BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Penelitian mengenai *stacking hybrid* telah diterapkan di berbagai domain dan secara umum menunjukkan bahwa penggabungan beberapa algoritma mampu meningkatkan akurasi dan stabilitas prediksi dibanding model tunggal. Di domain lingkungan, Di Nunno dkk. menggabungkan *MLP* dan *Random Forest* untuk memprediksi suhu permukaan delapan danau di Polandia selama 30 tahun. Model *MLP–RF* secara konsisten melampaui *MLP* tunggal maupun *WT-MLP* pada berbagai rentang waktu dan kondisi geografis (Di Nunno dkk., 2023). Pada keamanan jaringan, Shafieian dan Zulkernine mengusulkan *stacking* bertingkat dengan beberapa *base-learner* dan *MLP* sebagai *meta-learner*. Model tersebut mengungguli model tunggal, *boosting*, dan *bagging* pada akurasi, *AUC*, dan *F1-score*, dengan *false positive rate* serendah 0,001 dan biaya komputasi lebih rendah dibanding *deep learning* (Shafieian dan Zulkernine, 2023).

Di ranah peramalan beban listrik jangka pendek, Zhao dkk. mengusulkan *RF-TStacking* yang mengombinasikan *Random Forest* dan *LightGBM* dengan *Bayesian regression* sebagai *meta-learner*, disertai pemilihan fitur berbasis pentingnya fitur *RF* dan penyetelan *hyperparameter* berbasis *Bayesian optimization*; pendekatan ini meningkatkan akurasi hingga 3,53% dan menurunkan *error* 8,17% dibanding *XGBoost*, *LightGBM*, dan *CNN* (Zhao dkk., 2023). Pada domain *IDS*, Ahmed dkk. mengembangkan *HAEnID* yang menggabungkan *stacking ensemble*, *Bayesian Model Averaging*, dan *Conditional Ensemble Method* pada *CIC-IDS2017*, menggunakan *SMOTE* serta *SHAP* dan *LIME* untuk meningkatkan deteksi dan interpretabilitas; model ini mencapai akurasi *multiclass* 98,79% dengan *false positive* rendah, meskipun belum secara khusus mengkaji serangan web seperti *SQL Injection* dan *XSS* (Ahmed dkk., 2024).

Sejalan dengan bukti efektivitas *stacking hybrid*, sejumlah studi menempatkan *LightGBM* sebagai komponen penting karena kombinasi akurasi dan efisiensi komputasinya. Pada data klinis besar yang tidak seimbang, *LightGBM* dilaporkan melampaui *Decision Tree*, *Random Forest*, *XGBoost*, dan *CatBoost* dengan akurasi 78%, *AUC* 0,85, dan *F1-score* 0,79 setelah rekayasa fitur yang tepat (Boudali dkk., 2024). Pada deteksi serangan *WiFi realtime*, *LightGBM* juga menunjukkan keunggulan dari sisi akurasi dan kecepatan, dengan pelatihan hingga 26 kali lebih cepat dan pengujian 20% lebih cepat dibanding *XGBoost*, serta waktu inferensi rata-rata 121 mikrodetik sehingga layak untuk operasi berskala besar (Bhutta dkk., 2024).

Kinerja serupa muncul pada skenario keamanan jaringan lain; dalam arsitektur *SD-IoT* untuk deteksi *DDoS*, *LightGBM* mencapai akurasi 98,67% dengan *FAR* 0% dan *AUC* 0,996, mengungguli algoritma pembanding pada kontroler *SDN* (Chauhan dan Atulkar, 2023). Pada *IDS* berbasis *NSL-KDD*, *LightGBM* meraih akurasi 98,3%, *precision* 98,9%, *recall* 97,4%, AUC 0,981, dan *false positive* 0,9%, menunjukkan kemampuan menangani data jaringan yang besar dan kompleks secara cepat dan akurat (Khafajeh, 2020).

Random Forest juga banyak digunakan pada *IDS* karena ketangguhan dan efisiensinya untuk data *network flow*. Pada penilaian risiko siber dengan *CIC-IDS2017*, penyetelan jumlah pohon dan mekanisme *bagging* mampu mendorong akurasi hingga 96,94% dengan rata-rata F1-score 97,86% tanpa harus membesarkan ukuran hutan secara berlebihan (Jeamaon dan Khemapatapan, 2024). Kombinasi seleksi fitur dan *Random Forest* di *NSL-KDD* menunjukkan bahwa reduksi fitur sekitar 50% melalui *CFS* yang dioptimasi *PSO* tetap mempertahankan akurasi 98,78%, *detection rate* 98,84%, dan FAR 1,3%, melampaui *Naive Bayes*, *SGD*, *deep learning*, *KNN*, dan *SVM* (Safa dkk., 2024). Pendekatan *hybrid* yang memadukan *GOA* dan algoritma genetika sebelum pelatihan *Random Forest* di *UNSW-NB15*, *CIC-DDoS2019*, dan *CIC Bell DNS EXF 2021* bahkan mencapai akurasi hingga 98%, 99%, dan 92%, dengan peningkatan *true positive rate* dan penurunan *false positive rate* dibanding *SVM* dan *Logistic*

Regression (Bakro dkk., 2024). Pada skenario yang lebih ketat secara sumber daya, *Random Forest* tetap kompetitif; dalam pemantauan keamanan pada *drone* dengan *dataset UNSW-NB15*, *Random Forest* dilaporkan mencapai akurasi 81,59% dengan konsumsi energi rendah dan inferensi cepat di CPU, sehingga layak untuk perangkat dengan daya dan kapasitas komputasi terbatas ketika dibandingkan dengan *CNN* dan *DNN* (Slimani dkk., 2025).

Pada konteks serangan web, beberapa penelitian mulai berfokus khusus pada subset *Web Attack* di *CIC-IDS2017*. Mohammed Hashem Almourish, dkk. mengembangkan *IDS* berbasis pendekatan anomali menggunakan *CIC-IDS2017* yang memuat beberapa serangan web, termasuk *Brute Force, Cross-site Scripting (XSS)*, dan *SQL Injection (SQLI)*. Mereka mengevaluasi berbagai *classifier* klasik seperti *Decision Tree, Naive Bayes, k-Nearest Neighbors, Random Forest*, dan *AdaBoost*, dengan rerata akurasi masing-masing sekitar 95%, 98%, 96%, 96%, dan 98%, sehingga menunjukkan bahwa model ML tradisional sudah cukup kuat untuk mendeteksi serangan web pada skenario tertentu (Almourish dkk., 2022). Pendekatan yang berbeda diajukan oleh Tang, dkk. yang memfokuskan deteksi serangan terhadap situs web, khususnya ancaman umum dalam daftar *OWASP*. Mereka mengusulkan model berbasis DAE (*Deep Auto-Encoder*) yang digabungkan dengan konsep *Magnet Loss* di ruang laten untuk memaksimalkan margin antara wilayah normal dan serangan. Dengan menggunakan *dataset CIC-IDS2017* dan *CSE-CIC-IDS2018*, model ini berhasil mencapai peningkatan *Accuracy*, *Precision*, *Recall*, dan *F1-score* dibanding pendekatan sebelumnya, sekaligus menurunkan waktu komputasi pada fase pengujian (Tang dkk., 2024). Namun, studi ini belum mengintegrasikan kerangka stacking hybrid, penanganan ketidakseimbangan kelas yang eksplisit, maupun analisis interpretabilitas model.

