# BAB II
# LANDASAN TEORI

## 2.1 Tinjauan Pustaka

Penyakit Jantung Koroner (PJK) merupakan kondisi yang ditandai oleh penyumbatan pembuluh darah jantung atau arteri koroner akibat timbunan lemak. Akumulasi lemak yang berkelanjutan ini mengakibatkan penyempitan arteri, yang pada gilirannya mengurangi suplai darah ke jantung. Gejala PJK dapat timbul sebagai akibat dari proses ini. Penyakit ini diduga kuat dipicu oleh proses peradangan dan penumpukan kolesterol di dalam pembuluh darah. Dinding arteri koroner menjadi sempit atau tersumbat karena terbentuknya plak. Plak ini merupakan gabungan dari kolesterol berlebih dan berbagai zat lain yang beredar di dalam darah, termasuk protein, kalsium, dan sel-sel yang meradang. Seiring waktu, plak dapat membesar. Jika bagian luar plak yang mengeras ini pecah atau robek, platelet (partikel darah yang berfungsi dalam pembekuan) akan bergerak ke lokasi tersebut dan membentuk gumpalan darah di sekitar plak. Pembentukan gumpalan ini semakin mempersempit arteri, yang menyebabkan penurunan signifikan pada aliran darah. Proses menumpuknya plak dalam arteri koroner ini secara medis dikenal sebagai aterosklerosis atau sering disebut sebagai "pengerasan arteri". (Kemenkes, 2022).

Penyakit jantung atau *Cardiovascular Disease* (CDV) merupakan faktor pemicu kematian nomor satu di dunia. Meskipun penyakit jantung bukan penyakit menular, namun merupakan penyakit dengan resiko kematian terbesar. *World Health Organization* (WHO) 2020 melaporkan, penyakit jantung menyebabkan kematian sebesar 82 juta jiwa setiap tahunnya di seluruh dunia, angka kejadian penyakit jantung menunjukkan kecenderungan naik secara periodik/setiap tahun. Diperkirakan hingga di tahun–tahun berikutnya penyakit jantung akan akan menyebabkan kematian lebih banyak lagi (Purnama, 2020).

Jaringan saraf tiruan/ *neural network* pertama kali diusulkan sejak lebih dari 70 tahun yang lalu. Jaringan ini terinspirasi oleh jaringan saraf biologis pada otak

manusia, meniru strukturnya. Jaringan saraf merupakan salah satu metode pembelajaran mesin dan biasanya terdiri dari neuron, koneksi, dan bobot. neuron, atau disebut juga persepsi, menerima sinyal masukan dari neuron sebelumnya dan mengirimkan sinyal keluaran ke neuron berikutnya. Neuron-neuron ini biasanya terorganisir dalam lapisan-lapisan. Koneksi merupakan hubungan antara neuron pada lapisan yang berbeda. Setiap neuron dapat memiliki banyak koneksi dengan neuron di lapisan sebelum dan berikutnya, dan setiap koneksi memiliki bobot yang menentukan cara neuron memproses sinyal masukan untuk menghasilkan sinyal keluaran/*output* (Wang dkk, 2020).

Algoritma ELM pertama kali dipresentasikan pada tahun 2006 oleh Guang Bin Huang, Chee Kheong Siew, dan Qin Yu Zhu. ELM dirancang untuk jaringan saraf tiruan lapisan tersembunyi. SLFNs. Algoritma ini menggunakan pemilihan acak dari node tersembunyi dan penentuan bobot *output* SLFNs secara analitis. Algoritma ini dirancang untuk memberikan kinerja generalisasi yang sangat baik sambil belajar dengan sangat cepat. Bahkan pada aplikasi yang sangat besar dan rumit, algoritma baru ini menunjukkan kinerja generalisasi yang baik dan mampu belajar ribuan kali lebih cepat dibandingkan algoritma pembelajaran yang biasa digunakan oleh jaringan saraf tiruan *feedforward*.

Jaringan saraf tiruan (JST) dapat dilatih secara efektif menggunakan ELM, sebuah teknik yang dikenal luas karena kecepatan pelatihannya yang superior dan kemampuannya untuk mencapai akurasi tinggi hanya dengan membutuhkan sedikit sekali pengaturan parameter. ELM seringkali dapat mencapai akurasi yang sebanding, bahkan terkadang lebih baik, dari metode JST yang lebih kompleks yang memerlukan penyetelan parameter ekstensif. Ini dikarenakan ELM meminimalkan intervensi manusia dalam proses penyetelan, yang seringkali menjadi tugas yang membosankan dan membutuhkan keahlian domain yang mendalam. ELM sangat mudah digunakan dan diimplementasikan. Kemudahan ini tidak hanya menghemat waktu pengembangan tetapi juga mengurangi risiko kesalahan manusia yang dapat memengaruhi kinerja model. Kombinasi kecepatan dan akurasi dengan intervensi parameter yang minimal menjadikan ELM alat yang sangat kuat dan efisien dalam berbagai skenario aplikasi, mulai dari klasifikasi hingga regresi. Keunggulan ELM

ini berasal dari metodologinya yang unik: selama proses pelatihan, bobot koneksi antara neuron *input* dan neuron tersembunyi ditetapkan secara acak dan tidak diperbarui. Pendekatan revolusioner ini secara signifikan mengurangi kompleksitas komputasi yang biasanya melekat pada algoritma pelatihan JST tradisional. Berbeda dengan metode iteratif yang memakan waktu, ELM mampu menentukan bobot output secara analitis, menjadikannya pilihan yang sangat efisien untuk berbagai aplikasi, terutama di mana kecepatan dan kinerja real-time menjadi krusial (Saberi F.M dkk, 2020).

Beberapa keterbatasan yang dimiliki ELM seperti penentuan jumlah *hidden nodes* yang didasarkan pada metode *trial and error*, yang mempersulit pencapaian akurasi optimal. Selain itu, pemilihan nilai *bias* secara acak dalam ELM dapat menyebabkan perhitungan yang kurang maksimal. Untuk mengatasi kelemahan ini, algoritma PSO diintegrasikan ke dalam klasifikasi ELM dengan tujuan meningkatkan performa klasifikasi dibandingkan ELM konvensional. Hasil penelitian menunjukkan bahwa ELM yang dioptimalkan dengan PSO menghasilkan klasifikasi diagnosis penyakit jantung yang lebih akurat, dengan peningkatan yang signifikan. Akurasi ELM konvensional hanya 57,32% , sementara ELM yang dioptimalkan dengan PSO mencapai 83,74%. Peningkatan ini dikarenakan PSO membantu menentukan parameter ELM, seperti *input weight* dan *bias*, yang sangat memengaruhi hasil evaluasi. Pengujian parameter optimal menunjukkan bahwa ukuran populasi 300 memberikan akurasi rata-rata 83,30% , jumlah *hidden nodes* 9 menghasilkan akurasi rata-rata terbaik 84,18% , dan iterasi 3 mencapai skor *fitness* terbaik dengan akurasi rata-rata 83,64% (Ariyanti, P.A dkk, 2023).



Pada penelitian sebelumnya telah dikembangkan suatu model prediksi diabetes berbasis ELM dengan memanfaatkan data kuesioner kesehatan sebagai input utama. Model ini dirancang untuk mengoptimasi kecepatan pelatihan sekaligus meminimalkan kebutuhan sumber daya komputasi. Secara struktural, sistem ini terdiri atas dua tahap fundamental: (1) pra-pemrosesan data, dan (2) implementasi algoritma ELM. Pada tahap pra-pemrosesan, dilakukan transformasi data meliputi normalisasi nilai dan konversi variabel tekstual menjadi representasi

numerik. Model kemudian diimplementasikan dalam sebuah platform aplikasi berbasis antarmuka pengguna yang dirancang untuk memfasilitasi proses pengisian kuesioner kesehatan secara mandiri oleh pengguna. Data yang dimanfaatkan dalam studi ini dikumpulkan dari catatan klinis pasien di Rumah Sakit Diabetes Sylhet di Bangladesh, yang mencakup 17 parameter klinis meliputi indikator gejala dan tanda-tanda diabetes pada populasi pasien maupun subjek berisiko. Hasil eksperimen menunjukkan bahwa penggunaan fungsi aktivasi multikuadratik *Radial Basis Function* (RBF) menghasilkan performa optimal dengan tingkat akurasi mencapai 98,07%. Model ini juga mencatatkan nilai *false-positive* 0% dan *false-negative* 1,9%, yang mengindikasikan tingkat reliabilitas yang tinggi untuk aplikasi *skrining* awal diabetes (Elsayed dkk, 2022).

Berdasarkan studi yang dilakukan oleh Khan, M.A. dkk. pada tahun 2021, sebuah kerangka kerja yang mengintegrasikan *One Class Kernel* ELM (OCK-ELM) dengan *Selected Deep Features* (SDF) berhasil melampaui performa metode *Convolutional Neural Network* (CNN) 15 lapis, dengan menunjukkan akurasi yang lebih tinggi. Penambahan langkah fusi (penggabungan) turut berkontribusi dalam peningkatan kinerja sistem secara signifikan, serta meningkatkan akurasi rata-rata. Lebih lanjut, kombinasi SDF ELM mampu meningkatkan akurasi hingga 95% melalui pemilihan fitur-fitur yang paling diskriminatif. Meskipun demikian, terdapat satu keterbatasan krusial dari pendekatan ini, yaitu proses pemilihan fitur yang diterapkan terkadang menghapus beberapa informasi penting dan bermanfaat, bukan hanya informasi yang tidak diperlukan.

Penelitian ini mengintegrasikan estimasi magnitudo kesalahan, yang diperoleh melalui proses diagnosis dan rekonstruksi, sebagai data input krusial untuk pengembangan model prognosis. Sebanyak 700 sampel data magnitudo kesalahan dimanfaatkan untuk melatih dua pendekatan model, VAR dan ELM. Meskipun model VAR, dengan sifat liniernya, menunjukkan kendala dalam memprediksi perkembangan fault secara akurat untuk jangka panjang karena nonlinieritas *inheren* dari data *fault*, model ELM membuktikan keunggulannya. Kemampuan ELM dalam menangani kompleksitas nonlinier data *fault* memungkinkan prediksi magnitudo kesalahan yang jauh lebih akurat dan relevan

untuk prognosis jangka panjang. Oleh karena itu, estimasi magnitudo kesalahan dari proses rekonstruksi menjadi fondasi data yang esensial, dengan ELM muncul sebagai pilihan model yang lebih superior dalam mengantisipasi perkembangan *fault* di masa depan secara komprehensif dan akurat (Qi R dan Zhang J, 2020).

