Wibawa et al., 2018). Menurut (Septiani, 2017), Proses klasifikasi didasarkan pada empat komponen mendasar yaitu:

1. Kelas (*class*) adalah sebuah komponen yang berupa variabel kategori yang merepresentasikan atau mewakili label yang terdapat pada objek.
2. Prediktor yaitu sebuah variabel yang direpresentasikan oleh karakteristik (atribut) data.
3. *Training set* merupakan bagian dari dataset yang dilatih untuk membuat prediksi atau menjalankan fungsi dari sebuah algoritma *machine learning*.
4. Pengujian Dataset, merupakan data baru hasil klasifikasi yang akan diuji untuk melihat performa dan keakuratan dari klasifikasi.

Proses klasifikasi dicontohkan oleh (Han et al., 2012) seperti yang ditunjukkan pada Gambar 2.1.



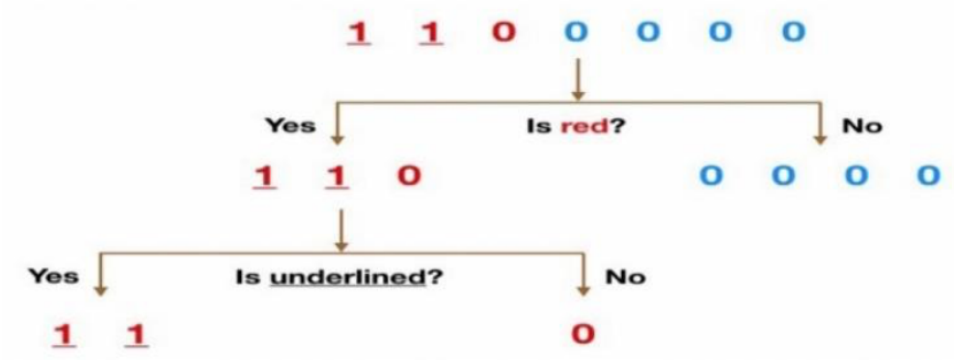
Gambar 2.1 Proses Klasifikasi Data

Sumber: Han et al., 2012

Seperti yang terlihat pada Gambar 2.1, (a) *Learning:* adalah proses pembelajaran data *training* yang dianalisis menggunakan algoritma klasifikasi. Di sini, atribut *loan_decision* berperan sebagai label kelas dan model pembelajaran atau pengklasifikasi direpresentasikan dalam bentuk aturan klasifikasi (*classification rules*). (b) *Classification:* adalah proses dimana *test data* digunakan untuk untuk melakukan estimasi keakuratan dari aturan klasifikasi (*classification rules)* yang dibangun. Jika akurasi dinilai dapat diterima, maka *rules* atau aturan yang diperoleh dapat diterapkan pada klasifikasi untuk data baru.

2.3.1 *Random Forest Classifier*

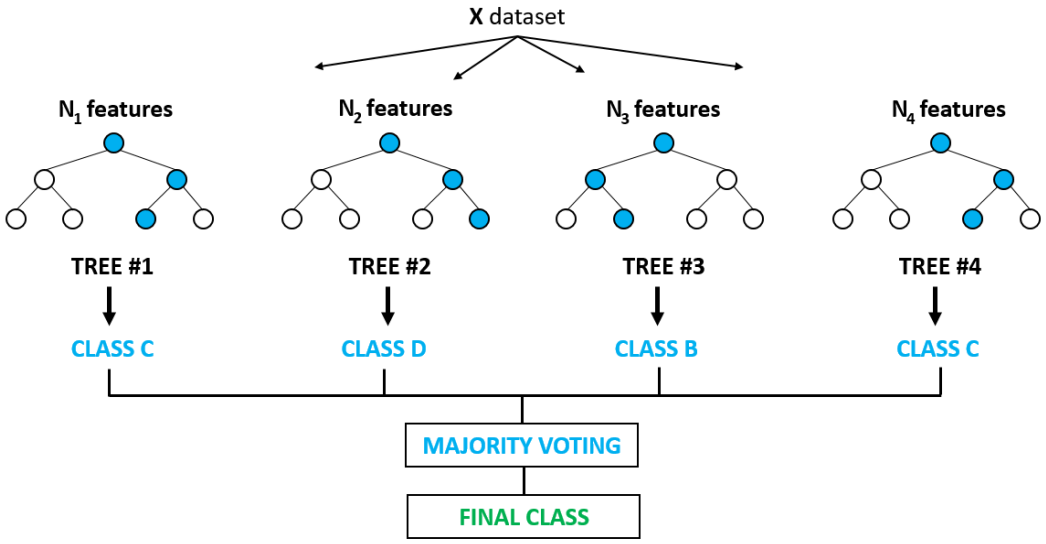
Random forest Classifier adalah algoritma pembelajaran yang berbasis gabungan dari beberapa algoritma atau disebut sebagai *Ensemble learning. Ensemble learning* pada *Random Forest* terdiri dari *n* kumpulan *Decision Tree* yang tidak memiliki hubungan atau tidak berkorelasi (Fadilah, 2018). Kinerja *Random Forest* ini diadaptasi dari algoritma *Decision Tree*, dimana setiap *tree* atau pohonnya dikembangkan dari sampel *bootstrap* berdasarkan data *training* (Renata & Ayub, 2020). *Decision Tree* sendiri adalah sebuah struktur algoritma yang digunakan untuk mempelajari klasifikasi dan prediksi pola dari data dan menggambarkan relasi dari variabel atribut x dan variabel target y dalam bentuk *tree* atau pohon. Setiap *internal node* atau simpul pada *decision tree* merupakan pengujian terhadap variabel atribut dan setiap cabangnya merupakan hasil dari pengujian tersebut, sedangkan *node* terluar yakni daun atau yang biasa disebut *leaf* menjadi labelnya (Sutoyo, 2018). Sebagai contoh, penelitian (Renata & Ayub, 2020) membuat sebuah visualisasi dari *decision tree* seperti yang tertera pada Gambar 2.2. Pada contoh ini, diberikan data yang berisi dua angka 1 dan lima angka 0 serta setiap angka memiliki warnanya masing-masing. Jika data ingin dikelompokkan dan fitur yang diambil dari data tersebut adalah warna dan garis bawah, maka *tree* akan seperti Gambar 2.2.



Gambar 2.2 Cara kerja *Decision Tree*

Sumber: Renata & Ayub, 2020

Algoritma *Random Forest* merupakan kumpulan dari *decision tree* yang beroperasi menjadi kombinasi fungsional dan membentuk model klasifikasi dalam bentuk kumpulan *tree* selama proses pelatihan dataset. Bila divisualisasikan, proses klasifikasi yang dilakukan pada *Random Forest* akan terbentuk seperti Gambar 2.3.



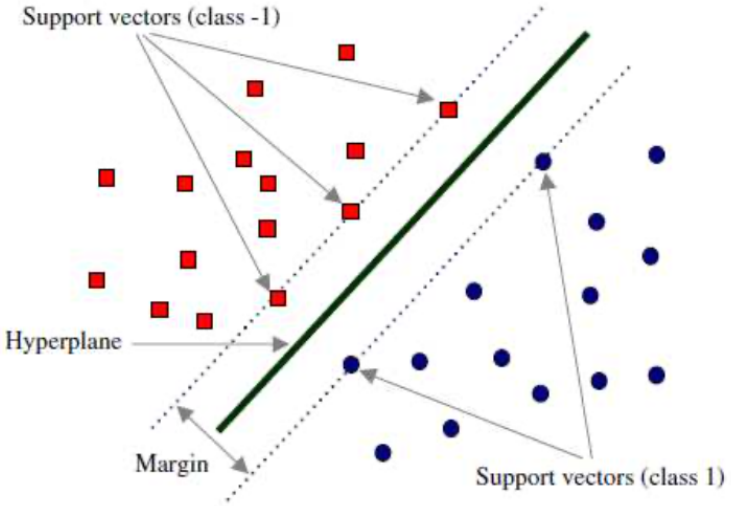
Gambar 2.3 Visualisasi Proses Klasifikasi Random Forest
Sumber: Ma'ruf, 2020

Seperti yang terlihat pada Gambar 2.3, setiap *tree* akan bekerja secara individu menggunakan atribut-atribut yang telah dipilih dari dataset secara acak. Setiap *decision tree* memiliki kesimpulan prediksi klasifikasi dan hasil prediksi akan digabungkan berdasarkan nilai rata-rata dari *tree* (regresi) atau menghitung *majority votes* yaitu mengambil keputusan yang dominan dari setiap pohon yang terbentuk (klasifikasi). Proses menggabungkan output dari beberapa model individu seperti di atas disebut *Ensemble Learning* (Kirasich et al., 2018). Alasan lain mengapa *Random Forest* menghasilkan *classifier* yang lebih baik dibandingkan dengan model individual lainnya dikarenakan algoritma ini menggunakan *decision tree* yang tidak memiliki korelasi. Kesalahan pada hasil prediksi dalam satu *decision tree* dapat diatasi karena ditutupi dengan kebenaran yang didapatkan dari *decision tree* lainnya asalkan arah pembuatan *decision tree* benar (Renata & Ayub, 2020).

2.3.2 Support Vector Machine (SVM)

Support Vector Machine (SVM) adalah salah satu metode *supervised learning* yang digunakan untuk proses klasifikasi dengan cara menganalisis dan mengenali pola yang ada pada data berdasarkan fungsi *hyperplane*. Pada ruang berdimensi tinggi, akan dicari *hyperplane* yang dapat memaksimalkan jarak (*margin*) antara kelas data (Baghaee et al., 2020). *Hyperplane* ini digunakan sebagai batas yang memisahkan *support vector* kelas satu dengan

support vector kelas lainnya seperti yang terlihat pada Gambar 2.4 (Nugraha & Purnamasari, 2019).



Gambar 2.4 Representasi *hyperplane* untuk klasifikasi umum dari algoritma SVM. Sumber: (Baghaee et al., 2020)

Pada dasarnya, metode SVM digunakan untuk proses klasifikasi data yang hanya memiliki dua kelas dan linearly separable seperti yang ditunjukkan Gambar 2.4 sehingga ketika suatu masalah tidak dapat dipisahkan secara linier dalam ruang input, SVM tidak dapat menemukan hyperplane pemisah yang kuat yang meminimalkan jumlah titik data yang salah klasifikasi dan dapat digeneralisasi dengan baik (Awad & Khanna, 2015). Untuk mengatasi masalah data tidak terpisah secara linear tersebut, digunakan fungsi kernel *Radial Basis Function* (RBF) (Ma'Ruf et al., 2019). RBF kernel ini memiliki dua parameter yaitu Gamma dan Cost. Parameter Cost (C) merupakan parameter yang berfungsi untuk menghindari misklasifikasi pada setiap sampel dalam training dataset. Parameter Gamma menentukan seberapa jauh pengaruh dari suatu sampel training. Jika nilai gamma rendah, titik yang berada jauh dari garis pemisah dipertimbangkan dalam perhitungan untuk garis pemisah. Ketika nilai gamma tinggi berarti titik – titik yang berada di sekitar garis akan dipertimbangkan dalam perhitungan (Ningrum, 2018). Persamaan 2.1 berikut merupakan persamaan dari RBF kernel:

$$Radial\ Basis\ Function\ (RBF): K\left(\vec{x}_i, \vec{x}_j\right) = exp\left(-\frac{\left|\left|\vec{x}_i - \vec{x}_j\right|\right|^2}{2\sigma^2}\right) \quad (2.1)$$

Di mana $\vec{x}_i$ dan $\vec{x}_j$ adalah vektor dari ruang fitur dan σ *gamma* adalah parameter bebas. Pemilihan parameter bebas sangat penting, karena nilai parameter yang dipilih dapat mengakibatkan *overfitting* pada data. Selain itu, SVM juga mempunyai teknik untuk melakukan klasifikasi pada *multiclass*, yaitu One *Against All* (OAA) atau *One Versus All* (OVA) (Awad & Khanna, 2015). *One Versus All* (OVA) ini yaitu membandingkan suatu kelas dengan semua selain dirinya yang dianggap menjadi satu kesatuan karena pada dasarnya SVM adalah *Machine Learning* yang hanya mengklasifikan dua kelas saja secara linear (Nugraha & Purnamasari, 2019).

2.4 Exploratory Data Analysis (EDA)

Exploratory data analysis (EDA) adalah salah satu strategi pada *data science* untuk mengenali atau melakukan analisis pada data (Komorowski et al., 2016). EDA juga merupakan teknik pencarian heuristik untuk menemukan hubungan yang signifikan antara variabel secara keseluruhan dalam dataset atau kumpulan data yang besar (Taboada & Han, 2020). EDA adalah langkah awal yang fundamental setelah pengumpulan data, dimana data hanya divisualisasikan, diplot, dan dimanipulasi tanpa asumsi apa pun. Hal ini dilakukan untuk membantu menilai kualitas data dalam membangun model.

Secara definitif, EDA merupakan suatu proses kritis dalam melakukan penyelidikan awal pada data untuk menemukan pola, anomali, menguji hipotesis, dan menguji asumsi dengan bantuan ringkasan dan representasi grafis. Sebagian besar teknik EDA bersifat grafis dengan beberapa teknik kuantitatif. Alasan ketergantungan yang tinggi pada grafik adalah bahwa pada dasarnya peran utama EDA adalah untuk mengeksplorasi data sehingga grafik mempunyai peran yang tinggi dalam memberikan informasi kepada para analis (Komorowski et al., 2016). Selain itu, menurut (Samosir et al., 2021), data statistik yang ditampilkan secara numerik saja dapat mengaburkan, menyembunyikan, atau bahkan salah dalam merepresentasikan struktur data sehingga dapat mengakibatkan penarikan kesimpulan yang salah.

EDA sendiri tidak bergantung hanya pada suatu tipe model atau prosedur baku yang sudah didefinisikan sebelumnya. Hal ini dikarenakan EDA memiliki karakteristik fleksibel yang diperlukan untuk melakukan identifikasi dan investigasi suatu fenomena yang muncul

pada saat melakukan penelitian empiris. Pengaplikasian teknik *Exploratory Data Analysis* (EDA) pada sebuah dataset dilakukan dengan berbagai perspektif tergantung pada konteks dan rincian analisis yang dibutuhkan (Wahyuni et al., 2019). Dengan melakukan EDA, analisis data dapat lebih mudah dilakukan dan kondisi dataset menjadi mudah dipahami. Memahami kondisi dataset pada EDA dapat merujuk pada setidaknya empat poin berikut (Samosir et al., 2021):

1. Mengekstrak variabel penting dan membuang variabel yang tidak berguna.
2. Mengidentifikasi *outliers*, nilai yang kosong atau hilang (*missing values*), dan kesalahan manusia (*human error*).
3. Memahami hubungan antar variabel, dan
4. Memaksimalkan pengetahuan yang kita miliki terhadap kondisi data dan meminimalkan potensi kesalahan di kemudian hari.

Dalam dua penelitian lain yang dilakukan oleh (Wahyuni et al., 2019) dan (Hoaglin, 2014), empat tema utama dari teknik *Exploratory Data Analysis* (EDA) dijelaskan sebagai berikut:

1. *Display*

   Diplay dapat memperlihatkan pola data dan struktur analisis melalui tampilan visual.
2. *Residual*

   Residual terfokus pada apa yang tersisa pada data setelah analisis dilakukan sesuai dengan *schematic equation*.
3. *Re-expression*

   Re-expression berfokus pada pertanyaan tentang skala apa saja yang akan membantu menyederhanakan analisis data. Dalam hal ini, fungsi matematika digunakan untuk menjelaskan perilaku dan karakteristik data.
4. *Resistance*

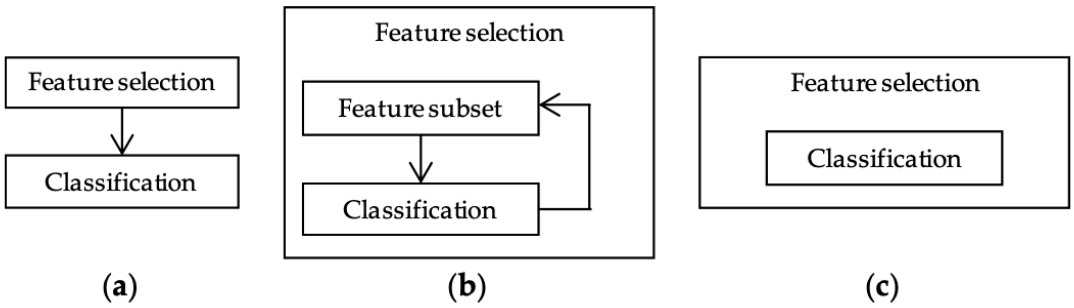
   Resistance yaitu memastikan bahwa hasil analisis data tidak hanya dipengaruhi oleh sekelompok data saja.

2.5 Seleksi Fitur

Seleksi fitur adalah salah satu teknik pra proses klasifikasi yang umum digunakan dalam *Machine Learning* dan ilmu Statistika untuk meningkatkan kinerja pembelajaran dan

mengatasi masalah pada *high dimensional data.* Pada *high dimensional data* atau data berdimensi tinggi, seleksi fitur digunakan untuk pemilihan subset fitur yang relevan dan menghapus fitur yang bersifat *redundant*, berlebihan, dan tidak diinginkan sebelum proses pembangunan model klasifikasi (Hameed et al., 2018). Fitur-fitur atau atribut yang bersifat *redundant* dihapus karena tidak berkontribusi dengan baik sebagai prediktor pada model pembelajaran karena informasi yang mereka berikan pada dasarnya telah disajikan atau diwakilkan oleh fitur lain (Yuan et al., 2017). Selain itu, fitur yang tidak relevan tidak hanya berdampak negatif pada keakuratan hasil klasifikasi, tetapi juga menambah kesulitan saat mencari informasi yang berguna pada data (Bielza & Larrañaga, 2020).

Terdapat tiga kategori metode seleksi fitur tergantung pada interaksinya dengan model pembelajaran, tiga kategori ini yaitu *filter method, wrapper method*, dan *embedded method* seperti yang terlihat pada Gambar 2.5 (Suppers et al., 2018).



Gambar 2.5 Interaksi Metode Seleksi Fitur Terhadap *Classifier*
Sumber: Suppers et al., 2018

Berdasarkan Gambar 2.5, (a) Metode *filter* melakukan pemilihan fitur secara independen tanpa konstruksi model klasifikasi, (b) Metode *wrapper* secara iteratif menambah atau menghilangkan sekumpulan fitur menggunakan akurasi prediksi dari model klasifikasi, dan (c) Dalam *Embedded Method*, seleksi fitur merupakan bagian integral (tidak terpisahkan) dari model klasifikasi.

2.5.1 Filter Method

Filter Method atau metode filter adalah metode seleksi fitur yang hanya berfokus pada karakteristik umum dan statistik dari dataset sehingga tidak bergantung pada metode pembelajaran apa pun (Lu et al., 2018). Karena independensinya dari algoritma induksi, metode filter ini tidak mahal secara komputasi dan memiliki kapasitas generalisasi yang baik.

Metode ini bertindak sebagai pemeringkat yang dapat mengurutkan fitur dari yang terbaik hingga yang terburuk. Pemeringkatan fitur tergantung pada kriteria statistik data, misalnya varians, konsistensi, jarak, informasi, korelasi, dll (Bielza & Larrañaga, 2020). Pada penelitian ini, akan diimplementasikan empat jenis algoritma seleksi fitur dari metode *filter* yaitu *Analysis of Variance* (ANOVA), *Mutual Information*, *Relief*, dan *Fisher Score.*

***Analysis of Variance* (ANOVA)**

Tujuan seleksi fitur ANOVA adalah untuk membandingkan apakah dua atau lebih kelompok populasi memiliki nilai rata-rata atau *mean* yang sama atau tidak untuk mengetahui perbedaan antara estimasi sampel dalam varians dan sampel antar varians (Kuswanto, 2020). Seleksi fitur ANOVA menggunakan uji-F untuk menentukan apakah variabilitas antara rata-rata kelompok fitur lebih besar daripada variabilitas pengamatan di dalam kelompok fitur. Uji-F sendiri adalah salah satu uji statistik yang memberikan nilai-f dengan cara menghitung rasio antar varians. Dalam penelitian ini, akan digunakan *One-way* ANOVA untuk menghitung rasio varians antar kelompok fitur dan varians di dalam kelompok fitur. Dalam hal ini, grup adalah *instance* dengan nilai target yang sama. Metode seleksi fitur ANOVA ini menggunakan uji-f, fitur-fitur tersebut diurutkan (rangking) berdasarkan nilai *f-score* yang lebih tinggi (Lu et al., 2018). Nilai-F dalam metode ANOVA dapat ditemukan menggunakan Persamaan 2.2:

$$RF = \frac{variance\ between\ classes}{variance\ within\ class} \tag{2.2}$$

variance between classes yaitu:

$$\sum_{i=1}^{K} n_i(\overline{Y}_i - \overline{Y})^2 / (K-1) \tag{2.3}$$

Pada Persamaan 2.3, $\overline{Y}_i$ menunjukkan *sample mean* pada kelompok ke-*i*, n_i merupakan total observasi pada kelompok ke-*i,* $\overline{Y}$ menunjukkan nilai rata-rata atau *mean* dari seluruh data, dan *K* menunjukkan total kelompok. *Variance within class* pada Persamaan 2.2 dijelaskan pada Persamaan 2.4 berikut:

$$\sum_{i=1}^{K}\sum_{j=1}^{n_i}(Y_{ij} - \overline{Y}_i)^2 / (N-K) \tag{2.4}$$

Pada Persamaan 2.4, $\overline{Y}_{ij}$ adalah observasi ke-*j* pada elemen ke-*i* dari kelompok *K* dan *N* adalah *sample size* secara keseluruhan pada kelompok ke-*i.*

Mutual Information

Seleksi fitur *Mutual Information* (MI) digunakan untuk melihat dan menghitung ukuran ketergantungan antara dua variabel acak yang digunakan dalam Teori Informasi, Statistik, dan *Machine Learning* (Noshad et al., 2019). Nilai *Mutual Information* (MI) antara dua variabel acak adalah non-negatif, yang mengukur ketergantungan antara variabel. Nilai *Mutual Information* (MI) adalah nol jika dan hanya jika dua variabel acak bersifat independen, dan semakin tinggi nilai MI berarti nilai ketergantungan lebih tinggi (Zhao et al., 2016). Nilai *Mutual Information* antara dua variabel acak X dan Y dapat dinyatakan secara formal sebagai Persamaan 2.5.

$$I(X\ ;\ Y) = H(X) - H(X\ |Y) \tag{2.5}$$

Pada Persamaan 2.5, $I(X\ ;\ Y)$ adalah informasi yang bersifat *mutual* untuk X dan Y. $H(X)$ adalah entropi untuk *X,* dan $H(X\ |Y)$ adalah nilai entropi bersyarat untuk *X* yang diberikan *Y.*

ReliefF

Sebagai salah satu metode seleksi fitur dari kategori *filter*, *ReliefF* menghitung statistik proksi untuk setiap fitur yang dapat digunakan dengan memperkirakan kualitas dari suatu fitur atau relevansinya dengan konsep target berdasarkan atribut statistik yang disebut sebagai *feature weight* (W [A] = bobot fitur 'A'), atau biasa disebut sebagai 'skor' fitur yang dapat berkisar dari -1 (terburuk) hingga +1 (terbaik) (Urbanowicz et al., 2018). Algoritma ini merupakan algoritma pemilihan atribut yang berbasis pada *instance* atau *record*. Pemilihan atribut dilakukan dengan menghitung perbedaan bobot untuk setiap *instance* dengan dipilih secara *random* sebagai *near hit* (tetangga terdekat terpilih pada kelas yang sama) dan *near miss* (tetangga terdekat terpilih pada kelas yang berbeda) (Fitriani et al., 2020). Cara kerja *ReliefF* dapat dilihat pada Gambar 2.6.

Algorithm 1 Pseudo-code for the original Relief algorithm

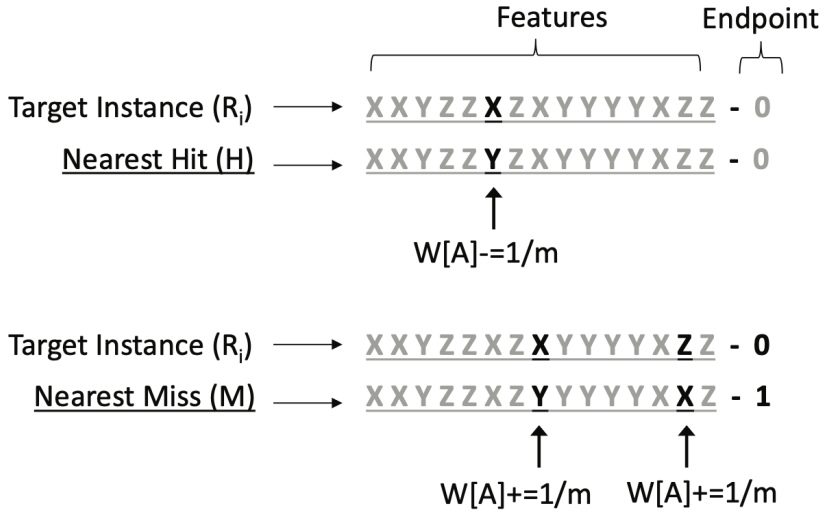
Require: for each training instance a vector of feature values and the class value
  $n \leftarrow$ number of training instances
  $a \leftarrow$ number of features (i.e. attributes)
  Parameter: $m \leftarrow$ number of random training instances out of n used to update W

  initialize all feature weights $W[A] := 0.0$
  for i:=1 **to** m **do**
    randomly select a 'target' instance R_i
    find a nearest hit 'H' and nearest miss 'M' (instances)
    for A:= 1 **to** a **do**
      $W[A] := W[A] - diff(A, R_i, H)/m + diff(A, R_i, M)/m$
    end for
  end for
  return the vector W of feature scores that estimate the quality of features

Gambar 2.6 *Pseudocode* dari Algoritma ReliefF

Sumber: Urbanowicz et al., 2018

Seperti yang tertera pada Gambar 2.6, *ReliefF* berputar melalui m pelatihan acak dari *instances* (Ri) yang dipilih tanpa penggantian, di mana m adalah parameter yang ditentukan pengguna. Pada setiap siklus, Ri adalah *instance* 'target' dan bobot skor fitur W diperbarui berdasarkan perbedaan nilai fitur yang diamati antara target dan instance tetangga. Oleh karena itu setiap siklus, jarak antara *instance* 'target' dan semua instance lainnya dihitung. *ReliefF* mengidentifikasi dua contoh tetangga terdekat dari target yaitu yang satu kelas disebut *nearest hit* (H) dan yang dengan kelas yang berlawanan disebut *nearest miss* (M). Selain itu, cara kerja *ReliefF* juga ditunjukkan oleh Gambar 2.7.



Gambar 2.7 Cara kerja algoritma ReliefF

Seperti yang terlihat pada Gambar 2.7, Relief memperbarui W[A] untuk *instance* target yang diberikan saat dibandingkan dengan *nearest miss* dan *nearest hit* nya. Dalam contoh ini, fitur adalah diskrit dengan kemungkinan nilai X, Y, atau Z, dan titik akhir adalah biner dengan nilai 0 atau 1. Perhatikan bahwa ketika nilai fitur berbeda, bobot fitur yang sesuai meningkat sebesar 1/m untuk nearest miss dan berkurang 1/m untuk *nearest hit*

Fisher Score

Fisher Score merupakan teknik seleksi fitur yang berbasis pada ranking dari ratio masing-masing fitur. Fisher Score bertujuan untuk menghilangkan fitur-fitur yang tidak relevan dan bersifat redundant menggunakan metode diskriminatif dan model statistik generatif. Nilai dari *Fisher Score* dapat dihitung berdasarkan nilai rata-rata atau mean dari varian fitur pada setiap kelas. Semakin tinggi nilai *Fisher Score* pada suatu fitur, semakin baik dalam membedakan objek antar kelas (Boonthong et al., 2016;Widagdo et al., 2020). Nilai dari *Fisher Score* dapat dihitung menggunakan Persamaan 2.6:

$$FScore_r = \frac{\sum_{i=1}^{c} n_i(\mu_r^i - \mu_r)^2}{\sum_{i=1}^{c} n_i(\sigma_r^i)^2} \tag{2.6}$$

Keterangan:

n_i menunjukkan jumlah sampel pada kelas ke i.

μ_r^i merupakan nilai rata-rata fitur ke$-r$ pada kelas ke$-i$.

σ_r^i merupakan nilai varian fitur $ke-r$ pada kelas $ke-i$.

μ_r merupakan nilai rata-rata fitur ke r.

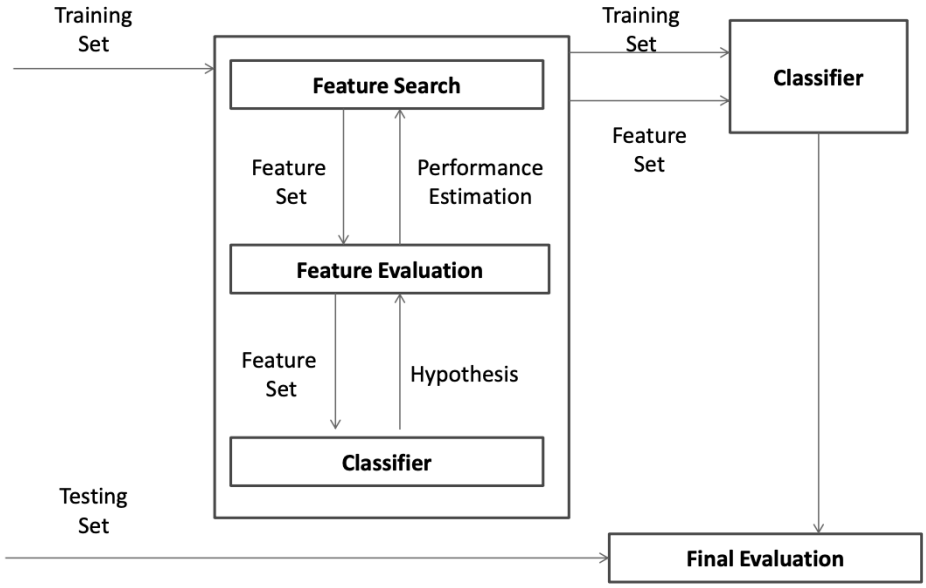
2.5.2 *Wrapper Method*

Metode *wrapper* membutuhkan sebuah algoritma *Machine Learning* untuk digunakan sebagai *black box evaluator* dalam proses seleksi atau menemukan *subset* fitur terbaik sehingga metode ini sangat bergantung pada jenis *classifier* (Malekipirbazari et al., 2021). Pada dasarnya, setiap kombinasi antara *search strategy* dan algoritma pemodelan dapat digunakan sebagai metode *wrapper (Effrosynidis & Arampatzis, 2021)*. Implementasi sebuah *wrapper* yang dilakukan dalam kumpulan data dengan banyak fitur biasanya menghabiskan sumber daya, komputasi dan juga waktu yang sangat banyak saat dijalankan. Namun, metode ini

mudah diterapkan dan dapat memodelkan dependensi fitur (Tang et al., 2014). Menurut buku (Tang et al., 2014), secara umum model pada metode *wrapper* akan bekerja seperti berikut:

1. Mencari subset fitur,
2. Mengevaluasi subset fitur yang dipilih dengan mengandalkan kinerja classifier,
3. Ulangi Langkah 1 dan Langkah 2 sampai kualitas yang diinginkan tercapai.

Kerangka umum dalam menerapkan metode wrapper dalam melakukan seleksi fitur untuk klasifikasi berisi tiga komponen utama dan ditunjukkan oleh Gambar 2.8



Gambar 2.8 Kerangka Umum Metode Seleksi Fitur *Wrapper* untuk Klasifikasi.
Sumber: Tang et al., 2014

Kerangka umum untuk seleksi fitur metode *wrapper* seperti yang ditunjukkan pada Gambar 2.8 berisi dua komponen utama, yaitu *Feature Search* dan *Feature Evaluation. Feature Search* merupakan cara mencari *subset* fitur dari semua kemungkinan *subset* fitur, *Feature Evaluation* adalah bagaimana pengembangan kinerja pengklasifikasi yang dipilih. Pada penelitian ini, akan digunakan tiga jenis algoritma seleksi fitur dari metode *wrapper*, yaitu *Sequential Forward Selection* (SFS)*, Backward Elimination* (BE), dan *Recursive Feature Elimination* (RFE).

***Sequential Forward Selection* (SFS)**

Sequential Forward Selection (SFS) adalah teknik seleksi fitur yang dimulai dengan mengevaluasi satu subset atribut input. Dalam seleksi fitur berdasarkan *Sequential Forward*

Selection, model *Machine Learning* dipertimbangkan dalam proses yang iteratif, yang berarti akan terus menambahkan fitur di setiap iterasi sampai akurasi model tidak meningkat dengan penambahan fitur lebih lanjut (Saha et al., 2020).

Menurut (Chandra, 2015), langkah-langkah dalam menerapkan seleksi fitur *Sequential Forward Selection* (SFS) adalah sebagai berikut:

1. Fitur terpenting $S_1 = f_i$ dipilih terlebih dahulu menggunakan beberapa kriteria.
2. Kemudian pasangan fitur dibentuk dengan f_i dan pasangan terbaik dipilih sebagai $S_2 = \{f_i, f_i\}$.
3. Kumpulan tiga fitur dibentuk menggunakan S_2 dan kumpulan tiga fitur terbaik dipilih sebagai $S_2 = \{f_i, f_i, f_k\}$.
4. Proses ini diulang sampai sejumlah fitur yang telah ditentukan dipilih.