Penanganan ketidakseimbangan kelas melalui *SMOTE* terbukti krusial untuk menjaga akurasi dan ketahanan deteksi *IDS*. Pada skenario *IoT realtime* dengan *dataset IoT-23*, penerapan *SMOTE* meningkatkan representasi kelas minor sehingga *XGBoost* mencapai akurasi 98,89% dan *Random Forest* 98,54% dengan waktu prediksi sekitar 0,0311 detik, menandakan kesiapan untuk respons cepat di lapangan (Alrefaei dan Ilyas, 2024). Efek serupa tampak pada *RF-SMOTE* di *N-BaIoT* dan *NSL-KDD*, di mana *oversampling* sintetis mendorong akurasi hingga 99,98% dan *AUC* 100% pada *N-BaIoT* dengan *FAR* 0,08%, serta akurasi 98,82% dan *AUC* 99,91% pada *NSL-KDD*, melampaui *Decision Tree* dan *SVM* (Dankan Gowda dkk., 2023). Pada konteks deteksi kejahatan siber di lingkungan *cloud*, kombinasi *SVM* dan *SMOTE* dalam kerangka *IDSEM* meningkatkan representasi kelas minor, menaikkan akurasi, dan menurunkan kesalahan deteksi untuk kasus seperti *phishing* dan *vishing* (Kishore dan Bhaskari, 2023).

Di sisi seleksi fitur, *ANOVA F-Test* dilaporkan efektif sebagai metode filter yang cepat dan interpretatif; pada skenario *IoT*, *ANOVA* digunakan untuk merangking fitur berdasarkan rasio keragaman antar dan intra kelas, lalu memilih fitur dengan *p-value* signifikan sehingga metrik klasifikasi meningkat dan waktu latih berkurang, termasuk pada pengujian di perangkat terbatas seperti *Raspberry Pi* (Musthafa dkk., 2024). Pada jaringan nirkabel, pemilihan fitur dengan *ANOVA F-value* menyisakan puluhan atribut paling informatif dan mendorong deteksi ancaman *WLAN* yang lebih cepat dan hemat energi (Natkaniec dan Bednarz, 2023).

Aspek interpretabilitas model *IDS* juga semakin banyak mendapat perhatian. Penggunaan *SHAP* dilaporkan mampu menyediakan penjelasan global dan lokal yang konsisten atas keputusan klasifikasi, membantu mengidentifikasi fitur lalu lintas paling berpengaruh dan menyejajarkannya dengan pola serangan yang relevan. Studi *XAI-IDS* menunjukkan bahwa penyajian pentingnya fitur berbasis *SHAP* pada *CIC-IDS2017* dan *NSL-KDD* memberikan wawasan yang dapat ditindaklanjuti bagi analis tanpa beban komputasi yang berarti, sehingga layak untuk skenario mendekati *realtime* (Arreche

dkk., 2024). Temuan komparatif lainnya menegaskan bahwa *SHAP* unggul sebagai alat interpretasi yang reliabel dalam konteks forensik keamanan karena meningkatkan keterlacakan, akuntabilitas, dan kesesuaian bukti pada *IDS* berbasis *machine learning* (Hermosilla dkk., 2025). Pada studi lain, integrasi *SHAP* dalam kerangka *ensemble* untuk *CIC-IDS2017* terbukti membantu memvalidasi *subset* fitur dan memperkuat konsistensi performa ketika dikombinasikan dengan teknik pra-proses modern (Ahmed dkk., 2024).

Berdasarkan sintesis literatur tersebut, dapat disimpulkan bahwa penelitian terdahulu telah membuktikan efektivitas *stacking hybrid* untuk meningkatkan akurasi dan stabilitas *IDS*, menunjukkan bahwa *LightGBM* dan *Random Forest* sama-sama kuat untuk data jaringan berskala besar, menegaskan peran penting *SMOTE*, seleksi fitur berbasis *ANOVA F-Test*, dan interpretabilitas dengan *SHAP* dalam meningkatkan kualitas deteksi intrusi. Namun demikian, hasil penelusuran sistematis yang peneliti lakukan pada *database Scopus*, *IEEE Xplore*, dan *ScienceDirect* dengan kombinasi kata kunci seperti "*CIC-IDS2017 Web Attack*", "*web attack classification*", "*stacking ensemble LightGBM Random Forest*", "*SHAP ANOVA IDS*" menunjukkan bahwa:

1) Belum ditemukan penelitian yang secara spesifik membangun kerangka *IDS* serangan web berbasis *stacking hybrid* dengan mengombinasikan *LightGBM* dan *Random Forest* sebagai *base-learner* serta *MLP* sebagai *meta-learner* yang ditujukan khusus untuk kelas *SQL Injection*, *XSS*, dan *Brute Force* pada *subset Web Attack* dari *CIC-IDS2017*.
2) Integrasi yang eksplisit antara *SMOTE* yang diterapkan secara ketat hanya pada data latih, seleksi fitur berbasis *ANOVA F-Test* dengan target pengurangan latensi, serta analisis interpretabilitas berbasis *SHAP* yang dipadukan secara sistematis dengan hasil *ANOVA* pada konteks serangan web belum tersedia.
3) Pelaporan metrik sumber daya seperti latensi inferensi dan penggunaan memori pada lingkungan *VPS* untuk *model stacking hybrid IDS* serangan web juga masih terbatas.

Pemetaan hubungan penelitian terdahulu dan posisi penelitian ini terhadap kombinasi dataset, pendekatan algoritmik, serta teknik pra-proses dan interpretabilitas disajikan secara ringkas pada Tabel 2.1.

Tabel 2.1 Pemetaan Penelitian Terdahulu

	Generic IDS	*Domain* / *Dataset* Khusus Bukan Web	Serangan Web
Model Klasik	• Khafajeh (2020): *LightGBM, NSL-KDD.*	• Slimani dkk. (2025): *RF* pada *drone, UNSW-NB15*.	• Almourish Mohammed Hashemand Abduljalil (2022): *ML* dan *Deep learning*.
Stacking Ensemble	• Shafieian & Zulkernine (2023): *Stacking, MLP meta-learner*.	• Zhao dkk. (2023): *RF-TStacking.* • Di Nunno dkk. (2023): *MLP–RF Stacking.*	-
Non-stacking + komponen lanjutan	• Dankan Gowda dkk. (2023): *RF-SMOTE, N-BaIoT, NSL-KDD.* • Arreche dkk. (2024): *SHAP* • Hermosilla dkk. (2025): *SHAP*	• Musthafa dkk. (2024): *ANOVA F-Test di IoT*. (Raspberry Pi). • Alrefaei & Ilyas (2024): *XGBoost, RF, SMOTE* pada *IoT-23*. • Kishore & Bhaskari (2023): *SVM, SMOTE* (*IDSEM*). • Natkaniec & Bednarz (2023): *ANOVA F-value, WLAN*.	• Tang dkk., (2024): *DMAE, RF classifiers.*
Stacking + komponen lanjutan	• Ahmed dkk. (2024): *HAEnID, SMOTE, SHAP, LIME*.	-	• Penelitian ini: *Stacking* (*LGBM, RF, MLP*), *SMOTE, ANOVA F-Test, SHAP, Latency Memory*.

Tabel 2.1 memetakan penelitian terdahulu berdasarkan dua dimensi, yaitu fokus *dataset* (*generic IDS*, *domain/dataset* khusus non-web, dan serangan web) serta jenis pendekatan (model klasik, *stacking, non-stacking* dengan komponen lanjutan, dan *stacking* dengan komponen lanjutan). Baris 1–3 menunjukkan bahwa berbagai studi telah mengeksplorasi model klasik, *stacking*, serta penambahan *SMOTE*, seleksi fitur, *XAI*, atau fokus *resource* pada *dataset generic IDS* maupun domain khusus bukan web, dan sebagian mulai menyentuh serangan web *CIC-IDS2017*, tetapi masih terpisah-pisah dan umumnya tanpa kerangka *stacking* yang lengkap. Baris ke-4 memperlihatkan bahwa integrasi *stacking* dengan komponen lanjutan secara utuh baru terlihat pada *HAEnID* (*generic IDS*) dan belum ditemukan pada domain khusus bukan web maupun serangan web. Pada sel paling kanan bawah, penelitian ini diposisikan sebagai penelitian yang mengisi gap tersebut karena secara khusus memodelkan serangan web *CIC-IDS2017 Web Attack* dengan *Stacking Hybrid LightGBM* dan *Random Forest* serta *MLP* yang terintegrasi dengan *SMOTE*, seleksi fitur *ANOVA F-Test*, *SHAP*, serta evaluasi latensi dan *memory footprint* di lingkungan *VPS*.