Pada penelitian yang dilakukan oleh Vashist.S dkk, 2022 memperkenalkan dan mengusulkan penggunaan improvisasi ELM sebagai metode yang efektif untuk prediksi hasil panen tanaman pertanian. Dengan penerapan data pra-pemrosesan menggunakan Kalman Filter dan ekstraksi fitur melalui *Linear Discriminant Analysis* (LDA), model ELM yang diimprovisasi mampu mencapai tingkat akurasi tertinggi sebesar 98,83%, yang mengungguli berbagai model lain seperti CNN, LeNet, dan kombinasi CNN&PSO. Dalam penelitian tersebut ditunjukkan bahwa ELM cukup handal dan efisien dalam memprediksi hasil panen berdasarkan berbagai data geografis dan musim, serta Berperan sebagai penunjang proses penentuan kebijakan/keputusan yang akurat dalam bidang pertanian. Hasil ini juga menguatkan bahwa peningkatan jumlah data dan data yang berkualitas sangat berpengaruh dalam meningkatkan akurasi prediksi. Secara keseluruhan, penelitian ini menegaskan bahwa metode *Deep* ELM yang diimprovisasi memiliki potensi besar sebagai solusi prediksi hasil panen yang akurat dan andal, serta dapat diterapkan dalam pengelolaan pertanian berbasis data untuk meningkatkan produktivitas dan efisiensi.

*Synthetic Minority Over-sampling Technique* (SMOTE) dengan *Edited Nearest Neighbor* (ENN) memberikan dampak positif yang signifikan dalam meningkatkan kinerja model deteksi kanker payudara melalui beberapa mekanisme kunci. Pertama, teknik ini menggabungkan *over-sampling* dengan SMOTE dan *under-sampling* dengan ENN untuk menyeimbangkan distribusi kelas minoritas (kanker) dan mayoritas (non-kanker), sehingga model dapat mempelajari kedua kelas secara lebih adil dan efektif. Kedua, keseimbangan data yang dihasilkan serta penghapusan *noise* oleh ENN berkontribusi pada peningkatan akurasi model hingga 99,42%, bersama dengan nilai *precision* dan *recall* yang tinggi, menunjukkan kemampuan klasifikasi yang lebih baik untuk kasus positif maupun negatif. Ketiga, ENN membantu mengurangi *overfitting* dengan

menghilangkan sampel ambigu atau tidak representatif, sehingga model menjadi lebih stabil dan memiliki generalisasi yang lebih baik ketika diterapkan pada data baru. Keempat, proses pembersihan dan penyeimbangan data membuat fitur-fitur yang dipelajari model lebih diskriminatif, meningkatkan ketangguhan (*robustness*) dalam membedakan tumor *benign* dan *malignant*. Secara keseluruhan, SMOTE-ENN tidak hanya meningkatkan akurasi dan keandalan model, tetapi juga mendukung diagnosis yang lebih akurat dan efektif dalam konteks klinis, menjadikannya solusi yang kuat untuk menangani ketidakseimbangan data dalam klasifikasi kanker payudara (Indu, dan Ajmer, 2025).

Pada penelitian yang dilakukan oleh Wang,S. dkk (2021) ketidakseimbangan data yang sering muncul dalam klasifikasi, khususnya pada data medis, di mana jumlah sampel antar kelas (mayoritas dan minoritas) tidak seimbang. Untuk mengatasinya, studi ini mengusulkan peningkatan pada algoritma SMOTE, yang kini mengintegrasikan konsep distribusi normal. Tujuan dari inovasi ini adalah untuk menghasilkan sampel sintetis yang lebih mewakili data minoritas, memastikan bahwa sampel yang dihasilkan lebih dekat ke pusat distribusi kelas minoritas aslinya. Pendekatan ini penting untuk menjaga karakteristik statistik data asli dan mengurangi risiko "marginalisasi" distribusi. Metode yang digunakan melibatkan standarisasi data, penghitungan pusat sampel minoritas, dan estimasi distribusi fitur, yang semuanya berkontribusi pada sintesis sampel baru yang lebih akurat. Hasil eksperimen menunjukkan bahwa algoritma SMOTE yang ditingkatkan ini secara signifikan meningkatkan efektivitas klasifikasi menggunakan metode *Random Forest* pada berbagai dataset tidak seimbang, termasuk dataset medis seperti Pima dan WDBC, melampaui kinerja SMOTE konvensional. Dengan demikian, penelitian ini menawarkan solusi yang menjanjikan untuk memperbaiki performa klasifikasi data tidak seimbang, khususnya dalam konteks medis, melalui pendekatan distribusi normal yang lebih stabil dan mempertahankan statistik data asli.

Beberapa penelitian meggunakan algoritma ELM menghasilkan akurasi yang berbeda-beda. Perbandingan akurasi tersebut disajikan dalam Tabel 2.1.

Tabel 2.1 Perbandingan Akurasi Beberapa Penelitian

| Studi | Algoritma | Akurasi |
|---|---|---|
| Ariyanti, dkk 2023 (diagnosa penyakit jantung) | ELM & PSO | 57,2% |
| Suchithra dkk, 2020 (klasifikasi kesuburan tanah) | ELM | 89% |
| Elsayed, dkk 2022(prediksi diabetes) | ELM | 97% |
| Penelitian ini | ELM | 82% |

Dari Tabel 2.1 menunjukkan bahwa tingkat akurasi dari tiga penelitian menggunakan algoritma ELM di atas 80%, dan hanya satu penelitian yang memperoleh akurasi jauh di bawah 80%. Hal ini dapat menunjukan bahwa Tingkat akurasi dari penelitian yang menggunakan algoritma ELM cukup baik.

## 2.2 Dasar Teori

### 2.2.1 Penyakit Jantung Koroner (PJK)

Penyakit Jantung Koroner (PJK) adalah salah satu akibat utama dari artheriosklerosis, yaitu pengerasan pembuluh darah nadi. Pada kondisi ini, arteri menyempit karena adanya penumpukan lemak di dinding pembuluh darah, yang dapat menghambat aliran oksigen ke jantung dan menyebabkan angina atau rasa nyeri serta ketidaknyamanan di dada. Jika tidak ditangani, kondisi ini dapat berkembang menjadi penyakit jantung koroner. PJK adalah salah satu bentuk penyakit kardiovaskular yang menjadi penyebab kematian utama di dunia. Penyakit ini bersifat degeneratif dan berkaitan dengan gaya hidup serta status sosial ekonomi masyarakat. PJK merupakan masalah kesehatan utama yang sering terjadi di negara-negara maju (Oktaviono, Y. H. 2023).

Menurut data dari *Nurses Health Study* di Amerika Serikat, gaya hidup sehat dapat mengurangi risiko berbagai penyakit, termasuk penyakit jantung, hingga sekitar 80%. Gaya hidup sehat tersebut mencakup beberapa kebiasaan seperti menjaga berat badan, tidak merokok, membatasi konsumsi alkohol, berolahraga setiap hari setidaknya 30 menit. Aspek lain dari gaya hidup sehat adalah pola makan yang seimbang, yang menekankan pada konsumsi makanan rendah lemak jenuh

dan tinggi serat, seperti buah-buahan, sayuran, biji-bijian, dan sumber protein sehat. Penelitian menunjukkan bahwa individu yang menerapkan kebiasaan-kebiasaan ini tidak hanya memiliki risiko lebih rendah untuk terkena penyakit kardiovaskular, tetapi juga mengalami peningkatan kualitas hidup secara keseluruhan. Konsistensi dalam menjalankan pola hidup sehat ini menjadi faktor utama dalam mengurangi risiko penyakit jantung, karena manfaat dari perubahan gaya hidup sering kali terlihat dalam jangka panjang. (Abufaraj dkk., 2019).

Menurut Jabri dkk, 2021, gejala penyakit atau serangan jantung dapat bervariasi antara individu, dengan beberapa mengalami nyeri ringan, sementara yang lain merasakan sakit hebat atau bahkan sama sekali tidak menunjukkan gejala. Gejala umum yang sering dialami oleh penderita penyakit jantung koroner meliputi nyeri dada yang dapat menyebar ke leher, lengan, dan dagu, sesak napas akibat penumpukan cairan di paru-paru, serta debaran jantung yang tidak biasa yang dapat disertai dengan aritmia. Gejala-gejala ini, seperti keringat dingin dan perasaan panik, dapat menandakan masalah kesehatan yang serius, sehingga penting bagi individu untuk mengenali dan segera mencari bantuan medis jika mengalami tanda-tanda tersebut.

Faktor risiko penyebab PJK terdiri dari dua jenis, yaitu faktor risiko yang tidak dapat diubah, seperti usia, jenis kelamin, dan genetik, adapun faktor risiko yang dapat diubah, seperti kebiasaan merokok, dislipidemia, hipertensi, kurang aktifitas fisik, kelebihan berat badan/obesitas, diabetes mellitus, stres, konsumsi alkohol, dan kebiasaan diet yang buruk. Pola konsumsi makanan dan minuman juga berperan dalam PJK yang dapat diubah, seperti hipertensi, dislipidemia, dan diabetes mellitus. Pola konsumsi makanan yang tidak sehat seperti makanan dengan kandungan karbohidrat, kolesterol, dan lemak yang tinggi akan memberikan pengaruh terhadap tubuh dan menjadi faktor risiko untuk terkena diabetes mellitus, dislipidemia, hipertensi, dan penyakit jantung koroner. Selain pola konsumsi makanan dan minuman, pendidikan dan pekerjaan juga mempunyai pengaruh terhadap kesehatan. Seseorang yang tingkat pendidikannya tinggi cenderung memiliki banyak pengetahuan tentang kesehatan. Bekerja dapat memberikan efek yang baik bagi kesehatan dan kesejahteraan pekerja, tetapi bekerja juga dapat

menimbulkan efek buruk bagi pekerja. Risiko penyakit jantung dapat meningkat hingga lebih dari 60% pada pekerja yang mengalami stress, terutama pekerja yang memiliki gaya hidup yang tidak sehat dan kurang berolahraga (Naomi W.S, dkk, 2021).

### 2.2.2 Tingkat Risiko Penyakit Jantung

Untuk menilai kondisi jantung, salah satu alat yang paling sering digunakan adalah klasifikasi dari *New York Heart Association* (NYHA). Klasifikasi ini tidak hanya sekadar mengukur penyakit, melainkan memetakan pasien ke dalam beberapa kelas, mengacu pada gejala yang mereka rasakan saat beraktivitas. Tingkatannya bervariasi, mulai dari pasien yang tidak merasakan apa pun bahkan saat melakukan aktivitas berat, hingga mereka yang mengalami gejala berat meskipun sedang beristirahat. Klasifikasi ini sangat berguna bagi dokter dan pasien untuk memahami sejauh mana penyakit jantung telah berdampak pada kualitas hidup sehari-hari.

Sistem klasifikasi ini dibagi menjadi empat kelas utama, di mana setiap kelasnya menggambarkan tingkat keparahan yang berbeda-beda. Misalnya, Kelas I merujuk pada pasien yang tidak memiliki batasan aktivitas fisik, sedangkan Kelas IV digunakan untuk pasien yang mengalami gejala bahkan saat istirahat dan tidak bisa melakukan aktivitas fisik tanpa ketidaknyamanan. Melalui klasifikasi ini, kita bisa melihat dengan jelas bagaimana penyakit jantung telah berdampak pada kualitas hidup seseorang, seperti yang terlihat pada Tabel 2.2.