***Backward Elimination* (BE)**

Backward Elimination melakukan proses seleksi fitur dengan cara yang sebaliknya dibandingkan dengan *Sequential forward Selection* (SFS). Seleksi fitur dilakukan dengan cara menguji semua variabel menggunakan *classifier* kemudian menghapus variabel-variabel yang dianggap tidak memberikan pengaruh yang signifikan. Setiap fitur akan diproses satu per satu, jika fitur dianggap tidak berpengaruh atau berpengaruh tapi tidak signifikan dalam model maka akan dihapus. Iterasi akan diulangi sampai penghapusan fitur tidak lagi memberikan perbaikan terhadap model. Kelemahan dari *Backward elimination* adalah jika fitur telah dibuang maka tidak dapat ditambahkan lagi. Kegunaan dari fiturnya tidak dapat dievaluasi lagi dan dibawa kembali ke subset fitur yang dipilih (Ary & Rismiati, 2019).

Langkah-langkah dalam menerapkan seleksi fitur *Backward Elimination* adalah sebagai berikut:

1. Untuk setiap n fitur, akan ada sebuah *Predefined Criterion Function*.
2. Pada setiap langkah, satu fitur akan dibuang dan fungsi *Criterion* akan ditemukan untuk semua subset yang berisi n - 1 fitur
3. Berdasarkan fungsi *Criterion* tersebut, fitur yang mempunyai performa yang paling rendah akan dibuang.
4. Prosedur ini akan terus diulang sampai tersisa fitur yang telah ditentukan sebelumnya tersisa.

***Recursive Feature Elimination* (RFE)**

Recursive Feature Elimination (RFE) merupakan salah satu metode *wrapper* sehingga seleksi fitur pada *Recursive Feature Elimination* (RFE) bekerja dengan algoritma pembelajaran mesin sesuai dengan kinerja keberhasilan dari algoritma klasifikasi yang digunakan. Metode Recursive Feature Elimination (RFE) bekerja dengan menghapus atribut secara rekursif dan membangun model pada atribut yang tersisa (Shetye, 2019). Metode ini menghilangkan fitur yang tidak perlu dan lemah atau fitur yang paling tidak mempengaruhi keberhasilan model klasifikasi dan pada saat yang sama mempertahankan fitur yang efektif dan kuat yang meningkatkan keberhasilan model. Metode ini menggunakan prosedur iteratif yang bekerja mirip dengan *Backward Elimination*. Metode ini pertama akan membuat model pada seluruh set fitur dan menilai setiap fitur sesuai dengan efek dan kepentingannya pada variabel target. Setelah itu, model akan dibangun kembali setelah menghapus fitur yang paling tidak penting pada setiap langkah dan menghitung ulang pentingnya setiap fitur hingga keberhasilan model tertinggi tercapai (Akkaya, 2021).

2.5.3 *Embedded Method*

Metode *Embedded* adalah metode seleksi fitur yang menjembatani kesenjangan antara metode *filter* dan metode *wrapper* dengan cara melakukan seleksi fitur dan klasifikasi secara bersamaan (Algamal & Lee, 2015). Metode ini menggabungkan perhitungan dan kriteria statistik seperti yang ada pada metode *filter* untuk memilih dan menyeleksi beberapa fitur dan kemudian menggunakan salah satu algoritma pembelajaran mesin untuk memilih subset dengan kinerja klasifikasi terbaik. Metode *Embedded* juga mengurangi kompleksitas dari komputasi *wrapper* dengan cara tidak melakukan klasifikasi ulang pada himpunan bagian di setiap iterasi dan dapat memodelkan dependensi fitur. Metode *Embedded* tidak melakukan iterasi sehingga seleksi fitur dilakukan pada fase pembelajaran. (Effrosynidis & Arampatzis, 2021). Pada penelitian ini, akan diimplementasikan dua jenis algoritma seleksi fitur dari metode *embedded* yaitu *Least Absolute Shrinkage and Selection Operator* (LASSO) dan *Elastic Net.*

***Least Absolute Shrinkage and Selection Operator* (LASSO)**

Metode *Least Absolute Shrinkage and Selection Operator* (LASSO) dikemukakan oleh Tibshirani pada tahun 1996 untuk estimasi parameter dan juga pemilihan variabel dan model secara bersamaan dalam analisis regresi (Fonti, 2017). Metode LASSO adalah metode reduksi dimensi berdasarkan model regresi linier dengan fungsi penalti L1 yang telah menarik perhatian luas di bidang seleksi fitur karena kinerjanya yang efisien (Muthukrishnan & Rohini, 2017). Estimasi LASSO dapat didefinisikan seperti pada Persamaan 2.7

$$\hat{\beta}^{lasso} = \arg\min_{\beta} \left\{ \frac{1}{2} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \tag{2.7}$$

Atau dapat juga ditulis seperti Persamaan 2.8:

$$\hat{\beta}^{lasso} = \arg\min_{\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2 \tag{2.8}$$
$$\text{yang dipengaruhi oleh } \sum_{j=1}^{p} |\beta_j| \leq t$$

LASSO mengubah setiap koefisien dengan komponen konstan λ, membuang fitur yang memiliki koefisien nol. Oleh karena itu LASSO termasuk metode seleksi fitur berjenis *forward-looking* untuk regresi. LASSO mengurangi residual *sum of squares* dengan jumlah nilai absolut dari koefisien menjadi kurang dari konstanta. LASSO awalnya didefinisikan dalam konteks *least square*, tetapi juga dapat diperluas ke berbagai model. LASSO meningkatkan akurasi prediksi dan interpretasi model dengan menggabungkan kualitas *Ridge Regression* dan teknik pemilihan subset. Jika ada korelasi tinggi dalam kelompok prediktor, LASSO hanya memilih satu di antara mereka dan mengecilkan yang lain menjadi nol. Metode ini mengurangi variabilitas perkiraan dengan mengecilkan beberapa koefisien tepat ke nol menghasilkan model yang mudah dijelaskan (Muthukrishnan & Rohini, 2017).

Elastic Net

Elastic Net adalah modifikasi dari pendekatan regresi linear berganda yang dirancang untuk memecahkan masalah pemilihan fitur yang berdimensi tinggi (Fukushima et al., 2019). *Elastic Net* memilih variabel secara otomatis dan melakukan penyusutan terus menerus untuk meningkatkan akurasi prediksi menggunakan dua istilah penalti yaitu regularisasi L1 dan L2 (Sabett et al., 2017). Menurut (Zou & Hastie, 2005), metode ini bekerja seperti jaring ikan yang

dapat diregangkan dan dapat menyimpan "semua ikan besar", yaitu prediktor penting dan menghilangkan yang tidak relevan.

Misalkan kita memiliki $p = 1,..., P$ prediktor yang dilambangkan dengan $x_1,,..., x_p$, estimasi variabel respon Y dapat ditulis sebagai $\hat{Y} = \beta_0 + \beta_1 x_1 + .. \beta_p x_p$, berdasarkan regresi linier. Koefisien dari ($\widehat{\beta} = [\beta_0, \ldots, \beta_p]^T$) dihitung dengan meminimalkan nilai *sum of square* dari kesalahan residual dalam Persamaan 2.9.

$$SSE = \|Y - \hat{\beta}\|^2 \tag{2.9}$$

$$Dimana: Y = \begin{bmatrix} y1 \\ y2 \\ \vdots \\ \vdots \\ \vdots \\ yN \end{bmatrix} \text{ dan } X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1P} \\ 1 & x_{21} & \cdots & x_{2P} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{N1} & \cdots & x_{NP} \end{bmatrix}$$

Dalam kasus di mana dimensi ruang fitur lebih tinggi daripada jumlah pengamatan seperti pada penelitian ini, koefisien dihitung dengan meminimalkan fungsi L Persamaan 2.10 dan bukan dengan meminimalkan SSE (Wei et al., 2019).

$$L = SSE + \alpha\rho\|\hat{\beta}\|_1 + \alpha(1-\rho)\|\hat{\beta}\|^2 \tag{2.10}$$

$\|\hat{\beta}\|_1$, $\|\beta\|^2$, α, dan ρ didefinisikan secara berurutan pada persamaan 2.11 - 2.13.

$$\|\hat{\beta}\|_1 = \sum_{p=0}^{P} |\beta_p| \tag{2.11}$$

$$\|\beta\|^2 = \sum_{p=0}^{P} \beta_p^2 \tag{2.12}$$

$$\alpha > 0, 0 \leq \rho \leq 1 \tag{2.13}$$

Derajat kompleksitas model yang diberikan penalti dikendalikan oleh istilah pembobotan α dan ρ. Karena hasil dari Elastic Net dipengaruhi oleh α dan ρ, konfigurasinya harus dilakukan dalam proses pembelajaran (Amini & Hu, 2021).

2.5.4 K-Fold Cross Validation

Cross Validation adalah salah satu metode *resampling data* yang paling banyak digunakan untuk menilai kemampuan generalisasi dari model prediktif dan untuk mencegah terjadinya *overfitting* pada model (Berrar, 2018). Dalam membangun model akhir untuk prediksi kasus nyata di masa depan, fungsi pembelajaran atau algoritma pembelajaran *f* biasanya diterapkan ke seluruh rangkaian pembelajaran. Tujuan dari *cross validation* dalam fase pembangunan model adalah untuk memberikan perkiraan kinerja model akhir ini pada data baru. Salah satu teknik dari *cross validation* adalah *k-fold cross validation,* yang mana memecah data menjadi k bagian set data dengan ukuran yang sama untuk menghilangkan bias (Tempola et al., 2018).

K-Fold Cross Validation adalah metode validasi dengan membagi data ke dalam *k-subset*, kemudian melakukan pengulangan sebanyak *k* kali untuk *training* dan *testing*. Dalam setiap iterasi atau pengulangan, digunakan satu *subset* sebagai data uji dan *subset* lainnya sebagai data pembelajaran (Azis et al., 2020). Alur kerja *K-fold cross-validation* diilustrasikan pada Gambar 2.9

Gambar 2.9 *10-fold Cross Validation*

Sumber: Ashfaque & Iqbal, 2019

Pada *K-fold Cross Validation*, dataset dibagi menjadi 10-*fold* independen. Jumlah sampel di setiap lipatan adalah sama. Setiap iterasi, 9 lipatan digunakan untuk pelatihan dan lipatan yang tersisa digunakan sebagai dataset pengujian, tanpa persimpangan data. Proses ini diulang sepuluh kali pada tingkat pembelajaran yang sama dan akan menghasilkan evaluasi rata-rata dari sepuluh hasil seperti yang ditunjukkan pada Gambar 2.9. Dalam setiap iterasi dalam *K-fold Cross Validation*, kombinasi data yang digunakan pada data *training* dan data *testing* selalu berbeda dan dapat diringkas sebagai Persamaan 2.14

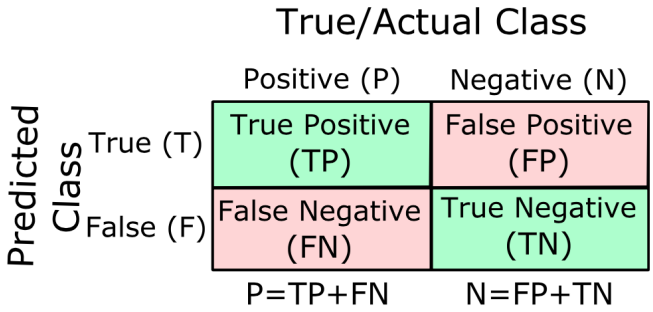
$$D = D_1 \cup D_2 \cup \ldots D_{10}, \qquad (2.14)$$

$$D_i \cap D_j = \emptyset \; (i \neq j)$$

Pada Persamaan 2.13, D merepresentasikan total dataset, lalu D_i dan D_j merepresentasikan *independent subset* (He et al., 2018).

2.6 Confusion Matrix

Confusion matrix merupakan salah satu metode pengujian yang dapat digunakan untuk mengukur kinerja dari sebuah metode klasifikasi. *Confusion matrix* ini mengandung informasi yang dapat membandingkan hasil klasifikasi yang telah dilakukan oleh model klasifikasi dengan hasil klasifikasi yang seharusnya (Karsito & Susanti, 2019). Terdapat empat kemungkinan keluaran yang merepresentasikan elemen dari *confusion matrix* atau *contingency table*. Keempat istilah tersebut adalah True Positive (TP), True Negative (TN), False Positive (FP) dan False Negative (FN) (Tharwat, 2018). *Confusion Matrix* dapat digambar seperti Gambar 2.10.



Gambar 2.10 *Confusion Matrix*

Diagonal hijau pada Gambar 2.10 menunjukkan prediksi yang benar dan diagonal merah muda menunjukkan prediksi yang salah.

1. **TP (True Positive) :** Nilai sebenarnya positif dan model memprediksi nilai positif
2. **FP (False Positive)** : Nilai sebenarnya positif tetapi model memprediksi nilai negatif (**Type 1 error**)
3. **FN (False Negative) :** Nilai sebenarnya negatif tetapi model memprediksi nilai positif (**Type 2 error**)
4. **TN (True Negative) :** Nilai sebenarnya negatif dan model memprediksi nilai negatif

Dari perhitungan *Confusion Matrix* dapat dihitung nilai *Accuracy, Precision,* dan *Recall* menggunakan Persamaan 2.15 - 2.17:

$$accuracy = \frac{TP + TN}{Total} \tag{2.15}$$

$$precision = \frac{TP}{TP + FP} \tag{2.16}$$

$$recall = \frac{TP}{TP + FN} \tag{2.17}$$

Accuracy merupakan ukuran untuk berapa banyak prediksi yang benar dari model yang telah dibangun terhadap dataset. Akurasi adalah metrik dasar yang baik untuk mengukur kinerja model. *Precision* memberi informasi tentang seberapa banyak kasus yang diprediksi dengan benar dan terbukti positif. Recall memberi informasi tentang berapa banyak kasus positif aktual yang dapat diprediksi dengan benar oleh model yang dibangun. Semakin tinggi nilai *Recall* berarti bahwa sebagian besar kasus positif (TP+FN) akan diberi label sebagai positif (TP). Sedangkan nilai *Recall* yang rendah berarti model memiliki jumlah FN yang tinggi (seharusnya positif tetapi diberi label sebagai negatif) (Ting, 2017).

2.7 Kajian Pustaka

Penelitian tentang implementasi seleksi fitur sebagai sebuah solusi untuk mengatasi masalah-masalah yang ada pada *high dimensional data* telah dilakukan oleh para peneliti sebelumnya. Beberapa penelitian dapat mencapai akurasi yang dibutuhkan hanya dengan menerapkan salah satu dari metode seleksi fitur saja, sedangkan beberapa dari penelitian tersebut butuh menggabungkan beberapa teknik seleksi fitur untuk mencapai target akurasi yang diinginkan.

Penelitian pertama dengan judul "Analisis Pengaruh *Kernel Support Vector Machine* (SVM) pada Klasifikasi Data Microarray untuk Deteksi Kanker" pada tahun 2017. Penelitian

ini dilakukan oleh (Diani, 2017) menggunakan algoritma seleksi fitur Analysis of Variance (ANOVA) dengan tujuan mengatasi masalah *curse of dimensionality* yaitu untuk mereduksi dimensi yang ada pada empat dataset yang digunakan, selain itu ANOVA digunakan untuk menemukan gen informatif dalam dataset dan mengetahui interaksi antar gen serta pengaruh terhadap suatu perilaku. Hasil penelitian ini menunjukkan bahwa ANOVA terbukti dapat menemukan pasangan gen informatif dengan cara menghitung korelasi antar gen sehingga dapat diurutkan dari yang paling informatif. Hasilnya diambil 10 fitur yang paling penting dari setiap dataset yang dapat membantu dalam proses pengklasifikasian yang dilakukan oleh Support Vector Machine (SVM). Selain itu, kernel trick pada SVM yang diterapkan saat *learning model* sangat membantu dalam mengatasi masalah feature space. Penelitian ini menghasilkan klasifikasi dengan akurasi yang tinggi dari empat dataset yang digunakan. Untuk dataset leukimia dan ovarian cancer, akurasi terbesar dihasilkan oleh kernel polynomial yaitu sebesar 100% dan 97,54% dengan nilai parameter C=1.5, d=1 dan C=1.5, d=2 . Sedangkan untuk dataset lung cancer akurasi terbesar diperoleh dari kernel linear yaitu sebesar 100% dengan nilai parameter C=1.0 dan untuk dataset colon tumor akurasi terbesar diperoleh dari kernel RBF sebesar 85,15% dengan nilai parameter C=1.5 dan σ=0.5.

Penelitian kedua dengan judul "Klasifikasi Dokumen *Menggunakan Support Vector Machine* dan *Mutual Information*" pada tahun 2019. Penelitian ini dilakukan oleh (Kristiani, 2019) dengan menggunakan teknik seleksi fitur Mutual Information untuk menentukan suatu kata yang menjadi ciri khas atau kata unik yang digunakan dalam dokumen abstrak tugas akhir yang akan diklasifikasikan. Hasil dari seleksi fitur MI dapat mengurangi 10 fitur yang kurang relevan dari total 147 fitur. Pengujian pada penelitian ini dilakukan dengan cara membandingkan klasifikasi dengan proses seleksi fitur MI maupun tanpa seleksi fitur. Hasilnya yaitu nilai akurasi metode SVM dengan menggunakan MI dan tidak menggunakan MI untuk klasifikasi dokumen abstrak adalah sama yakni sebesar 94%. Tidak ada perbedaan akurasi walaupun fitur sudah dikurangi sebanyak 10 fitur. Hal ini membuktikan kombinasi SVM dan MI dapat bekerja dengan baik.

Penelitian ketiga dengan judul “Seleksi Fitur *ReliefF* Pada Klasifikasi *Malware* Android Menggunakan *Support Vector Machine* (SVM)” pada tahun 2020. Penelitian ini dilakukan oleh (Fitriani et al., 2020) dengan tujuan melihat seberapa signifikan pengaruh seleksi fitur *ReliefF* terhadap hasil klasifikasi *malware* Android menggunakan *Support Vector Machine* (SVM).

Penelitian ini juga menggunakan metode seleksi fitur pembanding yaitu *Chi-Square* (CHI), *Correlation-based Feature Selection* (CFS), dan *Gain Ratio* (GR). Hasilnya klasifikasi jenis Malware Android dapat bekerja dengan optimal, tetapi metode Relieff yang digunakan sebagai metode seleksi fitur dalam penelitian ini tidak memberikan hasil akurasi yang diharapkan karena hasil akurasinya berada sangat jauh di bawah hasil akurasi dari data original (dataset tanpa seleksi fitur). Akurasi klasifikasi Seleksi Fitur *ReliefF*, *Chi-Square*,CFS, dan *Gain Ratio* semuanya menghasilkan hasil 33.33333%. Sedangkan hasil klasifikasi tanpa Seleksi Fitur memberikan hasil yang cukup tinggi yaitu 95%. Menurut penelitian ini, rendahnya akurasi hasil klasifikasi dengan seleksi fitur disebabkan karena seleksi fitur tidak cocok digunakan dengan data yang sedikit.

Penelitian keempat dengan judul "Kombinasi Feature Selection *Fisher Score* dan *Principal Component Analysis* Untuk Klasifikasi *Cervix Dysplasia* (PCA)" pada tahun 2020. Penelitian ini dilakukan oleh (Widagdo et al., 2020) dengan tujuan melihat pengaruh penerapan metode seleksi fitur *Fisher Score* dan ekstraksi fitur PCA dalam mengatasi fitur yang bersifat *irrelevant* dan *redudant* untuk meningkatkan kinerja klasifikasi *Cervix Dysplasia.* Pada penelitian ini, *Fisher Score* digunakan untuk memilih kandidat fitur berdasarkan perangkingan sementara *Principal Component Analysis* akan membentuk data baru dengan fitur tidak saling berkorelasi. Data set hasil ekstraksi Principal Component Analysis akan digunakan sebagai fitur metode jaringan syaraf tiruan *Backpropagation.* Penggunaan metode seleksi fitur *Fisher Score* dan PCA terbukti signifikan memberikan peningkatan dalam hal mengklasifikasikan cervix dysplasia. Penggunaan Fisher Score dan PCA menghasilkan kinerja terbaik dari semua eksperimen dengan akurasi 0,964, Sensitivity 0,990 dan specificity 0,889. Selain itu dari segi waktu komputasi, kombinasi Fisher Score dan PCA mampu mempercepat komputasi algoritma Backpropagation. Waktu yang dibutuhkan dalam proses klasifikasi sebesar 15,49 detik

Penelitian kelima dengan judul “Implementasi Seleksi Fitur dengan *Backward Elimination* untuk Klasifikasi Prediksi Perceraian” pada tahun 2021. Penelitian yang dilakukan oleh (Raihan et al., 2021) ini melakukan perbandingan akurasi dari metode seleksi fitur *Backward Elimination* yang diterapkan ke empat algoritma klasifikasi yaitu *Artificial Neural Network* (ANN), *Gaussian Naive Bayes*, *Multi-Layer Perceptron*, dan *Decision Tree*. Pada penelitian ini juga dilakukan preprocessing agar tidak terjadi bias pada data seperti menghapus data yang duplikat dan melihat missing values. Hasilnya diambil 6 dari 54 fitur yang menghasilkan performa terbaik bagi masing-masing algoritma dengan pembagian data hold-

out. Keenam fitur yang diperoleh dari seleksi fitur *Backward Elimination* yang telah divalidasi dengan *k-fold cross-validation*. Hasil klasifikasi yang dilakukan menggunakan *Gaussian naive Bayes* menghasilkan performa tertinggi dengan akurasi 97.34%, presisi, 100%, dan f-score 96.87%.

Penelitian keenam dengan judul "Implementasi Metode *Forward Selection* Pada Algoritma *Support Vector Machine* (SVM) dan *Naive Bayes Classifier Kernel Density*" pada tahun 2019. Penelitian ini dilakukan oleh (Sasongko & Arifin, 2019) untuk membandingkan implementasi dan pengaruh dari metode seleksi fitur *Forward Selection* pada algoritma klasifikasi SVM dan *Naïve Bayes Kernel Density*. Pembentukan model klasifikasi pada penelitian ini dilakukan dengan menganalisis perubahan *kernel*, faktor penalti (C) SVM, jumlah kernel *Naïve bayes kernel density*, dan hasil *feature subset forward selection*. Selain itu digunakan juga lima buah eksperimen *kernel* SVM yaitu *dot* (linear), *radial* (RBF), *polynomial*, *neural*, dan *dananova*. Implementasi metode seleksi fitur *forward selection* terbukti dapat bekerja dengan baik pada model baik pada algoritma SVM maupun *Naïve bayes kernel density*. Akurasi tertinggi pada klasifikasi dengan dataset sekolah ABC diperoleh oleh algoritma FS-SVM dengan parameter C sebesar 10.0 yaitu 99.29%. Pengujian model klasifikasi FS-SVM parameter dengan C sebesar 10.0 pada dataset peminatan sekolah XYZ memperoleh akurasi sebesar 95.17% dan nilai AUC 0.956.

Penelitian ketujuh dengan judul "Implementasi Metode *Random Forest* dan *Recursive Feature Elimination* Untuk Klasifikasi Berita" pada tahun 2021. Penelitian ini dilakukan oleh (Havis, 2021) untuk mengukur pengaruh dari implementasi metode seleksi fitur *Recursive Feature Elimination* (RFE) terhadap model *Random Forest* (RF) dalam mengklasifikasikan sub kategori berita pada situs merahputih.com. Pada penelitian ini dilakukan proses uji coba berdasarkan data yang telah tersedia dan parameterisasi model dengan metode TF-IDF dan *Random Forest* yang telah dilengkapi metode *Recursive Feature Elimination* serta *text preprocessing*. Implementasi *Recursive Feature Elimination* pada model klasifikasi *Random Forest* dapat mengurangi sejumlah 19% fitur dan mendapatkan performa F1-Score sebesar 93,44%.

Penelitian kedelapan dengan judul "*Feature selection with Lasso for classification of ischemic strokes based on EEG signals*" pada tahun 2020. Penelitian ini dilakukan oleh (Angga Yuwono et al., 2020) dengan menggunakan salah satu tugas pemodelan statistik utama yaitu seleksi fitur *Least Absolute Shrinkage and Selection Operator* (LASSO), dimana metode ini

dapat memilih fitur yang relevan dengan mengecilkan beberapa nilai koefisien menjadi nol. Penelitian ini menggunakan *Random Forest* sebagai *classifier*-nya. Hasil penelitian ini menunjukkan bahwa metode LASSO dapat mengoptimalkan kinerja *Random Forest* dengan menghasilkan nilai akurasi 75% dari mereduksi 45 fitur menjadi 24 fitur.

Penelitian kesembilan dengan judul "*Efficient and sparse feature selection for biomedical text classification via the elastic net: Application to ICU risk stratification from nursing notes*" pada tahun 2015. Penelitian ini dilakukan oleh (Marafino et al., 2015) dengan mengimplementasikan salah satu metode dari seleksi fitur *Embedded* yaitu *Elastic Net* dalam mengembangkan model pengklasifikasi berdasarkan teks bebas dari catatan keperawatan untuk memprediksi risiko kematian ICU dan untuk menemukan fitur teks paling kuat yang terkait dengan kematian menggunakan *Elastic Net*. Hasil penerapan *Elastic Net Regularization* terhadap dataset teks bebas klinis pada penelitian ini terbukti dapat mengurangi jumlah fitur menjadi 465 fitur. Hal ini mewakili hanya 0,00025% dari jumlah *input* fitur, atau hampir sepuluh ribu kali pengurangan, dan AUC untuk pengklasifikasi ini adalah 0,889, sehingga membuat pengklasifikasi tersebut lebih mudah ditafsirkan dan lebih menghemat kinerja. Fitur yang dipilih juga relevan secara klinis dan berkorelasi baik dengan apa yang saat ini diketahui tentang hasil ICU. Selain itu, dengan menghindari *overfitting*, mengklasifikasi teks reguler memiliki potensi untuk meningkatkan kegunaan dan portabilitas metode yang ada dalam bidang NLP klinis.

Tabel 2.1 Tabel perbandingan pustaka yang dikaji

Judul Penelitian, Peneliti	Metode Seleksi Fitur	Metode Klasifikasi	Dataset	Hasil
Analisis Pengaruh Kernel Support Vector Machine (SVM) pada Klasifikasi Data Microarray untuk Deteksi Kanker, (Diani, 2017)	ANOVA	*Support Vector Machine* (SVM)	*Dataset* yang digunakan dalam penelitian ini terdiri dari empat *dataset* DNA *microarray* yaitu leukimia, *colon tumor*, *lung cancer,* dan *ovarian cancer*	ANOVA terbukti dapat menemukan pasangan gen informatif dan model menghasilkan akurasi yang tinggi dari empat *dataset* yang digunakan. Untuk *dataset* leukimia dan *ovarian cancer*, akurasi terbesar

				dihasilkan oleh kernel polynomial yaitu sebesar 100% dan 97,54%, Sedangkan untuk *dataset lung cancer* akurasi terbesar diperoleh dari kernel linear yaitu sebesar 100% dan untuk *dataset colon tumor* akurasi terbesar diperoleh dari kernel RBF sebesar 85,15%.
Klasifikasi Dokumen Menggunakan Support Vector Machine dan Mutual Information (Kristiani, 2019)	*Mutual Information*	*Support Vector Machine* (SVM)	*Dataset* yang digunakan pada penelitian ini berisi 200 data abstrak.	Nilai akurasi metode SVM dengan menggunakan MI dan tidak menggunakan MI untuk klasifikasi dokumen abstrak adalah sama yakni sebesar 94%. Tidak ada perbedaan akurasi walaupun fitur sudah dikurangi sebanyak 10 fitur. Hal ini membuktikan kombinasi SVM dan MI dapat bekerja dengan baik.
Seleksi Fitur Relieff Pada Klasifikasi Malware Android Menggunakan Support Vector Machine(SVM), (Fitriani et al., 2020)	*ReliefF, Chi-Square, Correlation-based Feature Selection*(CFS), dan *Gain Ratio*	*Support Vector Machine* (SVM)	Data yang digunakan dalam penelitian ini adalah APK file yang didapatkan dari situs *VirusShare*	Klasifikasi jenis *Malware Android* dapat bekerja dengan optimal, tetapi metode *ReliefF* dan semua metode seleksi fitur perbandingan yang digunakan dalam penelitian ini tidak memberikan hasil akurasi yang bagus dan hanya mencapai 33.33%
Kombinasi Feature Selection *Fisher*	*Fishser Score*	*Backpropagation*	Dataset yang digunakan berupa citra sel tunggal pap	Penggunaan metode seleksi fitur terbukti

Score dan *Principal Component Analysis* Untuk Klasifikasi *Cervix Dysplasia* (PCA)			smear berasal dari Rumah Sakit Universitas Herlev (Denmark)	signifikan memberikan peningkatan dalam hal mengklasifikasikan cervix dysplasia. Penggunaan Fisher Score dan PCA menghasilkan kinerja terbaik dari semua eksperimen dengan akurasi 0,964, Sensitivity 0,990 dan specificity 0,889. Selain itu dari segi waktu komputasi, kombinasi Fisher Score dan PCA mampu mempercepat komputasi algoritma Backpropagation. Waktu yang dibutuhkan dalam proses klasifikasi sebesar 15,49 detik
Implementasi Seleksi Fitur dengan Backward Elimination untuk Klasifikasi Prediksi Perceraian, (Raihan et al., 2021)	*Backward Elimination*	ANN (*Artificial Neural Network*) *Gaussian Naive Bayes, Multi-Layer Perceptron*, dan *Decision Tree*	Dataset yang digunakan pada penelitian ini adalah data *Divorce Predictors Scale* (DPS) yang dapat diakses secara publik di UCI *Machine Learning Repository*	Hasil klasifikasi yang dilakukan menggunakan Gaussian naive Bayes menghasilkan performa tertinggi dengan akurasi 97.34%, presisi, 100%, dan f-score 96.87%.
Implementasi Metode Forward Selection Pada Algoritma Support Vector Machine (SVM) dan Naive Bayes Classifier Kernel Density,	*Forward Selection*	*Support Vector Machine* (SVM) dan *Naive Bayes Classifier Kernel Density*	Penelitian ini menggunakan dataset peminatan SMA dari dua sekolah berbeda	Implementasi metode forward selection terbukti dapat meningkatkan nilai akurasi dari hasil klasifikasi model baik pada

(Sasongko & Arifin, 2019)				algoritma SVM maupun Naïve bayes kernel density. Akurasi tertinggi pada klasifikasi dengan dataset sekolah ABC diperoleh oleh algoritma FS-SVM dengan parameter C sebesar 10.0 yaitu 99.29%. Pengujian model klasifikasi FS-SVM parameter dengan C sebesar 10.0 pada dataset peminatan sekolah XYZ memperoleh akurasi sebesar 95.17% dan nilai AUC 0.956.
Implementasi Metode Random Forest dan Recursive Feature Elimination Untuk Klasifikasi Berita, (Havis, 2021)	*Recursive Feature Elimination* (RFE)	*Random Forest*	Dataset pada penelitian ini diperoleh dengan melakukan proses *web crawling* pada situs merahputih.com milik PT. Merah Putih	Implementasi Recursive Feature Elimination pada model klasifikasi Random Forest dapat mengurangi sejumlah 4% fitur dan mendapatkan performa F1-Score sebesar 93,31%,
Feature selection with Lasso for classification of ischemic strokes based on EEG signals, (Yuwono et al., 2020)	LASSO	*Random Forest*	Dataset yang digunakan berisi data *stroke* berdasarkan sinyal EEG.	Hasil penelitian ini menunjukkan bahwa metode LASSO dapat mengoptimalkan kinerja Random Forest dengan menghasilkan nilai akurasi 75% dari mereduksi 45 fitur menjadi 24 fitur.
Efficient and sparse feature selection for biomedical text classification via	*Elastic Net*	*Support Vector Machine* (SVM)	Penelitian ini menggunakan catatan keperawatan 24 jam pertama dari 25.826 pasien dewasa yang masuk ICU	Penerapan Elastic Net Regularization terhadap dataset

the elastic net: Application to ICU risk stratification from nursing notes			dari *database* MIMIC-II.	teks bebas klinis pada penelitian ini terbukti dapat mengurangi jumlah fitur yang dipilih lebih dari seribu kali lipat, sehingga membuat pengklasifikasi tersebut lebih mudah ditafsirkan dan lebih menghemat kinerja serta dapat menghindari *overfitting*. Fitur yang dipilih juga relevan secara klinis dan berkorelasi baik dengan apa yang saat ini diketahui tentang hasil ICU.

Berdasarkan tinjauan pustaka dari Tabel 2.1 di atas, diketahui bahwa seleksi fitur sangat berpengaruh dalam mengatasi data yang memiliki fitur yang sangat banyak bahkan *high dimensional data*. Oleh karena itu akan digunakan masing-masing satu metode seleksi fitur dari setiap pustaka yang dibahas pada tinjauan pustaka di atas sebagai metode seleksi fitur pada data NIRS untuk penelitian ini. Tidak seperti penelitian-penelitian di atas yang hanya menggunakan satu metode seleksi fitur, Penelitian ini akan menggunakan sembilan metode dari tiga kategori seleksi fitur untuk melakukan perbandingan antar metode dan kategori seleksi fitur.

BAB III
METODOLOGI PENELITIAN

Metodologi penelitian adalah tahapan proses yang menjadi pedoman untuk melakukan penelitian agar tujuan dari penelitian dapat tercapai.

2.8 Tahapan Penelitian

Penelitian ini akan dilakukan melalui beberapa tahap yaitu pengumpulan dan analisis data, seleksi fitur, pembangunan model klasifikasi, pengujian dan perbandingan model, dan penarikan kesimpulan. Gambar 3.1 berikut akan menunjukkan tahapan dari penelitian.