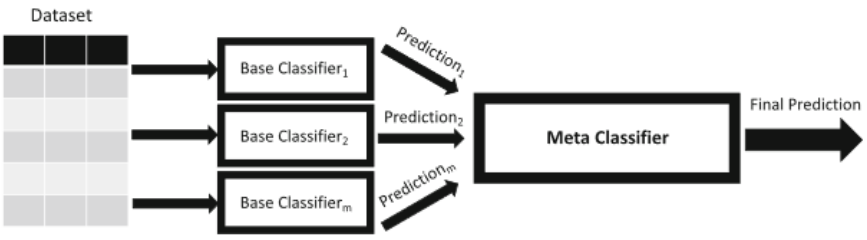
2.2 Dasar Teori

2.2.1 *Stacking Classifier*

Stacking Classifier adalah metode *ensemble* dalam *Machine Learning* yang menggabungkan beberapa *base-learners* dengan menggunakan sebuah *meta-learner* untuk meningkatkan akurasi prediksi. Pada tingkat pertama, model dasar membuat prediksi yang kemudian digunakan sebagai masukan untuk model meta. Model meta ini akan mempelajari bagaimana menggabungkan prediksi dari model dasar untuk membuat prediksi akhir yang lebih akurat. Pendekatan ini sering digunakan untuk meningkatkan performa klasifikasi dengan cara memanfaatkan kekuatan dari beberapa algoritma yang berbeda (Alexandropoulos dkk., 2019).

Keunggulan utama dari metode *Stacking Classifier* dalam berbagai penelitian adalah kemampuannya untuk menggabungkan kekuatan beberapa model dasar. Dibandingkan dengan metode *ensemble* lainnya, seperti *Bagging* atau *boosting*, *Stacking* memberikan fleksibilitas lebih besar dengan menggunakan *meta-learner* yang mampu mempelajari kelemahan dan kelebihan dari setiap *base classifier* yang berbeda. Hal ini menghasilkan kinerja yang lebih *robust*, terutama pada dataset yang kompleks dan bervariasi. Menurut beberapa studi, *Stacking* menunjukkan kinerja yang lebih baik daripada model tunggal atau bahkan beberapa metode *ensemble* lain dalam sebagian besar kasus. Keuntungan lain dari *Stacking* adalah kemampuannya untuk mengurangi *overfitting* pada data pelatihan dengan memanfaatkan informasi dari berbagai model. Selain itu, kombinasi model yang berbeda memungkinkan *Stacking* untuk menangkap pola yang mungkin tidak ditemukan oleh model tunggal (Džeroski dan Ženko, 2004).

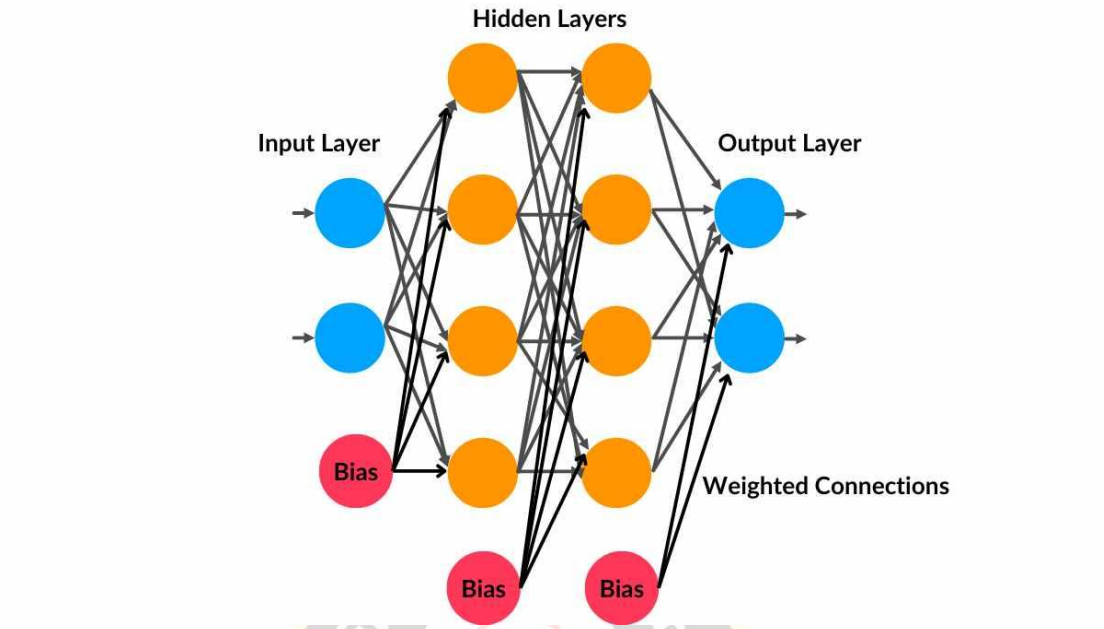
Stacking memiliki beberapa perbedaan utama dibandingkan dengan metode *Bagging* dan *boosting*. Salah satunya adalah jenis model yang digunakan. Dalam *Stacking*, model yang digunakan biasanya beragam atau heterogen, yang berarti setiap model dasar memakai algoritma yang berbeda-beda. Sebaliknya, *Bagging* dan *boosting* cenderung menggunakan model yang sama. Selain itu, cara menggabungkan hasil prediksi dari model dasar juga berbeda. *Bagging* dan *boosting* menggunakan metode seperti *majority* voting atau rata-rata, yang bersifat langsung. Di sisi lain, *Stacking* melibatkan pelatihan model tambahan yang disebut *meta-classifier*/*meta-learner*, yang bertugas menggabungkan hasil prediksi dari semua model dasar. Pada *Stacking* dua lapis, setiap model dasar membuat prediksi, dan hasilnya kemudian digabungkan oleh *meta-learner* untuk menghasilkan prediksi akhir, seperti yang diperlihatkan pada Gambar 2.1 (Shafieian dan Zulkernine, 2023).



Gambar 2.1 Diagram Model *Stacking* Secara Umum (Shafieian dan Zulkernine, 2023)

2.2.2 *Multi Layer Perceptron*

MLP merupakan salah satu jenis *Artificial Neural Network* (jaringan saraf tiruan) yang paling dasar dan populer. *MLP* terdiri dari tiga jenis lapisan utama yaitu lapisan *input*, *hidden layer* (lapisan tersembunyi), dan lapisan *output*. *MLP* mampu mempelajari hubungan non-linear antara fitur *input* dan target *output*, yang membuatnya efektif dalam tugas klasifikasi dan regresi. Salah satu keunggulan *MLP* adalah kemampuannya untuk memecahkan masalah yang tidak dapat dipisahkan secara linier (Du dan Swamy, 2019). Arsitektur *MLP* dengan lapisan *input*, lapisan *output*, dan dua lapisan tersembunyi ditujukan pada Gambar 2.2.



Gambar 2.2 Arsitektur *MLP* dengan Lapisan *Input*, Lapisan *Output*, dan Dua Lapisan Tersembunyi

Dalam *MLP*, neuron pada lapisan tersembunyi memiliki fungsi aktivasi *non-linear*, yang memungkinkan *MLP* untuk memproses dan memetakan data yang kompleks. Untuk mengoptimalkan kinerja *MLP*, proses pembelajaran melibatkan penyesuaian bobot dan bias di antara neuron menggunakan algoritma *backpropagation*. Persamaan yang digunakan untuk menghitung *output* dari neuron ke-j pada lapisan ke-i dapat dilihat pada Persamaan 2.1 (Li dkk., 2024).

$$y_j^{(i)} = f_j^{(i)}(w_j^{(i)}.y^{(i-1)}.b_j^{(i)}) \quad (2.1)$$

Dengan:

1) $y_j^{(i)}$ adalah *output* dari neuron ke-j pada lapisan ke-i,
2) $w_j^{(i)}$ adalah vektor bobot yang menghubungkan neuron ke-j pada lapisan ke-i dengan neuron di lapisan sebelumnya,
3) $y^{(i-1)}$ adalah *output* dari lapisan sebelumnya,

4) $b_j^{(i)}$ adalah bias untuk neuron ke-j pada lapisan ke-i,

5) $f_j^{(i)}$adalah fungsi aktivasi yang digunakan pada neuron tersebut.