Tabel 2.2 Klasifikasi Risiko Penyakit Jantung Menurut NYHA

| **Klasifikasi menurut NYHA** |
|---|
| **Stadium I**<br>Pasien memiliki penyakit jantung, namun tidak ada pembatasan aktivitas fisik. Aktivitas normal tidak menimbulkan gejala seperti kelelahan berlebihan, jantung berdebar, sesak napas, atau nyeri dada. |
| **Stadium II**<br>Penderita gangguan jantung ini menghadapi pembatasan minor pada kemampuan aktivitas fisik. Pasien tetap nyaman dalam kondisi istirahat. Namun, aktivitas pada tingkat normal akan memicu manifestasi seperti fatigue, palpitasi, dispnea, atau nyeri dada. |
| **Stadium III**<br>Pasien didiagnosis dengan gangguan jantung dan menunjukkan keterbatasan minor dalam menjalankan aktivitas fisik. Gejala tidak muncul saat beristirahat. Namun, aktivitas pada tingkat normal akan menimbulkan keluhan seperti *fatigue, palpitasi, dispnea,* atau *angina pectoris*. |
| **Stadium IV**<br>Pasien didiagnosis penyakit jantung yang menghalangi kemampuan mereka untuk melakukan aktivitas fisik apa pun tanpa gejala yang parah. Mereka mungkin mengalami manifestasi gagal jantung bahkan dalam kondisi istirahat. Tentu saja, aktivitas akan meningkatkan semua keluhan tersebut. |



Pada Tabel 2.2 disajikan klasifikasi gejala gagal jantung menurut *New York Heart Association (NYHA)*, yang membagi tingkat keparahan kondisi pasien ke dalam empat stadium berdasarkan keterbatasan aktivitas fisik dan gejala yang muncul.

### 2.2.3 Pembelajaran Mesin dan Klasifikasi

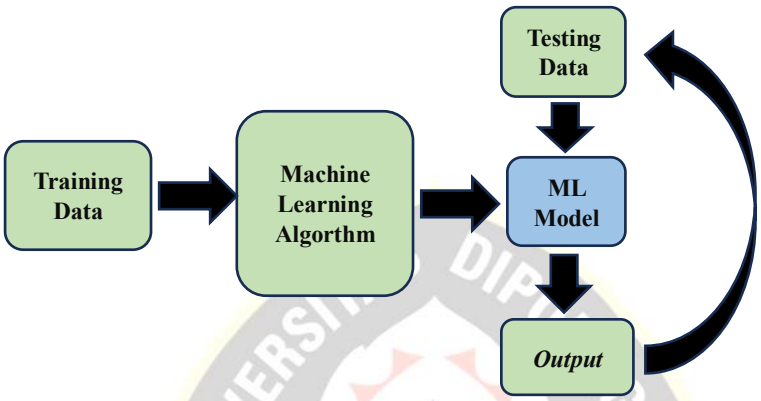
Pembelajaran mesin merupakan pendekatan dalam AI (*Artificial Intelegence*) yang banyak digunakan untuk meniru atau menggantikan tindakan yang dilakukan manusia untuk menyelesaikan persoalan atau mengotomatisasi pekerjaan/proses. Sesuai namanya, *machine learning* berupaya menirukan mekanisme makhluk cerdas (termasuk manusia) dalam mempelajari data dan kemudian menerapkan pengetahuan tersebut secara luas (generalisasi). Ciri khas dari *machine learning* adalah adanya proses pelatihan, pembelajaran atau pengujian. Oleh karena itu, *machine learning* membutuhkan data untuk dipelajari yang disebut sebagai data pelatihan.

Pembelajaran mesin/*Machine Learning* (ML) merupakan cabang dari kecerdasan buatan *(Artificial Intelligence*, AI) yang memungkinkan sistem untuk mendapatkan pengetahuan dan membuat keputusan berdasarkan data tanpa harus diprogram secara eksplisit. Dengan kata lain, sistem komputer dapat meningkatkan kinerjanya dari waktu ke waktu dengan memproses dan menganalisis sejumlah besar data. Pembelajaran mesin merupakan bagian dari bidang keilmuan kecerdassan buatan (*Artificial Intelligence*) yang banyak untuk diamati dan diteliti karena dapat berguna untuk memecahkan berbagai macam permasalahan kehidupan di berbagai bidang. Bidang tersebut seperti bidang kesehatan, kemasyarakatan, keuangan, maupun bidang lainnya. Ulasan dari berbagai macam bidang dengan machine learning dapat disajikan dalam berbagai macam algoritma yang dapat diimplementasikan. Algoritma tersebut dibagi menjadi tiga kategori dalam *machine learning* diantaranya *supervised learning, unsupervised learning dan reinforcement learning* (Praiss dkk, 2020).



Pembelajaran terawasi mencakup dua fungsi, yaitu fungsi klasifikasi dan regresi. Fungsi klasifikasi dengan jenis algoritma *Naïve Bayes Classifier, Decision Trees, Support Vektor Machine* (SVM)*, random forest,* dan *K-Nearest Neighbors* sedangkan fungsi regresi dengan jenis algoritma: regresi linier, regresi jaringan syaraf, SVM, regresi *Decision Tree*, regresi Laso, dan regresi *ridge*. Pembelajaran tidak terawasi hanya fokus pada pengelompokkan */clustering* yang terdiri dari lima algoritma yaitu: *K-Means Clustering, Mean Shift Clustering, DBSCAN Clustering,*

*Agglomeratif Hierarchical Clustering,* dan *Gaussian Mixture.* Jenis pembelajaran reinforcement fokus pada *Decission Making*, yang terdiri dari; *Q Learning, R Learning,* dan *TD Learning*. Ketiga jenis ML masing-masing memiiki fungsi, kelebihan, serta kekurangan. Adapun alur kerja pembelajaran mesin secara umum ditunjukkan pada Gambar 2.1.




Gambar 2.1. Alur Kerja Pembelajaran Mesin (Prais, 2020)

Gambar 2.1 menunjukkan proses berulang yang umum dalam pengembangan sistem pembelajaran mesin, di mana pengujian dan evaluasi dilakukan secara terus-menerus untuk menyempurnakan kualitas model.

### 2.2.4 Jaringan Saraf Tiruan (JST) dan *Multilayer Perceptron* (MLP)



Jaringan Saraf Tiruan (JST), atau yang dikenal juga sebagai *Artificial Neural Network* (ANN), pertama kali dikembangkan pada tahun 1943 oleh seorang ahli neurofisiologi bernama Warren Mc Culloch dan seorang pakar logika, Walter Pitts. Jaringan saraf, yang sering disebut sebagai jaringan saraf buatan, merupakan algoritma mesin pembelajaran yang terinspirasi oleh jaringan syaraf biologis. Algoritma ini pertama kali diusulkan pada abad yang lalu, sekitar 70 tahun yang lalu atau lebih. Konsep ini muncul sebagai upaya untuk menciptakan sistem komputasi yang dapat meniru kemampuan otak manusia dalam memproses

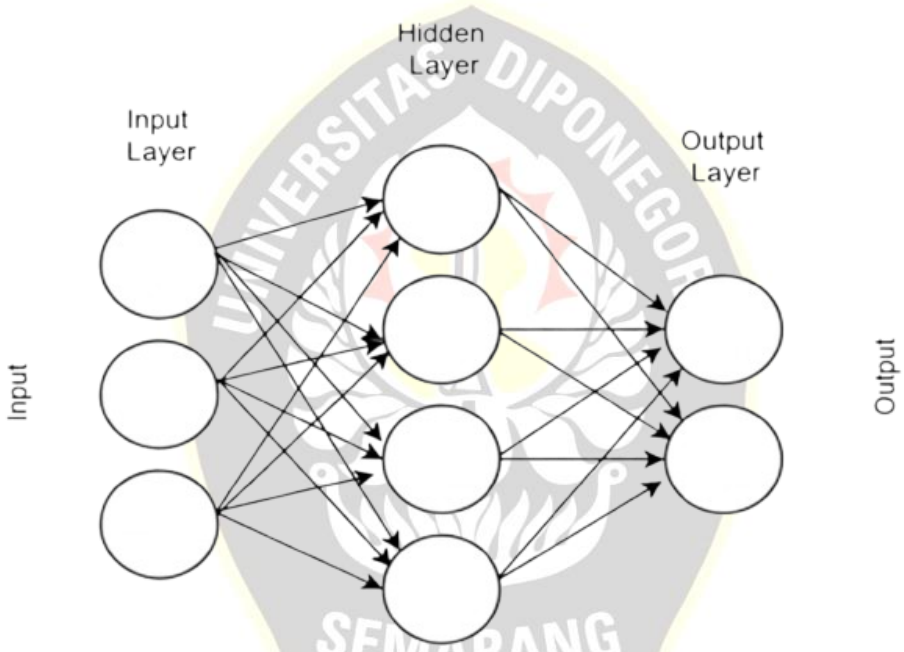
informasi. Jaringan saraf buatan terinspirasi oleh jaringan saraf biologis, yaitu kumpulan neuron yang saling terhubung dalam otak manusia. Struktur dasarnya dirancang untuk meniru cara kerja sistem saraf biologis, di mana neuron-neuron buatan saling berkomunikasi melalui koneksi yang disebut bobot sinaptik. Dengan pendekatan ini, jaringan saraf buatan mampu mempelajari pola, mengenali hubungan dalam data, dan membuat prediksi berdasarkan pengalaman yang telah diperoleh selama proses pelatihan. Penemuan ini menjadi landasan bagi perkembangan teknologi kecerdasan buatan yang semakin canggih hingga saat ini (Choi Y Rene dkk, 2025).

Sebagai salah satu metode pembelajaran mesin, jaringan saraf memiliki struktur khas yang terdiri dari neuron, koneksi, dan bobot. Neuron, yang sering juga disebut perseptron, berfungsi sebagai unit dasar dalam jaringan saraf. Setiap neuron menerima sinyal masukan dari neuron sebelumnya melalui koneksi yang diwakili oleh bobot, yang menentukan seberapa penting sinyal tersebut dalam proses komputasi. Setelah menerima sinyal, neuron memproses informasi tersebut dengan menerapkan fungsi aktivasi tertentu untuk menghasilkan sinyal keluaran, yang kemudian diteruskan ke neuron berikutnya. Dalam jaringan saraf, neuron biasanya diorganisir dalam lapisan-lapisan berbeda yang masing-masing memiliki fungsi tertentu.