Gambar 3.1. Diagram alur penelitian

2.8.1 Pengumpulan dan Analisis Data

Pada tahap ini, data yang dibutuhkan untuk membangun model klasifikasi akan dikumpulkan, dianalisis, dan dilakukan juga proses *preprocessing* pada data. Data pada penelitian ini didapatkan dari penelitian sebelumnya dengan judul "*Near infrared spectroscopic data for rapid and simultaneous prediction of quality attributes in intact mango fruits*" oleh (Samadi et al., 2020). Dataset ini berisi data spektrum NIRS yang diperoleh dari total 186 buah mangga utuh dari 4 kultivar yang berbeda. Data tersedia sebagai format ekstensi file XLS dan UNSB dengan jumlah data yang berisi 186 baris dan 1563 kolom. Kolom pada *dataset* ini terdiri dari spektrum data NIRS, label, dan kolom prediksi vitamin, kelarutan, dan keasaman dari kultivar buah mangga hasil penelitian sebelumnya.

Dataset yang didapatkan kemudian akan disiapkan dan dianalisis menggunakan *Exploratory Data Analysis* (EDA) untuk karakteristik data. EDA juga diterapkan untuk melihat ada atau tidaknya *missing value* pada atribut data, melihat jenis data bersifat linear atau non-linear dan melihat korelasi antar data agar datanya dapat lebih mudah dipahami dan siap untuk diproses. *Dataset* spektrum NIRS ini dapat di akses melalui situs Mendeley Data (https://data.mendeley.com/datasets/b9d6s7hr33/1). Setelah melakukan EDA, dilakukan juga proses *preprocessing* pada data agar data bisa digunakan dengan baik. Pada tahap ini, akan

dilakukan pembuangan pada beberapa kolom yang bukan termasuk data spektrum NIRS serta konversi nilai variabel *output* dari *string* ke *integer*.

2.8.2 Seleksi Fitur

Pada tahap ini, fitur-fitur penting dan relevan dari dataset yang sudah bersih akan diseleksi atau dipilih menggunakan 3 kategori seleksi fitur yaitu *filter, wrapper, dan embedded* dengan sembilan metode berbeda. Pada kategori *Filter*, digunakan metode ANOVA, *Mutual Information*, *Fisher Score*, dan *ReliefF*. Selanjutnya pada kategori seleksi fitur *Wrapper* digunakan tiga metode yaitu *Forward Selection*, *Backward Elimination*, dan *Recursive Feature Elimination* (RFE). Terakhir, seleksi fitur *Least Absolute Shrinkage and Selection Operator* (LASSO) dan *Elastic Net* akan diimplementasikan sebagai seleksi fitur kategori *Embedded*.

Selain itu, akan diterapkan *trial and error* dalam menentukan jumlah fitur yang akan digunakan untuk mendapatkan model dengan performa klasifikasi tertinggi. Pada kategori seleksi fitur yang melakukan pengurutan atau *ranking* pada seluruh fitur yang diperoleh berdasarkan koefisien dan urutan kepentingan fitur seperti kategori *filter* dan *embedded*, tahap *trial and error* ini akan menambahan fitur pada model klasifikasi mulai dari satu fitur sampai seratus fitur untuk melihat jumlah fitur optimal yang dapat menghasilkan performa terbaik. Untuk kategori *wrapper,* fitur yang akan digunakan pada model klasifikasi hanya kelipatan 20 dengan batas 100 fitur.

2.8.3 Pembangunan Model Klasifikasi

Dataset baru yang telah diperoleh dari hasil seleksi fitur akan diklasifikasi menggunakan *Random Forest* dan *Support Vector Machine* (SVM). Implementasi *Random Forest* sebagai metode klasifikasi dari salah satu *Ensemble Learning* dilakukan karena mampu membangun model dengan akurasi yang lebih tinggi daripada jika hanya menggunakan satu algoritma/model saja (Lee et al., 2020). Proses klasifikasi *Random Forest* ini dilakukan dengan cara membangun pohon keputusan sebanyak yang diperlukan. Selain itu, diimplementasikan juga metode klasifikasi *Support Vector Machine* (SVM) sebagai metode pembanding dengan menggunakan fungsi kernel RBF.

Pada penelitian ini, akan dilakukan empat skenario klasifikasi, yaitu *Random Forest* yang menggunakan 100, 150, dan 200 *tree* serta klasifikasi menggunakan SVM.

2.8.4 Pengujian dan Penarikan Kesimpulan

Pada tahap ini, *Confusion Matrix* akan diimplementasikan pada empat klasifikasi yang telah dilakukan untuk melakukan pengujian terhadap model klasifikasi yang telah dibangun. Dari *Confusion Matrix* ini nantinya akan didapatkan nilai *accuracy, precison_micro* dan *recall_micro*. Nilai-nilai ini nantinya akan digunakan untuk melihat performa klasifikasi terbaik dari empat klasifikasi yang dilakukan dengan membuat empat tabel berdasarkan masing-masing klasifikasi.

Selanjutnya dapat ditarik kesimpulan dari penelitian yang telah dilakukan dan saran-saran dari penulis sebagai acuan dalam pengembangan sistem berikutnya.

BAB IV
HASIL DAN PEMBAHASAN

3.1 Pengumpulan Data

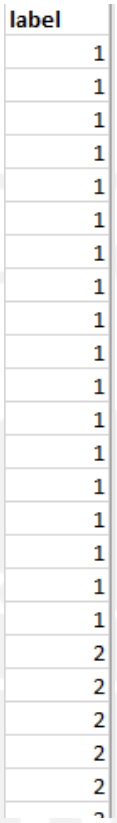
Data yang digunakan untuk penelitian ini adalah data spektrum NIRS 186 mangga (Munawar, 2019) yang dapat diunduh dari https://data.mendeley.com/datasets/b9d6s7hr33/1. Gambar 4.1 menampilkan data spektrum NIRS mangga yang telah diunduh.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	No	Mango Cultivars	Vit C (mg/100g)	TA (mg/100g)	SSC (oBrix)	999,9	1000,3	1000,7	1001,1	1001,4	1001,8	1002,2	1002,6	1003	1003,4	1003,8	1004,2	1004,5
2	1	Cengkir	62,51267	599,6819	8,695	0,517039	0,516867	0,516921	0,516366	0,516206	0,51566	0,515261	0,51472	0,514282	0,513759	0,513524	0,512883	0,51204
3	2	Cengkir	58,55433	488,5819	8,825	0,465913	0,465593	0,465691	0,465959	0,465898	0,464764	0,464363	0,464231	0,464284	0,464047	0,463307	0,462537	0,46209
4	3	Cengkir	62,096	549,249	9,225	0,550232	0,549902	0,549755	0,549763	0,54998	0,549185	0,548379	0,547819	0,547142	0,546779	0,546289	0,545881	0,545395
5	4	Cengkir	62,30433	464,366	8,965	0,462931	0,462902	0,4627	0,462785	0,462643	0,461748	0,461245	0,461259	0,461403	0,46089	0,460205	0,459927	0,459609
6	5	Cengkir	46,241	346,849	9,435	0,449824	0,449643	0,44987	0,450019	0,449672	0,44884	0,4487	0,448091	0,447702	0,447621	0,447355	0,446986	0,446363
7	6	Cengkir	35,941	437,093	10,135	0,449824	0,449643	0,44987	0,450019	0,449672	0,44884	0,4487	0,448091	0,447702	0,447621	0,447355	0,446986	0,446363
8	7	Cengkir	51,741	167,9413	13,705	0,449824	0,449643	0,44987	0,450019	0,449672	0,44884	0,4487	0,448091	0,447702	0,447621	0,447355	0,446986	0,446363
9	8	Cengkir	53,451	156,7747	20,815	0,499188	0,499168	0,499243	0,499276	0,498889	0,498166	0,498043	0,497774	0,497297	0,496689	0,496202	0,496262	0,495843
10	9	Cengkir	60,381	376,3858	11,295	0,499188	0,499168	0,499243	0,499276	0,498889	0,498166	0,498043	0,497774	0,497297	0,496689	0,496202	0,496262	0,495843
11	10	Cengkir	44,351	342,8858	14,425	0,46456	0,464569	0,46484	0,464797	0,464321	0,463564	0,46354	0,463491	0,46304	0,462485	0,462137	0,461851	0,461476
12	11	Cengkir	49,731	309,3858	13,387	0,628127	0,627072	0,62647	0,625762	0,624661	0,623522	0,623366	0,622498	0,621493	0,620496	0,619615	0,618739	0,617381
13	12	Cengkir	57,311	319,2947	13,517	0,628127	0,627072	0,62647	0,625762	0,624661	0,623522	0,623366	0,622498	0,621493	0,620496	0,619615	0,618739	0,617381
14	13	Cengkir	53,531	399,208	13,917	0,49887	0,498158	0,498572	0,498509	0,497802	0,497247	0,49715	0,497005	0,496722	0,495875	0,494925	0,494425	0,494688
15	14	Cengkir	67,131	339,1635	13,657	0,463472	0,462753	0,46289	0,462739	0,462209	0,461645	0,461649	0,461625	0,461443	0,461146	0,460213	0,459463	0,459557
16	15	Cengkir	63,841	186,5524	14,127	0,525084	0,524445	0,524628	0,524391	0,523918	0,523329	0,523013	0,522749	0,522371	0,521691	0,520743	0,520401	0,520401
17	16	Cengkir	57,7986	191,1991	14,827	0,491996	0,491499	0,491438	0,490788	0,490058	0,489565	0,489667	0,489497	0,489025	0,48854	0,487886	0,486877	0,486544
18	17	Cengkir	71,772	399,208	18,397	0,471459	0,471074	0,470934	0,470379	0,47026	0,46988	0,469497	0,469435	0,469454	0,468998	0,468351	0,467619	0,467762
19	18	Cengkir	55,7637	346,3951	25,507	0,513505	0,513086	0,513458	0,513126	0,512536	0,51202	0,51142	0,511253	0,511205	0,510674	0,509978	0,509548	0,509499
20	19	Kweni	57,029	429,208	15,987	0,363834	0,363489	0,363418	0,36327	0,363237	0,36238	0,362127	0,36199	0,361873	0,361192	0,360576	0,36027	0,359791
21	20	Kweni	77,8647	667,092	19,117	0,363834	0,363489	0,363418	0,36327	0,363237	0,36238	0,362127	0,36199	0,361873	0,361192	0,360576	0,36027	0,359791
22	21	Kweni	51,9651	339,892	19,32	0,359705	0,359462	0,359603	0,359467	0,359104	0,358377	0,357889	0,357575	0,357461	0,356991	0,356505	0,356167	0,355681
23	22	Kweni	54,1613	235,892	18,78	0,359705	0,359462	0,359603	0,359467	0,359104	0,358377	0,357889	0,357575	0,357461	0,356991	0,356505	0,356167	0,355681
24	23	Kweni	70,2785	697,892	10,31	0,398669	0,398369	0,398148	0,397953	0,398103	0,397841	0,397318	0,396783	0,396442	0,396151	0,395649	0,3951	0,394718
25	24	Kweni	67,1712	365,892	13	0,362831	0,362573	0,362676	0,36273	0,362519	0,361553	0,361198	0,360818	0,360541	0,360128	0,359768	0,359211	0,358673
26	25	Kweni	68,7086	231,692	8,9	0,362831	0,362573	0,362676	0,36273	0,362519	0,361553	0,361198	0,360818	0,360541	0,360128	0,359768	0,359211	0,358673
27	26	Kweni	71,3438	690,092	14,21	0,360536	0,360478	0,360434	0,360178	0,360036	0,359469	0,359102	0,358762	0,358303	0,357621	0,357175	0,356982	0,356351
28	27	Kweni	73,1505	388,532	7,91	0,360536	0,360478	0,360434	0,360178	0,360036	0,359469	0,359102	0,358762	0,358303	0,357621	0,357175	0,356982	0,356351

Gambar 4.1 Data spektrum NIRS mangga

Pada penelitian ini, akan diambil kolom nilai spektrum NIRS mangga beserta labelnya. Dapat dilihat bahwa kolom label pada *dataset* ini bertipe *String* dan harus diubah menjadi *integer* agar dapat dimasukkan ke dalam beberapa fungsi yang akan digunakan nantinya. Untuk itu akan diubah nilai kolom label secara manual dengan label satu (“Cengkir”), label dua

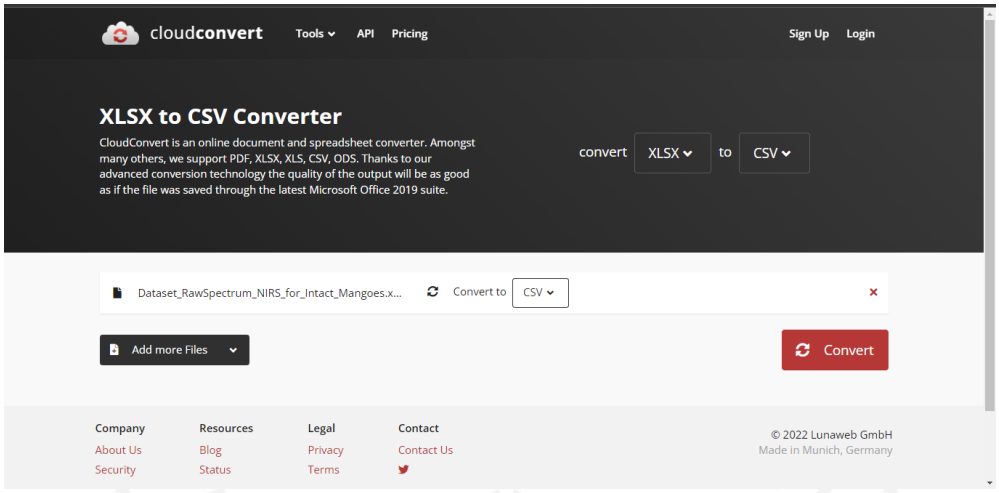
("Kweni"), label tiga ("Kent"), dan label empat ("Palmer"). Gambar 4.2 memperlihatkan kolom label yang telah diganti nilainya.



label
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
2
2
2
2
2

Gambar 4.2 Kolom label yang telah diganti nilainya bertipe *integer*

Data spektrum NIRS mangga yang diunduh ini berekstensi ".xls", oleh karena itu perlu dilakukan konversi ekstensinya menjadi ".csv" agar dapat digunakan. Konversi ini akan dilakukan menggunakan aplikasi konversi *online* cloudconvert (https://cloudconvert.com/xlsx-to-csv). Gambar 4.3 menampilkan tampilan situs aplikasi cloudconvert.



Gambar 4.3 Tampilan situs aplikasi cloudconvert

Setelah proses konversi selesai, data akan diunduh ke dalam penyimpanan lokal perangkat penulis dengan nama "nirsMangga.csv" dan akan digunakan untuk proses selanjutnya. Gambar 4.4 menampilkan data yang telah dikonversi menjadi ".csv"

Gambar 4.4 Data yang telah dikonversi menjadi ".csv"

3.2 *Exploratory Data Analysis* (EDA) dan *Preprocessing*

Untuk melihat karakteristik dari dataset NIR yang digunakan, dilakukan analisis data menggunakan *Exploratory Data Analysis* (EDA). Pertama *import* dataset serta *libraries* yang digunakan dengan menjalankan baris kode pada Gambar 4.5 dan Gambar 4.6.

```
#import libraries
import numpy as np
import pandas as pd
```

```
import seaborn as sns
import matplotlib.pyplot as plt
```

Gambar 4.5 Kode program *import libraries* untuk EDA dan *Preprocessing*

Akan digunakan *library* "numpy" untuk membantu proses yang berhubungan dengan operasi matematis, *library* "pandas" untuk membantu proses yang berhubungan dengan manipulasi data, *library* "seaborn" untuk membantu proses yang berhubungan dengan korelasi data, dan library "matplotlib.pyplot" untuk membantu proses pembuatan grafik.

```
#import dataset
df = pd.read_csv("nirsMangga.csv")
```

Gambar 4.6 Kode program untuk *import dataset*.

Pada Gambar 4.6, dataset dari *file* "nirsMangga.csv" akan di-*import* ke dalam *data frame* "df". Untuk melihat lima data pertama pada *data frame* "df" dapat digunakan fungsi head() seperti pada baris kode pada Gambar 4.7 dan Gambar 4.8 memperlihatkan lima data pertama dari *data frame* "df".

```
#lihat lima data pertama df
df.head()
```

Gambar 4.7 Kode program untuk melihat lima baris pertama "df"

	No	Mango Cultivars	Vit C (mg/100g)	TA (mg/100g)	SSC (oBrix)	999.9	1000.3	1000.7	1001.1	1001.4	1001.8	1002.2	1002.6	1003	1003.4	1003.8
0	1	1	62.51267	599.6819	8.695	0.517039	0.516867	0.516921	0.516366	0.516205	0.515660	0.515261	0.514720	0.514282	0.513759	0.513524
1	2	1	58.55433	488.5819	8.825	0.465913	0.465593	0.465691	0.465959	0.465898	0.464764	0.464363	0.464231	0.464284	0.464047	0.463306
2	3	1	62.09600	549.2490	9.225	0.550232	0.549902	0.549755	0.549763	0.549980	0.549185	0.548379	0.547819	0.547142	0.546779	0.546289
3	4	1	62.30433	464.3660	8.965	0.462931	0.462902	0.462700	0.462785	0.462643	0.461748	0.461245	0.461259	0.461403	0.460890	0.460205
4	5	1	46.24100	346.8490	9.435	0.449824	0.449643	0.449870	0.450019	0.449672	0.448840	0.448700	0.448091	0.447702	0.447621	0.447355

Gambar 4.8 Lima data pertama "df"

Dapat dilihat pada Gambar 4.8, terdapat fitur "No", "Mango Cultivars", "Vit C (mg/100g)", "TA (mg/100g)", dan "SSC (oBrix)" yang tidak dibutuhkan, oleh karena itu fitur-fitur ini akan dibuang/di-*drop* dengan menjalankan baris kode pada Gambar 4.5

```
#drop fitur-fitur yang tidak digunakan
df_fix = df.drop(labels=['No', 'Mango Cultivars', 'Vit C (mg/100g)', 'TA
(mg/100g)', 'SSC (oBrix)'], axis=1)
```

Gambar 4.9 Kode program untuk membuang fitur-fitur yang tidak digunakan

Pada Gambar 4.9, dijalankan baris tersebut untuk membuang fitur-fitur yang tidak digunakan. Fitur-fitur yang digunakan akan disimpan pada *data frame* "df_fix". Selanjutnya, untuk melihat lima data pertama setelah fitur-fitur yang tidak digunakan tersebut, dapat digunakan fungsi head() lagi seperti pada Gambar 4.10

```
#lihat lima data pertama df
df_fix.head()
```

Gambar 4.10 Kode program untuk melihat lima baris pertama "df_fix"

Selanjutnya, Gambar 4.11 menunjukkan lima data pertama *data frame* "df_fix" yang telah dipanggil pada Gambar 4.10.

	999.9	1000.3	1000.7	1001.1	1001.4	1001.8	1002.2	1002.6	1003	1003.4	1003.8	1004.2	1004.5	1004.9	1005.3
0	0.517039	0.516867	0.516921	0.516366	0.516205	0.515660	0.515261	0.514720	0.514282	0.513759	0.513524	0.512883	0.512040	0.511836	0.511466
1	0.465913	0.465593	0.465691	0.465959	0.465898	0.464764	0.464363	0.464231	0.464284	0.464047	0.463306	0.462537	0.462090	0.462184	0.462087
2	0.550232	0.549902	0.549755	0.549763	0.549980	0.549185	0.548379	0.547819	0.547142	0.546779	0.546289	0.545881	0.545395	0.545310	0.545080
3	0.462931	0.462902	0.462700	0.462785	0.462643	0.461748	0.461245	0.461259	0.461403	0.460890	0.460205	0.459927	0.459609	0.459211	0.458932
4	0.449824	0.449643	0.449870	0.450019	0.449672	0.448840	0.448700	0.448091	0.447702	0.447621	0.447355	0.446985	0.446363	0.446003	0.445648

2466.9	2469.3	2471.6	2474	2476.3	2478.7	2481.1	2483.5	2485.8	2488.2	2490.6	2493	2495.4	2497.8	2500.2	label
1.496648	1.498207	1.499908	1.501218	1.502628	1.503947	1.505065	1.505929	1.506978	1.507936	1.508755	1.509356	1.510223	1.510651	1.511547	1
1.372552	1.375687	1.378162	1.380810	1.384078	1.386533	1.388579	1.390588	1.393135	1.395065	1.396263	1.397289	1.398694	1.400154	1.401989	1
1.450506	1.451636	1.453299	1.455403	1.457336	1.458508	1.459362	1.460702	1.462801	1.463697	1.463799	1.464256	1.464685	1.465478	1.466563	1
1.377544	1.380604	1.383608	1.386249	1.388994	1.391381	1.393639	1.395964	1.398350	1.400225	1.401508	1.402762	1.404272	1.405578	1.406921	1
1.385177	1.388107	1.391341	1.394078	1.397283	1.400033	1.402181	1.404301	1.406622	1.408270	1.410406	1.411573	1.412805	1.414365	1.416021	1

Gambar 4.11 Lima data *input* dan *ouput* pertama "df_fix"

Untuk melihat dimensi (banyak baris dan kolom) dari data yang dipakai, dapat dipanggil atribut "shape" dari *data frame* "df" seperti baris kode pada Gambar 4.12 dan hasilnya pada Gambar 4.13.

```
#cek dimensi dataset
df_fix.shape
```

Gambar 4.12 Kode program untuk mengecek dimensi data

```
(186, 1558)
```

Gambar 4.13 Keluaran dari kode program yang memperlihatkan dimensi data

Dapat dilihat pada Gambar 4.13, terdapat 186 baris dan 1558 kolom pada data yang digunakan. Selanjutnya dilakukan pengecekan pada tipe data *input* dan *output* dari dataset

dengan memanggil atribut “dtypes” dari *data frame* “df” seperti pada kode program Gambar 4.14.

```
#cek tipe data dataset
df_fix.dtypes
```

Gambar 4.14 Kode program untuk melihat tipe data pada dataset

```
2459.9    float64
2462.2    float64
2464.6    float64
2466.9    float64
2469.3    float64
2471.6    float64
2474      float64
2476.3    float64
2478.7    float64
2481.1    float64
2483.5    float64
2485.8    float64
2488.2    float64
2490.6    float64
2493      float64
2495.4    float64
2497.8    float64
2500.2    float64
label       int64
```

Gambar 4.15 Tipe data dataset

Pada Gambar 4.15, dapat dilihat bahwa semua data *input* bertipe data float64 dan *output* bertipe data int64. Untuk melihat penggunaan memori/*memory usage* dataset, digunakan fungsi info() seperti pada baris kode pada Gambar 4.16

```
df_fix.info()
```

Gambar 4.16 Kode program untuk melihat penggunaan memori dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 186 entries, 0 to 185
Columns: 1558 entries, 999.9 to label
dtypes: float64(1557), int64(1)
memory usage: 2.2 MB
```

Gambar 4.17 Keluaran dari kode program pada gambar 4.16

Pada Gambar 4.17 dapat dilihat penggunaan memori dataset adalah sebesar 2.2 MB dan terdapat detail-detail *dataset* lainnya seperti banyak data yang bertipe float64 (sebanyak 1557 fitur) dan tipe int64 (sebanyak satu fitur). Dapat dilihat juga *range index* yaitu dari *index* 0

sampai 185. Selanjutnya dijalankan fungsi describe() untuk menghitung dan menampilkan ringkasan statistik dari dataset seperti Gambar 4.18.

```
df_fix.describe()
```

Gambar 4.18 Kode program untuk menampilkan ringkasan statistik dari *dataset*

	999.9	1000.3	1000.7	1001.1	1001.4	1001.8	1002.2	1002.6	1003	1003.4	1003.8	1004.2
count	186.000000	186.000000	186.000000	186.000000	186.000000	186.000000	186.000000	186.000000	186.000000	186.000000	186.000000	186.000000
mean	0.470990	0.470317	0.470073	0.469959	0.469614	0.468830	0.468439	0.468219	0.467915	0.467364	0.467033	0.466492
std	0.067623	0.067513	0.067419	0.067371	0.067330	0.067233	0.067149	0.067119	0.067103	0.067036	0.067028	0.066950
min	0.359705	0.359462	0.359603	0.359467	0.359104	0.358377	0.357889	0.357575	0.357461	0.356991	0.356505	0.356167
25%	0.438271	0.437551	0.437256	0.437129	0.436796	0.436155	0.435898	0.435692	0.435161	0.434624	0.434557	0.433944
50%	0.468262	0.467445	0.466986	0.467068	0.466896	0.466074	0.465598	0.465148	0.464806	0.464326	0.463948	0.463893
75%	0.499109	0.498421	0.498491	0.498465	0.497888	0.497192	0.496888	0.496721	0.496463	0.495692	0.495013	0.494785
max	0.759821	0.758425	0.757702	0.757295	0.756754	0.755493	0.754821	0.754700	0.754536	0.753567	0.752842	0.751998

2474	2476.3	2478.7	2481.1	2483.5	2485.8	2488.2	2490.6	2493	2495.4	2497.8	2500.2	label
186.000000	186.000000	186.000000	186.000000	186.000000	186.000000	186.000000	186.000000	186.000000	186.000000	186.000000	186.000000	186.000000
1.525908	1.527589	1.529063	1.530619	1.531952	1.533440	1.534393	1.535336	1.536066	1.536951	1.537809	1.538768	2.940860
0.127961	0.127685	0.127583	0.127405	0.127189	0.127041	0.126883	0.126851	0.126799	0.126729	0.126733	0.126758	0.913412
1.218640	1.220893	1.223759	1.226502	1.229511	1.232319	1.234361	1.235337	1.237196	1.238874	1.239891	1.241483	1.000000
1.459382	1.460661	1.461599	1.463149	1.465169	1.467178	1.468450	1.469866	1.470643	1.471730	1.473176	1.474589	2.250000
1.521762	1.523283	1.524657	1.525973	1.527454	1.528209	1.529202	1.530279	1.530254	1.531152	1.531543	1.532280	3.000000
1.568286	1.570308	1.571987	1.574000	1.575753	1.577730	1.579015	1.580222	1.580739	1.581087	1.581512	1.582513	4.000000
2.454628	2.454920	2.455570	2.457048	2.458371	2.458920	2.458715	2.459428	2.459773	2.461786	2.463613	2.465585	4.000000

Gambar 4.19 Ringkasan statistik dari *dataset*

Dengan menjalankan fungsi describe(), dapat dilihat pada Gambar 4.19 ringkasan statistik dari *dataset* seperti jumlah baris (*count*), nilai rata-rata (*mean*), standar deviasi (*std*), nilai paling rendah (*min*), nilai paling tinggi (*max*), kuartil bawah (25%), kuartil tengah (50%), dan kuartil atas (75%).

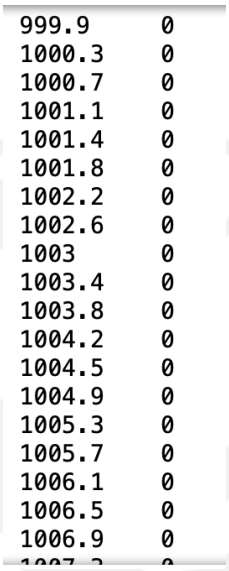
Dapat dilihat bahwa nilai spektrum NIRS mangga yang kolomnya berdekatan (misal kolom “999.9” dengan “1000.3”, “1000.7”, dan “1001.1”) mempunyai nilai yang hampir sama, hal ini dapat dilihat dari nilai *mean* kolom-kolom tersebut. Selain itu, dapat juga dilihat bahwa nilai standar deviasi, nilai paling rendah, nilai paling maksimum, dan semua nilai kuartil nilai spektrum NIRS mangga yang kolomnya berdekatan hampir sama.

Untuk melihat keberadaan nilai kosong atau *missing values* pada dataset, dijalankan fungsi isna() seperti Gambar 4.20 dan hasil dari menjalankan fungsi isna() terdapat pada Gambar 4.21.

```
#cek nilai NA pada setiap features Dataset
```

```
df_fix.isna().sum()
```

Gambar 4.20 Kode program untuk melihat keberadaan NA atau *missing values*.



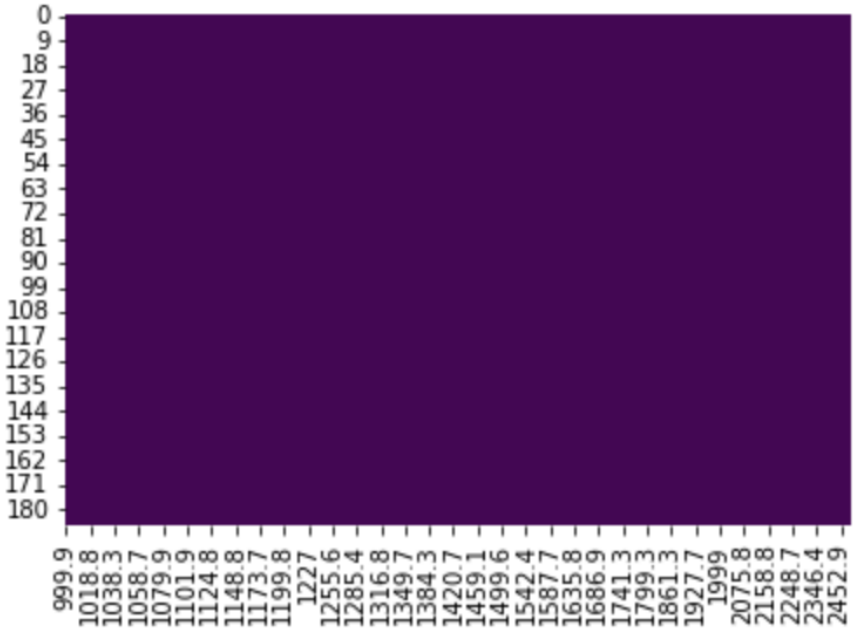
999.9	0
1000.3	0
1000.7	0
1001.1	0
1001.4	0
1001.8	0
1002.2	0
1002.6	0
1003	0
1003.4	0
1003.8	0
1004.2	0
1004.5	0
1004.9	0
1005.3	0
1005.7	0
1006.1	0
1006.5	0
1006.9	0

Gambar 4.21 Keluaran dari fungi isna()

Berdasarkan Gambar 4.21, dapat dilihat bahwa tidak terdapat nilai kosong atau *missing values* sama sekali pada *dataset*. Untuk memvisualisasikan keberadaan *missing values* pada dataset, digunakan fungsi heatmap() seperti gambar 4.22 dan hasilnya terlihat pada Gambar 4.23.

```
#grafik missingness map
sns.heatmap(df_fix.isnull(),cbar=False,cmap='viridis')
```

Gambar 4.22 Kode program untuk memvisualisasikan *missingness map*



Gambar 4.23 Missingness map

Untuk melihat nilai korelasi masing-masing fitur dapat digunakan fungsi corr() seperti pada baris kode pada Gambar 4.24 dengan hasil pada Gambar 4.25.