Proses pelatihan MLP dilakukan menggunakan algoritma *Backpropagation* dengan pendekatan *supervised learning*. Secara umum, *Backpropagation* menghitung gradien fungsi *loss* $\mathcal{L}$ terhadap bobot dan bias melalui dua tahap, yaitu *forward pass* untuk memperoleh prediksi $\hat{y}$ dan *backward pass* untuk menghitung *error* dan gradien pada setiap lapisan (Nikov dkk., 2025). Rumus *Backpropagation* pada lapisan *output* L didefinisikan pada Persamaan 2.2.

$$\delta^{(L)} = \frac{\partial \mathcal{L}}{\partial z^{(L)}} = \hat{y} - t \tag{2.2}$$

Dengan:

1) $\delta^{(L)}$ adalah vektor *error* (galat signal) pada lapisan *output*;

2) $z^{(L)}$ adalah vektor masukan (sebelum aktivasi) ke lapisan *output*;

3) $\hat{y}$ adalah vektor keluaran prediksi jaringan;

4) t adalah vektor target (*one-hot*) yang merepresentasikan kelas sebenarnya;

5) $\mathcal{L}$ adalah fungsi *loss*.

Error pada setiap lapisan tersembunyi $i = L-1, \ldots, 1$ dipropagasikan mundur menggunakan Persamaan 2.3.

$$\delta^{(i)} = (W^{(i+1)})^{\mathrm{T}} \delta^{(i+1)} \cdot f^{(i)'}(z^{(i)}) \tag{2.3}$$

Dengan:

1) $\delta^{(i)}$ adalah vektor *error* (galat signal) pada lapisan ke-*i*;

2) $W^{(i+1)}$ adalah matriks bobot dari lapisan ke-*I* ke lapisan ke-(i+1);

3) $f^{(i)'}$ turunan fungsi aktivasi pada lapisan ke-*i*;

4) $z^{(i)}$ vektor masukan (sebelum aktivasi) ke lapisan ke-*i*.

Setelah vektor error diperoleh untuk setiap lapisan, gradien fungsi *error* terhadap bobot dan bias pada lapisan ke-*i* dihitung dengan Persamaan 2.4.

$$\frac{\partial \mathcal{L}}{\partial W^{(i)}} = \delta^{(i)} (h^{(i-1)})^{\mathrm{T}}, \qquad \frac{\partial \mathcal{L}}{\partial b^{(i)}} = \delta^{(i)} \tag{2.4}$$

Dengan:

1) $h^{(i-1)}$ adalah vektor keluaran (aktivasi) pada lapisan ke-(i-1);
2) $\frac{\partial \mathcal{L}}{\partial W^{(i)}}$ adalah gradien *loss* terhadap matriks bobot lapisan ke-*i*;
3) $\frac{\partial \mathcal{L}}{\partial b^{(i)}}$ adalah gradien *loss* terhadap vektor bias lapisan ke-*i*.

Dalam penelitian ini digunakan beberapa *optimizer* berbasis *gradient descent*, yaitu *Stochastic Gradient Descent* (*SGD*), *Adam*, dan *L-BFGS*. *SGD* memperbarui bobot menggunakan gradien yang dihitung dari satu atau beberapa sampel (*mini-batch*), sehingga lebih efisien untuk *dataset* besar dan banyak digunakan untuk mengoptimasi MLP pada berbagai domain (Talebi dkk., 2023). *Adam* merupakan *optimizer* adaptif yang menyimpan rata-rata perubahan dari gradien dan kuadrat gradien untuk setiap parameter sehingga *learning rate* menjadi adaptif per-parameter (Rashedi dkk., 2024). Sementara itu, L-BFGS (Limited-memory BFGS) adalah metode kuasi-Newton yang mendekati informasi Hessian untuk mempercepat konvergensi, dan telah digunakan untuk melatih jaringan saraf tiruan pada berbagai masalah optimasi *nonlinier* (Wang dkk., 2023).

Untuk memperoleh kombinasi hiperparameter terbaik (termasuk pemilihan *optimizer Adam*, *SGD*, atau *L-BFGS*), penelitian ini menggunakan pendekatan *hyperparameter optimization* berbasis *TPE (Tree-structured Parzen Estimator)*. *TPE* memodelkan distribusi *hyperparameter* yang menghasilkan performa baik dan buruk secara terpisah, kemudian memilih kandidat baru yang memaksimalkan rasio manfaat antara kedua distribusi tersebut.

2.2.3 *LightGBM*

LightGBM adalah algoritma yang cepat dan efisien untuk mengolah data besar. Dengan dua teknik khusus, yaitu *GOSS* dan *EFB*, algoritma ini mampu mempercepat proses dan menghemat memori, namun tetap akurat. *GOSS* mempercepat proses pelatihan dengan mengecualikan data yang memiliki gradien kecil, sehingga fokus pada informasi penting tanpa kehilangan akurasi signifikan. Sementara itu, *EFB* mengurangi jumlah fitur dengan menggabungkan fitur yang jarang aktif bersamaan, sehingga penggunaan memori dan waktu komputasi menjadi lebih efisien (Bentéjac dkk., 2021).

LightGBM menunjukkan performa yang unggul dibandingkan dengan algoritma lain seperti *XGBoost* dan *CatBoost*, khususnya dalam hal kecepatan dan efisiensi penggunaan memori, tanpa mengurangi tingkat presisi model. Dalam beberapa kasus, *LightGBM* menggunakan hanya sepertiga dari memori yang dibutuhkan oleh metode lain dan tetap mempertahankan akurasi yang setara (Yan dkk., 2021).

LightGBM adalah versi efisien dari *GBDT* (*Gradient Boosting Decision Tree*). Seperti halnya *GBDT*, algoritma ini bekerja dengan menggunakan gradien negatif dari fungsi *loss* untuk menghitung sisa kesalahan dari *decision tree* saat ini. Sisa kesalahan ini kemudian digunakan untuk membangun *decision tree* baru. Pada setiap iterasi, model aslinya tetap sama, dan fungsi baru ditambahkan agar hasil prediksi semakin mendekati nilai yang sebenarnya. Tujuan dari proses pelatihan ini adalah meminimalkan kesalahan prediksi, seperti yang dijelaskan dalam Persamaan 2.5 (Zhang dkk., 2020):

$$\hat{y}^i(t) = \hat{y}^i(t-1) + \eta f_i(x_i) \tag{2.5}$$

Dengan:

1) $\hat{y}^i(t)$: Merupakan nilai prediksi dari contoh ke-i pada iterasi ke-t.

2) $\eta f_i(x_i)$: Merupakan selisih antara nilai prediksi yang dihasilkan oleh model dan nilai target dari pohon keputusan pada iterasi ke-t, yang ditambahkan ke prediksi sebelumnya untuk mengurangi kesalahan.

Pada setiap langkah, model memperbarui prediksi dengan menambahkan pohon baru yang berfokus memperbaiki kesalahan dari prediksi sebelumnya. Dalam *LightGBM*, proses ini dilakukan dengan lebih cepat dan efisien menggunakan dua teknik khusus yaitu *GOSS* dan *EFB*. Selain itu, fungsi objektif dari *LightGBM* juga dijelaskan pada Persamaan 2.6 (Zhang dkk., 2020)

$$Obj^k = \sum_{i=1}^{n} L(y_i, \hat{y}^i(t-1) + f_t(x_i)) + \sum_{t=1}^{T} \Omega(f_t) \quad (2.6)$$

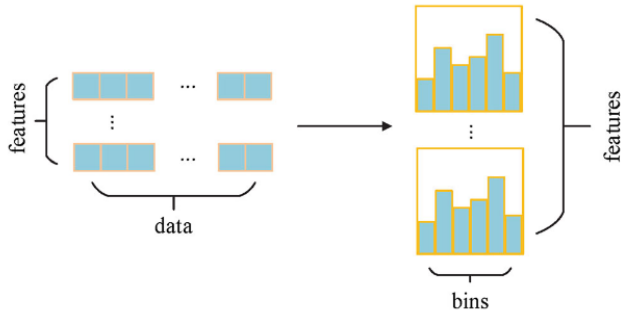
Dengan:

1) $L(y_i, \hat{y}^i(t-1) + f_t(x_i))$: Mengukur selisih antara nilai prediksi sebelumnya $\hat{y}^i(t-1)$ dan nilai aktual y_i.