Lapisan masukan (*input layer*) bertugas menerima data awal yang akan diproses oleh jaringan. Selanjutnya, lapisan tersembunyi (*hidden layer*) melakukan analisis dan transformasi data melalui interaksi antar-neuron, menangkap pola-pola kompleks dalam data. Terakhir, lapisan keluaran (*output layer*) menghasilkan hasil akhir, seperti klasifikasi atau prediksi, berdasarkan proses yang terjadi di lapisan sebelumnya. Organisasi lapisan ini memungkinkan jaringan saraf untuk menangani berbagai jenis masalah, mulai dari pengenalan gambar hingga analisis data yang lebih kompleks, menjadikannya salah satu alat paling kuat dalam pembelajaran mesin modern. Koneksi didefinisikan sebagai hubungan antara neuron-neuron dari lapisan yang berbeda. Sebuah neuron dapat memiliki banyak koneksi ke neuron-neuron dari lapisan sebelumnya dan lapisan berikutnya. Dan setiap koneksi mengirimkan parameter penting yang disebut bobot. Itulah bobot yang menentukan

bagaimana neuron memproses sinyal masukan untuk menghasilkan sinyal keluaran (Shui-Hua dkk, 2021).

Jaringan saraf tiruan ditentukan olah tiga hal, yaitu: arsitektur jaringan, metode *training/learning*/algoritma, dan fungsi aktivasi. Arsitektur jaringan merupakan pola hubungan antar neuron, metode *training/learning/*algoritma merupakan cara yang digunakan untuk menentukan bobot penghubung, sedangkan fungsi aktivasi merupakan tempat terkumpulnya neuron di dalam lapisan-lapisan yang dinamakan dengan *neuron layers* (Solihun dan Wahyudi , 2020). Adapun arsitektur JST ditunjukkan oleh Gambar 2.2.




Gambar 2.2 Arsitektur Jaringan Saraf Tiruan

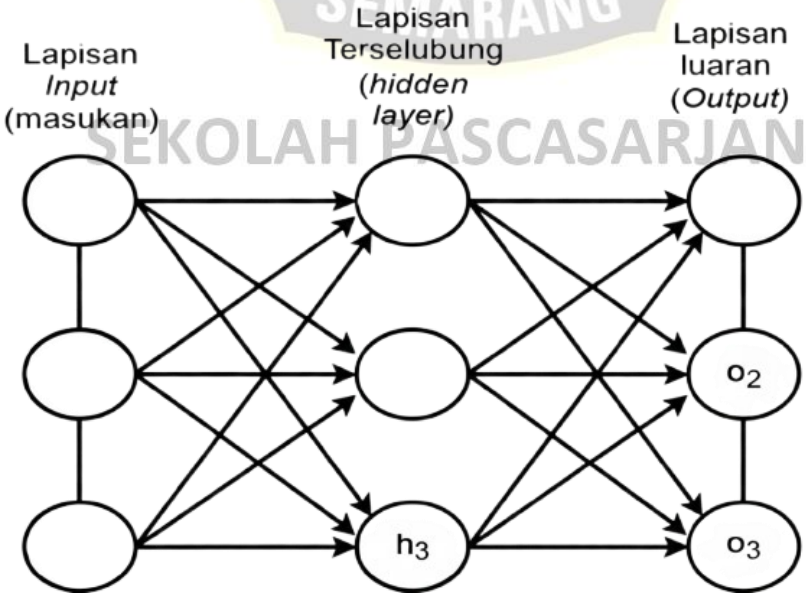


Gambar 2.2 merupakan arsitektur JST yang terdiri dari tiga lapisan/*layer*, yaitu: lapisan masukan/*input layer*, yang terdiri dari unit-unit *input* yang merupakan unit-unit di dalam lapisan *input*. Unit *input* berfungsi mengambil data atau pola dari lingkungan eksternal untuk mendeskripsikan suatu masalah. Di sisi lain, lapisan tersembunyi (*hidden layer*) terdiri dari unit-unit yang memproses data, tetapi hasil keluarannya tidak dapat diamati secara langsung. Dan lapisan keluaran/*output layer*. Unit-unit *output* merupakan unit-unit di dalam lapisan *output*. Pada lapisan *output* merupakan solusi dari jaringan saraf tiruan terhadap suatu permasalahan

(Effendi, 2018). Fungsi aktivasi yang umum digunakan pada JST meliputi: fungsi linier, fungsi ambang batas/*threshold*, fungsi ambang batas linier/*linear threshold*, dan fungsi sigmoid.

*Multilayer perceptron* (MLP) merupakan jenis jaringan saraf tiruan yang paling terkenal dan banyak digunakan. Jaringan ini dibentuk oleh unit-unit seperti yang ditunjukkan pada Gambar 2.2. Setiap unit menghitung jumlah terbobot dari input yang diterima, ditambah dengan suatu konstanta. Hasil penjumlahan ini kemudian diproses melalui fungsi nonlinear, yang sering disebut sebagai fungsi aktivasi. Umumnya, unit-unit ini saling terhubung secara *feedforward*, artinya tidak membentuk putaran (*loop*), seperti yang diperlihatkan dalam Gambar C2.2. Namun, untuk beberapa jenis aplikasi, jaringan *recurrent* (yang tidak *feedforward*) di mana sebagian koneksi membentuk *loop* juga digunakan (Almeida, 1997).

Algortima *Multi Layer Perceptron* (MLP) merupakan sebuah algoritma pembelajaran mendalam (*Deep Learning*) yang banyak dimanfaatkan untuk mengolah informasi guna melakukan identifikasi pola, klasifikasi, dan prediksi (Zahara & Sugianto, 2021). MLP merupakan salah satu algoritma Jaringan Syaraf Tiruan yang melakukan *feedforward* atau maju mundur (Khoirudin dkk., 2019). *Multi Layer Perceptron* terdiri atas lapisan masukan, lapisan luaran, dan lapisan terselubung (*hidden layer*), seperti yang ditunjukkan oleh Gambar 2.3 (Setialaksana dkk., 2021).



Gambar 2.3 Arsitektur Jaringan MLP

Gambar 2.3 menyajikan arsitektur dasar dari sebuah *Multilayer Perceptron (MLP)*, salah satu jenis jaringan saraf tiruan (*Artificial Neural Network*) yang paling umum. Struktur ini terdiri dari tiga lapisan utama yang bekerja secara sekuensial untuk memproses data. Lapisan pertama adalah lapisan masukan (*Input Layer*), yang terdiri dari neuron-neuron berlabel $x_1$, $x_2$, dan $x_3$. Lapisan ini berfungsi sebagai pintu masuk data awal ke dalam jaringan. Selanjutnya, data diteruskan ke lapisan tersembunyi (*Hidden Layer*), yang dilabeli dengan $h_1$, $h_2$, dan $h_3$. Di sinilah proses komputasi dan transformasi data yang paling kompleks terjadi, di mana setiap neuron di lapisan ini menerima masukan dari semua neuron di lapisan sebelumnya. Terakhir, setelah data diproses, hasilnya disampaikan ke lapisan keluaran (*Output Layer*), yang terdiri dari neuron $o_1$, $o_2$, dan $o_3$. Lapisan ini menghasilkan output akhir dari seluruh proses, yang dapat digunakan untuk prediksi atau klasifikasi. Koneksi panah yang mengalir dari kiri ke kanan menunjukkan bahwa data bergerak secara satu arah (*feedforward*), dari *input* hingga *output*, yang merupakan karakteristik utama dari arsitektur MLP.

### 2.2.5 *Single-Layer Feedforward Networks* (SLFNs)

Jaringan *Feedforward* Lapisan Tersembunyi Tunggal (SLFNs) adalah jenis jaringan *feedforward* yang paling sederhana, dengan arsitektur yang terdiri dari satu lapisan perseptron tersembunyi yang tujuan utamanya adalah mengidentifikasi pola dalam data agar proses pengenalan atau klasifikasi menghasilkan akurasi yang tinggi. Jaringan *Feedforward* bekerja dengan mengoptimalkan bobot melalui fungsi aktivasi untuk menemukan konfigurasi yang paling efektif. Secara spesifik, algoritma *Extreme Learning Machine* (ELM) memungkinkan parameter pembelajaran pada *hidden node* (seperti bobot *input* dan bias *input*) diatur secara acak. Sementara itu, bobot *output* jaringan dapat ditentukan secara analitis menggunakan operasi *generalized inverse* yang relatif mudah. (Wang J. dkk, 2021).

Fungsi aktivasi neuron SLFN menggunakan unit ambang linier. SLFN dapat membedakan antara dua hingga N pengamatan menggunakan fungsi aktivasi non-linier seperti tangen hiperbolik (tanh) atau batas keras (*hardlim*), tergantung pada jumlah neuron dalam lapisan tersembunyi. Ketika SLFN memiliki N neuron tersembunyi, ia dapat dengan tepat membedakan antara N pengamatan berbeda. Namun, kelemahan utama SLFN adalah masalah generalisasi, yang merupakan kekurangan signifikan ketika menggunakan algoritma pembelajaran tradisional (Elsayed dkk, 2022).

Diberikan himpunan pelatihan $S = \{(xi, ti)|xi = (xi_1, xi_2, \dots, xin)T \in Rn, ti = (ti_1,\ ti_2, \dots tim)T \in Rm\}$, di mana xi menunjukkan nilai masukan dan ti mewakili target. Keluaran o dari ELM yang memiliki b neuron tersembunyi dapat dinyatakan sebagai:

$$\sum_{i=1}^{N\wedge} \beta igi(\boldsymbol{wixj} + \boldsymbol{bi}) = \boldsymbol{oj}, j = 1, \dots, N \qquad (2.1)$$

Dengan

J = 1,2, …., N

Wi = $(Wi1, W12, \dots, Win)^T$ merupakan vektor dari bobot yang menghubungkan $i^{th}$ node tersembunyi dan node input.

$\beta i = (\beta i1, \beta 12, \dots, \beta in)^T$ Vektor bobot dari simpul tersembunyi ke-i ke lapisan *output,*

g(i) = fungsi aktivasi di lapisan tersembunyi,

bi = bias dari node tersembunyi ke-i

wi = Vektor bobot dari lapisan *input* ke node tersembunyi ke-i,

xi = Vektor *input* dari sampel ke-i

$N\wedge$ = Jumlah node tersembunyi dalam ELM,

Formula dari fungsi-fungsi tersebut disajikan dalam Tabel 2.3.