```
#nilai korelasi fitur
corrmat = df_fix.corr()
```

Gambar 4.24 Kode program untuk melihat nilai korelasi setiap fitur

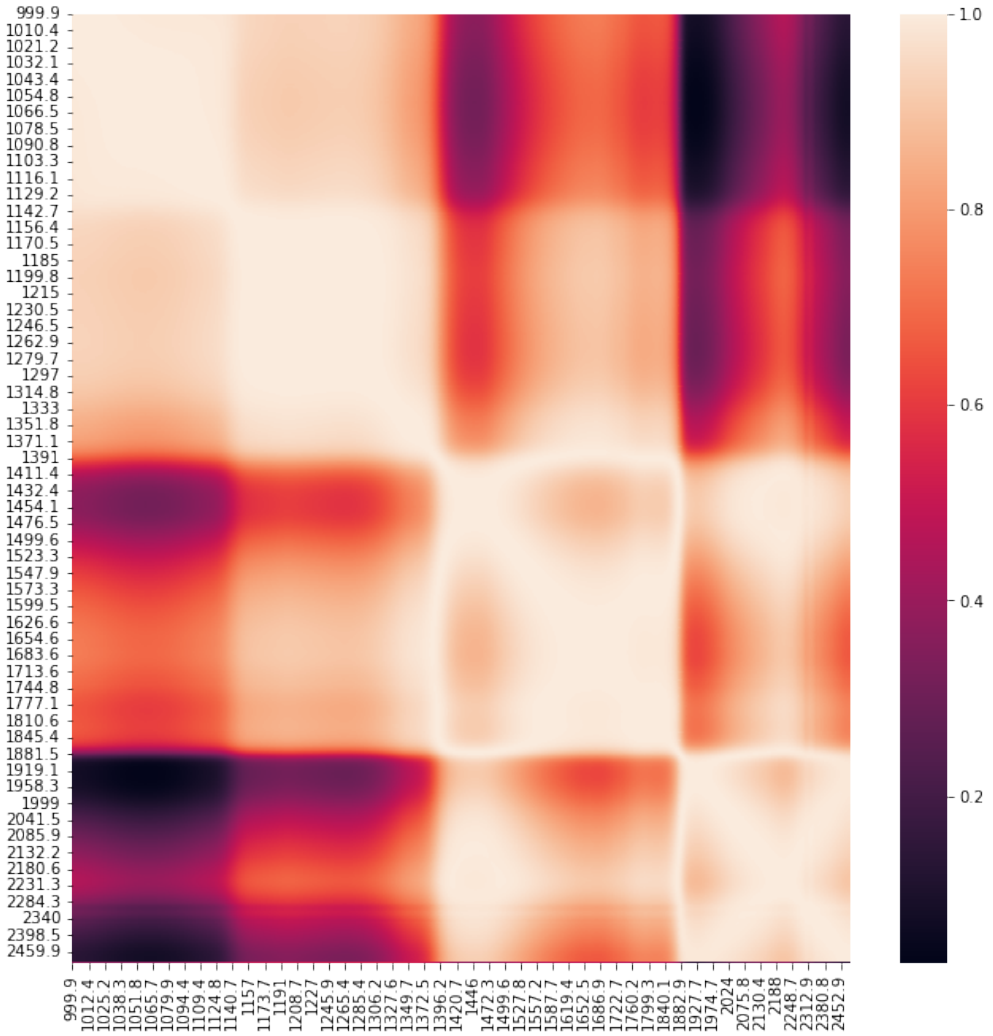
	999.9	1000.3	1000.7	1001.1	1001.4	1001.8	1002.2	1002.6	1003	1003.4	...	2481.1	2483.5	2485.8	2488.2
999.9	1.000000	0.999994	0.999984	0.999980	0.999968	0.999969	0.999971	0.999967	0.999954	0.999949	...	0.139708	0.139429	0.139227	0.139359
1000.3	0.999994	1.000000	0.999994	0.999988	0.999977	0.999979	0.999980	0.999975	0.999962	0.999957	...	0.139868	0.139589	0.139390	0.139523
1000.7	0.999984	0.999994	1.000000	0.999995	0.999983	0.999984	0.999987	0.999982	0.999972	0.999966	...	0.139617	0.139340	0.139143	0.139277
1001.1	0.999980	0.999988	0.999995	1.000000	0.999993	0.999991	0.999990	0.999986	0.999980	0.999977	...	0.138652	0.138375	0.138179	0.138313
1001.4	0.999968	0.999977	0.999983	0.999993	1.000000	0.999996	0.999988	0.999983	0.999984	0.999982	...	0.138113	0.137836	0.137639	0.137772
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2493	0.138989	0.139157	0.138912	0.137947	0.137408	0.137199	0.137255	0.137243	0.136460	0.135965	...	0.999909	0.999941	0.999965	0.999982
2495.4	0.139022	0.139191	0.138947	0.137984	0.137444	0.137235	0.137293	0.137280	0.136498	0.136003	...	0.999883	0.999922	0.999951	0.999971
2497.8	0.139163	0.139334	0.139091	0.138129	0.137592	0.137382	0.137440	0.137426	0.136645	0.136152	...	0.999853	0.999898	0.999932	0.999957
2500.2	0.138761	0.138935	0.138694	0.137732	0.137196	0.136986	0.137044	0.137026	0.136245	0.135752	...	0.999808	0.999860	0.999903	0.999934
label	0.364579	0.363127	0.361797	0.362486	0.363554	0.363185	0.362040	0.362337	0.363233	0.363284	...	0.155955	0.155235	0.154509	0.154150

Gambar 4.25 Nilai korelasi setiap fitur

Gambar 4.25 menunjukkan nilai korelasi setiap fitur pada *dataset*. Untuk memvisualisasikan nilai korelasi setiap fitur pada *dataset* dapat digunakan fungsi heatmap() seperti pada baris kode pada Gambar 4.26 dengan hasil pada Gambar 4.27.

```
#grafik visualisasi nilai korelasi fitur
fig, ax = plt.subplots()
fig.set_size_inches(11,11)
sns.heatmap(corrmat)
```

Gambar 4.26 Kode program visualisasi nilai korelasi fitur



Gambar 4.27 Grafik visualisasi nilai korelasi fitur

Pada gambar 4.27, dapat dilihat korelasi antar setiap fitur pada dataset. Warna yang lebih terang mengindikasikan bahwa dua variabel yang berkorelasi memiliki nilai korelasi yang

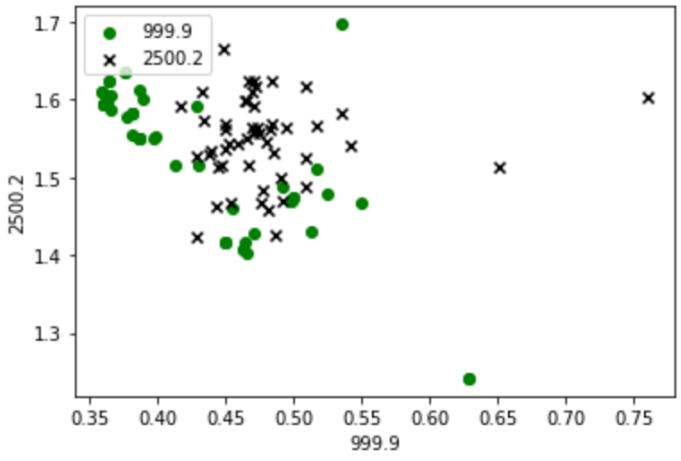
tinggi (semakin mendekati satu), sedangkan warna yang lebih rendah mengindikasikan bahwa dua variabel yang berkorelasi memiliki nilai korelasi yang rendah (semakin mendekati nol).

Untuk mengetahui apakah dataset yang digunakan *linear* atau *non-linear*, dapat digambar scatter plot 2 dimensi dari 2 variabel dataset. Apabila titik-titik data di *scatter plot* terbagi secara *linear* (*Linearly Seperable*) maka data tersebut adalah *linear.* Di sini akan dilakukan projeksi terhadap data fitur "2500.2" dan "999.9", akan diliat apakah data tersebut linier atau tidak dengan menjalankan program pada Gambar 4.28.

```
plt.scatter(df_fix.iloc[:50, 0], df_fix.iloc[:50, 1], color='green',
marker='o', label='999.9')
plt.scatter(df_fix.iloc[50:100, 0], df_fix.iloc[50:100, 1], color='black',
marker='x', label='2500.2')
plt.xlabel('999.9')
plt.ylabel('2500.2')
plt.legend(loc='upper left')
plt.show()
```

Gambar 4.28 Kode program untuk melihat plot

Fungsi scatter() pada Gambar 4.28 akan membuat scatter plot dua dimensi dari projeksi 50 data pertama dari kolom nol (fitur "999.9") dengan kolom satu (fitur "2500.2") sebagai projeksi data fitur "999.9", titik data projeksi ini akan ditandai dengan tanda "o", dan projeksi data ke-50 sampai ke-100 dari kedua kolom sebelumnya sebagai projeksi data fitur "2500.2", titik data projeksi ini akan ditandai dengan tanda "x". Setelah itu, scatter plot dua dimensi ini akan ditampilkan seperti pada Gambar 4.29 menggunakan fungsi show().



Gambar 4.29 *Output* dari *scatter plot*

Dari scatter plot pada Gambar 4.29, dapat dilihat bahwa data tidak dapat dibagi secara linier oleh karena itu dataset tersebut adalah dataset non-linier.

3.3 Seleksi Fitur

Seleksi fitur dilakukan menggunakan sembilan metode berbeda dengan menerapkan *trial and error* dalam menentukan jumlah fitur yang akan digunakan untuk mendapatkan model dengan performa klasifikasi tertinggi. Pada kategori seleksi fitur *filter* dan *embedded*, tahap *trial and error* ini akan menambahan fitur pada model klasifikasi mulai dari satu fitur sampai seratus fitur untuk melihat jumlah fitur optimal yang dapat menghasilkan performa terbaik. Untuk kategori *wrapper,* fitur yang akan digunakan pada model klasifikasi hanya kelipatan 20 dengan batas 100 fitur. Selanjutnya seluruh fitur hasil seleksi akan dilampirkan pada lampiran A.

3.3.1 ANOVA

Seleksi fitur ANOVA menggunakan nilai uji-F untuk mengevaluasi dan melakukan *scoring* terhadap fitur serta melihat keterkaitannya dengan label yang ada. Pertama akan di-*import* beberapa fungsi dari *library* "sklearn" yang digunakan untuk operasi yang berhubungan dengan proses-proses *Machine Learning* seperti baris kode Gambar 4.30 dan *library* "panda" seperti Gambar 4.5.

```
#import libraries
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import f_classif,SelectKBest
```

Gambar 4.30 Kode program *import libraries*

Dari *library* "sklearn" akan diambil beberapa fungsi di antaranya:

a. train_test_split(), fungsi yang membagi *dataset* menjadi *train set* dan *test set*.
b. f_classif(), fungsi untuk melakukan uji-F.
c. SelectKBest, fungsi yang akan memilih k fitur terbaik berdasarkan operasi seleksi fitur yang telah dilakukan.

Selanjutnya dataset di-*import* dan dilakukan operasi *drop* untuk membuang fitur-fitur yang tidak digunakan seperti pada baris kode pada Gambar 4.6 dan Gambar 4.9. Setelah itu dilakukan proses pembagian (*split*) terhadap *dataset* dengan besaran 70% *train set* dan 30% *test set* seperti baris kode pada Gambar 4.31.

```
# separate dataset into train and test
x_train, x_test, y_train, y_test = train_test_split(
    df.drop(labels=['No', 'Mango Cultivars', 'Vit C (mg/100g)', 'TA
(mg/100g)', 'SSC (oBrix)', 'label'], axis=1),
    df['label'],
    test_size=0.3,
    random_state=0)
```

Gambar 4.31 Kode program pembagian *dataset*

Terdapat empat parameter yang digunakan fungsi train_test_split() pada Gambar 4.31 yaitu data *input* (parameter pertama), data *output* (parameter kedua), test_size (parameter ketiga, besaran *test set*), dan random_state (parameter keempat, nilai *seed* yang menentukan *randomness* atau keacakan *train* dan *test set*). Fungsi train_test_split() akan mengembalikan empat nilai yaitu data *input* untuk pelatihan ("x_train"), data *input* untuk pengujian ("x_test"), data *output* untuk pelatihan ("y_train"), dan data *output* untuk pengujian ("y_test"). Berikut Gambar 4.32, Gambar 4.33, Gambar 4.34, dan Gambar 4.35 yang memaparkan lima data pertama "x_train", "x_test", "y_train", dan "y_test".

	999.9	1000.3	1000.7	1001.1	1001.4	1001.8	1002.2	1002.6	1003	1003.4	...	2478.7	2481.1	2483.5	2485.8	2488.2
16	0.471459	0.471074	0.470934	0.470379	0.470260	0.469880	0.469497	0.469435	0.469454	0.468998	...	1.413537	1.415740	1.417568	1.419698	1.421711
51	0.433239	0.432622	0.432626	0.432379	0.431620	0.430710	0.430836	0.430847	0.430188	0.429470	...	1.601232	1.602877	1.604524	1.605982	1.606778
183	0.545045	0.544204	0.543792	0.543596	0.543338	0.542534	0.541493	0.541139	0.541308	0.540831	...	1.524657	1.525973	1.527454	1.529518	1.530097
145	0.545846	0.544815	0.544524	0.544631	0.544169	0.543143	0.542535	0.542080	0.541842	0.541258	...	1.421962	1.422955	1.423717	1.424639	1.425080
40	0.381048	0.380483	0.380541	0.380151	0.379599	0.379188	0.379009	0.378722	0.378310	0.377719	...	1.571125	1.572674	1.574303	1.576075	1.577273

5 rows × 1557 columns

Gambar 4.32 Lima data pertama "x_train"

	999.9	1000.3	1000.7	1001.1	1001.4	1001.8	1002.2	1002.6	1003	1003.4	...	2478.7	2481.1	2483.5	2485.8	2488.2
106	0.515735	0.514788	0.514485	0.514438	0.514549	0.513389	0.512699	0.512385	0.512242	0.511751	...	1.480614	1.481897	1.483379	1.484372	1.485257
45	0.365708	0.365245	0.365290	0.364957	0.364485	0.363896	0.363671	0.363543	0.363154	0.362575	...	1.597865	1.599884	1.600029	1.601138	1.602895
158	0.447674	0.447100	0.446745	0.446732	0.446522	0.445682	0.445182	0.445093	0.445267	0.444695	...	1.369408	1.370934	1.372218	1.373669	1.374740
63	0.470157	0.469284	0.468909	0.468909	0.468549	0.467326	0.466983	0.466877	0.466525	0.466021	...	1.603899	1.605261	1.605737	1.606289	1.606965
135	0.454289	0.453789	0.453421	0.453163	0.452773	0.452016	0.451783	0.451579	0.451241	0.450911	...	1.461599	1.462719	1.463619	1.464288	1.465099

5 rows × 1557 columns

Gambar 4.33 Lima data pertama "x_test"

```
16     1
51     3
183    4
145    3
40     2
Name: label, dtype: int64
```

Gambar 4.34 Lima data pertama "y_train"

```
106    4
45     2
158    3
63     3
135    3
Name: label, dtype: int64
```

Gambar 4.35 Lima data pertama "y_test"

Selanjutnya metode seleksi fitur ANOVA didefinisikan dengan baris kode pada Gambar 4.36.

```
# Define ANOVA feature selection
fs = SelectKBest(score_func=f_classif,k=100)
```

Gambar 4.36 Kode program pendefinisian metode seleksi fitur ANOVA

Melalui fungsi SelectKBest() yang terlihat pada Gambar 4.36, didefinisikan sebuah metode seleksi fitur dengan metode ANOVA (melalui parameter "score_func" yang diberi nilai "f_classif", metode "f_classif" menggunakan nilai uji-F untuk melakukan seleksi fitur/ANOVA) dan banyaknya fitur yang diambil adalah sebanyak 100 dengan nilai uji-F tertinggi (melalui parameter k=100). Selanjutnya metode seleksi fitur tersebut diaplikasikan kepada data x_train dan y_train dengan baris kode pada Gambar 4.37.

```
# Apply feature selection
fs.fit(x_train,y_train)
```

Gambar 4.37 Kode program untuk melakukan seleksi fitur ANOVA

.

Fungsi fit() pada Gambar 4.37 akan melatih model seleksi fitur yang mengambil 10 fitur terbaik dari x_train dan y_train. Selanjutnya, akan dilihat fitur apa saja yang dipilih dengan metode ANOVA dengan membuat *data frame* yang berisi nama fitur, *F-score*, dan *p-value* seperti Gambar 4.38.

```
Create features score, features p_value, and features name
features_score = pd.DataFrame(fs.scores_)
features_pvalue = pd.DataFrame(np.round(fs.pvalues_,4))
features = pd.DataFrame(x_train.columns)
feature_score = pd.concat([features,features_score,features_pvalue],axis=1)

#reset index, supaya tidak ada nilai nan di dataframe yg dibikin
features_score.reset_index(drop=True, inplace=True)
features_pvalue.reset_index(drop=True, inplace=True)
features.reset_index(drop=True, inplace=True)

# Assign the column name
feature_score.columns = ["Input_Features","Score","P_Value"]
```

```
# Print features score
print(feature_score.nlargest(100,columns="Score"))
```

Gambar 4.38 Kode program untuk melihat 100 fitur terbaik

Pada kode program Gambar 4.38 akan dibuat *data frame* "feature_score" yang berisi nilai *F-score* setiap fitur (yang didapat dari nilai uji-F dari seleksi fitur ANOVA), *data frame* "p-value" yang berisi nilai *p-value* setiap fitur (yang didapat dari nilai pvalues dari seleksi fitur ANOVA), dan *data frame* "features" yang berisi nama seluruh fitur yang ada pada *train set* (yang didapat dari nama kolom data *input training*). *Data frame* "feature_score" akan digabungkan (*concatenate*) dengan *data frame* "p-value" dan "features". Kemudian agar tidak terdapat nilai NaN (*not a number*) pada *data frame* "features_score" akan digunakan fungsi reset_index(). Setelah itu akan di-*assign* nama kolom dari dua kolom (nama fitur dan nilai *mutual information*-nya) yang ada pada *data frames* tersebut. Terakhir akan ditampilkan isi dari lima fitur pertama dan terakhir dari *data frame* "feature_score" seperti pada Gambar 4.39.

```
     Input_Features      Score  P_Value
220          1092.6  46.849895      0.0
221          1093.1  46.840097      0.0
229          1096.8  46.819700      0.0
214          1089.8  46.809279      0.0
226          1095.4  46.798650      0.0
..              ...        ...      ...
271          1116.6  46.139463      0.0
175          1072.3  46.098527      0.0
274            1118  46.079169      0.0
174          1071.8  46.063684      0.0
169          1069.6  46.054953      0.0

[100 rows x 3 columns]
```

Gambar 4.39 Fitur terbaik berdasarkan seleksi fitur ANOVA

Data frame "feature_score" yang ditampilkan pada Gambar 4.39 mempunyai empat kolom yaitu kolom indeks urutan fitur pada *dataset* (misalnya fitur "1092.6" merupakan fitur ke-220 pada *dataset*), kolom nama fitur ("Input_Features"), kolom *F-score* ("Score"), dan kolom p-value ("P-Value").

3.3.2 *Mutual Information*

Seleksi fitur *Mutual Information* (MI) menggunakan nilai *mutual information* untuk mengukur ketergantungan antara dua variabel. Pertama, akan di-*import libraries* "pandas" dan fungsi mutual_info_classif() dari *library* "sklearn" seperti baris kode pada Gambar 4.5 dan 4.40.

```
#import library
from sklearnfrom sklearn.feature_selection import mutual_info_classif
```

Gambar 4.40 Kode program menentukan *mutual information*

Fungsi mutual_info_classif() akan digunakan untuk melakukan proses seleksi fitur *Mutual Information*. Setelah itu, *import dataset* (Gambar 4.6 dan Gambar 4.9) dan bagi dengan besaran 70% *train set* dan 30% *test set* dengan menjalankan baris kode seperti pada gambar 4.31. Setelah itu, tentukan *mutual information* dengan menjalankan baris berikut pada gambar 4.40.

```
# determine the mutual information
mutual_info = mutual_info_classif(x_train, y_train)
```

Gambar 4.41 Kode program menentukan *mutual information*

Untuk melihat nilai *mutual information* yang didapatkan setelah menjalankan baris kode pada Gambar 4.41 akan dibuat *data frame* "features_mi" yang berisikan fitur beserta nilai *mutual information*-nya dengan menjalankan baris kode pada Gambar 4.42.

```
# Create features_mi, and features name
features_mi = pd.DataFrame(mutual_info)
features = pd.DataFrame(x_train.columns)
features_mi = pd.concat([features, features_mi],axis=1)

#reset index, supaya tidak ada nilai nan di dataframe yg dibikin
features_mi.reset_index(drop=True, inplace=True)
features.reset_index(drop=True, inplace=True)

# Assign the column name
features_mi.columns = ["Input_Features", "Mutual Information Value"]

# Print features score
print(features_mi.nlargest(100,columns="Mutual  Information  Value"))Gambar
4.41 Kode program melihat nilai mutual information
```

Gambar 4.42 Kode program untuk membuat *data frame* "features_mi"

Pada kode program Gambar 4.42 akan dibuat *data frame* “features_mi” yang berisi nilai *mutual information* setiap fitur (yang didapat dari nilai *mutual information* dari seleksi fitur *Mutual Information*) dan data frame “features” yang berisi nama seluruh fitur yang ada pada *dataset* (yang didapat dari *data frame* “features”). *Data frame* “features_mi” akan digabungkan (*concatenate*) dengan *data frame* “features”. Kemudian agar tidak terdapat nilai NaN (*not a number*) pada *data frame* “fisher_ranking” akan digunakan fungsi reset_index(). Setelah itu akan di-*assign* nama kolom dari dua kolom (nama fitur dan nilai *mutual information*-nya) yang ada pada *data frames* tersebut. Selanjutnya akan ditampilkan isi dari 100 fitur pertama *data frame* “features_mi” yang menunjukkan 100 fitur pertama yang terpenting seperti pada Gambar 4.43.

```
     Input_Features  Mutual Information Value
318          1139.7                 0.709039
319          1140.2                 0.703579
322          1141.7                 0.702486
315          1138.2                 0.700575
316          1138.7                 0.699476
..              ...                      ...
697          1367.5                 0.633102
536          1260.4                 0.632364
0             999.9                 0.631941
569            1281                 0.631650
565          1278.5                 0.631549

[100 rows x 2 columns]
```

Gambar 4.43 Fitur terbaik berdasarkan seleksi fitur Mutual Information

3.3.3 ReliefF

ReliefF menghitung statistik proksi untuk setiap fitur yang dapat digunakan dengan memperkirakan 'kualitas' fitur atau relevansinya dengan konsep target berdasarkan atribut statistik yang disebut sebagai *feature weight*. Pertama, akan di-*import library* “pandas” seperti Gambar 4.5, *library* “sklearn” seperti Gambar 4.30 dan fungsi ReliefF() dari *libary* “ReliefF” seperti Gambar 4.44.

```
from ReliefF import ReliefF
```

Gambar 4.44 Kode program *import library* “ReliefF”

Selanjutnya, *import dataset* dan bagi dengan besaran 70% *train set* dan 30% *test set* dengan menjalankan baris kode seperti pada gambar 4.6 dan 4.31. Setelah itu, akan dilakukan proses seleksi fitur *ReliefF* dengan kode yang terdapat pada Gambar 4.45.

```
fs = ReliefF(n_neighbors=30, n_features_to_keep=100)
x_train_tf = fs.fit(x_train.to_numpy(), y_train.to_numpy())
```

Gambar 4.45 Kode program seleksi fitur ReliefF

Pada Gambar 4.45, dilakukan proses seleksi fitur untuk mendapatkan 100 fitur terbaik berdasarkan *score* setiap fitur yang didapatkan dari proses seleksi fitur ReliefF. Ditetapkan total tetangga (*neighbors*) adalah sebesar 30 tetangga. *Neighbors* adalah titik-titik data terdekat dari poin data yang sedang diukur *score* fitur-nya, semakin tinggi perbedaan nilai antar data fitur yang sedang diukur dengan titik-titik data terdekatnya dalam kelas yang berbeda, semakin tinggi pula nilai *score* fitur-nya, sedangkan semakin tinggi perbedaan nilai antar data fitur yang sedang diukur dengan titik-titik data terdekatnya dalam kelas yang sama, semakin rendah nilai *score* fitur-nya. Setelah itu, akan dilakukan proses seleksi fitur *ReliefF* dengan menggunakan fungsi fit() yang akan melatih data *training input* dan data *training output* dalam bentuk "numpy". Untuk melihat nilai *score* 100 fitur yang didapatkan setelah proses seleksi fitur ReliefF, akan dibuat *data frame* "features_scores" yang berisikan fitur beserta nilai *socre*-nya dengan menjalankan baris kode pada Gambar 4.46.

```
features_scores = pd.DataFrame(fs.feature_scores)
features = pd.DataFrame(x_train.columns)

features_scores = pd.concat([features, features_scores],axis=1)

#reset index, supaya tidak ada nilai nan di dataframe yg dibikin
features_scores.reset_index(drop=True, inplace=True)
features_scores.reset_index(drop=True, inplace=True)

# Assign the column name
features_scores.columns = ["Input_Features", "Features_Score"]

# Print features score
print(features_scores.nlargest(100,columns="Features_Score"))
```

Gambar 4.46 Kode program untuk membuat *data frame* "features_score"

Pada kode program Gambar 4.48 akan dibuat *data frame* "features_score" yang berisi nilai *scores* setiap fitur (yang didapat dari nilai *scores* dari seleksi fitur ReliefF) dan *data frame* "features" yang berisi nama seluruh fitur yang ada pada *dataset* (yang didapat dari *data frame* "features"). *Data frame* "features_scores" akan digabungkan (*concatenate*) dengan *data frame* "features". Kemudian agar tidak terdapat nilai NaN (*not a number*) pada *data frame* "fisher_ranking" akan digunakan fungsi reset_index(). Setelah itu akan di-*assign* nama kolom dari dua kolom (nama fitur dan nilai *scores*-nya) yang ada pada *data frames* tersebut.

Selanjutnya akan ditampilkan isi dari 100 fitur pertama *data frame* “features_scores” yang menunjukkan 100 fitur yang terpenting seperti pada Gambar 4.47.

	Input_Features	Features_Score
1153	1800.5	-670.0
1183	1838.8	-670.0
1258	1942.1	-670.0
1548	2481.1	-670.0
795	1442	-674.0
917	1547	-674.0
1143	1788.1	-674.0
1480	2329.5	-674.0
1225	1895.3	-676.0
0	999.9	-678.0
1	1000.3	-678.0
2	1000.7	-678.0
3	1001.1	-678.0
4	1001.4	-678.0
5	1001.8	-678.0
6	1002.2	-678.0
7	1002.6	-678.0
8	1003	-678.0
9	1003.4	-678.0
10	1003.8	-678.0
11	1004.2	-678.0
12	1004.5	-678.0
13	1004.9	-678.0
14	1005.3	-678.0
15	1005.7	-678.0
16	1006.1	-678.0
17	1006.5	-678.0
18	1006.9	-678.0
19	1007.3	-678.0
20	1007.7	-678.0

Gambar 4.47 Fitur terbaik berdasarkan seleksi fitur *ReliefF*.

3.3.4 Fisher Score

Seleksi fitur yang dilakukan *Fisher Score* berbasis pada *ranking* dari *ratio* masing-masing fitur. Pertama, akan di-*import library* “pandas”, dan “sklearn” seperti kode program Gambar 4.5 dan Gambar 4.30 serta *library* “skfeature” seperti kode program pada Gambar 4.48.

```
#import libraries
from skfeature.function.similarity_based import fisher_score
```

Gambar 4.48 Kode program *import* fungsi fisher_score() *library* “skfeature”

Selanjutnya, import dataset dan bagi dengan besaran 70% *train set* dan 30% *test set* dengan menjalankan baris kode seperti pada Gambar 4.6 dan 4.31. Setelah itu, cari *ranking* setiap fitur berdasarkan *fisher score*-nya seperti pada Gambar 4.48 dan hasilnya yang ditampilkan pada Gambar 4.49.

```
#cari ranking setiap fitur berdasarkan fisher score
score = fisher_score.fisher_score(x_train.to_numpy(), y_train.to_numpy())
```

Gambar 4.49 Kode program untuk mencari ranking setiap fitur berdasarkan fisher *score*

```
array([1336, 1335, 1342, 1330, 1338, 1327, 1337, 1349, 1343, 1339, 1329,
       1332, 1326, 1333, 1348, 1313, 1331, 1364, 1334, 1354, 1323, 1363,
       1314, 1352, 1328, 1315, 1341, 1351, 1362, 1318, 1365, 1355, 1319,
       1359, 1340, 1312, 1344, 1353, 1321, 1310, 1325, 1350, 1320, 1322,
       1347, 1324, 1316, 1366, 1369, 1311, 1361, 1360, 1305, 1317, 1376,
       1375, 1367, 1358, 1368, 1306, 1345, 1356, 1346, 1309, 1357, 1370,
       1374, 1373, 1307, 1377, 1372, 1308, 1304, 1371, 1379, 1380, 1378,
       1303, 1301, 1302, 1381, 1387, 1382, 1300, 1386, 1299, 1389, 1298,
       1392, 1383, 1391, 1388, 1390, 1297, 1294, 1385, 1293, 1393, 1296,
       1292, 1290, 1384, 1295, 1289, 1394, 1291, 1395, 1286, 1287, 1404,
       1399, 1288, 1398, 1403, 1400, 1405, 1396, 1285, 1409, 1402, 1408,
       1406, 1397, 1282, 1401, 1283, 1407, 1284, 1416, 1410, 1411, 1281,
       1417, 1425, 1280, 1279, 1412, 1415, 1426, 1418, 1278, 1421, 1414,
       1277, 1419, 1420, 1413, 1276, 1275, 1422, 1429, 1274, 1424, 1430,
       1428, 1273, 1427, 1423, 1437, 1272, 1433, 1434, 1271, 1438, 1431,
       1269, 1439, 1435, 1270, 1268, 1440, 1436, 1432, 1267, 1441, 1266,
       1450, 1447, 1443, 1265, 1445, 1446, 1442, 1444, 1448, 1264, 1449,
       1456, 1451, 1263, 1457, 1452, 1455, 1453, 1454, 1262, 1458, 1460,
       1461, 1459, 1260, 1462, 1261, 1463, 1259, 1464, 1468, 1467, 1258,
```

Gambar 4.50 Indeks ranking setiap fitur berdasarkan fisher *score*

Pada kode program Gambar 4.49 dicari nilai fisher *score* setiap fitur menggunakan fungsi fisher_score(). Fungsi fisher_score() menerima dua parameter yaitu data *input* sebagai parameter pertama dan data *output* sebagai parameter kedua. Karena fungsi fisher_score() hanya menerima data yang berjenis "numpy" maka data *input* dan *output* yang akan dimasukkan ke dalam fungsi fisher_score(), harus diubah dulu jenisnya menjadi "numpy" dengan menggunakan fungsi to_numpy().

Gambar 4.50 menunjukkan indeks ranking yang dimulai dari fitur pertama (999.9 dengan ranking ke-1336) dan seterusnya. Ranking ini diurutkan dari *fisher score* masing-masing fitur. Selanjutnya akan dibuat *data frame* yang menyimpan nama fitur serta ranking fisher *score*-nya dengan urutan dari paling rendah ke atas. Gambar 4.51 berisi kode program untuk membuat *data frame* tersebut.

```
# Create dataframe for the feature with its rank
fisher_ranking = pd.DataFrame(fisher_score)
features = pd.DataFrame(x_train.columns)
fisher_ranking = pd.concat([features,fisher_ranking],axis=1)

# Reset index, supaya tidak ada nilai nan di dataframe yg dibikin
```

```
fisher_ranking.reset_index(drop=True, inplace=True)
features.reset_index(drop=True, inplace=True)

# Assign the column name
fisher_ranking.columns = ["Input_Features","Fisher_Ranking"]

# Print features score
print(fisher_ranking.nlargest(100,columns="Fisher_Ranking"))
```

Gambar 4.51 Kode program untuk mengurutkan ranking fitur

Kode pada Gambar 4.51 akan membuat *data frame* "fisher_ranking" yang berisi ranking fitur dan "features" yang berisi nama semua fitur yang ada di *train* set. *Data frame* "features" akan digabungkan (*concatenate*) dengan *data frame* "fisher_ranking". Kemudian agar tidak terdapat nilai NaN pada *data frame* "fisher_ranking" akan digunakan fungsi reset_index(). Setelah itu akan di-*assign* nama kolom dari dua kolom (nama fitur dan ranking fisher *score*-nya) yang ada pada *data frames* tersebut. Terakhir akan ditampilkan 100 nama fitur beserta ranking *fisher score*-nya yang tertinggi seperti pada Gambar 4.52.

```
     Input_Features  Fisher_Ranking
1556         2500.2               0
1555         2497.8               1
1554         2495.4               2
1553           2493               3
1552         2490.6               4
...             ...             ...
1463         2294.5              95
1442         2252.6              96
1431         2231.3              97
1428         2225.5              98
1426         2221.7              99

[100 rows x 2 columns]
```

Gambar 4.52 *Data frame* fisher *ranking* 30 fitur tertinggi

3.3.5 *Sequential Forward Selection* (SFS)

Sequential Forward Selection (SFS) dimulai dengan mengevaluasi semua subset fitur yang hanya terdiri dari satu atribut input. Pada penelitian ini, penulis menggunakan model *Random Forest Classifier* dan *Support Vector Machine* untuk melakukan seleksi fitur SFS. Pertama, akan di-*import library* "pandas" seperti Gambar 4.5, *library* "sklearn" seperti Gambar

4.30, serta fungsi SequentialFeatureSelector() dari *library* "mlxtend" menggunakan kode program pada Gambar 4.53.

```
#import libraries
from mlxtend.feature_selection import SequentialFeatureSelector as sfs
```

Gambar 4.53 Kode program *import libraries*

Selanjutnya, dataset di-*import* dan dibagi dengan besaran 70% *train set* dan 30% *test set* dengan menjalankan baris kode seperti pada Gambar 4.6 dan 4.31. Setelah itu, akan dilatih model *Random Forest Classifier* untuk melakukan seleksi fitur SFS ini seperti pada baris kode pada Gambar 4.54. Untuk melatih model *Support Vector Machine* untuk melakukan seleksi fitur SFS, ganti nilai variabel "mod" menjadi "svm.SVC()".

```
# calling the model
mod = RandomForestClassifier()
sfs1 = sfs(lreg, k_features=30, forward=True, verbose=2,
scoring='accuracy', n_jobs=-1)

# train the model
sfs1 = sfs1.fit(x_train, y_train)
```

Gambar 4.54 Kode program seleksi fitur dengan melatih model yang digunakan pada seleksi fitur SFS

Pada gambar 4.54 dilakukan proses SFS dengan melatih model *Random Forest Classifier* untuk menyeleksi 30 fitur yang mampu membangun model dengan akurasi sebagus mungkin (parameter k_features=30). Parameter "forward" bernilai "True" artinya proses seleksi fitur akan dilakukan dengan metode SFS. Parameter "verbose" diatur nilainya menjadi dua agar dapat diperlihatkan ringkasan (*summary*) dari tiap proses iterasi pelatihan. Parameter "scoring" menentukan cara pengujian model, karena pada penelitian ini akan digunakan nilai akurasi sebagai nilai pengujian model, maka nilai parameter "scoring" diatur menjadi "accuracy". Parameter "n_jobs" menentukan seberapa banyak CPU *cores* yang digunakan untuk menjalankan proses seleksi fitur SFS, ditentukan nilai -1 yang artinya akan digunakan semua CPU *cores* yang tersedia untuk menjalankan proses seleksi fitur SFS ini. Dengan digunakannya semua CPU *cores*, waktu yang dibutuhkan untuk melakukan proses seleksi fitur SFS akan berkurang.

Seleksi fitur *Sequential Forward Selection* pada penelitian ini akan dilakukan dengan melatih model *Random Forest Classifier* hingga 20, 40, 60, 80, dan 100 iterasi dalam mendapatkan 20, 40, 60, 80, dan 100 fitur terbaik untuk melatih model *Random Forest*

Classifier dengan akurasi tertinggi. Perlu diperhatikan bahwa fitur hasil dari masing-masing proses pelatihan 20, 40, 60, 80, dan 100 iterasi akan berbeda. Pada iterasi pertama, akan digunakan satu fitur terlebih dahulu dari ke-1557 fitur yang ada pada *dataset*. Selanjutnya, pada iterasi kedua akan dipilih dua fitur yang mampu menghasilkan model *Random Forest* dengan akurasi tertinggi dan begitu juga dengan iterasi-iterasi selanjutnya sampai iterasi ke-100. Gambar 4.55 hingga 4.57 akan menunjukkan 20 fitur yang dipilih berdasarkan seleksi fitur *Sequential Forward Selection* dengan menggunakan model *Random Forest Classifier*. Fitur-fitur dengan jumlah kelipatan 20 lainnya akan ditampilkan pada Lampiran A.