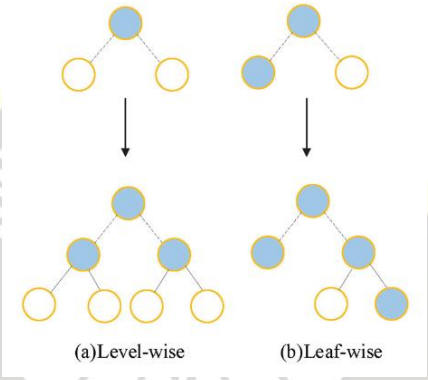
2) $\Omega(f_t)$: Merupakan penalti yang digunakan untuk mengontrol kompleksitas model, menghindari *overfitting*.

Keseluruhan proses ini bertujuan untuk menemukan fungsi prediksi (f_t) yang terbaik pada setiap iterasi guna meminimalkan kesalahan total dan meningkatkan akurasi prediksi (Zhang dkk., 2020).

LightGBM menggunakan beberapa teknik yang membuatnya lebih efisien dibandingkan algoritma *GBDT* tradisional. *LightGBM* membagi nilai kontinu menjadi beberapa interval menggunakan algoritma histogram seperti yang ditujukan pada Gambar 2.3, sehingga proses pelatihan lebih cepat dan penggunaan memori lebih efisien. Pohon keputusan dalam *LightGBM* juga tumbuh dengan strategi berbasis daun (*Leaf-wise*), yang memungkinkan pertumbuhan lebih fokus pada bagian yang memiliki nilai pembagian terbaik seperti yang ditujukan pada Gambar 2.4 (Yang dkk., 2024).



Gambar 2.3 Algoritma Histogram *LightGBM (Yang dkk., 2024)*



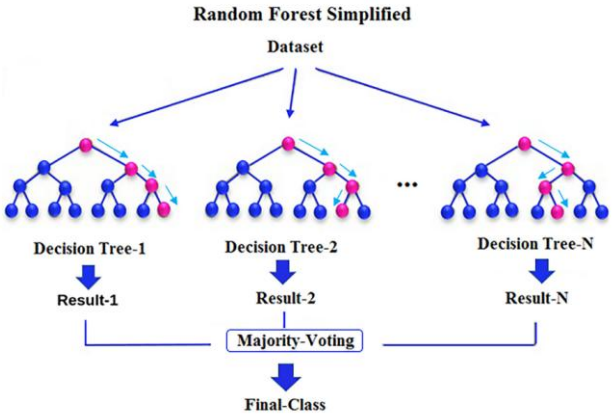
Gambar 2.4 *Level-Wise Strategy* dan *Leaf-Wise Strategy LightGBM* (Yang dkk., 2024)

2.2.4 *Random Forest*

Random Forest adalah algoritma pembelajaran yang digunakan untuk melakukan klasifikasi dan regresi. Algoritma ini diperkenalkan oleh Leo Breiman pada tahun 2001 dan bekerja dengan cara menggabungkan banyak pohon keputusan. Setiap pohon dibangun dari sampel acak data pelatihan, dan hasil akhir diperoleh dengan cara menghitung rata-rata prediksi dari semua pohon (untuk regresi) atau dengan mengambil keputusan mayoritas dari semua pohon (untuk klasifikasi) (Attanasi dan Coburn, 2021).

Keunggulan *Random Forest* adalah kemampuannya dalam menangani hubungan data yang tidak linier, sehingga lebih fleksibel dibandingkan model regresi standar yang memiliki asumsi-asumsi seperti distribusi normal atau kesetaraan varians. Dengan menggabungkan banyak pohon keputusan, *Random Forest* juga dapat mengurangi risiko *overfitting*, yang sering terjadi pada model pohon keputusan tunggal. Algoritma ini mampu menangani *dataset* besar dengan efisien dan sangat efektif dalam mengolah data yang memiliki banyak fitur atau variabel (Iranzad dan Liu, 2025).

Random Forest adalah model yang terdiri dari banyak pohon keputusan. Model ini disebut acak karena dua prinsip yaitu penggunaan pengambilan sampel acak dari titik data, dan pembagian *node* berdasarkan kategori fitur. Untuk membagi setiap *node*, setiap pohon dalam *Random Forest* mengambil sampel data dan variabel *input* secara acak dari *dataset*. Setelah banyak pohon keputusan dibuat, model memilih kelas yang paling umum berdasarkan mayoritas hasil dari pohon-pohon tersebut. Pengklasifikasi dengan satu pohon keputusan mungkin lebih baik dibandingkan memilih kelas secara acak, tetapi *Random Forest* menghasilkan model yang jauh lebih kuat dengan menggabungkan banyak pohon dan melakukan pemilihan *input* secara acak seperti yang ditujukan pada Gambar 2.5 (Yousefi dkk., 2024).



Gambar 2.5 Algoritma Sederhana *Random Forest* (Yousefi dkk., 2024)

Algoritma *Random Forest* ini bekerja dengan menggabungkan beberapa pohon keputusan yang dilatih secara independen menggunakan sampel acak dari data latih. Dengan menggunakan teknik *Bagging*, *Random Forest* secara acak memilih *subset* dari data latih dan membangun pohon keputusan untuk setiap *subset*. Persamaan prediksi dalam *Random Forest* dapat dinyatakan seperti pada Persamaan 2.7 (Hu dkk., 2023).

$$f(x) = \frac{1}{M} \sum_{m=1}^{M} h_m(x) \tag{2.7}$$

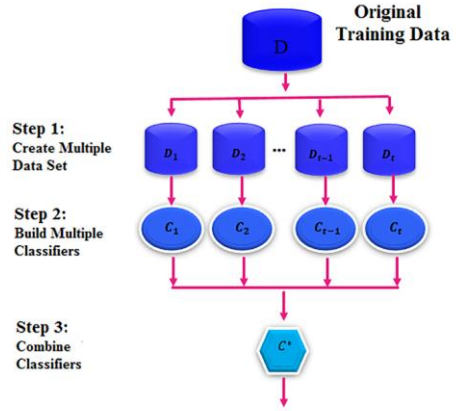
Dengan:

1) $f(x)$ adalah hasil prediksi akhir;
2) M adalah jumlah total pohon keputusan;
3) $h_m(x)$ adalah hasil prediksi dari pohon keputusan ke-m;
4) x adalah input vektor (fitur).

Langkah-langkah dalam membangun *Random Forest* meliputi (Hu dkk., 2023):

1) Memilih sampel data secara acak dari *dataset* asli dengan penggantian (*bootstrap*);
2) Membangun pohon keputusan untuk setiap *subset* data dengan memilih secara acak fitur yang digunakan untuk memisahkan *node*;
3) Mengulang langkah-langkah di atas hingga sejumlah pohon keputusan terbentuk;
4) Untuk prediksi akhir, mengambil rata-rata (untuk regresi) atau mengambil keputusan mayoritas dari semua pohon (untuk klasifikasi) dari semua hasil prediksi pohon.

Random Sampling dalam algoritma *Random Forest* adalah proses di mana setiap pohon keputusan dilatih dengan menggunakan sampel acak dari *dataset*. Teknik ini dikenal sebagai *bootstrapping*, di mana data dipilih secara acak dengan penggantian (*replacement*), sehingga beberapa sampel dapat muncul lebih dari satu kali dalam pelatihan satu pohon. Dengan menggunakan sampel acak ini, model dapat mengurangi bias dari data dan meningkatkan kemampuan dalam membuat prediksi yang lebih baik pada data baru yang belum pernah dilihat. Gambaran teknik *bootstrapping* dapat dilihat pada Gambar 2.6 (Yousefi dkk., 2024).