Tabel 2.3 Fungsi Aktivasi pada ELM

| Fungsi | Formula |
|---|---|
| Fungsi sigmoid | $G(a, b, x) = \dfrac{1}{1 + \exp(-(ax + b))}$ |

| | |
|---|---|
| Fungsi tangen hiperbolik | $G(a,b,x) = \dfrac{1-\exp\{-(ax+b)\}}{1+\exp\{-(ax+b)\}}$ |
| Fungsi basis radial | $G(a,b,x) = \exp(-b\|\|x-a\|\|)$ |
| Fungsi multi-kuadratik | $G(a,b,x) = (\|\|x-a\|\| + b^2)^{\frac{1}{2}}$ |
| Fungsi batas keras | $G(a,b,x) = \{1, \text{ax} + \text{b} \leq 0\ 0$, atau |
| Fungsi kosinus | $G(a,b,x) = \cos(ax+b)$ |

Tujuan dari pelatihan adalah untuk meminimalkan kesalahan antara target dan output ELM. Fungsi objek yang paling umum digunakan adalah *mean squared error* (MSE) yang ditujukkan oleh persamaan 2.2.

$$MSE = \sum_{i=1}^{N}(tij - Oij)^2, j = 1, \dots, m \tag{2.2}$$

Dengan

$ti$ = Vektor target dari sampel ke-i

Oi = nilai prediksi dari sampel ke-i,

di mana N adalah jumlah sampel pelatihan, dan *i* serta *j* adalah indeks untuk sampel pelatihan dan *node* lapisan keluaran. Dapat dibuktikan bahwa SLFN mampu mendekati semua sampel pelatihan ketika jumlah node tersembunyi N mendekati tak terhingga sebagaimana ditunjukkan oleh persamaaan 3 yang disebut kemampuan aproksimasi *universal*, jadi harus ada sekumpulan wi, bi, dan βi yang cukup sebagaimana ditunjukkan oleh persamaan 4.

$$\sum_{j=1}^{N}\|oj - tj\| = 0 \tag{2.3}$$



Dengan

Oj = *Output* prediksi (*predicted output*) untuk sampel ke-j*j* dari model.

*tj* adalah vektor lengkap untuk sampel ke-j*j* (semua *node output*)

*tij* adalah komponen spesifik (node ke-j*j*, sampel ke-i*i*)

$$\sum_{i=1}^{N^{\wedge}} \beta ig(wixj + -bi), = tj, j = 1, \dots, m \tag{2.4}$$

Dengan

$\sum_{i=1}^{\widehat{N}}$

: Menunjukkan penjumlahan dari i=1*i*=1 hingga i=n*i*=*n*, di mana n*n* adalah jumlah node pada lapisan tersembunyi (*hidden layer*).

βi : Vektor bobot (*weight*) yang menghubungkan node ke-i pada lapisan tersembunyi dengan *node output*.

*g*i : Fungsi aktivasi (*activation function*) untuk *node* ke-*i* pada lapisan tersembunyi.

*Wi* : Vektor bobot yang menghubungkan *node input* ke node tersembunyi ke-i.

*X* *i*: Vektor input data ke-i

*bi* : Nilai bias (*threshold*) untuk node tersembunyi ke-i

*tj* **:** Target atau nilai *output* yang diharapkan untuk sampel data ke-j.

*j* *=1,...,N* Indeks yang menunjukkan bahwa persamaan ini berlaku untuk semua sampel data dari ke-1 hingga ke-N.

Rumus pada persamaan 2.4 dapat disingkat sebagai

$$H\beta = T \tag{2.5}$$

Dengan H = matriks keluaran lapisan tersembunyi ELM (matriks acak)

T = matiks dari target atau *output*

$$H(w_1, \dots, w_{N\wedge}, b1, \dots b_N, X_1 \dots X_N = \begin{bmatrix} g(w_i x_0 + b_1) & \cdots & g(w_{N\wedge} x_1 + b_{N\wedge}) \\ \vdots & \ddots & \vdots \\ g(w_i x_N + b_1) & \cdots & g(w_{N\wedge} x_N + b_{N\wedge}) \end{bmatrix} \tag{2.6}$$

Dengan

wi,..wn : vektor bobot (*weights*) yang menghubungkan *input layer* ke *hidden layer*.

*wi* : Bobot untuk neuron ke-i di *hidden layer*.

*wi*·*xj* : Perkalian *dot product* antara bobot dan *input*

*b*1,..*bN* : bias (*threshold*) untuk setiap neuron di *hidden layer*

*bi* : Nilai bias yang ditambahkan ke hasil perkalian *wixj*

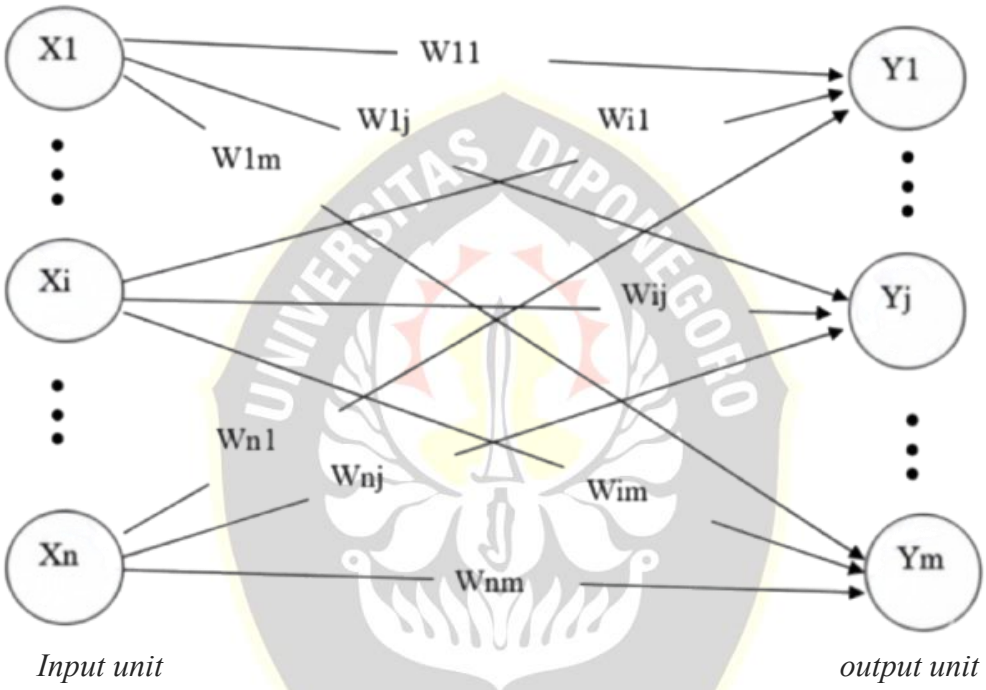
*x*1,..,*xN* : Vektor *input* data (fitur).

*xj* : Sampel *input* ke-j dengan dimensi tergantung jumlah fitur

*g* : Fungsi aktivasi (sigmoid, ReLU, tanh) yang diterapkan pada hasil

kombinasi linier *input* dan bobot.

Menurut Widotomo dkk, 2021, sebuah *Single-Layer Feedforward Neural Networks* (SLFNs) konvensional terdiri dari tiga lapisan: lapisan *input*, lapisan tersembunyi, dan lapisan *output*. Struktur SLFN ditunjukkan oleh Gambar 2.4.




Gambar 2.4 Struktur SLFNs (Widotomo dkk, 2021)

Gambar 2.4 merupakan struktur dari *Single-Layer Feedforward Neural Networks* (SLFNs). Dengan komponen utama: lapisan masukan (*input layer*), lapisan tersembunyi (*hidden layer*), dan lapisan keluaran (*output layer*). Input diterima oleh lapisan masukan dan diteruskan ke lapisan tersembunyi. Lapisan tersembunyi memproses *input* dengan mengalikan bobot dan menambahkan bias, kemudian menerapkan fungsi aktivasi. *Output* dari lapisan tersembunyi diteruskan ke lapisan *output*, di mana proses serupa dilakukan untuk menghasilkan *output* akhir.

### 2.2.6 *Extreme Learning Machine* ( ELM)

*Extreme Learning Machine* (ELM) merupakan sebuah algoritma pelatihan untuk Jaringan Saraf Tiruan (JST) *feedforward* dengan satu lapisan tersembunyi (*hidden layer*). Metode ini dikembangkan untuk mengatasi kelemahan utama JST konvensional, yaitu kecepatan pelatihan (*learning speed*) yang rendah akibat proses penyesuaian bobot dan bias secara iteratif. ELM mengeliminasi kebutuhan iterasi tersebut dengan cara menginisialisasi bobot *input* dan bias secara acak, lalu menentukah bobot *output* secara analitis menggunakan *Moore-Penrose Generalized Inverse*. Dengan mekanisme ini, ELM mampu mencapai kecepatan pelatihan yang jauh lebih superior (Huang, 2006).

Algoritma ini merupakan neuron lapisan tunggal yang tersembunyi dan dikenal sebagai *Single-Layer Feedforward Neural Networks* (SLFNs). Beberapa parameter neuron yang tersembunyi termasuk bobot *multilayer perceptron* (MLP). Metode ini membutuhkan waktu latihan yang sangat cepat dan memiliki performa yang sangat baik. Namun, metode ini memiliki beberapa kekurangan, seperti kemungkinan *overfitting* model, terhadap model, dapat melemahkan kapasitas kontrol karena metode ini menghitung secara langsung solusi kuadrat terkecil norma minimum, dan dapat memberikan akurasi yang tidak sempurna. ELM ditemukan guna melatih SLFN, yang merupakan struktur jaringan saraf buatan yang paling banyak digunakan. Melatih jaringan adalah untuk menentukan parameter-parameter ini yang mencapai solusi yang optimal.

Keunggulan utama ELM yang miliki dibandingkan metode *deep learning* tradisional seperti *Deep Belief Network* (DBN) dan *deep Boltzmann machine* (DBM), terutama dalam hal kecepatan pelatihan dan efisiensi komputasi. Tidak hanya lebih cepat, ELM sering kali menghasilkan performa yang setara atau bahkan lebih baik dalam berbagai kasus. Sebagai contoh, beberapa penelitian menunjukkan bahwa ELM dan varian *multilayer*-nya mampu menyaingi atau mengungguli DBN dan DBM dalam tugas klasifikasi, dengan waktu pelatihan yang jauh lebih singkat. Selain itu, ELM juga menonjol dalam hal kemudahan implementasi dan keberlanjutan, menjadikannya pilihan menarik sebagai alternatif atau pelengkap untuk metode *deep learning* yang umumnya membutuhkan sumber daya komputasi

tinggi dan waktu pelatihan lama. Meskipun strukturnya relatif lebih sederhana, ELM tetap unggul dalam skenario yang memprioritaskan kecepatan dan efisiensi. (Wang J. dkk, 2021).

Algortima ELM merupakan alat pembelajaran mesin yang kuat dan efisien. Keunggulan fundamental ELM adalah mekanisme pembelajarannya yang tidak iteratif; bobot *input* dan bias pada lapisan tersembunyi tidak perlu disetel berulang kali, sehingga menghasilkan kecepatan yang jauh lebih tinggi dan biaya yang lebih rendah dibandingkan jaringan konvensional (SLFN). Karena keunggulan ini, ELM seringkali menjadi pilihan utama dibandingkan metode-metode sebelumnya (Huang dkk, 2006). Beberapa kelebihan lain yang membuatnya populer meliputi: tingkat akurasi dan kemampuan generalisasi yang baik, algoritma pembelajaran yang sederhana dan efisiensi yang tinggi, solusi yang terpadu untuk berbagai aplikasi praktis, membutuhkan lebih sedikit optimasi dan memiliki biaya komputasi yang setara dengan SVM (Zhang dkk, 2016).