```
Fitur yang diambil adalah:
['1001.1', '1001.4', '1018.4', '1031.7', '1038.3', '1052.2', '1080.8', '1120', '1129.7', '1153.9', '1175.8', '1234.
6', '1549.8', '1580.9', '1616.4', '1629.6', '1712.5', '1909.2', '1958.3', '1977.7']
```

Gambar 4.55 20 Fitur yang diambil dengan SFS

3.3.6 Backward Elimination

Backward Elimination melakukan proses seleksi fitur dengan cara yang sebaliknya dibandingkan dengan *Sequential forward Selection* (SFS). Pada penelitian ini, penulis menggunakan model *Random Forest Classifier* untuk melakukan seleksi fitur *Backward Elimination*. Sebelum dilakukan seleksi fitur *Backward Elimination*, akan diambil terlebih dahulu 300 fitur terpenting dengan menggunakan seleksi fitur *Mutual Information*, hal ini dilakukan untuk meringankan proses komputasi yang dilakukan pada proses seleksi fitur *Backward Elimination*.

Pertama, akan di-*import library* "pandas" dan "numpy" seperti pada Gambar 4.5, *library* "sklearn" seperti pada Gambar 4.40 untuk melakukan proses seleksi fitur *Mutual Information*, serta fungsi SequentialFeatureSelector() dari *library* "mlxtend" untuk melakukan proses seleksi fitur *Backward Elimination* menggunakan kode program pada Gambar 4.53.

Selanjutnya, dataset di-*import* dan dibagi dengan besaran 70% *train set* dan 30% *test set* dengan menjalankan baris kode seperti pada gambar 4.6 dan 4.31. Setelah itu akan dilakukan proses seleksi fitur *Mutual Information* dengan memilih 300 fitur terpenting dengan menjalankan kode program Gambar 4.56 dan ke-300 fitur tersebut dapat dilihat pada Gambar 4.57.

```
# determine the mutual information
mutual_info = mutual_info_classif(x_train, y_train)
```

```
mutual_info

# Apply feature selection
fs.fit(x_train,y_train)

# Create features score, features p_value, and features name
features_mi = pd.DataFrame(mutual_info)
features = pd.DataFrame(x_train.columns)
features_mi = pd.concat([features, features_mi],axis=1)

#reset index, supaya tidak ada nilai nan di dataframe yg dibikin
features_mi.reset_index(drop=True, inplace=True)
features.reset_index(drop=True, inplace=True)

# Assign the column name
features_mi.columns = ["Input_Features", "Mutual Information Value"]

# Print features score
print(features_mi.nlargest(100,columns="Mutual Information Value"))
```

Gambar 4.56 Kode program *Mutual Information* untuk memilih 200 fitur terpenting

```
    Input_Features  Mutual Information Value
318          1139.7                 0.709039
319          1140.2                 0.703579
322          1141.7                 0.702486
315          1138.2                 0.700575
316          1138.7                 0.699476
..              ...                      ...
697          1367.5                 0.633102
536          1260.4                 0.632364
0             999.9                 0.631941
569            1281                 0.631650
565          1278.5                 0.631549

[100 rows x 2 columns]
```

Gambar 4.57 Fitur terpenting dari seleksi fitur *Mutual Information*

Pada kode program Gambar 4.56, dilakukan seleksi fitur *Mutual Information* untuk memilih 300 fitur terpenting sama seperti yang terdapat pada kategori *filter*. Fungsi mutual_info_classif() akan digunakan untuk melakukan proses seleksi fitur *Mutual Information*. Setelah itu 300 fitur yang telah diseleksi tersebut akan ditampilkan seperti pada Gambar 4.57.

Selanjutnya akan dilakukan proses seleksi fitur *Backward Elimination* dengan model *Random Forest Classifier* dan 300 fitur yang telah didapatkan melalui seleksi fitur *Mutual*

Information sebelumnya. Proses seleksi fitur *Backward Elimination* pada penelitian ini dijelaskan pada Gambar 4.58.

```
# take the 300 selected feature from Mutual Information
x_train_selected = x_train[features_mi.nlargest(300,columns="Mutual Information
Value").iloc[0:300, 0]]
x_test_selected = x_test[features_mi.nlargest(300,columns="Mutual Information
Value").iloc[0:300, 0]]

# calling the Random Forest Classifier model
lreg = RandomForestClassifier(n_estimators=100)
sfs1 = sfs(lreg, k_features=20, forward=False, verbose=2, scoring='accuracy',
n_jobs=-1)

# Backward Elimination
sfs1 = sfs1.fit(x_train_selected, y_train)

# print the selected feature
feat_names = list(sfs1.k_feature_names_)
print("Fitur yang diambil adalah: ")
print(feat_names)
len(feat_names)
```

Gambar 4.58 Kode program seleksi fitur *Backward Elimination*

Pada Gambar 4.58, akan diambil terlebih dahulu 300 fitur terpenting yang didapatkan dari seleksi fitur *Mutual Information* sebelumnya dan ke-300 fitur ini akan disimpan pada *Data Frame* "x_train_selected" dan "x_test_selected". Selanjutnya akan di-*define* seleksi fitur *Backward Elimination* dengan fungsi sfs(), parameter pertama diisi dengan model yang akan digunakan pada proses seleksi fitur *Backward Elimination* ini yaitu *Random Forest Classifier*, parameter kedua ("k_features") akan diisi nilai 20 yang artinya pada seleksi fitur *Backward Elimination* ini akan dipilih 20 fitur terbaik, parameter ketiga ("forward") akan diisi dengan nilai *False* yang artinya akan dilakukan proses seleksi fitur *Backward Elimination*, parameter keempat ("verbose") akan diisi dengan nilai dua yang artinya akan diperlihatkan ringkasan dari setiap proses iterasi pelatihan, parameter kelima ("scoring") akan diisi nilai accuracy yang artinya akan digunakan nilai akurasi untuk mengukur performa model *Random Forest Classifier* yang telah dibangun pada setiap iterasinya, dan parameter keenam ("n_jobs") yang akan diisi nilai -1 yang artinya akan digunakan semua CPU *cores* yang tersedia untuk menjalankan proses seleksi fitur *Backward Elimination* ini. Untuk proses seleksi fitur *Backward Elimination* pada penelitian ini akan dilakukan sebanyak lima kali, dengan total fitur yang diambil berbeda-beda yaitu kelipatan 20 (20, 40, 60, 80, dan 100). Untuk melakukan seleksi fitur *Backward Elimination* dengan total fitur yang berbeda-beda, hanya diperlukan penggantian nilai parameter "k_features" menjadi 20, 40, 60, 80, dan 100. Hasil dari seleksi

fitur *Backward Elimination* untuk 20 fitur ini dapat dilihat pada Gambar 4.59 sedangkan untuk 40, 60, 80, dan 100 lainnya dapat dilihat pada Lampiran.

```
Fitur yang diambil adalah:
['1138.2', '1139.2', '1132.2', '1358.9', '1134.2', '1361', '1276', '1270.9', '1273.4', '1266.6', '1263.5', '1269.7'
, '1662', '1657.8', '1000.3', '1282.9', '1629.6', '1669.5', '1002.2', '1678.1']
```

Gambar 4.59 Dua puluh fitur terpenting dari seleksi fitur *Backward Elimination*

3.3.7 Recursive Feature Elimination (RFE)

Metode *Recursive Feature Elimination* (RFE) bekerja dengan menghapus atribut secara rekursif dan membangun model pada atribut yang tersisa. Pada metode ini digunakan matriks akurasi untuk mengukur kepentingan suatu fitur, kepentingan fitur ini akan dijadikan dasar untuk ranking fitur terpenting. Pertama, akan di-*import library* "pandas" dan "numpy" seperti pada Gambar 4.5, *library* "sklearn" seperti pada Gambar 4.30, dan fungsi RFE() dari *library* "sklearn" seperti baris kode pada Gambar 4.60.

```
#import library
from sklearn.feature_selection import RFE
```

Gambar 4.60 Kode program *import* fungsi RFE()

Selanjutnya, dataset di-*import* dan dibagi dengan besaran 70% *train set* dan 30% *test set* dengan menjalankan baris kode seperti pada gambar 4.6 dan 4.31. Seleksi fitur RFE ini akan membangun model *Random Forest Classifier* dalam proses seleksi fiturnya. Setelah itu, dilakukan seleksi fitur RFE untuk mendapatkan 30 fitur dengan menjalankan baris kode seperti pada Gambar 4.61 beserta keluarannya pada Gambar 4.62.

```
cols = list(x.columns)
model = RandomForestClassifier()

#Initializing RFE model
rfe = RFE(model, 20)

#Transforming data using RFE
X_rfe = rfe.fit_transform(x_train, y_train)

#Fitting the data to model
model.fit(X_rfe,y_train)
temp = pd.Series(rfe.support_,index = cols)
selected_features_rfe = temp[temp==True].index
print(selected_features_rfe)
```

Gambar 4.61 Kode program seleksi fitur RFE untuk mencari 20 fitur terbaik

```
Index(['1088', '1125.3', '1126.8', '1129.7', '1133.7', '1135.2', '1136.2',
       '1136.7', '1138.7', '1139.2', '1141.2', '1329.6', '1335.8', '1874.7',
       '1876.1', '1877.4', '1878.8', '1882.9', '2038.3', '2111.3'],
      dtype='object')
```

Gambar 4.62 Fitur hasil seleksi fitur RFE

Baris kode program Gambar 4.61 akan melakukan proses seleksi fitur RFE dengan model *Random Forest Classifier* untuk mendapatkan 20 fitur terbaik. Selanjutnya, 20 fitur yang diseleksi akan ditampilkan menggunakan fungsi print() dan ke-20 fitur ini dapat dilihat pada Gambar 4.62. Seleksi fitur RFE ini juga akan dilakukan untuk mencari fitur dengan kelipatan 20 lainnya seperti pada dua metode *wrapper* sebelumnya dan hasilnya akan ditampilkan pada Lampiran.

3.3.8 LASSO (*Least Absolute Shrinkage and Selection Operator*)

Metode LASSO dilakukan berdasarkan model regresi linier dengan fungsi penalti L1. Pada metode ini digunakan matriks akurasi untuk mengukur kepentingan suatu fitur, kepentingan fitur ini akan dijadikan dasar untuk rangking fitur terpenting. Pertama, akan di-*import library* "pandas" dan "numpy" seperti pada Gambar 4.5, *library* "sklearn" pada Gambar 4.30 serta fungsi GridSearchCV() untuk melakukan perhitungan pengujian CV, fungsi StandardScaler() untuk melakukan operasi *Scalling dataset*, fungsi Pipeline() untuk membuat sebuah *pipeline* (alur kerja) dari proses seleksi fitur LASSO yang akan dilakukan, serta fungsi Lasso() untuk melakukan proses seleksi fitur LASSO. Keempat fungsi ini dapat di-*import* dengan menjalankan kode program pada Gambar 4.63.

```
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.linear_model import Lasso
```

Gambar 4.63 Kode program *import libraries*

Selanjutnya, dataset di-*import* dan dibagi dengan besaran 70% *train set* dan 30% *test set* dengan menjalankan baris kode seperti pada gambar 4.6 dan 4.31. Sebelum dilakukan proses seleksi fitur LASSO, perlu dilakukan *scalling* terhadap *dataset*. *Scalling* perlu dilakukan agar fitur memiliki nilai *importance* yang lebih besar dari nol. Oleh karena itu, dibangun objek *Pipeline* yang mampu memudahkan proses *Scalling* seperti pada Gambar 4.64.

```
#scaling
pipeline = Pipeline([
                    ('scaler',StandardScaler()),
                    ('model',Lasso())
])
```

Gambar 4.64 Kode program *scalling* dataset

Objek "Pipeline" yang terdapat pada Gambar 4.64 terdiri dari *StandardScaler* dan objek LASSO. Setelah di-*scalling*, perlu dilakukan proses optimasi *hyperparameter* α LASSO *Regression*. *Hyperparameter* α adalah nilai yang mengatur "intensitas" fungsi penalti L1. Optimisasi nilai α ini ditujukan agar didapatkan fungsi penalti L1 yang mampu dengan seefektif mungkin melakukan proses seleksi fitur. Untuk optimisasi ini, akan diuji nilai α dari 0,1 sampai 10 dengan nilai *step* sebesar 0,1. Untuk setiap nilai α dihitung nilai rata-rata *negative mean square error* dari 5-*folds cross validation* dan akan dipilih nilai α yang dapat meminimalkan nilai *negative mean square error* tersebut. Proses optimisasi nilai α akan dilakukan seperti yang tertera pada Gambar 4.65 dan hasilnya pada Gambar 4.66.

```
#buat objek GridSearchCV
search = GridSearchCV(pipeline,
                      {'model__alpha':np.arange(0.1,10,0.1)},
                      cv = 5, scoring="neg_mean_squared_error",verbose=3
                      )

#train objek GridSearchCV
search.fit(x_train,y_train)

#nilai alpha terbaik
search.best_params_
```

Gambar 4.65 Kode program untuk optimisasi nilai α

```
{'model__alpha': 0.1}
```

Gambar 4.66 Nilai α terbaik

Diketahui dari Gambar 4.66 bahwa nilai α terbaik adalah 0,1 yang artinya nilai α sebesar 0,1 dapat mengatur proses efektifitas LASSO *Regression* sampai ke titik yang paling maksimal atau optimum. Nilai α sebesar 0,1 ini dijadikan sebagai parameter dari proses LASSO *Regression* yang akan menghasilkan nilai koefisien dari setiap fitur. Selanjutnya, untuk mengambil nilai koefisien yang didapatkan dari proses LASSO *Regression*, dapat dilakukan dengan menjalankan baris kode pada Gambar 4.67.

```
coefficients = search.best_estimator_.named_steps['model'].coef_
```

Gambar 4.67 Kode program untuk mengambil nilai koefisien dari LASSO

Nilai *importance* yang digunakan untuk seleksi fitur Lasso Regression merupakan nilai absolut dari nilai koefisien di atas, oleh karena itu untuk mendapatkan nilai *importance* dapat dijalankan baris kode seperti pada Gambar 4.68.

```
importance = np.abs(coefficients)
```

Gambar 4.68 Kode program untuk mengambil nilai *importance*

Pada LASSO *Regression*, fitur yang memiliki nilai *importance* sama dengan nol akan dibuang dan untuk fitur yang memiliki nilai *importance* lebih dari nol akan disimpan. Untuk mendapatkan fitur apa saja yang mempunyai nilai *importance* yang lebih dari nol, dapat dijalankan baris kode seperti pada Gambar 4.69 dan hasil dapat dilihat pada Gambar 4.70.

```
np.array(features)[importance > 0]
```

Gambar 4.69 Kode program untuk mengambil fitur dengan nilai *importance* lebih dari nol

```
array(['1323.5', '1326.2', '1326.9', '1328.9', '1331', '1331.7', '1337.1',
       '1397.7', '1398.5'], dtype='<U6')
```

Gambar 4.70 Fitur dengan nilai *importance* lebih dari nol

Seperti yang terlihat pada Gambar 4.70, diperoleh sembilan fitur yang dari proses seleksi fitur LASSO.

3.3.9 *Elastic Net*

Elastic Net memilih variabel secara otomatis dan melakukan penyusutan terus menerus untuk meningkatkan akurasi prediksi dengan menggunakan dua istilah penalti yaitu regularisasi L1 dan L2. Pertama, akan di-*import library* "pandas", "numpy", dan "matplotlib" seperti pada Gambar 4.5, *library* "sklearn" seperti pada Gambar 4.30, dan fungsi ElasticNet() dari *library* "sklearn" seperti Gambar 4.71.

```
#import libraries
from sklearn.linear_model import ElasticNet
```

Gambar 4.71 Kode program *import* fungsi ElasticNet()

Selanjutnya, import *dataset* dan bagi dengan besaran 70% *train set* dan 30% *test set* dengan menjalankan baris kode seperti pada gambar 4.6 dan 4.31. Setelah itu, akan dicari nilai RMSE dari model yang dibangun menggunakan pasangan nilai parameter "alphas" dan "l1_ratio", kedua nilai ini merupakan parameter fungsi ElasticNet() yang nantinya digunakan

untuk melakukan seleksi fitur *Elastic Net*, untuk melakukan hal ini dapat dijalankan baris kode seperti pada Gambar 4.72.

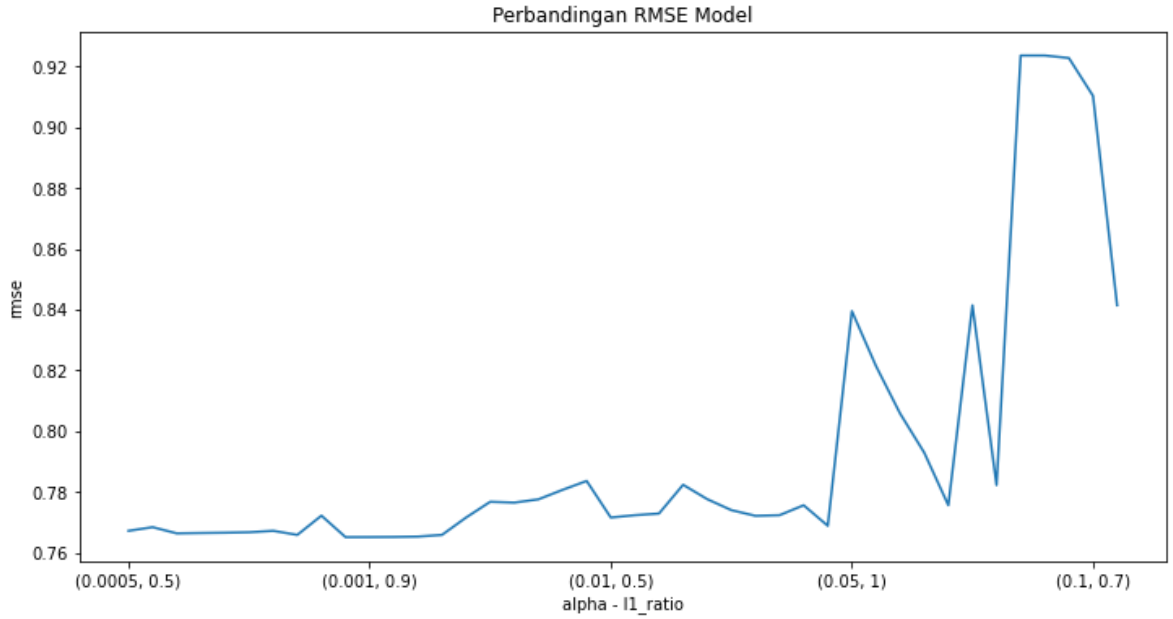
```
#cari nilai alpha dan l1_ratios terbaik
def rmse_cv(model):
    rmse= np.sqrt(-cross_val_score(model, X_train, y_train,
                  scoring="neg_mean_squared_error", cv = 5))
    return(rmse)

alphas = [0.0005, 0.001, 0.01, 0.03, 0.05, 0.1]
l1_ratios = [1.5, 1.1, 1, 0.9, 0.8, 0.7, 0.5]

cv_elastic = [rmse_cv(ElasticNet(alpha = alpha,
             l1_ratio=l1_ratio,tol=0.9)).mean()
             for (alpha, l1_ratio) in product(alphas, l1_ratios)]
```

Gambar 4.72 Kode program seleksi fitur *Elastic Net*

Pada Gambar 4.72, fungsi rmse_cv() akan melakukan perhitungan nilai *root mean squared error* (RMSE) dari model *Elastic Net* yang dibangun menggunakan pasangan nilai "alphas" dan "l1_ratios " yang diatur secara manual. Penulis menggunakan nilai "alphas" dan "l1_ratios" sebanyak enam nilai. Pasangan nilai "alphas" dan "l1_ratios" yang menghasilkan nilai RMSE paling kecil akan dipilih dan dijadikan parameter untuk menjalankan fungsi ElasticNet(). Perhitungan nilai RMSE ini akan dilakukan sampai semua kombinasi pasangan nilai "alphas" dan "l1_ratios" untuk membangun model *Elastic Net* tersebut, sudah berhasil dilakukan semua. Selanjutnya, nilai RMSE dari masing-masing model akan divisualisasikan menggunakan baris kode pada Gambar 4.73 dan hasil akan ditampilkan pada Gambar 4.74.

```
#visualisasi perbandingan RMSE
matplotlib.rcParams['figure.figsize'] = (12.0, 6.0)
idx = list(product(alphas, l1_ratios))
p_cv_elastic = pd.Series(cv_elastic, index = idx)
p_cv_elastic.plot(title = "Perbandingan RMSE Model")
plt.xlabel("alpha - l1_ratio")
plt.ylabel("rmse")
```

Gambar 4.73 Kode program visualisasi nilai RMSE dan pasangan nilai "alphas" dan "l1_ratio"



Gambar 4.74 Visualisasi nilai RMSE (sumbu y) dan pasangan nilai "alphas" dan "l1_ratio" (sumbu x)

Kode program Gambar 4.74 akan memvisualisasikan perbandingan nilai RMSE dan pasangan nilai "alphas" dan "l1_ratio" dengan menggunakan fungsi plot(). Nilai RMSE akan di-*plot*-kan menjadi sumbu y dan pasangan nilai "alphas" dan "l1_ratio" akan di-*plot*-kan menjadi sumbu x. Hasil dari visualisasi kode program Gambar 4.73 dapat dilihat pada Gambar 4.74 yang menunjukkan beberapa nilai pasangan nilai "alphas" dan "l1_ratio" dengan nilai RMSE. Dapat dilihat bahwa nilai RMSE terbesar ada pada pasangan nilai "alphas" sebesar 0,1 dan nilai "l1_ratio" sebesar 0,7. Untuk nilai RMSE terkecil (yaitu sekitar 0,76) ada pada pasangan nilai "alphas" sebesar 0,001 dan "l1_ratio" sebesar 0,9, pasangan nilai inilah yang akan diambil sebagai parameter fungsi ElasticNet(). Selanjutnya, pasangan nilai "alphas" dan "l1_ratio" ini akan digunakan sebagai parameter fungsi ElasticNet() untuk melakukan seleksi fitur *Elastic Net* seperti pada Gambar 4.75.

```
#seleksi fitur Elastic Net
elastic = ElasticNet(alpha=0.001, l1_ratio=0.9, tol=0.9)

elastic.fit(X_train, y_train)
```

Gambar 4.75 Kode program seleksi fitur Elastic Net

Proses seleksi fitur *Elastic Net* pada Gambar 4.75 akan menghasilkan nilai koefisien setiap fitur. Nilai koefisien ini akan dijadikan sebagai penentu apakah sebuah fitur layak untuk diambil (nilai koefisien lebih dari 0) atau dibuang (nilai koefisien sama dengan 0). Untuk

melihat berapa total fitur yang dibuang dan disimpan dapat dijalankan baris kode seperti pada Gambar 4.76 dan hasilnya akan ditampilkan pada Gambar 4.77.

```
#lihat total fitur yang diambil
coef = pd.Series(elastic.coef_, index = X_train.columns)

print("Elastic Net mengambil " + str(sum(coef != 0)) + " variabel dan
menghilangkan " +  str(sum(coef == 0)) + " variabel")
```

Gambar 4.76 Kode program untuk melihat total fitur yang diambil

```
Elastic Net mengambil 1009 variabel dan menghilangkan 548 variabel
```

Gambar 4.77 Total fitur yang diambil dan dibuang

Dari Gambar 4.77 dijelaskan bahwa terdapat 1009 variabel (*fitur*) hasil dari seleksi fitur *Elastic Net*, ke-1009 fitur ini artinya memiliki nilai koefisien yang tidak sama dengan nol dan dijelaskan juga bahwa terdapat 548 fitur yang dibuang, ini artinya ke-548 fitur tersebut mempunyai nilai koefisien yang sama dengan nol. Untuk melihat sebagian variabel yang diambil, dapat dijalankan baris kode pada Gambar 4.78 dan hasilnya akan ditampilkan pada Gambar 4.79.

```
#buat data frame yg nyimpan fitur terpilih beserta coef-nya
df_coef = pd.DataFrame(coef, columns=["Coeficients"])
df_fitur_terpilih = df_coef[(df_coef != 0).all(1)]

#print fitur yang dipilih
print("Fitur yang dipilih : " + str(df_fitur_terpilih.index.tolist()))
```

Gambar 4.78 Kode program untuk melihat fitur yang diambil

```
Fitur yang dipilih : ['999.9', '1000.3', '1000.7', '1079.4', '1079.9', '1080.8', '1082.1', '1082.6', '1083.9', '1084.4', '1085.
3', '1085.7', '1086.2', '1086.6', '1087.1', '1088.9', '1089.4', '1089.8', '1091.2', '1091.7', '1092.1', '1092.6', '1093.1', '10
93.5', '1094', '1094.4', '1094.9', '1095.4', '1095.8', '1096.3', '1096.8', '1097.2', '1098.2', '1098.6', '1099.1', '1099.5', '1
100', '1100.5', '1100.9', '1101.4', '1101.9', '1102.4', '1102.8', '1103.3', '1103.8', '1104.2', '1104.7', '1105.6', '1106.1',
'1106.6', '1107.1', '1108.5', '1109', '1109.4', '1109.9', '1110.4', '1110.9', '1111.8', '1112.3', '1112.8', '1113.2', '1113.7',
'1114.2', '1114.7', '1115.2', '1115.6', '1116.1', '1117.6', '1118', '1119', '1119.5', '1120', '1120.5', '1120.9', '1121.4', '11
21.9', '1122.9', '1123.4', '1124.3', '1124.8', '1125.3', '1125.8', '1126.3', '1126.8', '1127.3', '1127.8', '1128.3', '1128.7',
'1129.2', '1129.7', '1130.2', '1130.7', '1131.2', '1131.7', '1132.2', '1132.7', '1133.2', '1133.7', '1134.2', '1134.7', '1135.
2', '1135.7', '1136.2', '1136.7', '1137.2', '1137.7', '1138.2', '1138.7', '1139.2', '1139.7', '1140.2', '1140.7', '1141.2', '11
41.7', '1142.2', '1142.7', '1143.2', '1143.7', '1144.2', '1144.7', '1145.2', '1145.7', '1146.2', '1146.7', '1147.2', '1147.7',
'1148.2', '1148.8', '1149.3', '1149.8', '1150.3', '1150.8', '1151.3', '1151.8', '1152.3', '1152.8', '1153.4', '1153.9', '1154.
4', '1154.9', '1155.4', '1155.9', '1156.4', '1157', '1157.5', '1158', '1158.5', '1159', '1159.5', '1160.1', '1160.6', '1161.1',
'1161.6', '1162.7', '1163.2', '1163.7', '1165.3', '1165.8', '1166.3', '1166.8', '1167.4', '1167.9', '1168.4', '1169', '1170',
'1170.5', '1171.1', '1171.6', '1172.1', '1173.2', '1173.7', '1174.2', '1175.3', '1175.8', '1176.4', '1176.9', '1177.4', '1178',
'1178.5', '1179.1', '1179.6', '1180.1', '1180.7', '1181.2', '1181.7', '1182.8', '1183.4', '1185', '1185.5', '1186.1', '1186.6',
'1187.1', '1188.2', '1188.8', '1189.3', '1191.5', '1192.1', '1192.6', '1193.2', '1193.7', '1194.8', '1197', '1197.6', '1199.2',
'1199.8', '1205.4', '1205.9', '1206.5', '1207', '1208.7', '1209.3', '1210.4', '1211', '1211.6', '1212.1', '1213.8', '1214.4',
'1216.7', '1217.2', '1217.8', '1218.4', '1219', '1219.5', '1220.1', '1220.7', '1222.4', '1223', '1224.1', '1224.7', '1225.3',
'1225.9', '1226.5', '1227', '1228.2', '1228.8', '1229.4', '1229.9', '1230.5', '1231.1', '1232.9', '1233.5', '1234', '1234.6',
'1235.2', '1235.8', '1236.4', '1239.4', '1239.9', '1241.1', '1241.7', '1242.3', '1242.9', '1244.7', '1247.1', '1247.7', '1248.
3', '1249.5', '1250.1', '1250.7', '1251.3', '1251.9', '1252.5', '1253.7', '1254.3', '1256.2', '1256.8', '1257.4', '1258', '125
9.2', '1259.8', '1261.7', '1262.3', '1262.9', '1264.7', '1265.4', '1266', '1267.2', '1267.8', '1268.5', '1269.7', '1270.3', '12
70.9', '1271.6', '1272.2', '1272.8', '1273.4', '1275.3', '1276', '1277.2', '1277.8', '1278.5', '1279.1', '1279.7', '1280.4', '1
281', '1281.6', '1282.3', '1284.2', '1284.8', '1285.4', '1286.1', '1286.7', '1288', '1288.6', '1289.3', '1289.9', '1290.6', '12
91.2', '1291.8', '1292.5', '1293.1', '1293.8', '1294.4', '1295.1', '1295.7', '1296.4', '1297', '1297.7', '1298.3', '1299', '129
9.6', '1300.3', '1300.9', '1301.6', '1302.2', '1302.9', '1303.5', '1304.2', '1304.8', '1305.5', '1306.2', '1306.8', '1307.5',
```

Gambar 4.79 Sebagian dari total fitur yang diambil dari seleksi fitur *Elastic Net*

Setelah melakukan seluruh proses seleksi fitur, selanjutnya akan dilihat fitur-fitur yang paling sering digunakan pada seluruh seleksi fitur yang telah dilakukan. Untuk melihatnya, Pertama akan dibuat list untuk menyimpan semua fitur yang telah diseleksi oleh masing-masing metode seperti pada Gambar 4.80.

```
list_anova = ['1092.6', '1093.1', '1096.8', '1089.8', '1095.4', '1095.8',
              '1097.2', '1103.3', '1102.8', '1102.4', '1094.4', '1091.7',
              '1092.1', '1094', '1089.4', '1091.2', '1098.6', '1096.3',
              '1103.8', '1094.9', '1093.5', '1107.1', '1100.9', '1100.5',
              '1086.6', '1087.1', '1104.7', '1101.9', '1099.5', '1100']
list_mi = ['1139.7', '1140.2', '1141.7', '1138.2', '1138.7', '1139.2',
           '1140.7', '1141.2', '1142.2', '1132.2', '1137.7', '1131.7',
           '1137.2', '1359.6', '1358.9', '1142.7', '1134.2', '1360.3',
           '1133.7', '1361', '1131.2', '1132.7', '1272.8', '1358.2',
           '1133.2', '1357.5', '1356.7', '1267.2', '1275.3', '1262.3']
```

Gambar 4.80 *List* fitur-fitur yang telah terseleksi oleh setiap metode

Karena terlalu banyak, kode program untuk menyimpan list fitur hasil dari seleksi fitur lainnya tidak ditampilkan pada Gambar 4.80. Untuk *list* yang menyimpan fitur hasil dari seleksi fitur lainnya juga dibuat dengan kode program yang sama seperti Gambar 4.80, menyesuaikan dengan isinya. Setelah itu, semua *list* tersebut akan digabungkan menjadi satu *list* dan dimasukkan ke sebuah *data frame* dengan menjalankan kode program seperti Gambar 4.81

```
list_semua_fitur = list_anova + list_mi + list_fisher + list_reliefF +
list_sfs + list_backward + list_rfe + list_ElasticNet + list_Lasso
df_semua_fitur = pd.DataFrame(list_semua_fitur, columns=['fitur'])
```

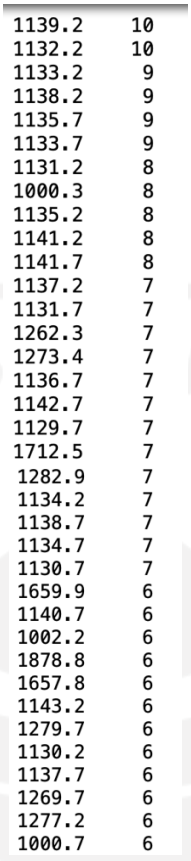
Gambar 4.81 Gabungkan semua *list* menjadi satu *data frame*

Selanjutnya akan dicari fitur-fitur yang setidaknya muncul lebih dari lima kali dari *data frame* tersebut dengan kode program pada Gambar 4.82.