Gambar 2.6 Teknik *Bootstrapping Random Forest* (Yousefi dkk., 2024)

Langkah-langkah utama dalam proses *bootstraping* ini meliputi (Yousefi dkk., 2024):

1) Menciptakan *dataset* acak: *Dataset* baru dibuat dengan mengambil sampel acak dari *dataset* asli. *Subset* dari kolom dan baris juga dipilih, yang diatur sebagai parameter dalam model. Menggunakan bagian yang lebih kecil dari baris dan kolom membantu menghindari *overfitting*;

2) Membangun beberapa pohon keputusan: Setelah *dataset* acak dibuat, pohon keputusan dilatih menggunakan *subset* data yang berbeda;

3) Menggabungkan hasil prediksi: Setelah semua pohon terbentuk, prediksi akhir dihitung dengan menggabungkan hasil dari setiap pohon melalui rata-rata (untuk regresi) atau mengambil keputusan mayoritas dari semua pohon (untuk klasifikasi).

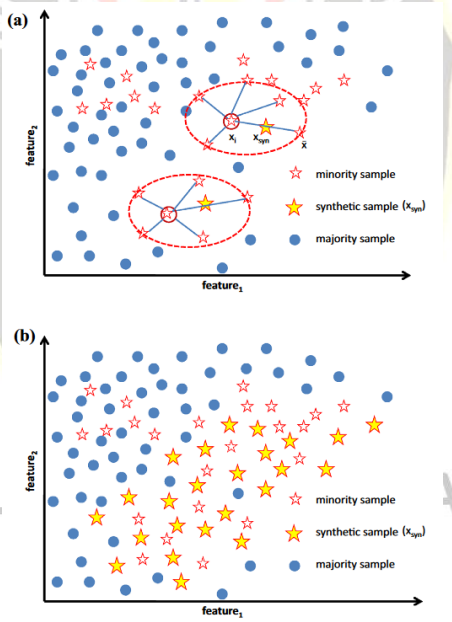
2.2.5 *SMOTE*

SMOTE adalah metode *oversampling* yang digunakan untuk menangani ketidakseimbangan kelas dalam *dataset*, terutama dalam kasus di mana satu kelas (minoritas) jauh lebih sedikit dibandingkan kelas lainnya (mayoritas). Metode ini banyak digunakan dalam masalah klasifikasi untuk meningkatkan performa model *Machine Learning* pada *dataset* yang tidak seimbang, dengan tujuan mencegah bias terhadap kelas mayoritas (Elreedy dkk., 2024).

SMOTE menciptakan data sintetis dari kelas minoritas dengan melakukan interpolasi antara sampel minoritas yang ada dan tetangganya. Hal ini lebih efektif daripada sekadar menduplikasi data, karena membantu model mengenali pola secara lebih akurat. Keunggulan *SMOTE* terletak pada kemampuannya untuk mencegah *overfitting* dan meningkatkan performa model pada data tidak seimbang, terutama dalam kasus-kasus seperti deteksi serangan jaringan dan klasifikasi medis. Beberapa

studi menunjukkan bahwa *SMOTE* secara signifikan dapat memperbaiki akurasi dan kinerja model dibandingkan metode lain (Jadwal dkk., 2022).

Proses utama *SMOTE* ini dilakukan dengan memilih secara acak salah satu dari tetangga terdekat (*KNN*) dari sampel kelas minoritas, lalu melakukan interpolasi acak antara dua sampel untuk menciptakan sampel baru seperti yang terlihat pada Gambar 2.7. *SMOTE* menggunakan teknik pencarian dan pemilihan berulang untuk menghasilkan sampel sintetis ini. *Confidence value* dari sampel-sampel dalam kelompok tetangga terdekat digunakan untuk menentukan berapa banyak sampel sintetis yang harus dibuat. Proses ini diulang hingga jumlah sampel sintetis yang diinginkan tercapai. *SMOTE* menciptakan sampel baru dalam ruang fitur, bukan ruang data, untuk menyeimbangkan distribusi kelas yang tidak seimbang (Raghuwanshi dan Shukla, 2021).



Gambar 2.7 *SMOTE* Melakukan Interpolasi Linier pada Contoh Kelas Minoritas yang Dipilih Secara Acak (Raghuwanshi dan Shukla, 2021)

Persamaan *SMOTE* untuk menghasilkan sampel sintetis baru dituliskan pada Persamaan 2.8 (He dkk., 2023a).

$$h_{syntetic} = x_i + \beta \,.(x_j - x_i) \tag{2.8}$$

Dengan:

1) x_i adalah sampel minoritas asli;
2) x_j adalah tetangga terdekat dari x_i yang juga berasal dari kelas minoritas;
3) β adalah bilangan acak dalam interval (0,1) yang menentukan posisi interpolasi di antara dua titik.

Proses ini menghasilkan data baru yang berada pada garis lurus antara dua titik dari kelas minoritas, dengan tujuan untuk memperluas representasi kelas minoritas tanpa menciptakan bias yang berlebihan akibat duplikasi langsung dari sampel yang sudah ada (He dkk., 2023a).

2.2.6 *One-Way ANOVA F-Test*

One-Way ANOVA (*Analysis of Variance*) *F-Test* adalah metode yang digunakan untuk menilai pengaruh signifikan atau perbedaan antara fitur tertentu dengan variabel target atau kelas dalam konteks klasifikasi maupun regresi. Prosesnya melibatkan pemisahan data menjadi kelompok-kelompok berdasarkan kategori target, lalu menghitung nilai *F* untuk setiap fitur guna menilai sejauh mana rata-rata nilai fitur tersebut berbeda antar kelompok. Fitur-fitur yang memiliki nilai *F* tinggi kemudian dipilih karena dianggap relevan dalam membedakan kelompok atau kelas. Metode ini membantu mengidentifikasi fitur yang memiliki kontribusi signifikan terhadap model (Hasanah dkk., 2024).

Penggunaan *ANOVA F-Test* bertujuan untuk memilih fitur-fitur penting dalam proses klasifikasi. Metode ini dilakukan untuk mengevaluasi apakah suatu fitur

menunjukkan perbedaan yang signifikan antar kelompok data (kelas). Proses ini melibatkan langkah-langkah berikut (ZHUANG dkk., 2021).

a) Menentukan hipotesis; Hipotesis ini membantu menentukan apakah sebuah fitur relevan untuk membedakan kelas. Hipotesis setia fitur dapat dilihat pada Persamaan 2.9 dan Persamaan 2.10.

$$H0:\ Tidak\ ada\ perbedaan\ rata-rata\ antar\ kelas. \quad (2.9)$$

$$H1:\ Ada\ perbedaan\ rata-rata\ setidaknya\ satu\ kelas. \quad (2.10)$$

b) Menghitung rata-rata total; Langkah ini bertujuan untuk menghitung rata-rata keseluruhan dari semua data di *dataset*. Persamaan untuk menghitung rata-rata total dapat dilihat pada Persamaan 2.11.

$$\theta = \frac{1}{2} \sum_{j=1}^{s} n_j \theta_j \quad (2.11)$$

Dengan:

1) n_j adalah jumlah sampel di kelas ke-j;
2) θ_j adalah rata-rata sampel di kelas ke- j;
3) n adalah total jumlah sampel.

c) Menghitung rata-rata setiap kelas; Persamaan untuk menghitung rata-rata nilai untuk masing-masing kelas dapat dilihat pada Persamaan 2.12.

$$\theta_j = \frac{1}{n_j} \sum_{i=1}^{n_j} x_{ij} \quad (2.12)$$

Dengan:

1) x_{ij} adalah nilai sampel ke-i dalam kelas ke-j;
2) n_j adalah jumlah sampel di kelas ke- j.

d) Menghitung variasi; Persamaan untuk menghitung variasi dalam kelompok untuk mengukur perbedaan di dalam setiap kelas dapat dilihat pada Persamaan 2.13, sedangkan persamaan untuk menghitung variasi antar kelompok untuk mengukur perbedaan rata-rata antar kelas dapat dilihat pada Persamaan 2.14.

$$SE = \sum_{j=1}^{s} \sum_{i=1}^{n_j} \left(x_{ij} - \theta_j\right)^2 \quad (2.13)$$

$$SA = \sum_{j=1}^{s} n_j \left(\theta_j - \theta\right)^2 \quad (2.14)$$

e) Menghitung nilai *F*; Berdasarkan nilai *F*, lalu mencari *P-value* dari tabel distribusi *F*:

   1) Jika $P > 0.05$, hipotesis nol diterima (tidak ada perbedaan signifikan antar kelompok untuk fitur tersebut);
   2) Jika $P \leq 0.05$, hipotesis nol ditolak (ada perbedaan signifikan antar kelompok untuk fitur tersebut).

f) Memilih fitur yang relevan; Hanya fitur dengan $P \leq 0.05$ yang dipertahankan, karena menunjukkan perbedaan yang signifikan dan relevan untuk klasifikasi.