Metodologi ELM terdiri dari empat fase utama. Fase pertama adalah partisi data ke dalam set pelatihan dan set pengujian. Fase kedua adalah tahap pelatihan, di mana bobot model dihitung secara analitis dalam satu komputasi tunggal berdasarkan data pelatihan. Fase ketiga adalah validasi, yaitu pengujian model terhadap data pengujian. Fase terakhir adalah evaluasi, di mana hasil prediksi dianalisis untuk mengukur performa model. Proses ini secara fundamental bersifat non-iteratif; tidak ada mekanisme pengulangan untuk optimasi bobot jika *error* belum memenuhi target. Notasi yang digunakan dalam algoritma pelatihan dan pengujian menurut Huang dkk, 2006 ditunjukkan oleh Tabel 2.4.

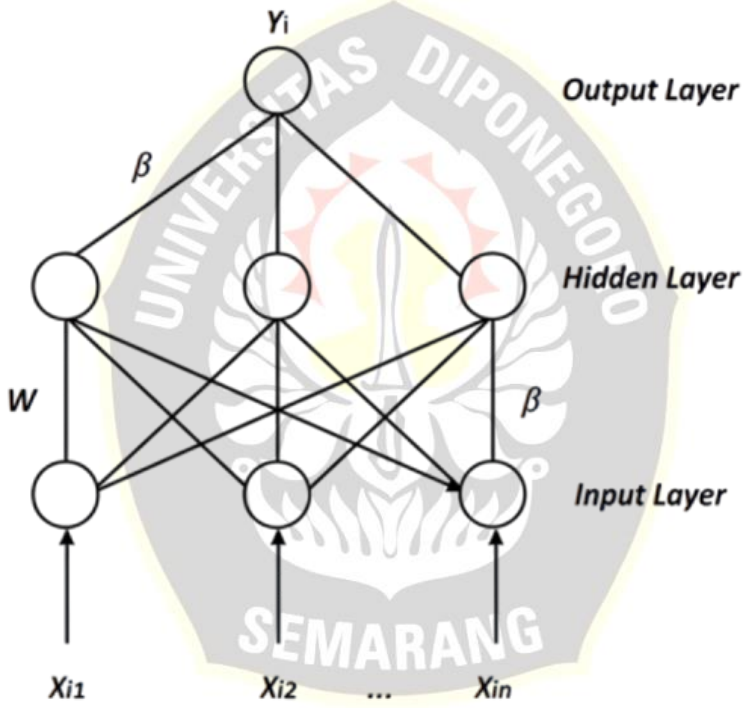


Tabel 2.4 Notasi Algortima Pelatihan dan Pengujian.

| No | Notasi | Arti/Keterangan |
|---|---|---|
| 1 | $x_i$ | Vektor data *input* |
| 2 | $T$ | Vektor target |
| 3 | $H$ | Matriks yang dihasilkan dari keluaran semua lapisan tersembunyi. |

| | | |
|---|---|---|
| 4 | $H^+$ | Matriks Moore-Penrose dari matriks H |

Sebagai bagian dari Jaringan Saraf Tiruan (JST), *Extreme Learning Machine* (ELM) merupakan algoritma untuk JST jenis *feedforward* yang memiliki ciri khas berupa penggunaan satu lapisan tersembunyi. Karena itu, ELM juga dikenal sebagai *Single Hidden Layer Feedforward Networks* (SLFNs), yang arsitekturnya terdiri dari tiga lapisan: *input*, satu *hidden layer*, dan *output* (Aditya dkk, 2020). Arsitektur dari ELM dapat dilustrasikan pada Gambar 2.5.






Gambar 2.5 Arsitektur ELM (Aditya dkk, 2020)

Gambar ini menunjukkan arsitektur *Extreme Learning Machine* (ELM), model jaringan saraf tiruan *single-hidden layer* yang dirancang untuk pelatihan yang sangat cepat. Arsitektur ini, mirip dengan jaringan saraf lainnya, memiliki tiga lapisan: lapisan *Input* yang menerima data masukan ($Xi_1, Xi_2, \ldots Xi_n$), lapisan tersembunyi, dan lapisan *Output* (Y). Keunikan utama ELM terletak pada cara latihannya. Bobot koneksi (W) antara lapisan *input* dan lapisan tersembunyi

ditugaskan secara acak dan tidak diubah selama pelatihan. Sebaliknya, hanya bobot koneksi (β) antara lapisan tersembunyi dan lapisan *output* yang dihitung secara analitis menggunakan metode seperti *least squares*. Pendekatan ini menghilangkan kebutuhan untuk proses optimasi iteratif yang panjang seperti *backpropagation*, sehingga secara signifikan mempercepat proses pelatihan dan membuat ELM sangat efisien dalam menyelesaikan masalah klasifikasi dan regresi.

Proses prediksi menggunakan ELM melibatkan beberapa tahapan. Pertama, data dinormalisasi, lalu digunakan untuk melatih (*training*) dan menguji (*testing*) model. Setelah itu, hasil prediksi dari model akan melalui proses denormalisasi untuk mengembalikannya ke skala nilai yang asli. Terakhir, tingkat kesalahan prediksi dievaluasi dengan menggunakan metode *Root Mean Square Error* (RMSE). Normalisasi merupakan proses penskalaan nilai dari atribut data sehingga dapat memiliki *range* yang sama. Berikut persamaan untuk menghitung Z-score *Normalization*.

$$X' = \frac{(X - mean)}{SD} \tag{2.7}$$

Dengan

$X'$ : hasil dari normalisasi

$X$ : nilai asli data yang digunakan,

mean : nilai rata-rata dari $X$ , dan

*SD* : nilai standar deviasi dari $X$.

Langkah-langkah yang dilakukan dalam proses pelatihan menggunakan metode ELM adalah sebagai berikut:

1. Menginisialisasi bobot $w$ dan bias secara acak.
2. Mencari keluaran dari lapisan tersembunyi dengan persamaan berikut:

$$Y_{in\ train} = X_{train} W^T + b \tag{2.8}$$

$Y_{in\ train}$ : keluaran dari bobot tersembunyi,

$X_{train}$ : matriks data *training*,

$W^T$ : transpose matriks bobot, dan

$b$ : bobot bias dari lapisan tersembuyi.

3. Mengaktivasi keluaran lapisan tersembunyi, dengan persamaan:

$$Y = \frac{1}{1+\exp(-H_{in\ train})} \quad (2.9)$$

Dengan Y = matriks keluaran lapisan tersembunyi,

$H_{in\ train}$ = matriks dari bobot tersembunyi.

4. Hitung matriks Moore-Penrose *Generalized Inverse* menggunakan persamaan berikut.

$$Y^{+} = (Y^{T}.Y)^{-1}Y^{T} \quad (2.10)$$

Dengan $Y^{+}$ merupakan matriks *Moore-Penrose Generalized Inverse* dari matriks Y, sedangkan matirks Y merupakan matriks keluaran lapisan tersembunyi.

5. Hitung *output weight* dengan persamaan berikut:

$$\beta = Y^{+}.Z \quad (2.11)$$

Dengan $Y^{+}$ merupakan matriks Moore-Penrose *Generalized Inverse* dan Z adalah matriks target.

Secara ringkas, proses pelatihan menggunakan metode ELM ditunjukkan pada Tabel 2.5.

Tabel 2.5 Pelatihan ELM

| Pelatihan ELM |
|---|
| Training Set $S = [(xi, yi)\|xi \in R^{m}, i = 1, \dots N]$<br>Inisialisasi:<br>Tetapkan nilai acak untuk bobot tersembunyi wi dan bias *bi* serta hitung matriks keluaran dari lapisan tersembunyi H menggunakan set pelatihan.<br>Solusi analitis<br>Dapatkan $\beta$ dari $H\beta$ dengan invers Moore Penrose. $\beta = HT$ di mana H adalah matriks invers umum Moore Penrose H |

Tujuan melatih SLFNs adalah untuk menemukan *wi, bi*, dan βi terbaik. Pelatihan dasar ELM terdiri dari dua tahap: inisialisasi acak dan penyelesaian parameter linier. Pada tahap pertama, ELM menerapkan parameter acak di lapisan tersembunyi yang tetap konstan selama proses pelatihan, memetakan vektor *input* ke dalam ruang fitur acak dengan menggunakan fungsi aktivasi nonlinier yang lebih efisien. Dengan adanya fungsi aktivasi kontinyu potongan nonlinier, ELM menunjukkan kemampuan aproksimasi *universal*. Pada tahap kedua, parameter βi dihitung menggunakan invers Moore-Penrose, yang menyelesaikan masalah linier tersebut. ELM mampu memberikan generalisasi yang lebih baik tanpa perlu menyetel parameter tersembunyi secara bertahap, dan dapat mendekati batas klasifikasi kompleks dengan jumlah *node* tersembunyi yang cukup banyak.

Langkah-langkah yang dilakukan dalam proses pengujian menggunakan algoritma ELM sebagai berikut:

1. Bobot W dan bias serta β diperoleh berdasarkan hasil pelatihan,
2. Hitung matriks keluaran *hidden layer* dengan persamaan berikut:

$$Y_{in\ test} = X_{test} W^T + b \tag{2.12}$$

Dengan

$Y_{in\ test}$ : keluaran dari bobot tersembunyi,

$X_{test}$ : matriks data *testing*,

$W^T$ : *transpose* matriks bobot, dan

$b$ : bobot bias dari lapisan tersembuyi



3. Mengaktivasi keluaran lapisan tersembunyi, dengan persamaan:

$$Y = \frac{1}{1+\exp(-H_{in\ test})} \tag{2.13}$$

Dengan Y = keluaran lapisan tersembunyi,

$H_{in\ train}$ = matriks dari bobot tersembunyi.

4. Hitung *output weight* dengan persamaan berikut:

$$H = Y.\beta \tag{2.14}$$

Dengan H adalah hasil *output layer* sedangkan Y adalah matriks keluaran *hidden layer* dan β adalah matriks keluaran *hidden layer* pada proses pelatihan.

5. Melakukan denormalisasi untuk mendapatkan nilai asli dengan persamaan berikut.

$$X = \left(x'.\sqrt{\frac{(xi-\mu)^2}{N}}\right) + \mu \tag{2.15}$$

Dimana $X$ adalah nilai asli dan x' adalah hasil *output layer*.

Evaluasi hasil prediksi dengan RMSE, berikut persamaan yang digunakan:

$$RMSE = \left(\frac{\sum(yi-xi)}{n}\right)^{1/2} \tag{2.16}$$

Dengan n adalah jumlah data yang diprediksi dan y adalah nilai hasil prediksi.