```
df_semua_fitur['fitur'].value_counts()[:36]
```

Gambar 4.82 Cari fitur-fitur yang muncul lebih dari dua kali

Fungsi value_counts() akan mengembalikan fitur yang paling banyak muncul pada *data frame* yang berisikan semua fitur yang didapatkan dari hasil proses seleksi fitur yang telah dilakukan. Terdapat 36 fitur yang muncul lebih dari lima kali, hasilnya dapat dilihat pada Gambar 4.83.



1139.2	10
1132.2	10
1133.2	9
1138.2	9
1135.7	9
1133.7	9
1131.2	8
1000.3	8
1135.2	8
1141.2	8
1141.7	8
1137.2	7
1131.7	7
1262.3	7
1273.4	7
1136.7	7
1142.7	7
1129.7	7
1712.5	7
1282.9	7
1134.2	7
1138.7	7
1134.7	7
1130.7	7
1659.9	6
1140.7	6
1002.2	6
1878.8	6
1657.8	6
1143.2	6
1279.7	6
1130.2	6
1137.7	6
1269.7	6
1277.2	6
1000.7	6

Gambar 4.83 Fitur-fitur yang paling banyak muncul

Gambar 4.83 menunjukkan fitur-fitur serta jumlah kemunculannya dari yang paling banyak terpilih pada kesembilan seleksi fitur yang telah dilakukan.

3.3.10 Waktu Eksekusi Kode dari Setiap Metode Seleksi Fitur

Waktu eksekusi yang dibutuhkan untuk melakukan proses seleksi fitur berbeda tergantung dari jenis metodenya. Untuk mendapatkan waktu eksekusi kode program dari setiap proses seleksi fitur yang telah dilakukan, akan ditambahkan *magic command* "%%time" pada setiap sel yang menjalankan kode program seleksi fitur. Tabel 4.1 akan menujukkan waktu yang dibutuhkan dari setiap proses seleksi fitur yang digunakan pada penelitian ini dalam satuan detik (s).

Tabel 4.1 Waktu Eksekusi Kode dari Setiap Metode Selekesi Fitur

Metode Seleksi Fitur	Jumlah Fitur	Waktu Eksekusi
ANOVA	100	0.40
Mutual Information	100	5.3
Fisher Score	100	0.41
ReliefF	100	0.43

Forward Selection	20	1762.8
	40	3600
	60	4680
	80	8280
	100	8244
Backward Elimination	20	2839.2
	40	2784.6
	60	2658.6
	80	2546.4
	100	2409
Recursive Feature Elimination	20	240
	40	246
	60	260
	80	210
	100	205
LASSO	9	84
Elastic Net	1009	45

Berdasarkan Tabel 4.1, dapat dilihat bahwa waktu eksekusi yang dibutuhkan oleh metode seleksi fitur *Filter* dan *Embedded* jauh lebih singkat daripada metode *Wrapper*. Hal ini terjadi karena pada metode *wrapper,* proses pembelajaran dilakukan berulang kali.

3.4 Proses Klasifikasi Menggunakan Atribut Data Hasil Seleksi Fitur

Setelah dilakukan proses seleksi fitur, akan dilakukan proses klasifikasi menggunakan metode *Random Forest* dan *Support Vector Machine* dengan fitur yang telah diseleksi. Untuk fitur yang didapatkan dari seleksi fitur jenis *Filter* dan *Embedded*, pembangunan model klasifikasi *Random Forest Classifier* dan SVM ini akan dilakukan dengan mencoba menggunakan satu demi satu fitur hasil seleksi (hingga 100 fitur) untuk melihat puncak terbaik dari performa klasifikasi yang dilakukan. Sedangkan untuk fitur hasil seleksi kategori *wrapper,* pembangunan model klasifikasi akan dilakukan menggunakan fitur-fitur dengan kelipatan 20 dengan batas 100 fitur hasil seleksi *wrapper*.

3.4.1 Klasifikasi Random Forest Menggunakan Atribut Data Hasil Seleksi Fitur

Penelitian ini menggunakan *estimator* atau *trees* berjumlah 100, 150, dan 200 dalam membangun model klasifikasi *Random Forest.* Hal ini berarti klasifikasi menggunakan *Random Forest* akan dilakukan tiga kali sesuai dengan jumlah *tree* yang digunakan. Penggunaan tiga nilai *estimator* berbeda ini dilakukan untuk melihat penggunaan jumlah *trees* terbaik pada klasifikasi data NIR. Untuk pengujian model klasifikasi *Random Forest Classifier*, digunakan nilai rata-rata *accuracy*, *precision_micro*, dan *recall_micro* dari 10-*fold cross*

validation train set. Untuk *test set* nilai pengujian yang dihitung sama dengan yang digunakan *train set* tetapi tidak akan diaplikasikan *cross validation* kepada *test set*. Dalam melakukan pembangunan dan pengujian model klasifikasi *Random Forest Classifier*, pertama akan di-*import* fungsi cross_validate() untuk melakukan proses pengujian dengan *k-fold cross validation*, metrics() untuk menghitung pengujian *test set*, KFold() untuk membuat objek *k-fold cross validation*, *library* "time" untuk menghitung waktu eksekusi klasifikasi, dan RandomForestClassifier() untuk melakukan proses klasifikasi *Random Forest Classifier* dari *library* "sklearn" seperti pada kode program Gambar 4.84.

```
#import libraries
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_validate
from sklearn import metrics
import time
```

Gambar 4.84 Kode program *import libraries* untuk klasifikasi *Random Forest Classifier*

Selanjutnya akan dibuat objek "cv" yang mengatur proses *k-fold cross validation* dengan menggunakan fungsi KFold() seperti pada Gambar 4.85.

```
#cross validation 10-fold
cv = KFold(n_splits=10, random_state=1, shuffle=True)
```

Gambar 4.85 Kode program untuk membuat objek "cv"

Ditentukan nilai parameter "fold" yang digunakan adalah sepuluh (ini berarti terdapat sepuluh fold yang akan digunakan dalam proses klasifikasi *Random Forest Classifier* ini), "random_state" yang digunakan adalah satu (ini berarti nilai satu akan digunakan sebagai *seed* atau pengatur keacakan data yang dibagi menjadi sepuluh *fold*), dan "shuffle" bernilai "True" (ini berarti akan dilakukan pengacakan saat data dibagi menjadi sepuluh *fold*). Selanjutnya akan dilakukan proses pelatihan sekaligus pengujian model menggunakan *Random Forest Classifier* seperti pada kode program Gambar 4.86.

```
#tentukan metode scoring yang digunakan
scoring_rfe = {'acc': 'accuracy',
               'prec_micro': 'precision_micro',
               'rec_micro': 'recall_micro'}

#tentukan total fitur yang digunakan dalam proses klasifikasi Random Forest
ini
n_feat = range(1, 101)
n_trees = [100]
```

```
max_acc = 0
max_prec = 0
max_rec = 0

for nfeat in n_feat:

        print("================================================")
        start_time = time.time()
#ambil n fitur input hasil seleksi fitur ANOVA
x_train_selected                                                    =
x_train[features_score.nlargest(100,columns="Score").iloc[0:nfeat, 0]]
x_test_selected                                                     =
x_test[features_score.nlargest(100,columns="Score").iloc[0:nfeat, 0]]

#Create a Gaussian Classifier
clf_rfe = RandomForestClassifier(n_estimators=ntrees)

#Train the model using the training sets
clf_rfe.fit(x_train_selected, y_train)
y_pred_rfe=clf_rfe.predict(x_test_selected)

#hitung score model dari data train
scores_rfe    =    cross_validate(clf_rfe,    x_train_selected,    y_train,
scoring=scoring_rfe, cv=cv, return_train_score=True)

print("Akurasi model RF data Train dengan " + str(nfeat) + " fitur dan " +
str(ntrees) + " trees:"
      + str(round(scores_rfe['train_acc'].mean(), 2)))
print("Akurasi model RF data Test dengan " + str(nfeat) + " fitur dan " +
str(ntrees) + " trees:"
      + str(round(metrics.accuracy_score(y_test, y_pred_rfe), 2)))
print("Precision model RF data Train dengan " + str(nfeat) + " fitur dan "
+ str(ntrees) + " trees:"
      + str(round(scores_rfe['train_prec_micro'].mean(), 2)))
print("Precision model RF data Test dengan " + str(nfeat) + " fitur dan " +
str(ntrees) + " trees:"
      +       str(round(metrics.precision_score(y_test,        y_pred_rfe,
average='micro'), 2)))
print("Recall model RFE data Train dengan " + str(nfeat) + " fitur dan " +
str(ntrees) + " trees:"
      + str(round(scores_rfe['train_rec_micro'].mean(), 2)))
print("Recall model RFE data Test dengan " + str(nfeat) + " fitur dan " +
str(ntrees) + " trees:"
      +         str(round(metrics.recall_score(y_test,         y_pred_rfe,
average='micro'), 2)))
print(" ")

   if(round(metrics.accuracy_score(y_test, y_pred_rfe), 2) > max_acc):
            max_acc = round(metrics.accuracy_score(y_test, y_pred_rfe), 2)

   if(round(metrics.precision_score(y_test,  y_pred_rfe,  average='micro'),
2) > max_prec):
      max_prec    =    round(metrics.precision_score(y_test,    y_pred_rfe,
average='micro'), 2)

   if(round(metrics.recall_score(y_test, y_pred_rfe, average='micro'), 2) >
max_rec):
```

```
      max_rec     =     round(metrics.recall_score(y_test,     y_pred_rfe,
average='micro'), 2)

        end_time = time.time()
        print("Total waktu: ", end_time - start_time)

print("====Nilai Performa Tertinggi====")
print("Nilai akurasi model tertinggi: " + str(max_acc))
print("Nilai presisi model tertinggi: " + str(max_prec))
print("Nilai Recall model tertinggi: " + str(max_rec))
```

Gambar 4.86 Kode program pelatihan dan pengujian model *Random Forest Classifier*

Pada kode program Gambar 4.86, pertama akan dibuat deskripsi metode pengujian model *Random Forest Classifier* yang akan dibuat, ditentukan metode pengujian dengan mencari akurasi, *recall micro* dan *precision micro* dari sebuah model *Random Forest Classifier*. Deskripsi metode pengujian ini akan disimpan dalam *dictionary* "scoring_rfe". Kemudian untuk pembangunan model *Random Forest Classifier* dengan fitur hasil seleksi fitur *Filter* dan *Embedded* jumlah fitur yang digunakan akan ditambahkan satu demi satu dengan menggunakan perulangan *for*. Untuk pembangunan model *Random Forest Classifier* dengan fitur hasil seleksi *wrapper,* nilai dari variabel n_feat akan ditentukan sendiri dan diisi dengan nilai kelipatan 20. Ditentukan juga total *trees* yang digunakan adalah 100, 150, dan 200. Nilai total fitur yang digunakan akan disimpan dalam *list* "n_feat" dan untuk nilai total *trees* yang digunakan akan disimpan dalam *list* "n_trees".

Pertama akan dibuat *data frame* yang berisikan data *input training* ("x_train_selected") dan *testing* ("x_test_selected"). Kedua *data frame* ini memiliki *syntax* yang berbeda sesuai dengan seleksi fitur yang digunakannya. Kode program Gambar 4.87 berisi beberapa *syntax* untuk mengambil data berdasarkan seleksi fitur yang digunakan.

```
#ambil n fitur input hasil seleksi fitur jenis Filter
x_train_selected =
                  x_train[fisher_ranking.nlargest(100,columns="Fisher_Scor
                  e").iloc[0:nfeat, 0]]
x_test_selected =
                  x_test[fisher_ranking.nlargest(100,columns="Fisher_Score
                  ").iloc[0:nfeat, 0]]

#ambil n fitur input hasil seleksi fitur jenis Wrapper
x_train_selected = x_train[feat_names].iloc[:,0:nfeat]
x_test_selected = x_test[feat_names].iloc[:,0:nfeat]

#ambil n fitur input hasil seleksi fitur LASSO
x_train_selected = x_train[np.array(features)[importance > 0]]
x_test_selected = x_test[np.array(features)[importance > 0]]
```

```
#ambil n fitur input hasil seleksi fitur Elastic Net
x_train_selected = x_train[df_fitur_terpilih.index]
x_test_selected = x_test[df_fitur_terpilih.index]
```

Gambar 4.87 Beberapa *syntax* untuk mengambil data

Pada Gambar 4.87 dipaparkan beberapa *syntax* untuk mengambil data dan menyimpannya dalam *data frame* "x_train_selected" dan "x_test_selected", berikut penjelasannya:

a. Untuk data dengan fitur yang didapatkan dari seleksi fitur jenis *Filter* akan digunakan100 fitur terpenting yang telah diseleksi. Pada setiap iterasi, akan dipilih fitur dengan nilai pengujian seleksi fitur tertinggi. Pada Gambar 4.87 diberikan contoh untuk mengambil data dari fitur yang telah diseleksi oleh seleksi fitur *Fisher* yang menggunakan nilai *Fisher score* sebagai nilai pengujiannya.
b. Untuk data dengan fitur yang didapatkan dari seleksi fitur *Wrapper,* akan ditentukan fitur dengan kelipatan 20 yaitu 20, 40, 60, 80, dan 100 fitur. Indeks "feat_names" berisikan semua nama fitur hasil seleksi sementara nfeat berisi jumlah fitur yang akan digunakan.
c. Untuk data dengan fitur yang didapatkan dari seleksi fitur *Elastic Net* akan digunakan indeks dari *data frame* "df_fitur_terpilih" seperti yang telah dijelaskan pada Gambar 4.78.
d. Untuk data dengan fitur yang didapatkan dari seleksi fitur LASSO akan dipilih fitur yang memiliki nilai *importance* lebih dari nol seperti yang telah dijelaskan pada Gambar 4.69.

Setelah itu, akan dibuat objek "clf_rfe" yang berisikan deskripsi proses klasifikasi *Random Forest Classifier* yang akan dilakukan. Pada objek "clf_rfe", ditentukan total *trees* yang akan digunakan dalam pembangunan model klasifikasi *Random Forest Classifier* yang akan dilakukan melalui parameter "n_estimators". Nilai dari parameter "n_estimators" akan berubah setiap iterasi sesuai dengan nilai "n_trees" yang telah ditetapkan.

Selanjutnya akan dilakukan proses pelatihan model klasifikasi *Random Forest Classifier* menggunakan data *input training* ("x_train_selected") serta data *output training* ("y_train") yang telah dijelaskan pada Gambar 4.31. Proses pelatihan model klasifikasi *Random Forest Classifier* ini akan dilakukan dengan menggunakan fungsi fit(). Setelah itu, akan dibuat variabel yang menyimpan hasil prediksi dari data *input testing* ("x_test_selected") dengan menggunakan model *Random Forest Classifier* yang telah dilatih sebelumnya. Proses prediksi ini dilakukan menggunakan fungsi predict().

Selanjutnya akan dihitung nilai performa prediksi data *training* oleh model dengan 10-*fold cross validation* menggunakan fungsi cross_validate(). Akan dihitung juga nilai performa prediksi data *testing* oleh model dengan menggunakan fungsi accuracy_score(), precision_score,() dan recall_score(). Hasil dari fungsi ini akan ditampilkan menggunakan fungsi print() seperti pada Gambar 4.88.

```
Total waktu:  1.3671367168426514
================================================
Akurasi model RFE data Train dengan 44 fitur dan 100 trees:1.0
Akurasi model RFE data Test dengan 44 fitur dan 100 trees:0.89
Precision model RFE data Train dengan 44 fitur dan 100 trees:1.0
Precision model RFE data Test dengan 44 fitur dan 100 trees:0.89
Recall model RFE data Train dengan 44 fitur dan 100 trees:1.0
Recall model RFE data Test dengan 44 fitur dan 100 trees:0.89

Total waktu:  1.3695893287658691
================================================
Akurasi model RFE data Train dengan 45 fitur dan 100 trees:1.0
Akurasi model RFE data Test dengan 45 fitur dan 100 trees:0.91
Precision model RFE data Train dengan 45 fitur dan 100 trees:1.0
Precision model RFE data Test dengan 45 fitur dan 100 trees:0.91
Recall model RFE data Train dengan 45 fitur dan 100 trees:1.0
Recall model RFE data Test dengan 45 fitur dan 100 trees:0.91
```

Gambar 4.88 Nilai pengujian model *Random Forest Classifier*

Seperti yang terlihat pada Gambar 4.88, proses klasifikasi SVM dalam menggunakan 44 dan 45 fitur dari ANOVA menghasilkan performa berbeda. Proses klasifikasi ini akan diulang sebanyak 100 kali untuk melihat jumlah fitur optimal yang dapat menghasilkan performa tertinggi dari penggunaan seleksi fitur ANOVA. Proses yang sama juga dilakukan untuk delapan seleksi fitur lainnya.

3.4.2 Klasifikasi *Support Vector Machine* (SVM) Menggunakan Atribut Data Hasil Seleksi Fitur

Akan dilakukan juga pelatihan dan pengujian model klasifikasi menggunakan metode *Support Vector Machine* (SVM) dengan fitur yang telah diseleksi. Penentuan jumlah fitur hasil seleksi fitur dari tiga kategori yang digunakan pada klasifikasi menggunakan SVM ini sama seperti yang dilakukan pada *Random Forest.* Untuk pengujian model klasifikasi SVM, digunakan nilai rata-rata *accuracy*, *precision_micro*, dan *recall_micro* dari 10-*fold cross validation train set*. Untuk *test set* nilai pengujian yang dihitung sama dengan yang digunakan *train set* tetapi tidak akan diaplikasikan *cross validation* kepada *test set*.

Dalam melakukan pembangunan dan pengujian model klasifikasi SVM, pertama akan di-*import* fungsi cross_validate() untuk melakukan proses pengujian dengan *k-fold cross*

validation, metrics() untuk menghitung pengujian *test set*, *library* "time" untuk menghitung waktu eksekusi dan KFold() untuk membuat objek *k-fold cross validation*, semua *library* yang disebutkan sudah dijelaskan cara import-nya pada Gambar 4.84. Untuk membangun model SVM diperlukan fungsi svm.SVC() yang ada pada *library* "sklearn", library ini dapat di-*import* dengan kode program Gambar 4.89.

```
#import libraries
from sklearn import svm
```

Gambar 4.89 Kode program *import* fungsi svm.SVC() dari *library* "sklearn"

Selanjutnya akan dibuat objek "cv" yang mengatur proses *k-fold cross validation* dengan menggunakan fungsi KFold() seperti pada Gambar 4.85. Setelah itu, akan dilakukan proses pelatihan sekaligus pengujian model SVM seperti pada kode program Gambar 4.90.

```
#tentukan metode scoring yang digunakan
scoring_svm = {'acc': 'accuracy',
               'prec_micro': 'precision_micro',
               'rec_micro': 'recall_micro'}

#tentukan total fitur yang digunakan dalam proses klasifikasi SVM ini
n_feat = range(1, 101)
max_acc = 0
max_prec = 0
max_rec = 0

for nfeat in n_feat:
    print("================================================")
    start_time = time.time()

    #ambil n fitur input hasil seleksi fitur MI
    x_train_selected=
   x_train[fisher_ranking.nlargest(100,columns="Fisher_Ranking").iloc[0:
   nfeat, 0]]
    x_test_selected=
   x_test[fisher_ranking.nlargest(100,columns="Fisher_Ranking").iloc[0:
   nfeat, 0]]
    #Create a Support Vector Classifier
    clf_svm = svm.SVC()

    #Train the model using the training sets
    clf_svm.fit(x_train_selected, y_train)
    y_pred_svm = clf_svm.predict(x_test_selected)

    #hitung score model dari data train
    scores_svm = cross_validate(clf_svm, x_train_selected, y_train,
                scoring=scoring_svm, cv=cv, return_train_score=True)

    print("akurasi model SVM data Train dengan " + str(nfeat) + " fitur: "
          + str(round(scores_svm['train_acc'].mean(), 2)))
    print("akurasi model SVM data Test dengan " + str(nfeat) + " fitur: "
```

```
          + str(round(metrics.accuracy_score(y_test, y_pred_svm), 2)))
    print("Precision model SVM data Train dengan " + str(nfeat) + " fitur:"
          + str(round(scores_svm['train_prec_micro'].mean(), 2)))
    print("Precision model SVM data Test dengan " + str(nfeat) + " fitur:"
          + str(round(metrics.precision_score(y_test, y_pred_svm,
          average='micro'), 2)))
    print("Recall model SVM data Train dengan " + str(nfeat) + " fitur:"
          + str(round(scores_svm['train_rec_micro'].mean(), 2)))
    print("Recall model SVM data Test dengan " + str(nfeat) + " fitur:"
          + str(round(metrics.recall_score(y_test, y_pred_svm,
          average='micro'), 2)))
    print(" ")

   if(round(metrics.accuracy_score(y_test, y_pred_svm), 2) > max_acc):
            max_acc = round(metrics.accuracy_score(y_test, y_pred_svm), 2)

     if(round(metrics.precision_score(y_test, y_pred_svm, average='micro'),
2) > max_prec):
        max_prec   =   round(metrics.precision_score(y_test,   y_pred_svm,
average='micro'), 2)

     if(round(metrics.recall_score(y_test, y_pred_svm, average='micro'), 2)
> max_rec):
        max_rec    =    round(metrics.recall_score(y_test,    y_pred_svm,
average='micro'), 2)

     end_time = time.time()
     print("Total waktu: ", end_time - start_time)

print("====Nilai Performa Tertinggi====")
print("Nilai akurasi model tertinggi: " + str(max_acc))
print("Nilai presisi model tertinggi: " + str(max_prec))
print("Nilai Recall model tertinggi: " + str(max_rec))
```

Gambar 4.90 Kode program pelatihan dan pengujian model SVM

Pada kode program Gambar 4.90, pertama akan dibuat deskripsi metode pengujian model SVM yang akan dibuat, ditentukan metode pengujian dengan mencari akurasi, *recall micro* dan *precision micro* dari sebuah model SVM. Deskripsi metode pengujian ini akan disimpan dalam *dictionary* "scoring_svm". Kemudian untuk pembangunan model SVM akan ditentukan total fitur yang digunakan sesuai dengan metode yang digunakan. Seperti pada contoh Gambar 4.90 yang akan menggunakan 100 fitur terbaik dari metode *fisher score*. Fitur-fitur ini nantinya akan ditambahkan satu demi satu pada model klasifikasi svm menggunakan perulangan *for*. Nilai total fitur yang digunakan akan disimpan dalam *list* "n_feat".

Akan dibuat *data frame* yang berisikan data *input training* ("x_train_selected") dan *testing* ("x_test_selected"). Kedua *data frame* ini memiliki *syntax* yang berbeda sesuai dengan seleksi fitur yang digunakannya. Kode program Gambar 4.87 berisi beberapa *syntax* untuk mengambil data berdasarkan seleksi fitur yang digunakan.

Setelah itu, akan dibuat objek " clf_svm" yang berisikan deskripsi proses klasifikasi SVM yang akan dilakukan dengan menggunakan fungsi SVC(). Selanjutnya akan dilakukan proses pelatihan model klasifikasi SVM menggunakan data *input training* ("x_train_selected") serta data *output training* ("y_train") yang telah dijelaskan pada Gambar 4.31. Proses pelatihan model klasifikasi SVM ini akan dilakukan dengan menggunakan fungsi fit(). Setelah itu, akan dibuat variabel yang menyimpan hasil prediksi dari data *input testing* ("x_test_selected") dengan menggunakan model SVM yang telah dilatih sebelumnya. Proses prediksi ini dilakukan menggunakan fungsi predict().

Selanjutnya akan dihitung nilai performa prediksi data *training* oleh model dengan 10-*fold cross validation* menggunakan fungsi cross_validate(). Akan dihitung juga nilai performa prediksi data *testing* oleh model dengan menggunakan fungsi accuracy_score(), precision_score,() dan recall_score(). Hasil dari fungsi ini akan ditampilkan menggunakan fungsi print() seperti pada Gambar 4.91.

```
Total waktu:  0.10031914710998535
================================================
akurasi model SVM data Train dengan 32 fitur: 0.71
akurasi model SVM data Test dengan 32 fitur: 0.64
Precision model SVM data Train dengan 32 fitur:0.71
Precision model SVM data Test dengan 32 fitur:0.64
Recall model SVM data Train dengan 32 fitur:0.71
Recall model SVM data Test dengan 32 fitur:0.64