2.2.7 *Outlier*

Outlier adalah observasi atau nilai data yang secara signifikan menyimpang dari pola umum dalam suatu himpunan data. Nilai ini dapat muncul karena berbagai alasan, termasuk kesalahan pencatatan, variasi alami yang ekstrem, atau kejadian langka yang layak diteliti lebih lanjut (Mellenbergh, 2019). Selain berpotensi menyebabkan distorsi terhadap data statistik, *outlier* juga sering kali mengandung informasi penting, seperti indikasi penipuan, gangguan sistem, atau anomali perangkat keras, sehingga penanganan yang tepat terhadap *outlier* tidak hanya meningkatkan kualitas analisis, tetapi juga memberikan manfaat strategis dalam konteks aplikasi nyata (Angiulli, 2019)

Salah satu metode populer untuk mendeteksi *outlier* adalah pendekatan statistik *robust* berbasis *IQR* (*Interquartile Range*). *IQR* dihitung sebagai selisih antara kuartil ketiga ($Q3$) dan kuartil pertama ($Q1$), sebagaimana dirumuskan dalam Persamaan 2.15.

$$IQR = Q3 - Q1 \tag{2.15}$$

Suatu nilai dikategorikan sebagai *outlier* apabila berada di bawah batas bawah atau di atas batas atas, yang masing-masing dihitung dengan Persamaan 2.16 dan Persamaan 2.17.

$$Batas\ bawah = Q1 - 1.5 \times IQR \tag{2.16}$$

$$Batas\ atas = Q3 + 1.5 \times IQR \tag{2.17}$$

Dengan demikian, semua data yang terletak di luar rentang tersebut dianggap menyimpang secara signifikan dari pola umum dan perlu dianalisis lebih lanjut. Pendekatan ini terbukti efektif dalam konteks keamanan jaringan, di mana deteksi *outlier* digunakan sebagai bagian dari sistem deteksi intrusi berbasis anomali. Studi yang dilakukan oleh Beulah dan Punithavathani pada tahun 2020 yang mengembangkan metode deteksi intrusi berbasis *clustering* dan *outlier* untuk membedakan pola normal dan intrusif pada data jaringan menunjukkan bahwa deteksi *outlier* dapat secara signifikan meningkatkan akurasi klasifikasi dan menurunkan tingkat kesalahan deteksi dalam sistem *IDS* (Beulah dan Punithavathani, 2020).

Setelah proses deteksi, salah satu teknik koreksi yang sering digunakan adalah *Winsorization*. Teknik ini tidak menghapus *outlier*, tetapi mengganti nilai-nilai ekstrem tersebut dengan nilai pada batas persentil tertentu, seperti persentil ke-5 dan ke-95, untuk menjaga stabilitas distribusi data. Formulasi matematisnya dituliskan pada Persamaan 2.18.

$$x_i \begin{cases} P_a, jika\ x_i < P_a \\ x_i, jika\ P_a \leq x_i \leq P_{1-a} \\ P_{1-a}, jika\ x_i > P_{1-a} \end{cases} \tag{2.18}$$

Winsorization banyak diterapkan dalam berbagai bidang seperti kesehatan, keuangan, dan *e-commerce* karena kemampuannya dalam menjaga integritas data sambil

meminimalkan pengaruh nilai ekstrem terhadap hasil analisis (Dash dkk., 2023). Studi juga menunjukkan bahwa kombinasi metode *IQR* untuk deteksi dan *Winsorization* untuk koreksi dapat meningkatkan performa analisis secara signifikan. Pendekatan gabungan ini terbukti mampu mengurangi *overfitting* dan meningkatkan akurasi prediksi, khususnya dalam model *machine learning* yang beroperasi pada data yang mengandung *noise* (Ch'ng dan Mahat, 2020).

2.2.8 *Min-Max Scalling*

Min-Max Scaling adalah metode normalisasi yang mengubah nilai-nilai dalam sebuah fitur agar berada dalam rentang [0, 1] dengan cara memetakan nilai minimum menjadi 0 dan nilai maksimum menjadi 1. Teknik ini sering digunakan dalam *preprocessing data* karena menjaga bentuk distribusi data dan mempercepat konvergensi dalam model *machine learning* (Muhammad Ali, 2022). Perhitungan matematis *Min-Max Scaling* dituliskan pada Persamaan 2.19.

$$x^i = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.19}$$

dengan x adalah nilai asli, x_{min} dan x_{max} adalah nilai minimum dan maksimum dalam fitur tersebut serta x^i adalah hasil nilai yang telah dinormalisasi ke rentang [0, 1].

2.2.9 *Bootstrap Paired T-Test*

Bootstrap Paired T-Test merupakan metode statistik *non*-parametrik yang digunakan untuk menguji perbedaan antara dua kondisi yang berpasangan, terutama saat asumsi distribusi normal tidak dapat dipenuhi. Tidak seperti *Paired T-Test* tradisional yang bergantung pada asumsi normalitas, *Bootstrap Paired T-Test* menggunakan pendekatan *resampling* terhadap pasangan data asli untuk menghasilkan distribusi empiris dari statistik uji. Proses ini meningkatkan keandalan pengujian

terutama pada data berdistribusi *non*-normal atau berukuran kecil. Metode ini bekerja dengan mengambil sampel acak berulang kali dari data berpasangan dan menghitung selisih rata-rata untuk masing-masing sampel, sehingga menghasilkan estimasi interval kepercayaan atau nilai *p* tanpa bergantung pada distribusi teoritis (Guo dkk., 2020).

Proses *Bootstrap Paired T-Test* dimulai dengan menghitung selisih antara setiap pasangan data, misalnya antara nilai sebelum dan sesudah perlakuan. Selisih tersebut dinyatakan sebagai $d_i = x_i - y_i$, di mana x_i dan y_i adalah nilai dari pasangan ke-i. Selanjutnya, dilakukan proses *resampling* sebanyak *B* kali dengan memilih secara acak elemen-elemen dari vektor selisih $D = \{d_1, d_2, \dots, d_n$ dengan pengembalian. Tujuan penelitian adalah membuktikan bahwa model usulan lebih unggul dari *baseline*, digunakan uji satu arah dengan Persamaan 2.20 dan 2.21.

$$H_0 = \Delta[d] \leq 0 \quad \textit{(tidak ada peningkatan atau lebih buruk dari baseline)} \tag{2.20}$$

$$H_1 = \Delta[d] > 0 \ (peningkatan\ dibanding\ baseline) \tag{2.21}$$

Prosedur uji yang dilakukan adalah sebagai berikut:

1) Menghitung selisih observasi rata-rata $\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i$;

2) Melakukan *resampling* berulang sebanyak *B* kali terhadap elemen-elemen *D* dengan pengembalian untuk membentuk sampel *bootstrap*;

3) Pada setiap iterasi ke-b, menghitung rata-rata selisih *bootstrap* dengan Persamaan 2.22;

$$\partial_b^* = \frac{1}{n}\sum_{i=1}^{n} \partial_{i,b}^* \tag{2.22}$$

4) Membentuk distribusi empiris $\{\overline{d_b^*}\}_{b=1}^{B}$. Nilai-*p* satu arah dihitung sebagai proporsi $\overline{d_b^*}$ yang tidak mendukung keunggulan model usulan, yaitu bagian massa di sisi ≤ 0 setelah dilakukan centering terhadap $\bar{d}$ (mengurangkan $\bar{d}$ dari setiap $\overline{d_b^*}$) sehingga pengujian merepresentasikan $H_0 : \Delta[d] = 0$. Keputusan H_0 ditolak jika *p* $p < \alpha$;

5) Selain nilai *p*, interval kepercayaan 95% untuk $\Delta[d]$ diperoleh dengan metode persentil dari $\{\overline{d_b^*}\}$; jika batas bawah CI > 0, maka peningkatan kinerja dinyatakan signifikan secara statistik pada α = 0,05.