### 2.2.7 *Synthetic Minority Over-sampling Technique* (SMOTE)

Menurut Elreedy dkk, 2022, suatu dataset dianggap tidak seimbang jika terdapat perbedaan proporsi yang signifikan dalam jumlah sampel antar kelas. Kondisi ini menyebabkan model klasifikasi (*classifier*) menjadi bias, karena model tersebut secara alami akan lebih condong memprediksi kelas mayoritas untuk mencapai akurasi keseluruhan yang tinggi. Akibatnya, kemampuan model untuk mengenali sampel dari kelas minoritas (sensitivitasnya) menjadi sangat buruk, meskipun performanya pada kelas mayoritas tampak baik. Untuk mengatasi hal ini dan meningkatkan performa model secara adil, diperlukan penerapan metode khusus yang dirancang untuk menangani ketidakseimbangan data. Untuk menangani *dataset* yang tidak seimbang, metode SMOTE (*Synthetic Minority Over-sampling Technique*) merupakan salah satu teknik paling populer dan efektif untuk mengatasi masalah ketidakseimbangan data. Teknik ini bekerja dengan cara menciptakan sampel-sampel sintetik baru untuk kelas minoritas. Pembuatan data sintetik ini dilakukan melalui proses interpolasi linier di antara sebuah sampel minoritas dengan beberapa tetangga terdekatnya (*K-nearest neighbors*).

Sebagai teknik *oversampling,* SMOTE dibuat untuk menambah populasi kelas minoritas bukan dengan pengambilan sampel ulang, melainkan dengan menghasilkan sampel sintetik. Proses ini memanfaatkan informasi dari K-tetangga terdekat di dalam kelas minoritas untuk menciptakan data baru di antara titik-titik data yang sudah ada. Dengan demikian, SMOTE mampu memperbesar dan memperlebar daerah keputusan untuk kelas minoritas. Hasil akhirnya adalah sebuah model klasifikasi yang lebih general dan memiliki kinerja yang lebih baik dalam mengidentifikasi sampel dari kelas minoritas yang sebelumnya kurang terwakili (Chawla, 2002).

Menurut Chawla dkk, 2002, SMOTE menghasilkan sampel minoritas sintetis dengan cara memanipulasi fitur-fitur dari data minoritas yang ada untuk menciptakan contoh baru yang realistis secara statistik. Desripsi algoritma over sampling–SMOTE yang dikembangkan oleh Chawla dkk, 2002 untuk menyeimbangkan data meliputi beberapa tahapan utama sebagai berikut:

1. Identifikasi Sampel Minoritas Pertama:

   Pilih sampel minoritas sebanyak $x_0$ secara acak.

   Tentukan dan identifikasi semua sampel dari kelas minoritas dalam *dataset*. Sampel ini menjadi basis untuk pembuatan contoh sintetik. Kelas minoritas adalah kelas dengan jumlah sampel paling sedikit dalam dataset.

2. Parameter *Over-sampling*

   Tentukan jumlah kelas minoritas yang diinginkan, biasanya dalam bentuk persentase, misalnya 100%, 200%, dan seterusnya. Parameter ini menentukan berapa banyak sampel sintetik yang akan dibuat.

3. Mencari tetangga terdekat

   Untuk setiap sampel minoritas, cari sejumlah tetangga terdekat (k tetangga, biasanya dipilih secara tetap). Biasanya, ini dilakukan menggunakan algoritma jarak *Euclidean* dan kriteria *k-nearest neighbors*.

   Jarak Euclidean merupakan ukuran jarak yang paling umum digunakan untuk mengukur kedekatan antara dua titik dalam ruang multidimensi. Secara matematis, jarak ini dihitung berdasarkan panjang garis lurus yang menghubungkan kedua titik tersebut.

Dalam konteks data fitur, jika kita memiliki dua sampel dengan vektor fitur $x = (x_1, x_2, \dots, x_d)$ dan $y = (y_1, y_2, \dots, y_d)$, maka jarak *Euclidean* antara keduanya ($d(x,y)$) dihitung sebagai:

$$d(x,y) = \sqrt{\sum_{i=1}^{d}(x_i - i)^2} \quad (2.17)$$

Jarak *Euclidean* sering digunakan dalam algoritma *machine learning* dan *data mining*, termasuk dalam proses pencarian tetangga terdekat (*nearest neighbor*) seperti dalam SMOTE, karena sifatnya yang intuitif dan efisien dalam ruang fitur berdimensi rendah hingga sedang.

Adapun langkah-langkah seleksi tetangga menurut Elreedy dkk, 2022 adalah sebagai berikut:

a. Menghitung jarak antar sampel

   Untuk setiap sampel minoritas $x_0$, dihitung jaraknya ke semua sampel minoritas lainnya menggunakan jarak *Euclidean*:

   $$r_i = \|x_i - x_0\| \quad (2.18)$$

   Dengan

   $x_i$= sampel minoritas lain

   $r_i$ = jarak dari $x_0$ ke $x_i$

b. Mengurutkan jarak dan memilih tetangga terdekat, dengan cara: menurutkan semua jarak *ri* diurutkan dari yang terkecil ke terbesar, kemudian, dipilih *K* tetangga terdekat berdasarkan jarak yang telah diurutkan tersebut. Tetangga-tetangga ini biasanya diindeks dari 1 sampai *K*.

c. Pemilihan tetangga secara acak dari *K* tetangga terdekat untuk proses interpolasi. Pada setiap iterasi pembuatan sampel sintetis, dapat dipilih tetangga secara acak dari antara *K* tetangga yang ada. Pemilihan ini dilakukan dengan probabilitas yang sama yaitu, setiap tetangga memiliki peluang $\frac{1}{K}$.

d. Setelah tetangga dipilih, dilakukan proses interpolasi antara sampel minoritas $x_0$ dan tetangga yang terpilih. Koordinat dari sampel sintetis $z$ dihitung sebagai campuran linier.

$$Z = (1 - w)x_0 + w_{xk} \quad (2.19)$$

Dengan $Z$ : sampel sintetis

$xk$ : tetangga yang dipilih

$w$ : adalah faktor interpolasi yang acak antara 0 dan 1.

Dalam proses seleksi tetangga terdekat (k-*nearest neighbors* atau k-NN), kriteria utama yang digunakan adalah jarak antara sampel yang sedang diproses dengan sampel lainnya dalam ruang fitur.

4. Pembuatan sampel sintetik untuk setiap sampel minoritas yang diproses:
   a. Pilih salah satu tetangga terdekat secara acak.
   b. Buat sampel sintetik baru dengan cara interpolasi linier antara sampel asal dan tetangga yang dipilih. Itu dilakukan dengan menambahkan ke semua fitur nilai hasil perkalian selisih antara keduanya dan angka acak antara 0 dan 1, sehingga sampel sintetik berada di antara keduanya dalam ruang fitur.
5. Buat sampel sintetik baru dengan cara interpolasi linier antara sampel asal dan tetangga yang dipilih. Itu dilakukan dengan menambahkan ke semua fitur nilai hasil perkalian selisih antara keduanya dan angka acak antara 0 dan 1, sehingga sampel sintetik berada di antara keduanya dalam ruang fitur.

   Lakukan interpolasi linier antara $X_0$ dan tetangga yang dipilih untuk membuat sampel sitetis baru z sesuai dengan

   $$z = (X_0 + (X_k - X_0) \quad (2.20)$$

   Dengan

   w adalah variabel acar seragam dalam rentang (0,1)

   $X_0$ adalah sampel minoritas

   $X_k$ adalah tetangga yang dipilih

6. Penggabungan Data Setelah sampel sintetik dibuat,

   Gabungkan keduanya dengan dataset asli sehingga jumlah data class minoritas meningkat sesuai kebutuhan. Hal ini akan menyeimbangkan data dengan memperbesar representasi kelas minoritas dalam *dataset*.

Dengan cara ini, SMOTE tidak hanya memperbanyak data minoritas secara numerik tetapi juga memperluas daerah *decision boundary* dari kelas minoritas, sehingga memperbaiki performa *classifier* terhadap kelas yang kurang direpresentasikan.

Misalkan,

$x_i$ adalah vektor fitur dari sampel minoritas asli,

$x_{zi}$ adalah vektor fitur dari salah satu tetangga terdekat $z$ dari $x_i$,

δ adalah bilangan acak dari distribusi *uniform* antara 0 dan 1.

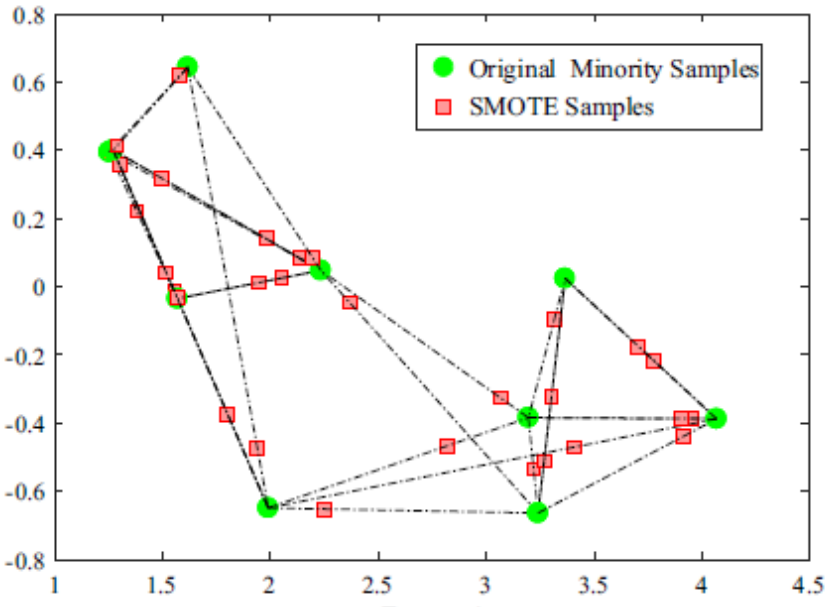
Maka, sampel sintetik $x_{new}$ dihitung dengan rumus:

$$x_{new} = x_i + \delta \times (x_{zi} - x_i) \tag{2.21}$$

Secara komponen, untuk setiap fitur j:

$$X_{new}^{(j)} = X_i^{(j)} + \delta \times (X_{zi}^{(j)} - X_i^{(j)}) \tag{2.22}$$

Rumus ini menghasilkan sampel baru di antara $x_i$ dan $x_{zi}$ dalam ruang fitur, tergantung pada nilai acak δ. Jika δ=0, maka sampel baru sama dengan$x_i$; jika δ=1, sama dengan tetangganya $x_{zi}$.



Gambar 2.6 Mekanisme interpolasi SMOTE (Elreedy dan Atiya, 2019)

Berdasarkan visualisasi pada Gambar 2.6, menampilkan sampel minoritas asli dan pola yang dihasilkan SMOTE. Terlihat bahwa mekanisme SMOTE menghasilkan sampel baru yang terletak tepat pada garis penghubung antara data minoritas dan tetangga terdekatnya (K-*nearest neighbors*). Penempatan pola ini mengakibatkan distribusi data menjadi lebih terkontraksi dibandingkan distribusi aslinya, selaras dengan temuan Elreedy dan Atiya (2019). Hal ini memperkuat premis bahwa sampel yang dihasilkan SMOTE tidak selalu merepresentasikan kepadatan asli dari kelas minoritas tersebut.