Total waktu:  0.10565447807312012
================================================
akurasi model SVM data Train dengan 33 fitur: 0.71
akurasi model SVM data Test dengan 33 fitur: 0.66
Precision model SVM data Train dengan 33 fitur:0.71
Precision model SVM data Test dengan 33 fitur:0.66
Recall model SVM data Train dengan 33 fitur:0.71
Recall model SVM data Test dengan 33 fitur:0.66
```

Gambar 4.91 Nilai pengujian model SVM

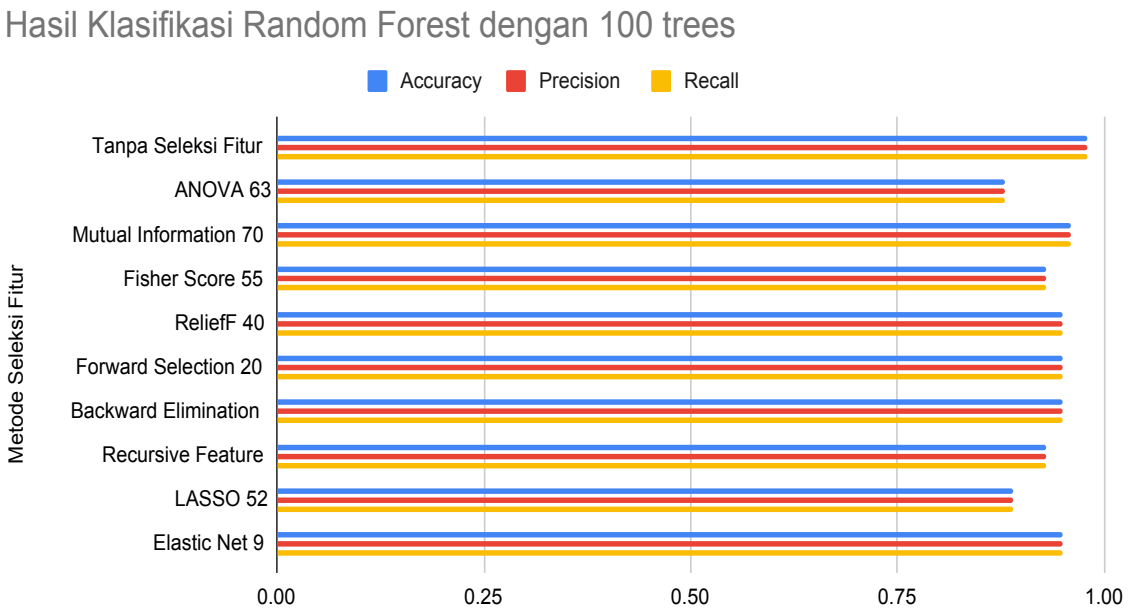
Seperti yang terlihat pada Gambar 4.91, proses klasifikasi SVM dalam menggunakan 32 dan 33 fitur dari *fisher score* menghasilkan performa berbeda. Proses klasifikasi SVM ini akan diulang sebanyak 100 kali untuk melihat jumlah fitur optimal yang dapat menghasilkan performa tertinggi dari penggunaan seleksi fitur *fisher score.* Proses yang sama juga dilakukan untuk delapan seleksi fitur lainnya.

3.5 Hasil Pengujian Model Klasifikasi

Setelah mendapatkan hasil dari pengujian terhadap empat skenario klasifikasi terhadap data NIRS, selanjutnya adalah memasukkan seluruh data yang diperoleh ke dalam empat tabel berbeda sesuai dari klasifikasi yang dilakukan agar hasil kombinasi seleksi fitur dan metode klasifikasi terbaik dapat dilihat dan dibandingkan. Untuk lebih memudahkan dalam melihat nilai tertinggi dan terendah dari setiap proses klasifikasi, nilai tertinggi akan ditandai tengan warna hijau sedangkan nilai terendah akan ditandai dengan warna merah. Tabel 4.1 akan menunjukkan hasil klasifikasi menggunakan Random Forest dengan 100 *trees* dan akan divisualisasikan pada Gambar 4.92.

Tabel 4.3 Hasil Pengujian dari Klasifikasi Random Forest 100 *tree*

Metode Seleksi Fitur	Jumlah Fitur Terbaik	Accuracy	Precision	Recall	Waktu Eksekusi
Tanpa Seleksi Fitur	1557	0.98	0.98	0.98	2.38
ANOVA	63	0.88	0.88	0.88	1.32
Mutual Information	70	0.96	0.96	0.96	1.36
Fisher Score	55	0.93	0.93	0.93	1.45
ReliefF	40	0.95	0.95	0.95	1.40
Forward Selection	20	0.95	0.95	0.95	1.47
Backward Elimination	60	0.95	0.95	0.95	1.39
Recursive Feature Elimination	80	0.93	0.93	0.93	1.32
LASSO	9	0.89	0.89	0.89	1.53
Elastic Net	52	0.95	0.95	0.95	1.38



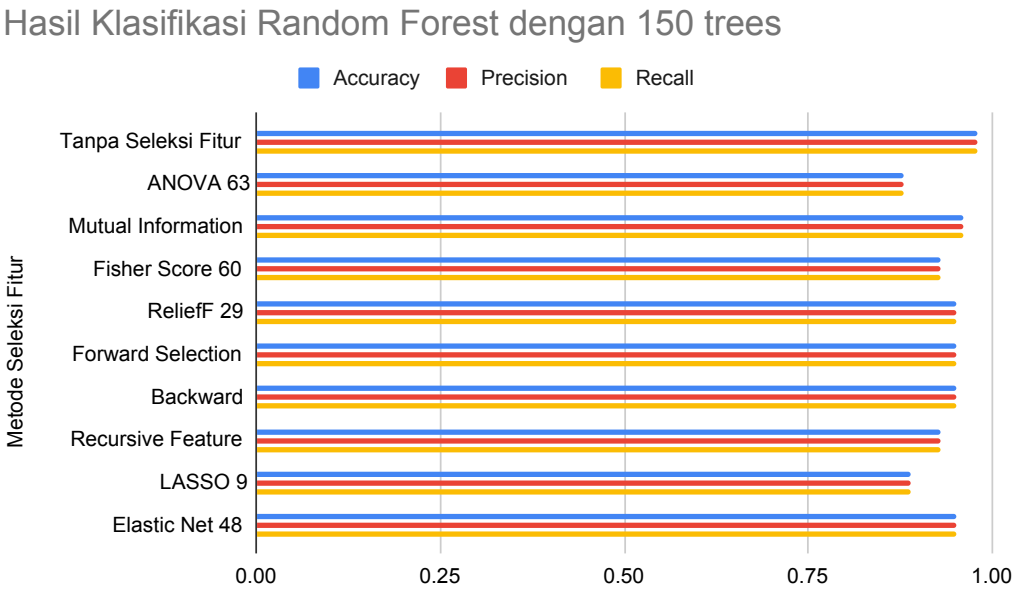
Gambar 4.92 Visualisasi hasil pengujian dari klasifikasi Random Forest 100 *trees*

Berdasarkan Tabel 4.3 dan Gambar 4.96, dapat dilihat bahwa kombinasi 100 *tree* dari Random Forest dan 70 fitur dari *Mutual Information* dapat mencapai performa tertinggi pada skenario pertama ini yaitu dengan menghasilkan 0.96 *accuracy,* 0.96 *recall*, dan 0.96 *precision*. Selain dapat menghasilkan performa tertinggi, waktu yang dibutuhkan metode *Mutual Information* untuk menghasilkan 100 fitur terpenting juga sangat cepat yaitu hanya 5.3 detik seperti yang diperlihatkan pada Tabel 4.1 sebelumnya. Sementara itu, nilai terendah pada skenario klasifikasi ini diperoleh oleh klasifikasi *Random Forest* 100 *trees* menggunakan 63 fitur dari metode ANOVA yang menghasilkan 0.88 *accuracy,* 0.88 *recall*, dan 0.88 *precision*. Selanjutnya Tabel 4.4 akan menunjukkan hasil klasifikasi dari Random Forest yang menggunkaan 150 *trees* dan akan divisualisasikan pada Gambar 4.93.

Tabel 4.4 Hasil Pengujian dari Klasifikasi Random Forest 150 *trees*

Metode Seleksi Fitur	Jumlah Fitur Terbaik	Accuracy	Precision	Recall	Waktu Eksekusi
Tanpa Seleksi Fitur	1557	0.98	0.98	0.98	2.80
ANOVA	63	0.88	0.88	0.88	1.93
Mutual Information	69	0.96	0.96	0.96	2.07

Fisher Score	60	0.93	0.93	0.93	2.08
ReliefF	29	0.95	0.95	0.95	1.99
Forward Selection	20	0.95	0.95	0.95	2.11
Backward Elimination	20	0.95	0.95	0.95	1.99
Recursive Feature Elimination	80	0.93	0.93	0.93	1.93
LASSO	9	0.89	0.89	0.89	2.44
Elastic Net	48	0.95	0.95	0.95	1.95



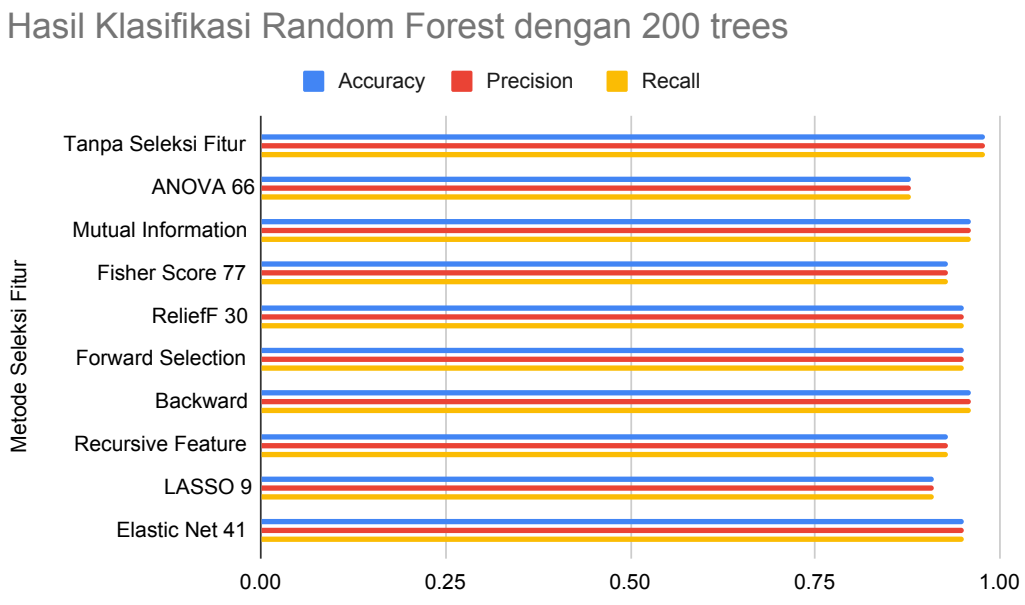
Gambar 4.93 Visualisasi hasil pengujian dari klasifikasi Random Forest 150 *tree*

Berdasarkan Tabel 4.4 dan Gambar 4.93, sama seperti klasifikasi dengan 100 *tree*, dapat dilihat bahwa kombinasi 150 *trees* dari *Random Forest* dan metode *Mutual Information* juga memberikan hasil klasifkasi tertinggi yaitu dengan 0.96 *accuracy,* 0.96 *recall*, dan 0.96 *precision*. Selain itu pada skenario kedua ini hanya dibutuhkan 69 fitur atau lebih sedikit satu fitur dari skenario sebelumnya, namun waktu klasifikasi yang dibutuhkan meningkat. Sementara itu, nilai terendah pada skenario ini juga dihasilkan dari *klasifikasi Random Forest* 150 *tree* yang menggunakan 63 fitur dari ANOVA yang tetap menghasilkan 0.88 *accuracy,* 0.88 *recall*, dan 0.88 *precision*. Dari sini dapat dilihat bahwa peningkatan jumlah *tree* pada *Random Forest* berpengaruh terhadap performa klasifikasi dan waktu eksekusi klasifikasi.

Selanjutnya, Tabel 4.5 akan menunjukkan hasil klasifikasi dari *Random Forest* yang menggunkaan 200 *tree* dan akan divisualisasikan pada Gambar 4.94.

Tabel 4.5 Hasil Pengujian dari Klasifikasi Random Forest 200 *trees*

Metode Seleksi Fitur	Jumlah Fitur Terbaik	Accuracy	Precision	Recall	Waktu Eksekusi
Tanpa Seleksi Fitur	1557	0.98	0.98	0.98	3.20
ANOVA	66	0.88	0.88	0.88	2.59
Mutual Information	72	0.96	0.96	0.96	2.75
Fisher Score	77	0.93	0.93	0.93	2.81
ReliefF	30	0.95	0.95	0.95	2.57
Forward Selection	20	0.95	0.95	0.95	2.77
Backward Elimination	60	0.96	0.96	0.96	2.69
Recursive Feature Elimination	60	0.93	0.93	0.93	2.67
LASSO	9	0.91	0.91	0.91	3.22
Elastic Net	41	0.95	0.95	0.95	2.65

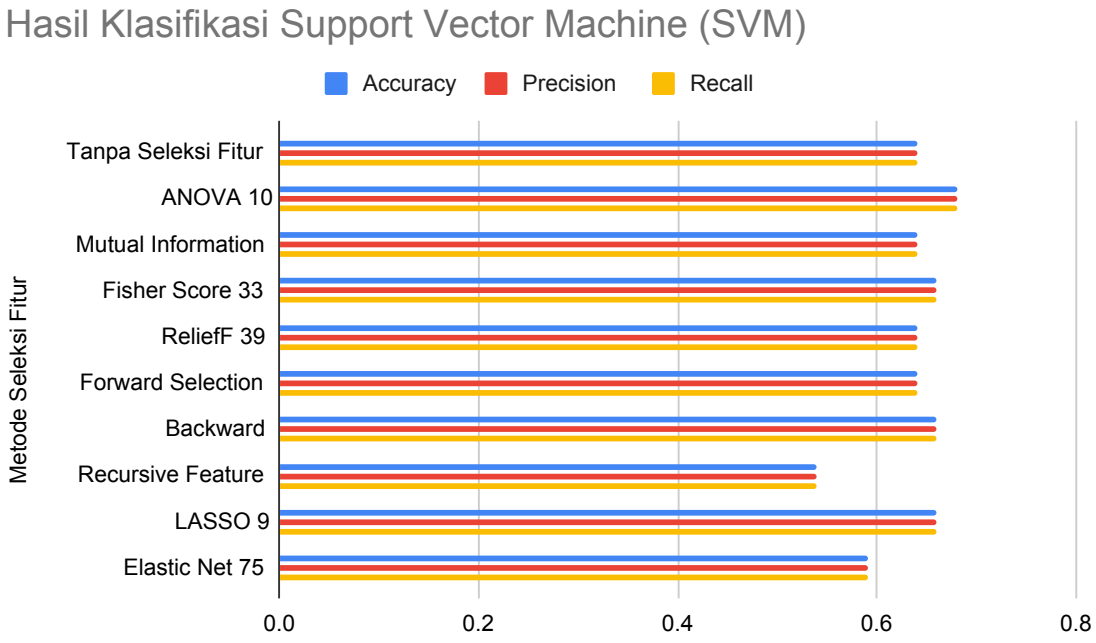


Gambar 4.94 Visualisasi hasil pengujian dari klasifikasi Random Forest 200 *trees*

Berdasarkan Tabel 4.5 dan Gambar 4.94, kombinasi 200 *tree* dari *Random Forest* ternyata membutuhkan jumlah fitur yang lebih banyak dari dua skenario sebelumnya. Hasil tertinggi pada skenario ini didapatkan dengan menggunakan 72 fitur hasil seleksi *Mutual Information*. Dapat dilihat juga bahwa performa yang diperoleh dari *Backward Elimination* pada skenario ini menghasilkan performa yang sama dengan *Mutual Information* bahkan dengan jumlah fitur yang lebih sedikit. Namun berdasarkan Tabel 4.1, waktu yang dibutuhkan untuk memperoleh 60 fitur dari *Backward Elimination* adalah 2658,6 detik, hal ini berbeda sangat jauh dari *Mutual Information* yang hanya membutuhkan waktu 5,3 detik saja. Berdasarkan pertimbangan ini, diputuskan bahwa kombinasi 200 *tree Random Forest* dan *Mutual Information* dapat menghasilkan performa terbaik. Klasifikasi *Random Forest* dengan menggunakan 72 fitur hasil seleksi *Mutual Information* ini menghasilkan 0.96 *accuracy,* 0.96 *recall*, dan 0.96 *precision*. Dari sini dapat dilihat bahwa peningkatan jumlah *tree* pada *Random Forest* tidak selalu meningkatkan nilai performa klasifikasi namun selalu meningkatkan waktu eksekusi klasifikasi. Performa terendah pada skenario ini juga dihasilkan dari klasifikasi *Random Forest* yang menggunakan fitur-fitur dari ANOVA. Namun terdapat perbedaan pada jumlah fitur yang digunakan, dimana dua skenario sebelumnya hanya membutuhkan 63 fitur saja untuk mencapat nilai *accuracy, recall,* dan *precision* sebesar 0.88. Pada skenario ini, dibutuhkan 66 fitur ANOVA untuk mencapai nilai yang sama seperti dua skenario sebelumnya. Setelah melakukan tiga skenario klasifikasi *Random Forest* menggunakan tiga *tree* yang berbeda. Dapat dilihat bahwa secara umum hasil pengujian klasifikasi yang menggunakan data hasil seleksi fitur *Filter* dapat mengungguli hasil klasifikasi dari *Wrapper* dan *Embedded* baik dari segi nilai dan waktu. Dari tiga tabel yang telah dibuat juga dapat dilihat bahwa penambahan jumlah *tree* pada *Random Forest* tidak selalu meningkatkan performa klasifikasi seperti yang terjadi pada metode *Mutual Information.* Penambahan jumlah *tree* ini terkadang dapat berpengaruh dalam mengurangi fitur yang dibutuhkan seperti yang terjadi pada LASSO dan RFE namun juga dapat menambah fitur yang harus digunakan untuk mencapai performa tertinggi seperti pada ANOVA. Selanjutnya Tabel 4.6 akan menunjukkan hasil dari klasfikasi menggunakan metode SVM dan akan divisualisasikan pada Gambar 4.95.

Tabel 4.6 Hasil Pengujian dari Klasifikasi SVM

Metode Seleksi Fitur	Jumlah Fitur Terbaik	Accuracy	Precision	Recall	Waktu Eksekusi
Tanpa Seleksi Fitur	1557	0.64	0.64	0.64	0.68
ANOVA	10	0.68	0.68	0.68	0.10
Mutual Information	19	0.64	0.64	0.64	0.10
Fisher Score	33	0.66	0.66	0.66	0.10
ReliefF	39	0.64	0.64	0.64	0.13
Forward Selection	40	0.64	0.64	0.64	0.13
Backward Elimination	80	0.66	0.66	0.66	0.11
Recursive Feature Elimination	20	0.54	0.54	0.54	0.11
LASSO	9	0.66	0.66	0.66	0.10
Elastic Net	75	0.59	0.59	0.59	0.19



Gambar 4.95 Visualisasi hasil pengujian dari klasifikasi Random Forest 150 *trees*

Pada Tabel 4.6 dan Gambar 4.95 terlihat bahwa fitur-fitur hasil seleksi ANOVA dapat mendapatkan hasil klasifikasi tertinggi saat digunakan bersama klasifikasi SVM yaitu sebesar 0.68 *accuracy,* 0.68 *recall,* dan 0.68 *precision*. Hal ini berbanding terbalik dari tiga klasifikasi sebelumnya dimana ANOVA selalu menjadi yang paling rendah. Walaupun ANOVA menjadi yang terbaik pada skenario ini, namun performa yang dihasilkan masih jauh lebih rendah

daripada tiga skenario sebelumnya. Pada skenario klasifikasi menggunakan SVM ini, seleksi fitur RFE yang menggunakan 20 fitur menghasilkan *accuracy, recall,* dan *precision* yang paling rendah yaitu 0.54 *accuracy,* 0.54 *recall,* dan 0.54 *precision*. Selain itu, hasil klasifikasi SVM dengan fitur hasil seleksi ANOVA mendapatkan hasil yang lebih tinggi daripada klasifikasi SVM yang dilakukan menggunakan 1557 atau seluruh fitur yang hanya menghasilkan 0.64 *accuracy*. Waktu eksekusi yang dibutuhkan SVM juga tidak ada yang mencapai satu detik, hal ini jauh lebih cepat dari waktu eksekusi dari *Random Forest* seperti yang terlihat pada Tabel 4.6. Untuk mempermudah dalam membandingkan hasil dari setiap skenario klasifikasi, Tabel 4.7 akan menampilkan rangkuman dari performa tertinggi dari masing-masing skenario klasifikasi.

Tabel 4.7 Rangkuman Performa Tertinggi dari Setiap Skenario Klasifikasi

Skenario Klasifikasi	Metode Seleksi Fitur	Jumlah Fitur	*accuracy*	*recall*	*precision*	Waktu Eksekusi(s)
Random Forest dengan 100 trees	*Mutual Information*	70	0.96	0.96	0.96	1.36
Random Forest dengan 150 trees	*Mutual Information*	69	0.96	0.96	0.96	2.07
Random Forest dengan 200 trees	*Mutual Information*	72	0.96	0.96	0.96	2.75
Support Vector Machine (SVM)	ANOVA	10	0.68	0.68	0.68	0.10

Dari Tabel 4.7 dapat dilihat bahwa seluruh skenario klasifikasi yang menggunakan *Random Forest* dapat mengungguli performa klasifikasi yang dihasilkan SVM. Seperti yang terlihat pada Tabel 4.7, skenario klasifikasi kedua dapat disebut menjadi skenario terbaik karena dapat menghasilkan performa tertinggi dengan menggunakan 150 *tree* dan 69 fitur dari *Mutual Infrormation* dengan waktu eksekusi yang dibutuhkan tidak berbeda jauh dari skenario satu yang hanya menggunakan 100 *tree*. Dapat dilihat juga bahwa pada seluruh skenario

klasifikasi *Random Forest* dan SVM, seleksi fitur dari kategori *Filter* dapat mengungguli performa yang dihasilkan oleh kategori *Wrapper* dan *Embedded* baik dari segi waktu dan nilai yang didapatkan. Berdasarkan penjabaran di atas, *Random Forest* dan kategori *Filter* dapat disebut sebagai kombinasi *classifier* dan seleksi fitur terbaik dalam klasifikasi data NIRS mangga.

BAB V
KESIMPULAN DAN SARAN

4.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan, diperoleh kesimpulan sebagai berikut:

a. Seluruh klasifikasi yang dilakukan menggunakan *Random Forest* dengan tiga *tree* yang berbeda dapat mengungguli hasil klasifikasi yang dilakukan menggunakan SVM.
b. Hasil yang diperoleh dari masing-masing klasifikasi berbeda tergantung dari model klasifikasi, jumlah *tree* (*pada Random Forest*), metode seleksi fitur, dan jumlah fitur yang digunakan. Seluruh skenario klasifikasi yang menggunakan *Random Forest* dapat mencapai performa tertinggi dengan menggunakan fitur-fitur *Mutual Information*, perbedaan hanya terdapat pada jumlah fitur yang dibutuhkan. Pada klasifikasi *Random Forest* dengan 100 *tree*, performa terbaik diperoleh dari penggunaan 70 fitur dari *Mutual Information* yang menghasilkan 0.96 *accuracy,* 0.96 *recall*, dan 0.96 *precision* sedangkan yang terendah dihasilakan menggunakan 63 fitur dari ANOVA yang menghasilkan 0.88 *accuracy,* 0.88 *recall*, dan 0.88 *precision.* Pada klasifikasi *Random Forest* yang menggunakan 150 *tree,* hasil klasifikasi terbaik diperoleh dengan menggunakan 69 fitur *Mutual Information* yang menghasilkan 0.96 *accuracy,* 0.96 *recall*, dan 0.95 *precision*. Pada skenario klasifikasi ini, hasil terendah juga diperoleh dari penggunaan 63 fitur dari ANOVA yang mendapatkan 0.88 accuracy, 0.88 *recall,* dan 0.88 *precision.* Selanjutnya, klasifikasi *Random Forest* yang menggunakan 200 *trees* mendapatkan performa tertinggi dengan menggunakan 72 fitur dari metode *Mutual Information* dan menghasilkan 0.96 *accuracy,* 0.96 *recall*, dan 0.96 *precision*. Sedangkan performa terendah diperoleh dengan menggunakan 66 fitur hasil seleksi ANOVA yang menghasilkan 0.88 *accuracy*, 0.88 *recall,* dan 0.88 *precision.* Berdasarkan ini, ANOVA menjadi metode seleksi fitur yang menghasilkan performa paling rendah pada tiga skenario klasifikasi menggunakan *Random Forest.*
c. Klasifikasi SVM yang menggunakan RBF kernel memperoleh hasil klasifikasi tertinggi dengan menggunakan 10 fitur hasil seleksi dari ANOVA yaitu sebesar 0.68 *accuracy,* 0.68 *recall,* dan 0.68 *precision*. Hasil terendah pada klasifikasi menggunakan SVM diperoleh dari penggunaan 20 fitur RFE yang hanya menghasilkan 0.54 *accuracy,* 0.54 *recall*, dan 0.54 *precision*.

d. Penambahan jumlah *tree* pada *Random Forest* terkadang dapat berpengaruh dalam mengurangi fitur yang dibutuhkan seperti yang terjadi pada LASSO dan RFE namun juga dapat menambah fitur yang harus digunakan untuk mencapai performa tertinggi seperti pada ANOVA.
e. Skenario kedua dengan 150 *tress* dari *Random Forest* dan 69 fitur dari *Mutual Information* menjadi skenario yang dapat menghasilkan performa terbaik sehingga *Random Forest* dan kategori *Filter* dapat disebut sebagai kombinasi *classifier* dan seleksi fitur terbaik dalam klasifikasi menggunakan data NIRS mangga.

4.2 Saran

Untuk menyempurnakan penelitiain ini, penulis memberikan beberapa saran berupa:

a. Menggunakan metode seleksi fitur lainnya untuk mendapatkan fitur pilihan yang lebih baik.
b. Menambahkan variasi jumlah fitur yang digunakan pada proses *trial and error* seleksi fitur.
c. Menambahkan perbandingan metode klasifika

DAFTAR PUSTAKA

Akkaya, B. (2021). The Effect of Recursive Feature Elimination with Cross-Validation Method on Classification Performance with Different Sizes of Datasets. *IV International Conference on Data Science and Applications (ICONDATA'21)*, *June*, 4–6. https://www.researchgate.net/publication/354253728_The_Effect_of_Recursive_Feature_Elimination_with_Cross-Validation_Method_on_Classification_Performance_with_Different_Sizes_of_Datasets

Algamal, Z. Y., & Lee, M. H. (2015). Regularized logistic regression with adjusted adaptive elastic net for gene selection in high dimensional cancer classification. *Computers in Biology and Medicine*, *67*, 136–145. https://doi.org/10.1016/j.compbiomed.2015.10.008

Amini, F., & Hu, G. (2021). A two-layer feature selection method using Genetic Algorithm and Elastic Net. *Expert Systems with Applications*, *166*(September 2020), 114072. https://doi.org/10.1016/j.eswa.2020.114072

Angga Yuwono, H., Kusuma Wijaya, S., & Prajitno, P. (2020). Feature selection with Lasso for classification of ischemic strokes based on EEG signals. *Journal of Physics: Conference Series*, *1528*(1), 6–12. https://doi.org/10.1088/1742-6596/1528/1/012029

Ary, M., & Rismiati, D. A. F. (2019). Ukuran Akurasi Klasifikasi Penyakit Mesothelioma Menggunakan Algoritma K-Nearest Neighbor dan Backward Elimination. *SATIN - Sains Dan Teknologi Informasi*, *5*(1), 11–18. https://doi.org/10.33372/stn.v5i1.444

Ashfaque, J. M., & Iqbal, A. (2019). *Introduction to Support Vector Machines and Kernel Methods*. *April*, 1–9.

Awad, M., & Khanna, R. (2015). Efficient learning machines: Theories, concepts, and applications for engineers and system designers. *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*, *January*, 1–248. https://doi.org/10.1007/978-1-4302-5990-9

Ayu, P. . (2017). *Pengembangan Model Penentuan Kandungan Kimia Utama Pembentuk Flavor Biji Kopi java Preanger Menggunakan FT NIR*.

Azis, H., Fattah, F., & Putri, P. (2020). *Performa Klasifikasi K-NN dan Cross-validation pada Data Pasien Pengidap Penyakit Jantung*. *12*(2), 81–86.

Baghaee, H. R., Mlakic, D., Nikolovski, S., & Dragicevic, T. (2020). Support Vector Machine-Based Islanding and Grid Fault Detection in Active Distribution Networks. *IEEE Journal of Emerging and Selected Topics in Power Electronics*, *8*(3), 2385–2403. https://doi.org/10.1109/JESTPE.2019.2916621

Bani-Fwaz, M. Z. (2020). Introduction to Spectroscopy. *Introduction to Spectroscopy: Spectroscopic Identification of Organic Compounds*, *19*(October 2016), 1–27. https://doi.org/10.13140/RG.2.2.29741.59360/1

Berrar, D. (2018). Cross-validation. *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*, *1–3*(April), 542–545. https://doi.org/10.1016/B978-0-12-809633-8.20349-X

Bielza, C., & Larrañaga, P. (2020). Data-Driven Computational Neuroscience. In *Data-Driven Computational Neuroscience*. https://doi.org/10.1017/9781108642989

Boonthong, P., Kulkasem, P., Rasmequan, S., Rodtook, A., & Chinnasarn, K. (2015). Fisher feature selection for emotion recognition. *ICSEC 2015 - 19th International Computer Science and Engineering Conference: Hybrid Cloud Computing: A New Approach for Big Data Era*, 2987-13167-3-PB.pdf. https://doi.org/10.1109/ICSEC.2015.7401452

Chandra, B. (2015). Gene Selection Methods for Microarray Data. In *Applied Computing in Medicine and Health*. Elsevier Inc. https://doi.org/10.1016/B978-0-12-803468-2.00003-5

Deng, K. (1999). Chapter 7 Feature Selection. *Phd Thesis*.

Effrosynidis, D., & Arampatzis, A. (2021). An evaluation of feature selection methods for environmental data. *Ecological Informatics*, *61*(April 2020), 101224. https://doi.org/10.1016/j.ecoinf.2021.101224

Fadilah, L. (2018). *Klasifikasi Random Forest pada Data Imbalanced Program Studi Matematika Universitas Islam Negeri Syarif Hidayatullah 2018 / 1439 H Klasifikasi Random Forest*.

Fitriani, I., Basuki, S., & Minarno, A. E. (2020). Seleksi Fitur Relieff Pada Klasifikasi Malware Android Menggunakan Support Vector Machine(SVM). *Jurnal Repositor*, *2*(11), 1529. https://doi.org/10.22219/repositor.v2i11.901

Fonti, V. (2017). Feature Selection using LASSO. *VU Amsterdam*, 1–26.

Fukushima, A., Sugimoto, M., Hiwa, S., & Hiroyasu, T. (2019). Elastic net-based prediction of IFN-β treatment response of patients with multiple sclerosis using time series microarray gene expression profiles. *Scientific Reports*, *9*(1), 1–11. https://doi.org/10.1038/s41598-018-38441-2

Hameed, S. S., Petinrin, O. O., Hashi, A. O., & Saeed, F. (2018). Filter-wrapper combination and embedded feature selection for gene expression data. *International Journal of Advances in Soft Computing and Its Applications*, *10*(1), 90–105.

Han, J., Kamber, M., & Pei, J. (2012). Data mining: Data mining concepts and techniques. In *Proceedings - 2013 International Conference on Machine Intelligence Research and Advancement, ICMIRA 2013*. https://doi.org/10.1109/ICMIRA.2013.45

He, Z., Zhang, X., Cao, Y., Liu, Z., Zhang, B., & Xiaoyan Wang. (2018). *LiteNet : Lightweight Neural Network for Detecting*. 1–18. https://doi.org/10.3390/s18041229

Hendrian, S. (2018). Algoritma Klasifikasi Data Mining Untuk Memprediksi Siswa Dalam Memperoleh Bantuan Dana Pendidikan. *Faktor Exacta*, *11*(3), 266–274. https://doi.org/10.30998/faktorexacta.v11i3.2777

Hoaglin, D. C. (2016). Exploratory data analysis. *Modern Approaches in Solid Earth Sciences*, *12*, 207–219. https://doi.org/10.1007/978-3-319-39264-6_15

Karsito, & Susanti, S. (2019). Klasifikasi Kelayakan Peserta Pengajuan Kredit Rumah Dengan Algoritma Naïve Bayes Di Perumahan Azzura Residencia. *Jurnal Teknologi Pelita Bangsa*, *9*, 43–48.

Kirasich, K., Smith, T., & Sadler, B. (2018). Random Forest vs Logistic Regression: Binary Classification for Heterogeneous Datasets. *Recommended Citation Kirasich*, *1*(3), 9. https://scholar.smu.edu/datasciencereviewhttp://digitalrepository.smu.edu.Availableat:https://scholar.smu.edu/datasciencereview/vol1/iss3/9

Komorowski, M., Marshall, D. C., Salciccioli, J. D., & Crutain, Y. (2016). Secondary Analysis of Electronic Health Records. *Secondary Analysis of Electronic Health Records*, *September*, 1–427. https://doi.org/10.1007/978-3-319-43742-2

Kristiani, S. A. (2019). *Klasifikasi Dokumen Menggunakan Support Vector Machine Dan Mutual Information*. 1–6. https://elibrary.unikom.ac.id/id/eprint/2453/8/UNIKOM_SHERLY AMANDA KRISTIANI_BAB 2.pdf

Kusumiyati, Munawar, A. A., & Suhandy, D. (2020). Prediksi Vitamin C, Total Asam Tertitrasi, Dan Total Padatan Terlarut Pada Buah Mangga Menggunakan Near-Infrared Reflectance Spectroscopy. *Jurnal Teknologi Pertanian*, *21*(3), 145–154. https://doi.org/10.21776/ub.jtp.2020.021.03.1

KUSWANTO, D. (2020). *COMPARATION OF F, BARTLETT AND LEVENE TEST FOR HOMOGENEITY VARIANCE OF HBA1C AND LIPID PROFILE ON PATIENT IN ISLAMIC HOSPITAL SURABAYA 2018-2019*. http://repository.unair.ac.id/id/eprint/108514

Lee, S., KC, B., & Choeh, J. Y. (2020). Comparing performance of ensemble methods in predicting movie box office revenue. *Heliyon*, *6*(6), e04260. https://doi.org/10.1016/j.heliyon.2020.e04260

Lu, S., Shen, S., Huang, J., Dong, M., Lu, J., & Li, W. (2018). Feature selection of laser-induced breakdown spectroscopy data for steel aging estimation. *Spectrochimica Acta - Part B Atomic Spectroscopy*, *150*(October), 49–58. https://doi.org/10.1016/j.sab.2018.10.006

Luqyana Z. T. M, & Husni, P. (2019). Aktivitas Farmakologi Tanaman Mangga (Mangifera indica L.): Review. *Jurnal Farmaka*, *17*(2), 187.

Ma'Ruf, F. A., Adiwijaya, & Wisesty, U. N. (2019). Analysis of the influence of Minimum Redundancy Maximum Relevance as dimensionality reduction method on cancer classification based on microarray data using Support Vector Machine classifier. *Journal of Physics: Conference Series*, *1192*(1), 1499–1506. https://doi.org/10.1088/1742-6596/1192/1/012011

Ma'ruf, H. (2020). *Penerapan Data Science pada Marketing (Customer Churn Prediction-Python) - Part 2*. https://hafizmrf3.medium.com/penerapan-data-science-pada-marketing-customer-churn-prediction-python-part-2-c34cdd4bb790

Malekipirbazari, M., Aksakalli, V., Shafqat, W., & Eberhard, A. (2021). Performance comparison of feature selection and extraction methods with random instance selection. *Expert Systems with Applications*, *179*(December 2020), 115072. https://doi.org/10.1016/j.eswa.2021.115072

Marzatillah, F. (2021). *TEKNOLOGI NIRS UNTUK KLASIFIKASI CAMPURAN MINYAK NILAM HASIL FRAKSINASI DENGAN MINYAK KERUING MENGGUNAKAN METODE PRINCIPAL COMPONENT ANALYSIS*.

Muthukrishnan, R., & Rohini, R. (2017). LASSO: A feature selection technique in predictive modeling for machine learning. *2016 IEEE International Conference on Advances in Computer Applications, ICACA 2016*, 18–20. https://doi.org/10.1109/ICACA.2016.7887916

Naufal, S. A., Adiwijaya, A., & Astuti, W. (2020). Analisis Perbandingan Klasifikasi Support Vector Machine (SVM) dan K-Nearest Neighbors (KNN) untuk Deteksi Kanker dengan Data Microarray. *JURIKOM (Jurnal Riset Komputer)*, *7*(1), 162. https://doi.org/10.30865/jurikom.v7i1.2014

Noshad, M., Zeng, Y., & Hero, A. O. (2019). Scalable Mutual Information Estimation Using Dependence Graphs. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, *2019-May*(Mi), 2962–2966. https://doi.org/10.1109/ICASSP.2019.8683351

Renata, E., & Ayub, M. (2020). Penerapan Metode Random forest untuk Analisis Risiko pada dataset Peer to peer lending. *Jurnal Teknik Informatika Dan Sistem Informasi*, *6*(3), 462–474. https://doi.org/10.28932/jutisi.v6i3.2890

Romadloni, N. T., & Pardede, H. F. (2019). Seleksi Fitur Berbasis Pearson Correlation Untuk Optimasi Opinion Mining Review Pelanggan. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *3*(3), 505–510. https://doi.org/10.29207/resti.v3i3.1189

Rozy, F., Rangkuti, S., Fauzi, M. A., Sari, Y. A., Dewi, E., & Sari, L. (2018). Analisis Sentimen Opini Film Menggunakan Metode Naïve Bayes dengan Ensemble Feature dan Seleksi Fitur Pearson Correlation Coefficient. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer (J-PTIIK) Universitas Brawijaya*, *2*(12), 6354–6361.

Sabett, C., Hafftka, A., Sexton, K., & Spencer, R. G. (2017). L1, Lp, L2, and elastic net penalties for regularization of Gaussian component distributions in magnetic resonance relaxometry. *Concepts in Magnetic Resonance Part A: Bridging Education and Research*, *46A*(2), 1–20. https://doi.org/10.1002/cmr.a.21427

Saha, P., Patikar, S., & Neogy, S. (2020). A correlation - Sequential forward selection based feature selection method for healthcare data analysis. *2020 IEEE International Conference on Computing, Power and Communication Technologies, GUCON 2020*, 69–72. https://doi.org/10.1109/GUCON48875.2020.9231205

Samadi, Wajizah, S., & Munawar, A. A. (2020). Near infrared spectroscopy (NIRS) data analysis for a rapid and simultaneous prediction of feed nutritive parameters. *Data in Brief*, *29*, 105211. https://doi.org/10.1016/j.dib.2020.105211

Samosir, F. V. P., Mustamu, L. P., Anggara, E. D., Wiyogo, A. I., & Widjaja, A. (2021). Exploratory Data Analysis terhadap Kepadatan Penumpang Kereta Rel Listrik. *Jurnal Teknik Informatika Dan Sistem Informasi*, *7*(2), 449–467. https://doi.org/10.28932/jutisi.v7i2.3700

Samuels, P. (2014). *Pearson Correlation*. *April 2014*, 1–5. https://www.researchgate.net/publication/274635640

Sembiring, M. B., Rahmi, D., Maulina, M., Tari, V., Rahmayanti, R., & Suwardi, A. B. (2020). Identifikasi Karakter Morfologi dan Sensoris Kultivar Mangga (Mangifera Indica L.) di Kecamatan Langsa Lama, Aceh, Indonesia. *Jurnal Biologi Tropis*, *20*(2), 179–184. https://doi.org/10.29303/jbt.v20i2.1876

Septiani, W. D. (2017). KOMPARASI METODE KLASIFIKASI DATA MINING ALGORITMA C4.5 DAN NAIVE BAYES UNTUK PREDIKSI PENYAKIT HEPATITIS. *Jurnal Pilar Nusa Mandiri*, *13*(1), 76–84. http://archive.ics.uci.edu/ml/.

Shetye, A. (2019). *Feature Selection with sklearn and Pandas*.