Pendekatan ini memungkinkan pengujian superioritas model usulan terhadap *baseline* tanpa mengasumsikan bentuk distribusi tertentu, serta menyediakan ukuran ketidakpastian melalui nilai-*p* dan interval kepercayaan berbasis *bootstrap* (Guo dkk., 2020).

2.2.10 *SHAP (SHapley Additive Explanations)*

SHAP (SHapley Additive exPlanations) merupakan metode yang digunakan untuk menjelaskan hasil prediksi dari model *machine learning* secara transparan. Metode ini menghitung seberapa besar pengaruh setiap fitur *input* terhadap hasil prediksi model. *SHAP* dapat memberikan penjelasan baik pada tingkat individu (prediksi satu data) maupun secara menyeluruh terhadap seluruh data yang digunakan dalam model (Mosca dkk., 2022). Penggunaan *SHAP* membantu dalam memahami bagaimana model membuat keputusan, sehingga meningkatkan kepercayaan terhadap hasil prediksi. Metode ini telah banyak diterapkan dalam berbagai bidang seperti kesehatan, sistem tenaga listrik, dan pusat data, karena kemampuannya dalam memberikan penjelasan yang akurat dan mudah dipahami (Gebreyesus dkk., 2024).

SHAP merupakan metode yang memberikan nilai kontribusi masing-masing fitur terhadap hasil prediksi model *machine learning*. Secara matematis, nilai *SHAP* dihitung berdasarkan rata-rata perubahan prediksi saat suatu fitur dimasukkan ke dalam berbagai kombinasi *subset* fitur lainnya. Rumus umum *SHAP* dituliskan pada Persamaan 2.23.

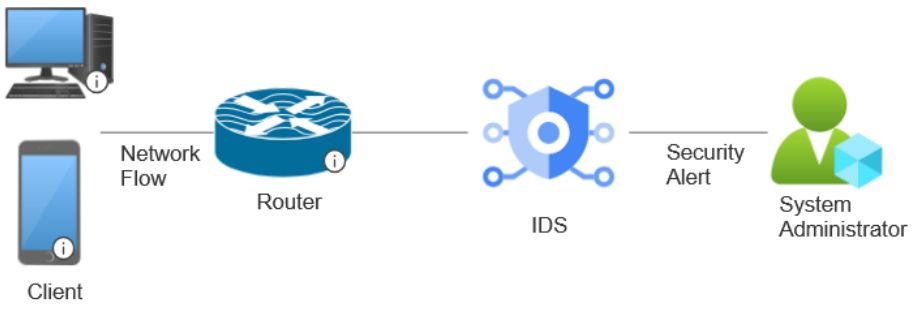
$$\emptyset_i(f,x) = \sum_{S \subseteq N\{i\}} \frac{|S|!(|N|-|S|-1)!}{|N|!}[f(S \cup \{i\}) - f(S)] \quad (2.23)$$

Rumus ini menghitung nilai kontribusi fitur i ($\emptyset_i$) dengan mempertimbangkan semua kemungkinan subset fitur lainnya (S) dari total fitur N, lalu mengukur seberapa besar pengaruh penambahan fitur i terhadap prediksi model. Nilai ini kemudian dirata-ratakan berdasarkan bobot kombinatorial, sehingga menghasilkan interpretasi yang adil dan konsisten untuk setiap fitur (Nohara dkk., 2019).

Dalam konteks keamanan siber, khususnya pada sistem deteksi dan pencegahan intrusi (*IDS/IPS*), *SHAP* berperan penting dalam meningkatkan transparansi dan keandalan model *machine learning* yang digunakan. Model deteksi ancaman yang kompleks, seperti *Random Forest*, *XGBoost*, atau *Deep Neural Networks*, sering kali dianggap sebagai "*black box*" karena sulit dipahami cara kerjanya. Dengan menggunakan *SHAP*, setiap prediksi dari model dapat dijelaskan secara terperinci, yaitu dengan menunjukkan fitur-fitur apa saja yang paling berkontribusi terhadap pendeteksian suatu serangan. Penjelasan berbasis *SHAP* ini tidak hanya memudahkan analis keamanan dalam memahami perilaku model, tetapi juga membantu dalam melakukan validasi dan pengambilan keputusan yang lebih cepat dan akurat di lingkungan sistem yang dinamis (Assegie, 2022).

2.2.11 *IDS*

IDS adalah kombinasi komponen perangkat lunak dan perangkat keras yang berperan penting dalam keamanan jaringan dengan memantau aktivitas pengguna serta perilaku di jaringan untuk mendeteksi pelanggaran keamanan. Sistem ini berfungsi mengidentifikasi aktivitas yang mencurigakan dan mengirim peringatan keamanan kepada administrator jaringan segera setelah serangan terdeteksi. *IDS* dapat berupa *HIDS* (*Host Intrusion Detection System*) yang memantau lalu lintas dari *host* tertentu atau *NIDS* (*Network Intrusion Detection System*) yang memantau semua lalu lintas dalam jaringan. *IDS* bekerja dengan dua pendekatan utama: teknik berbasis tanda tangan yang membandingkan data lalu lintas aktual dengan pola serangan yang tersimpan dalam basis data, dan teknik berbasis anomali yang mendeteksi perilaku yang tidak biasa dengan membandingkan lalu lintas dengan profil normal yang telah ditentukan. Teknik berbasis tanda tangan efektif dalam mendeteksi serangan yang telah dikenal, sementara teknik berbasis anomali lebih baik dalam mendeteksi serangan baru namun cenderung menghasilkan *false alarm* yang lebih tinggi. Diagram *IDS* dapat dilihat pada Gambar 2.8 (Hajj dkk., 2021).



Gambar 2.8 Diagram *IDS* (Hajj dkk., 2021)

2.2.12 Koefisien *Jaccard*

Pengukuran kemiripan himpunan pada tesis ini menggunakan koefisien *Jaccard* yang memusatkan penilaian pada proporsi irisan terhadap gabungan sehingga hanya pasangan fitur yang sama-sama hadir yang berkontribusi sementara absensi ganda diabaikan karena tidak informatif. Formulasi ini berakar pada karya klasik *Jaccard* di bidang ekologi yang memperkenalkan kemiripan berbasis irisan dan gabungan dan kemudian diadopsi luas untuk data biner modern (Jaccard, 1901).

Berbagai kajian terbaru menegaskan bahwa *Jaccard* banyak dipakai untuk mengukur kemiripan dan membantu proses *klastering*, khususnya pada data kategorikal, dan penggunaannya berjalan berdampingan dengan metrik lain seperti *Euclidean* dan *Cosine*. Rumus *Jaccard* ditulis pada Persamaan 2.24 (Gao dkk., 2023).

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, d_j = 1 - J \tag{2.24}$$

$|A \cap B|$ adalah banyaknya elemen yang keduanya sama-sama punya, sedangkan $|A \cup B|$ adalah banyaknya elemen unik yang muncul di salah satu atau keduanya.

Dalam praktik sekarang, nilai *Jaccard* di atas 0,5 umumnya dipandang sudah kuat karena melewati ambang baku yang dipakai luas. Di visi komputer, banyak studi menilai deteksi benar pada $IoU = Jaccard \geq 0{,}5$ sehingga apa pun di atas itu dianggap tumpang-tindih yang baik (El Akrouchi dkk., 2025).