### 2.2.8. Optimasi *Hyperparameter*

Sebuah proses untuk mengidentifikasi nilai *hyperparameter* yang paling optimal dari suatu algoritma *machine learning* untuk menciptakan model dengan performa maksimal. *Hyperparameter* yang tepat sangat penting karena dapat mengendalikan perilaku algoritma pelatihan dan memengaruhi kinerja model secara signifikan. Nilai optimal untuk satu masalah belum tentu menjadi nilai terbaik untuk masalah lain, karena nilai-nilai ini dapat berubah tergantung pada *dataset* yang digunakan. Tujuan dari optimasi *hyperparameter* adalah untuk meminimalkan kesalahan generalisasi model yang diharapkan. Beberapa strategi optimasi *hyperparameter* yang umum digunakan adalah:

1. Metode *Grid Search,*

   *Grid Search* merupakan metode konvensional yang paling mudah dipahami untuk mengoptimalkan *hyperparameter*. Cara kerjanya adalah dengan menguji setiap kombinasi nilai. *hyperparameter* yang telah ditetapkan dalam sebuah "grid". Algoritma ini akan melatih model untuk semua kombinasi tersebut, dan kinerjanya diukur menggunakan *cross-validation* pada data latih. Tujuannya adalah untuk memastikan model dapat mempelajari sebagian besar pola dari dataset. Keunggulan *Grid Search* adalah kemampuannya menjamin penemuan *hyperparameter* terbaik. Akan tetapi, metode ini memiliki kelemahan signifikan, yaitu konvergensi yang lambat dan masalah dimensionalitas, karena

kompleksitas komputasinya meningkat secara eksponensial seiring dengan bertambahnya jumlah parameter dan nilai yang diuji.

2. Metode *Random Search,*

   *Random Search* adalah alternatif dari *Grid Search* yang melakukan pencarian tidak menyeluruh. Dalam metode ini, algoritma mengambil sampel kombinasi *hyperparameter* secara acak dari distribusi probabilitas yang telah ditentukan. Tujuannya adalah untuk menemukan solusi terbaik menggunakan kombinasi acak *hyperparameter*. *Random Search* adalah O(n) untuk n evaluasi. Namun, kelemahan utamanya adalah tidak menggunakan informasi dari percobaan sebelumnya untuk memilih set berikutnya, sehingga tidak memiliki strategi untuk memprediksi percobaan selanjutnya.

3. Metode *Bayesian Optimization*

   *Bayesian Optimization* adalah algoritma pencarian yang belajar dari setiap iterasi sebelumnya. Meskipun mirip dengan *Random Search* karena sama-sama mengambil sampel *subset* kombinasi *hyperparameter*, keduanya berbeda dalam cara setiap kombinasi dipilih.

   Metode ini menganggap proses *hyperparameter tuning* sebagai optimasi dari sebuah "fungsi kotak hitam" (*black-box function*). Fungsi yang dioptimalkan adalah skor prediksi akhir model (misalnya, akurasi) pada set pengujian yang disimpan. Algoritma ini membangun model probabilistik (*surrogate model*) dari fungsi tujuan, dan yang paling umum digunakan adalah *Gaussian Process*. Prosesnya dimulai dengan memilih dan menguji beberapa kombinasi *hyperparameter* secara acak. Nilai-nilai ini kemudian digunakan untuk membuat draf pertama dari model fungsi tujuan. Setelah model awal ini dibuat, ada dua cara untuk memilih kombinasi berikutnya: memilih nilai yang dekat dengan nilai tertinggi yang sudah didapat (eksploitasi) atau menjelajahi *subset* lain dari ruang pencarian *hyperparameter* untuk mencari titik maksimum lainnya (eksplorasi).

### 2.2.9 Evaluasi Kinerja Model

Menurut Sathyanarayanan dan Tantri R, 2024, dalam berbagai masalah klasifikasi, sumber utama yang digunakan untuk mengukur akurasi model adalah *Confusion Matrix* (matriks konfusi), yang juga dikenal sebagai matriks klasifikasi. Matriks ini merupakan alat penting dalam evaluasi kinerja algoritma klasifikasi, karena memberikan gambaran yang jelas tentang bagaimana model telah melakukan prediksi terhadap data yang diuji. Matriks Konfusi menyajikan informasi dalam bentuk tabel yang menunjukkan jumlah prediksi yang benar dan salah, dibagi berdasarkan kelas yang berbeda. Dengan demikian, matriks ini memungkinkan analisis mendetail mengenai berbagai kategori, seperti *true positives* (TP), *true negatives* (TN), *false positives* (FP), dan *false negatives* (FN). Matriks konfusi 2x2 ditunjukkan oleh Tabel 2.6.

Tabel 2.6 Matriks Konfusi 2x2

| | | Aktual | |
|---|---|---|---|
| | | 1 (positive) | 0 (negative) |
| Prediksi | 1(positive) | **TP** *(True Positif)* | **FP** *(False Positif)* |
| | 0 (negative) | **FN** *(False Negatif)* | **TN** *(True Negative)* |



Matriks konfusi 2x2 terdiri atas *true positive (TP), false negatif (FN), false positive (FP)* dan *true negatif (TN)*. Pada TP merupakan data positif yang terdeteksi benar. FN merupakan kebalikan dari *true positive*, sehingga data positif, namun terdeteksi sebagai data negatif. Untuk FP sendiri, merupakan data negatif namun terdeteksi sebagai data positif, sedangkan TP merupakan data negatif yang terdeteksi dengan benar. Maka dapat dihitung nilai *accuracy, precission, recall, dan F-1 score*.

a) *Accuracy* menggambarkan seberapa akurat model dalam mengklasifikasikan dengan benar. Nilai *accuracy* ditunjukkan oleh persamaan (2.23).

$$Accuracy = \frac{(TP+TN)}{(TP+FP+FN+TN)} \tag{2.23}$$

b) *Precision* menggambarkan akurasi antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Nilai *Precision* ditunjukkan oleh persamaan (2.24).

$$Precission = \frac{(TP)}{(TP+FP)} \tag{2.24}$$

c) *Recall* atau *sensitivity*: menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Nilai *recall* ditunjukkan oleh persamaan (2.25).

$$Recall = \frac{(TP)}{(TP+FN)} \tag{2.25}$$

d) *F-1 Score* menggambarkan perbandingan rata-rata *precision* dan *recall* yang dibobotkan. *Accuracy* tepatnya digunakan sebagai acuan performansi algoritma jika *dataset* memiliki jumlah data *False Negatif* dan *False Positif* yang sangat mendekati (*symmetric*). Namun jika jumlahnya tidak mendekati, maka sebaiknya menggunakan *F1 Score* sebagai acuan. Nilai *F1 Score* ditunjukkan oleh persamaan (2.26).

$$F-1\ SCORE = \frac{(2*Recall*Precission)}{(Recall+Precission)} \tag{2.26}$$

Data angka-angka pada diagonal dari kiri ke kanan bawa adalah hasil prediksi yang benar, dan angka-angka di luar diagonal adalah hasil prediksi yang salah. Salah satu cara untuk melakukan klasifikasi adalah dengan menggunakan algoritma klasifikasi.

Matriks konfusi dapat diperluas menjadi ukuran yang lebih besar (multilevel matriks konfusi) untuk mengevaluasi kinejra model klasifikasi dengan lebih banyak kelas. Matriks $n \times n$, dimana $n$ adalah jumlah kelas serta data ($\boldsymbol{i}$,$\boldsymbol{j}$) merepresentasikan jumlah elemen dari kelas $\boldsymbol{i}$ dan terprediksi sebagai sebagai kelas $\boldsymbol{j}$ . Misalya dalam matriks konfusi 5x5 terdapat 5 baris dan 5 kolom yang mempresentasikan 5 kelas

yang berbeda, sehingga memungkinka analisis terjadinya kesalahan kasifikasi yang lebih kompleks dan detail. Matriks konfusi 5x5 ditunjukkan oleh Tabel 2.7.

Tabel 2.7 Matriks Konfusi 5x5

| | | AKTUAL | | | | |
|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 |
| PREDIKSI | 0 | C0,0 (TP) | C0,1 | C0,2 | C0,3 | C0,4 |
| | 1 | C1,0 | C1,1 (TP) | C1,2 | C1,3 | C1,4 |
| | 2 | C2,0 | C2,1 | C2,2 (TP) | C2,3 | C2,4 |
| | 3 | C3,0 | C3,1 | C3,2 | C3,3 (TP) | C3,4 |
| | 4 | C4,0 | C4,1 | C4,2 | C4,3 | C4,4 (TP) |

Tabel 2.7 mempresentasikan struktur teoretis matriks konfusi untuk klasifikasi multi-kelas dengan lima kategori (stadium 0 hingga stadium 4). Matriks ini berfungsi sebagai alat ukur untuk memetakan hasil prediksi model terhadap label aktual dari data riil. Format baris dalam tabel ini merepresentasikan Kelas Aktual (kondisi medis pasien yang sesungguhnya), sedangkan kolom merepresentasikan Kelas Prediksi (hasil klasifikasi yang dikeluarkan oleh algoritma *Extreme Learning Machine*).

Elemen-elemen yang terdapat di dalam tabel tersebut dapat diinterpretasikan sebagai berikut:

1. Diagonal Utama: Sel yang ditarik dari pojok kiri atas ke kanan bawah menunjukkan jumlah data yang berhasil diklasifikasikan dengan benar (*True Positive* untuk setiap kelas). Semakin besar nilai pada diagonal ini, semakin tinggi tingkat akurasi model dalam membedakan setiap stadium risiko PJK.
2. Diagonal utama (dari kiri atas ke kanan bawah mempresentasikan jumlah prediksi yang benar untuk setiap kelas.
   - (0,0) : jumlah prediksi benar kelas 0
   - (1,1) : jumlah prediksi benar kelas 1
   - (2,2) : jumlah prediksi benar kelas 2
   - (3,3) : jumlah prediksi benar kelas 3
   - (4,4) : jumlah prediksi benar kelas 4
3. Elemen di Luar Diagonal: Sel-sel di luar garis diagonal menunjukkan adanya kesalahan klasifikasi (*misclassification*). yaitu prediksi yang salah untuk setiap kelas. Sel (*i,j*) jumlah sampel kelas aktual *i* yang diprediksi sebagai kelas *j*. Sel (*j,i*) jumlah sampel kelas aktual *j* yang diprediksi sebagai kelas *i*. Misalnya, sel pada Baris 1 Kolom 2 menunjukkan jumlah pasien yang secara aktual berada di Stadium I namun diprediksi secara keliru oleh sistem sebagai Stadium II.
4. Jumlah baris dan kolom mempresentasikan total sampel utuk satiap kelas aktual dan prediksi.
5. Analisis Error: Melalui tabel ini, peneliti dapat mengidentifikasi kecenderungan model dalam melakukan kesalahan, apakah kesalahan tersebut

terjadi secara acak atau cenderung menumpuk pada stadium-stadium tertentu yang memiliki kemiripan fitur klinis.