Suppers, A., van Gool, A. J., & Wessels, H. J. C. T. (2018). Integrated chemometrics and statistics to drive successful proteomics biomarker discovery. *Proteomes*, *6*(2). https://doi.org/10.3390/PROTEOMES6020020

Sutoyo, I. (2018). IMPLEMENTASI ALGORITMA DECISION TREE UNTUK KLASIFIKASI DATA PESERTA DIDIK. *Jurnal Ilmiah Ilmu Komputer*, *7*(2), 45–51. https://doi.org/10.35329/jiik.v7i2.203

Taboada, G. L., & Han, L. (2020). Exploratory data analysis and data envelopment analysis of urban rail transit. *Electronics (Switzerland)*, *9*(8), 1–29. https://doi.org/10.3390/electronics9081270

Tang, J., Alelyani, S., & Liu, H. (2014). Feature Selection for Classification: A Review. *Data Classification: Algorithms and Applications*, *Forward st*, 571–605. https://doi.org/10.1201/b17320

Tempola, F., Muhammad, M., & Khairan, A. (2018). Perbandingan Klasifikasi Antara KNN dan Naive Bayes pada Penentuan Status Gunung Berapi dengan K-Fold Cross Validation. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, *5*(5), 577. https://doi.org/10.25126/jtiik.201855983

Tharwat, A. (2018). Classification assessment methods. *Applied Computing and Informatics*, *17*(1), 168–192. https://doi.org/10.1016/j.aci.2018.08.003

Ting, K. M. (2017). Confusion Matrix. *Encyclopedia of Machine Learning and Data Mining*, *October*, 260–260. https://doi.org/10.1007/978-1-4899-7687-1_50

Urbanowicz, R. J., Meeker, M., La Cava, W., Olson, R. S., & Moore, J. H. (2018). Relief-based feature selection: Introduction and review. *Journal of Biomedical Informatics*, *85*, 189–203. https://doi.org/10.1016/j.jbi.2018.07.014

Wahyuni, E. D., Arifiyanti, A. A., & Kustyani, M. (2019). Exploratory Data Analysis dalam Konteks Klasifikasi Data Mining. *Prosiding Nasional Rekayasa Teknologi Industri Dan Informasi XIV Tahun 2019 (ReTII)*, *2019*(November), 263–269. http://journal.itny.ac.id/index.php/ReTII

Wibawa, A. P., Purnama, M. G. A., Akbar, M. F., & Dwiyanto, F. A. (2018). Metode-metode Klasifikasi. *Prosiding Seminar Ilmu Komputer Dan Teknologi Informasi*, *3*(1), 134.

Widagdo, K. A., Adi, K., & Gernowo, R. (2020). Kombinasi Feature Selection Fisher Score dan Principal Component Analysis (PCA) untuk Klasifikasi Cervix Dysplasia. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, *7*(3), 565. https://doi.org/10.25126/jtiik.2020702987

Wiradinata, R. (2019). Modifikasi Instrumen NIR untuk Penentuan Kandungan Kimia Bahan Organik secara Cepat dan Non Destruktif. *Jurnal Keteknikan Pertanian*, *7*(1), 49–56.

Yuan, M., Yang, Z., Huang, G., & Ji, G. (2017). Feature selection by maximizing correlation information for integrated high-dimensional protein data. *Pattern Recognition Letters*, *92*, 17–24. https://doi.org/10.1016/j.patrec.2017.03.011

Zhao, J., Zhou, Y., Zhang, X., & Chen, L. (2016). Part mutual information for quantifying direct associations in networks. *Proceedings of the National Academy of Sciences of the United States of America*, *113*(18), 5130–5135. https://doi.org/10.1073/pnas.1522586113

LAMPIRAN

Seluruh *subset* hasil seleksi fitur:

Kategori Filter:

1. ANOVA:

100 fitur - ['1092.6', '1093.1', '1096.8', '1089.8', '1095.4', '1095.8', '1097.2', '1103.3', '1102.8', '1102.4', '1094.4', '1091.7', '1092.1', '1094', '1089.4', '1091.2', '1098.6', '1096.3', '1103.8', '1094.9', '1093.5', '1107.1', '1100.9', '1100.5', '1086.6', '1087.1', '1104.7', '1101.9', '1099.5', '1100', '1104.2', '1099.1', '1106.6', '1097.7', '1090.3', '1098.2', '1101.4', '1084.4', '1090.8', '1085.3', '1105.2', '1085.7', '1106.1', '1107.5', '1105.6', '1088.9', '1109', '1087.6', '1086.2', '1083.9', '1079.9', '1108.5', '1084.8', '1108', '1080.3', '1088.5', '1079.4', '1082.1', '1088', '1080.8', '1109.4', '1109.9', '1110.4', '1081.7', '1083.5', '1079', '1081.2', '1082.6', '1083', '1110.9', '1077.6', '1074.5', '1078.5', '1074.9', '1112.3', '1112.8', '1078.1', '1076.3', '1077.2', '1075.8', '1114.2', '1076.7', '1111.3', '1074', '1075.4', '1113.2', '1114.7', '1116.1', '1113.7', '1115.6', '1111.8', '1073.2', '1072.7', '1073.6', '1115.2', '1116.6', '1072.3', '1118', '1071.8', '1069.6']

2. Mutual Information

100 fitur - ['1139.7', '1140.2', '1141.7', '1138.2', '1138.7', '1139.2', '1140.7', '1141.2', '1142.2', '1132.2', '1137.7', '1131.7', '1137.2', '1359.6', '1358.9', '1142.7', '1134.2', '1360.3', '1133.7', '1361', '1131.2', '1132.7', '1272.8', '1358.2', '1133.2', '1357.5', '1356.7', '1267.2', '1275.3', '1262.3', '1276', '1272.2', '1270.9', '1143.2', '1267.8', '1136.7', '1273.4', '1266', '1270.3', '1266.6', '1265.4', '1274.7', '1269.1', '1369.6', '1370.4', '1263.5', '1271.6', '1361.7', '1368.9', '1135.7', '1276.6', '1268.5', '1269.7', '1264.7', '1261.7', '1134.7', '1277.2', '1662', '1274.1', '1264.1', '1279.7', '1262.9', '1135.2', '1659.9', '1660.9', '1130.7', '1136.2', '1632.7', '1663.1', '1368.2', '1279.1', '1654.6', '1349.7', '1651.4', '1664.1', '1631.7', '1657.8', '1652.5', '1658.8', '1000.3', '1628.6', '1653.5', '1297.7', '1277.8', '1630.6', '1280.4', '1655.6', '1350.4', '1353.9', '1259.8', '1000.7', '1665.2', '1282.9', '1656.7', '1371.1', '1367.5', '1260.4', '999.9', '1281', '1278.5']

3. Fisher:

100 fitur - ['2500.2', '2497.8', '2495.4', '2493', '2490.6', '2488.2', '2485.8', '2483.5', '2481.1', '2478.7', '2476.3', '2474', '2471.6', '2469.3', '2466.9', '2464.6', '2462.2', '2459.9', '2457.6', '2455.2', '2452.9', '2448.3', '2427.7', '2414.1', '2405.1', '2398.5', '2389.6', '2380.8', '2372.1', '2365.6', '2357', '2348.5', '2340', '2331.6', '2321.2', '2314.9', '2308.8', '2302.6', '2300.6', '2288.4', '2280.3', '2284.3', '2278.3', '2266.4', '2268.4', '2274.3', '2270.3', '2264.4', '2272.3', '2260.4', '2256.5', '2258.5', '2250.6', '2240.9', '2233.2', '2237', '2235.1', '2239', '2242.8', '2244.8', '2246.7', '2248.7', '2262.4', '2306.7', '2342.1', '2361.3', '2359.2', '2346.4', '2335.8', '2333.7', '2323.2', '2319.1', '2312.9', '2310.8', '2304.7', '2296.5', '2292.4', '2286.4', '2282.3', '2298.5', '2327.4', '2352.8', '2387.4', '2439.1', '2450.6', '2436.8', '2418.6', '2409.6', '2402.9', '2394', '2385.2', '2374.3', '2363.5', '2344.2', '2325.3', '2294.5', '2252.6', '2231.3', '2225.5', '2221.7']

4. ReliefF:

100 fitur - ['1800.5', '1838.8', '1942.1', '2481.1', '1442', '1547', '1788.1', '2329.5', '1895.3', '999.9', '1000.3', '1000.7', '1001.1', '1001.4', '1001.8', '1002.2', '1002.6', '1003', '1003.4', '1003.8',

'1004.2', '1004.5', '1004.9', '1005.3', '1005.7', '1006.1', '1006.5', '1006.9', '1007.3', '1007.7', '1008.1', '1008.5', '1008.8', '1009.2', '1009.6', '1010', '1010.4', '1010.8', '1011.2', '1011.6', '1012', '1012.4', '1012.8', '1013.2', '1013.6', '1014', '1014.4', '1014.8', '1015.2', '1015.6', '1016', '1016.4', '1016.8', '1017.2', '1017.6', '1018', '1018.4', '1018.8', '1019.2', '1019.6', '1020', '1020.4', '1020.8', '1021.2', '1021.6', '1022', '1022.4', '1022.8', '1023.2', '1023.6', '1024', '1024.4', '1024.8', '1025.2', '1025.6', '1026', '1026.4', '1026.8', '1027.2', '1027.6', '1028', '1028.5', '1028.9', '1029.3', '1029.7', '1030.1', '1030.5', '1030.9', '1031.3', '1031.7', '1032.1', '1032.6', '1033', '1033.4', '1033.8', '1034.2', '1034.6', '1035', '1035.4', '1035.9']

Kategori *Wrapper*:

1. Sequential Forward Selection

20 fitur - ['1001.1', '1001.4', '1018.4', '1031.7', '1038.3', '1052.2', '1080.8', '1120', '1129.7', '1153.9', '1175.8', '1234.6', '1549.8', '1580.9', '1616.4', '1629.6', '1712.5', '1909.2', '1958.3', '1977.7']

40 fitur - ['1000.3', '1004.9', '1007.7', '1008.8', '1010.8', '1012.8', '1013.6', '1016', '1018.4', '1027.6', '1036.7', '1039.2', '1040.4', '1042.1', '1072.7', '1085.3', '1096.8', '1108', '1109.9', '1128.7', '1129.7', '1132.2', '1152.3', '1164.8', '1172.1', '1174.2', '1174.8', '1204.8', '1227.6', '1249.5', '1326.9', '1331.7', '1638.9', '1712.5', '1819.5', '1900.8', '1905', '1906.4', '1962.7', '2414.1']

60 fitur - ['999.9', '1001.4', '1002.2', '1007.3', '1007.7', '1008.5', '1010.4', '1014', '1014.8', '1016', '1024', '1028.9', '1040', '1040.4', '1047.6', '1055.7', '1068.3', '1074.5', '1076.3', '1085.3', '1087.1', '1094.4', '1097.2', '1105.2', '1143.2', '1150.3', '1159.5', '1161.6', '1162.7', '1164.2', '1166.3', '1183.9', '1185.5', '1204.8', '1211', '1225.9', '1241.1', '1256.2', '1277.2', '1282.9', '1308.1', '1340.6', '1352.5', '1357.5', '1374.7', '1409.1', '1562.8', '1573.3', '1683.6', '1712.5', '1796.8', '1820.7', '1900.8', '1961.2', '1974.7', '2016.1', '2123.5', '2380.8', '2391.8', '2448.3']

80 fitur - ['1001.1', '1002.2', '1006.1', '1008.8', '1010.8', '1020', '1023.6', '1024.8', '1035.4', '1039.2', '1039.6', '1042.5', '1042.9', '1053.5', '1054.4', '1055.7', '1059.6', '1063.9', '1068.3', '1075.4', '1088.9', '1089.8', '1100.5', '1104.7', '1118.5', '1120.5', '1127.3', '1133.2', '1135.2', '1151.3', '1154.4', '1157.5', '1160.1', '1161.6', '1164.8', '1171.1', '1171.6', '1179.1', '1179.6', '1181.7', '1219.5', '1257.4', '1267.8', '1273.4', '1277.2', '1283.5', '1292.5', '1302.2', '1317.4', '1331', '1338.5', '1343.4', '1352.5', '1370.4', '1376.2', '1397.7', '1400.7', '1409.1', '1412.2', '1464.8', '1478.2', '1545.1', '1583.8', '1615.4', '1650.4', '1712.5', '1715.9', '1822', '1902.2', '1927.7', '1955.3', '1964.2', '1967.2', '1999', '2197.2', '2206.6', '2361.3', '2407.4', '2471.6', '2485.8']

100 fitur - ['1000.3', '1000.7', '1001.8', '1002.2', '1007.3', '1008.1', '1008.8', '1009.2', '1010.8', '1011.6', '1015.6', '1020', '1021.6', '1025.6', '1027.6', '1028', '1031.3', '1031.7', '1033.4', '1034.2', '1035', '1035.4', '1035.9', '1036.3', '1038.8', '1039.6', '1042.1', '1050.1', '1054', '1056.1', '1070.1', '1077.2', '1079.4', '1082.1', '1098.6', '1102.8', '1110.9', '1125.3', '1127.3', '1128.3', '1131.7', '1133.7', '1140.7', '1141.7', '1144.7', '1153.4', '1160.6', '1161.6', '1164.8', '1173.2', '1179.1', '1190.4', '1198.1', '1207', '1207.6', '1219', '1226.5', '1227.6', '1242.9', '1245.3', '1247.1', '1260.4', '1261.1', '1264.7', '1274.1', '1277.8', '1327.6', '1380.6', '1419.9', '1426.9', '1427.7', '1445.2', '1472.3', '1667.4', '1686.9', '1699', '1712.5', '1714.8', '1902.2', '1906.4', '1910.6', '1917.7',

'1936.3', '1939.2', '1961.2', '1962.7', '1964.2', '2022.4', '2035.1', '2096', '2099.4', '2137.5', '2151.6', '2167.8', '2217.9', '2274.3', '2280.3', '2335.8', '2455.2', '2488.2']

2. Backward Elimination

20 fitur - ['1138.2', '1139.2', '1132.2', '1358.9', '1134.2', '1361', '1276', '1270.9', '1273.4', '1266.6', '1263.5', '1269.7', '1662', '1657.8', '1000.3', '1282.9', '1629.6', '1669.5', '1002.2', '1678.1']

40 fitur - ['1138.2', '1138.7', '1139.2', '1140.7', '1132.2', '1131.7', '1134.2', '1360.3', '1361', '1131.2', '1272.8', '1358.2', '1262.3', '1270.9', '1273.4', '1266', '1274.7', '1369.6', '1269.7', '1279.7', '1660.9', '1279.1', '1349.7', '1651.4', '1657.8', '1000.3', '1353.9', '1278.5', '1130.2', '1673.8', '1298.3', '1258.6', '1701.3', '1178.5', '1681.4', '1318.1', '1321.5', '1364.6', '1688', '1258']

60 fitur - ['1140.2', '1141.7', '1138.2', '1139.2', '1141.2', '1142.2', '1132.2', '1131.7', '1137.2', '1359.6', '1142.7', '1134.2', '1133.7', '1131.2', '1272.8', '1357.5', '1356.7', '1262.3', '1143.2', '1136.7', '1273.4', '1266', '1265.4', '1370.4', '1368.9', '1135.7', '1268.5', '1264.7', '1261.7', '1134.7', '1277.2', '1274.1', '1279.7', '1659.9', '1660.9', '1130.7', '1632.7', '1631.7', '1657.8', '1665.2', '1282.9', '999.9', '1278.5', '1294.4', '1282.3', '1354.6', '1690.2', '1363.9', '1172.7', '1670.6', '1178', '1316.1', '1349', '1178.5', '1299', '1284.8', '1338.5', '1363.2', '1293.8', '1339.9']

80 fitur - ['1139.7', '1140.2', '1141.7', '1138.2', '1140.7', '1141.2', '1142.2', '1137.7', '1131.7', '1137.2', '1142.7', '1133.7', '1361', '1131.2', '1272.8', '1133.2', '1356.7', '1275.3', '1262.3', '1276', '1136.7', '1273.4', '1266', '1270.3', '1266.6', '1274.7', '1269.1', '1369.6', '1263.5', '1271.6', '1361.7', '1135.7', '1276.6', '1268.5', '1134.7', '1662', '1274.1', '1264.1', '1279.7', '1135.2', '1659.9', '1660.9', '1632.7', '1663.1', '1368.2', '1654.6', '1651.4', '1657.8', '1652.5', '1628.6', '1653.5', '1000.7', '1665.2', '1282.9', '1371.1', '1650.4', '1356', '1001.8', '1668.4', '1694.6', '1001.1', '1693.5', '1373.3', '1296.4', '1348.3', '1330.3', '1299', '1674.9', '1318.1', '1686.9', '1321.5', '1331', '1351.1', '1648.3', '1338.5', '1316.8', '1322.1', '1646.2', '1684.7', '1203.1']

100 fitur - ['1140.2', '1141.7', '1138.2', '1139.2', '1141.2', '1142.2', '1132.2', '1137.7', '1137.2', '1359.6', '1358.9', '1360.3', '1133.7', '1358.2', '1133.2', '1356.7', '1275.3', '1262.3', '1267.8', '1266.6', '1369.6', '1263.5', '1361.7', '1368.9', '1135.7', '1276.6', '1268.5', '1269.7', '1264.7', '1261.7', '1134.7', '1277.2', '1274.1', '1279.7', '1135.2', '1659.9', '1130.7', '1632.7', '1368.2', '1279.1', '1654.6', '1349.7', '1658.8', '1000.3', '1628.6', '1297.7', '1277.8', '1280.4', '1353.9', '1000.7', '1282.9', '1371.1', '1260.4', '1281', '1278.5', '1282.3', '1626.6', '1629.6', '1339.2', '1666.3', '1627.6', '1001.4', '1649.3', '1694.6', '1001.1', '1667.4', '1693.5', '1172.7', '1348.3', '1330.3', '1002.2', '1670.6', '1355.3', '1130.2', '1004.2', '1673.8', '1366', '1178', '1701.3', '1178.5', '1346.9', '1299', '1681.4', '1329.6', '1686.9', '1321.5', '1331', '1351.1', '1177.4', '1173.2', '1679.2', '1316.8', '1322.1', '1844', '1285.4', '1207', '1647.2', '1612.4', '1624.5', '1153.4']

3. Recursive Feature Elimination (RFE)

20 fitur - ['1088', '1125.3', '1126.8', '1129.7', '1133.7', '1135.2', '1136.2', '1136.7', '1138.7', '1139.2', '1141.2', '1329.6', '1335.8', '1874.7', '1876.1', '1877.4', '1878.8', '1882.9', '2038.3', '2111.3']

40 fitur - ['1084.4', '1084.8', '1088', '1127.3', '1128.3', '1129.7', '1130.2', '1132.7', '1133.2', '1134.2', '1134.7', '1135.2', '1135.7', '1137.2', '1138.7', '1139.2', '1141.2', '1141.7', '1142.7', '1145.2', '1257.4', '1282.9', '1295.1', '1304.2', '1609.4', '1638.9', '1874.7', '1876.1', '1878.8',

'1881.5', '1885.6', '1889.7', '1900.8', '2038.3', '2041.5', '2043.1', '2044.7', '2106.2', '2178.8', '2206.6']

60 fitur - ['1086.6', '1087.1', '1091.7', '1093.1', '1126.3', '1127.3', '1128.3', '1129.7', '1130.2', '1130.7', '1131.2', '1131.7', '1132.2', '1132.7', '1133.2', '1133.7', '1134.2', '1135.7', '1136.2', '1137.7', '1138.2', '1138.7', '1140.7', '1141.7', '1142.7', '1143.2', '1145.2', '1199.2', '1253.7', '1269.7', '1301.6', '1314.8', '1322.1', '1322.8', '1693.5', '1710.2', '1824.6', '1872', '1876.1', '1878.8', '1880.2', '1881.5', '1905', '1934.9', '1937.8', '1959.7', '2031.9', '2038.3', '2039.9', '2065.9', '2087.5', '2096', '2113.1', '2118.2', '2121.7', '2123.5', '2186.1', '2225.5', '2239', '2268.4']

80 fitur - ['1090.3', '1091.7', '1092.1', '1092.6', '1095.4', '1096.8', '1098.2', '1100', '1106.6', '1109', '1118.5', '1126.3', '1126.8', '1127.8', '1130.7', '1131.2', '1132.2', '1132.7', '1133.2', '1133.7', '1135.7', '1136.7', '1137.2', '1138.7', '1139.2', '1139.7', '1142.7', '1144.2', '1151.8', '1153.9', '1208.7', '1235.8', '1248.9', '1258', '1265.4', '1285.4', '1289.3', '1295.1', '1298.3', '1318.8', '1334.4', '1387.2', '1659.9', '1670.6', '1695.7', '1712.5', '1872', '1876.1', '1878.8', '1880.2', '1882.9', '1888.4', '1900.8', '1905', '1919.1', '1924.8', '1936.3', '1974.7', '1980.7', '1991.3', '2033.5', '2035.1', '2043.1', '2044.7', '2048', '2054.5', '2075.8', '2101.1', '2104.5', '2123.5', '2142.8', '2144.5', '2158.8', '2166', '2176.9', '2193.5', '2202.8', '2210.3', '2231.3', '2246.7']

100 fitur - ['1081.7', '1082.1', '1084.8', '1085.3', '1086.6', '1088', '1088.9', '1089.4', '1091.7', '1100', '1101.4', '1102.4', '1103.3', '1113.2', '1122.9', '1125.8', '1126.8', '1128.3', '1129.7', '1130.2', '1130.7', '1131.2', '1132.2', '1133.2', '1134.7', '1135.2', '1135.7', '1136.2', '1136.7', '1137.7', '1138.2', '1139.2', '1139.7', '1141.2', '1143.2', '1160.6', '1207', '1259.8', '1260.4', '1261.7', '1262.3', '1264.1', '1275.3', '1284.2', '1286.7', '1289.3', '1291.2', '1298.3', '1320.8', '1326.2', '1326.9', '1327.6', '1613.4', '1618.4', '1689.1', '1706.9', '1736.6', '1877.4', '1878.8', '1882.9', '1885.6', '1887', '1891.1', '1895.3', '1903.6', '1910.6', '1917.7', '1922', '1936.3', '1967.2', '1983.7', '1986.8', '1988.3', '1992.9', '1995.9', '2017.7', '2020.8', '2030.3', '2036.7', '2038.3', '2039.9', '2041.5', '2044.7', '2062.6', '2064.3', '2077.5', '2090.9', '2106.2', '2111.3', '2120', '2128.7', '2130.4', '2137.5', '2162.4', '2164.2', '2166', '2173.3', '2175.1', '2189.8', '2204.7']

Kategori *Embedded*:

1. Elastic Net:

1009 fitur - '999.9', '1000.3', '1000.7', '1079.4', '1079.9', '1080.8', '1082.1', '1082.6', '1083.9', '1084.4', '1085.3', '1085.7', '1086.2', '1086.6', '1087.1', '1088.9', '1089.4', '1089.8', '1091.2', '1091.7', '1092.1', '1092.6', '1093.1', '1093.5', '1094', '1094.4', '1094.9', '1095.4', '1095.8', '1096.3', '1096.8', '1097.2', '1098.2', '1098.6', '1099.1', '1099.5', '1100', '1100.5', '1100.9', '1101.4', '1101.9', '1102.4', '1102.8', '1103.3', '1103.8', '1104.2', '1104.7', '1105.6', '1106.1', '1106.6', '1107.1', '1108.5', '1109', '1109.4', '1109.9', '1110.4', '1110.9', '1111.8', '1112.3', '1112.8', '1113.2', '1113.7', '1114.2', '1114.7', '1115.2', '1115.6', '1116.1', '1117.6', '1118', '1119', '1119.5', '1120', '1120.5', '1120.9', '1121.4', '1121.9', '1122.9', '1123.4', '1124.3', '1124.8', '1125.3', '1125.8', '1126.3', '1126.8', '1127.3', '1127.8', '1128.3', '1128.7', '1129.2', '1129.7', '1130.2', '1130.7', '1131.2', '1131.7', '1132.2', '1132.7', '1133.2', '1133.7', '1134.2', '1134.7', '1135.2', '1135.7', '1136.2', '1136.7', '1137.2', '1137.7', '1138.2', '1138.7', '1139.2', '1139.7', '1140.2', '1140.7', '1141.2', '1141.7', '1142.2', '1142.7', '1143.2', '1143.7', '1144.2', '1144.7',

'1145.2', '1145.7', '1146.2', '1146.7', '1147.2', '1147.7', '1148.2', '1148.8', '1149.3', '1149.8',
'1150.3', '1150.8', '1151.3', '1151.8', '1152.3', '1152.8', '1153.4', '1153.9', '1154.4', '1154.9',
'1155.4', '1155.9', '1156.4', '1157', '1157.5', '1158', '1158.5', '1159', '1159.5', '1160.1', '1160.6',
'1161.1', '1161.6', '1162.7', '1163.2', '1163.7', '1165.3', '1165.8', '1166.3', '1166.8', '1167.4',
'1167.9', '1168.4', '1169', '1170', '1170.5', '1171.1', '1171.6', '1172.1', '1173.2', '1173.7', '1174.2',
'1175.3', '1175.8', '1176.4', '1176.9', '1177.4', '1178', '1178.5', '1179.1', '1179.6', '1180.1',
'1180.7', '1181.2', '1181.7', '1182.8', '1183.4', '1185', '1185.5', '1186.1', '1186.6', '1187.1',
'1188.2', '1188.8', '1189.3', '1191.5', '1192.1', '1192.6', '1193.2', '1193.7', '1194.8', '1197',
'1197.6', '1199.2', '1199.8', '1205.4', '1205.9', '1206.5', '1207', '1208.7', '1209.3', '1210.4', '1211',
'1211.6', '1212.1', '1213.8', '1214.4', '1216.7', '1217.2', '1217.8', '1218.4', '1219', '1219.5',
'1220.1', '1220.7', '1222.4', '1223', '1224.1', '1224.7', '1225.3', '1225.9', '1226.5', '1227', '1228.2',
'1228.8', '1229.4', '1229.9', '1230.5', '1231.1', '1232.9', '1233.5', '1234', '1234.6', '1235.2',
'1235.8', '1236.4', '1239.4', '1239.9', '1241.1', '1241.7', '1242.3', '1242.9', '1244.7', '1247.1',
'1247.7', '1248.3', '1249.5', '1250.1', '1250.7', '1251.3', '1251.9', '1252.5', '1253.7', '1254.3',
'1256.2', '1256.8', '1257.4', '1258', '1259.2', '1259.8', '1261.7', '1262.3', '1262.9', '1264.7',
'1265.4', '1266', '1267.2', '1267.8', '1268.5', '1269.7', '1270.3', '1270.9', '1271.6', '1272.2',
'1272.8', '1273.4', '1275.3', '1276', '1277.2', '1277.8', '1278.5', '1279.1', '1279.7', '1280.4', '1281',
'1281.6', '1282.3', '1284.2', '1284.8', '1285.4', '1286.1', '1286.7', '1288', '1288.6', '1289.3',
'1289.9', '1290.6', '1291.2', '1291.8', '1292.5', '1293.1', '1293.8', '1294.4', '1295.1', '1295.7',
'1296.4', '1297', '1297.7', '1298.3', '1299', '1299.6', '1300.3', '1300.9', '1301.6', '1302.2', '1302.9',
'1303.5', '1304.2', '1304.8', '1305.5', '1306.2', '1306.8', '1307.5', '1308.1', '1308.8', '1309.5',
'1310.1', '1310.8', '1311.4', '1312.1', '1312.8', '1313.4', '1314.1', '1314.8', '1315.4', '1316.1',
'1316.8', '1317.4', '1318.1', '1318.8', '1319.5', '1320.1', '1320.8', '1321.5', '1322.1', '1322.8',
'1323.5', '1324.2', '1324.9', '1325.5', '1326.2', '1326.9', '1327.6', '1328.2', '1328.9', '1329.6',
'1330.3', '1331', '1331.7', '1332.3', '1333', '1333.7', '1334.4', '1335.1', '1335.8', '1336.5', '1337.1',
'1337.8', '1338.5', '1339.2', '1339.9', '1340.6', '1341.3', '1342', '1342.7', '1343.4', '1344.1',
'1344.8', '1345.5', '1346.2', '1346.9', '1347.6', '1348.3', '1349', '1349.7', '1350.4', '1351.1',
'1351.8', '1352.5', '1353.2', '1353.9', '1354.6', '1355.3', '1356', '1356.7', '1357.5', '1358.2',
'1358.9', '1359.6', '1360.3', '1361', '1362.4', '1363.2', '1363.9', '1365.3', '1366', '1366.8', '1367.5',
'1368.2', '1368.9', '1369.6', '1370.4', '1371.1', '1371.8', '1372.5', '1373.3', '1374', '1374.7',
'1375.5', '1376.2', '1376.9', '1377.6', '1378.4', '1379.1', '1379.8', '1380.6', '1381.3', '1382.1',
'1382.8', '1383.5', '1384.3', '1385', '1385.7', '1386.5', '1387.2', '1388', '1388.7', '1389.5', '1390.2',
'1391', '1391.7', '1392.4', '1393.2', '1393.9', '1394.7', '1395.4', '1396.2', '1396.9', '1397.7',
'1398.5', '1399.2', '1400', '1400.7', '1401.5', '1402.2', '1403', '1403.8', '1404.5', '1405.3', '1406',
'1406.8', '1407.6', '1408.3', '1409.1', '1409.9', '1410.6', '1411.4', '1412.2', '1412.9', '1413.7',
'1414.5', '1415.2', '1416', '1416.8', '1417.6', '1418.3', '1419.1', '1419.9', '1420.7', '1421.5',
'1422.2', '1423', '1423.8', '1424.6', '1425.4', '1426.1', '1426.9', '1427.7', '1428.5', '1429.3',
'1430.1', '1430.9', '1431.7', '1432.4', '1433.2', '1434', '1434.8', '1435.6', '1436.4', '1437.2', '1438',
'1438.8', '1439.6', '1440.4', '1441.2', '1442', '1442.8', '1443.6', '1444.4', '1445.2', '1446', '1446.8',
'1447.6', '1448.5', '1449.3', '1450.1', '1450.9', '1451.7', '1457.4', '1458.2', '1459.1', '1459.9',
'1493.5', '1494.4', '1495.2', '1496.1', '1497', '1497.8', '1498.7', '1499.6', '1500.4', '1501.3',
'1502.2', '1503', '1503.9', '1504.8', '1505.7', '1506.5', '1507.4', '1508.3', '1509.2', '1510', '1510.9',
'1511.8', '1512.7', '1513.6', '1514.4', '1515.3', '1516.2', '1517.1', '1518', '1518.9', '1519.8',
'1520.7', '1521.6', '1522.5', '1523.3', '1524.2', '1525.1', '1526', '1526.9', '1527.8', '1528.7',
'1529.6', '1530.5', '1531.4', '1532.3', '1533.3', '1534.2', '1535.1', '1536', '1536.9', '1537.8',
'1538.7', '1539.6', '1540.5', '1541.5', '1542.4', '1543.3', '1544.2', '1545.1', '1546.1', '1547',
'1547.9', '1548.8', '1549.8', '1550.7', '1551.6', '1552.5', '1553.5', '1554.4', '1555.3', '1556.3',
'1557.2', '1558.1', '1559.1', '1560', '1560.9', '1561.9', '1562.8', '1563.8', '1564.7', '1565.7',
'1566.6', '1567.6', '1568.5', '1569.5', '1570.4', '1571.4', '1572.3', '1573.3', '1574.2', '1575.2',

'1576.1', '1577.1', '1578', '1579', '1580', '1580.9', '1581.9', '1582.9', '1583.8', '1584.8', '1585.8', '1586.7', '1587.7', '1588.7', '1589.7', '1590.6', '1591.6', '1592.6', '1593.6', '1594.5', '1595.5', '1596.5', '1597.5', '1598.5', '1599.5', '1600.5', '1601.4', '1602.4', '1603.4', '1604.4', '1605.4', '1606.4', '1607.4', '1608.4', '1609.4', '1610.4', '1611.4', '1612.4', '1613.4', '1614.4', '1615.4', '1616.4', '1617.4', '1618.4', '1619.4', '1620.5', '1621.5', '1622.5', '1623.5', '1624.5', '1625.5', '1626.6', '1627.6', '1628.6', '1629.6', '1630.6', '1631.7', '1632.7', '1633.7', '1634.8', '1635.8', '1636.8', '1637.9', '1638.9', '1639.9', '1641', '1642', '1643.1', '1644.1', '1645.1', '1646.2', '1647.2', '1648.3', '1649.3', '1650.4', '1651.4', '1652.5', '1653.5', '1654.6', '1655.6', '1656.7', '1657.8', '1658.8', '1659.9', '1660.9', '1662', '1663.1', '1664.1', '1665.2', '1666.3', '1667.4', '1668.4', '1669.5', '1670.6', '1672.7', '1673.8', '1674.9', '1699', '1700.2', '1701.3', '1702.4', '1703.5', '1704.6', '1705.7', '1706.9', '1708', '1709.1', '1710.2', '1711.4', '1712.5', '1713.6', '1714.8', '1715.9', '1717', '1718.2', '1719.3', '1720.5', '1721.6', '1722.7', '1723.9', '1725', '1727.3', '1728.5', '1729.6', '1768.6', '1769.8', '1771', '1772.2', '1773.4', '1774.6', '1775.8', '1777.1', '1778.3', '1779.5', '1780.7', '1781.9', '1783.2', '1784.4', '1785.6', '1788.1', '1789.3', '1790.6', '1833.6', '1834.9', '1836.2', '1837.5', '1838.8', '1840.1', '1841.4', '1842.7', '1844', '1845.4', '1846.7', '1848', '1849.3', '1850.6', '1852', '1853.3', '1854.6', '1855.9', '1857.3', '1858.6', '1859.9', '1861.3', '1862.6', '1863.9', '1865.3', '1866.6', '1868', '1869.3', '1870.7', '1872', '1873.4', '1874.7', '1876.1', '1877.4', '1878.8', '1880.2', '1881.5', '1882.9', '1884.3', '1885.6', '1887', '1888.4', '1889.7', '1891.1', '1900.8', '1902.2', '1903.6', '1905', '1906.4', '1907.8', '1909.2', '1910.6', '1912', '1913.5', '1914.9', '1916.3', '1917.7', '1919.1', '1920.5', '1922', '1923.4', '1924.8', '1926.3', '1964.2', '1965.7', '1967.2', '1968.7', '1970.2', '1971.7', '1973.2', '1974.7', '1976.2', '1977.7', '1979.2', '1980.7', '1982.2', '1983.7', '1985.2', '1986.8', '1988.3', '1989.8', '1991.3', '1992.9', '1994.4', '1995.9', '1997.5', '1999', '2000.6', '2002.1', '2003.7', '2005.2', '2006.8', '2008.3', '2009.9', '2011.4', '2013', '2014.6', '2016.1', '2017.7', '2019.3', '2020.8', '2022.4', '2024', '2062.6', '2064.3', '2065.9', '2067.6', '2069.2', '2070.9', '2072.5', '2074.2', '2075.8', '2077.5', '2079.2', '2080.8', '2082.5', '2084.2', '2085.9', '2087.5', '2089.2', '2090.9', '2092.6', '2094.3', '2096', '2097.7', '2099.4', '2101.1', '2102.8', '2104.5', '2106.2', '2107.9', '2109.6', '2111.3', '2113.1', '2114.8', '2116.5', '2118.2', '2120', '2121.7', '2123.5', '2125.2', '2126.9', '2128.7', '2130.4', '2132.2', '2133.9', '2135.7', '2137.5', '2139.2', '2141', '2142.8', '2144.5', '2146.3', '2148.1', '2149.9', '2151.6', '2153.4', '2155.2', '2157', '2158.8', '2160.6', '2162.4', '2164.2', '2166', '2167.8', '2169.7', '2171.5', '2173.3', '2175.1', '2176.9', '2178.8', '2180.6', '2182.4', '2184.3', '2186.1', '2189.8', '2195.4', '2197.2', '2199.1', '2244.8', '2246.7', '2248.7', '2250.6', '2252.6', '2254.6', '2256.5', '2258.5', '2260.4', '2262.4', '2264.4', '2266.4', '2268.4', '2270.3', '2272.3', '2274.3', '2276.3', '2278.3', '2280.3', '2282.3', '2284.3', '2286.4', '2288.4', '2290.4', '2292.4', '2294.5', '2296.5', '2298.5', '2300.6', '2302.6', '2304.7', '2323.2', '2325.3', '2327.4', '2357', '2359.2', '2361.3', '2363.5', '2365.6', '2367.8', '2370', '2372.1', '2374.3', '2376.5', '2394', '2396.2', '2398.5', '2400.7', '2402.9', '2405.1', '2407.4', '2409.6', '2466.9', '2469.3', '2471.6', '2474', '2476.3', '2478.7', '2481.1', '2483.5', '2485.8', '2488.2', '2493', '2495.4', '2500.2'.

2. LASSO:

9 fitur - '1323.5', '1326.2', '1326.9', '1328.9', '1331', '1331.7', '1337.1', '1397.7', '1398.5']