BAB II
LANDASAN TEORI

2.1 Penelitian Terdahulu

Beberapa kajian telah dilakukan terhadap penelitian terdahulu yang relevan untuk mendukung penelitian yang sedang dilakukan. Penelitian-penelitian tersebut memberikan landasan teoritis dan metodologis yang memperkuat argumen serta arah penelitian ini. Rangkuman penelitian terdahulu yang dijadikan acuan dapat dilihat pada Tabel 2.1.

Tabel 2.1 Perbandingan Penelitian Terdahulu

ID (Sitasi)	Penulis	Jurnal	Metode	Hasil
[14]	Kardiyasa et al.	*Jurnal Teknik Mesin, Industri, Elektro dan Informatika (2025)*	*Support Vector Machine (SVM)*	*Akurasi 96.5%, Precision 94.8%, Recall 92.3%. Model SVM efektif deteksi anomali real-time parameter listrik mesin.*
[13]	*Haque et al.*	MDPI	*Transformer Based Auto Encoder, Isolation Forest - XGBoost*	Akurasi 95%; *Recall* 100% untuk intrusion (minim false negative); mampu membedakan malfunction vs intrusion dan menurunkan *false positive*; performa stabil di kondisi sensor yang berubah.
[15]	*Mehmood et al.*	*Expert Systems with Applications* (Elsevier), 2025, Vol. 289,	*iForest–DBSCAN + LOESS*	Presisi 0.98; akurasi 99.8% (offshore-1), 99.1% (offshore-2), 99.9% (onshore); NBPC CI 95%; komputasi ringan vs NN
[16]	Kristanti et al.	*Jurnal Teknologi Industri (2023)*	Random Forest, XGBoost, SVM, LSTM	*Random Forest* dan LSTM unggul dengan *F1-score* >90% dalam deteksi tren anomali dan prediksi kerusakan mesin.
[17]	GinniSrinivasa et al.	*International Journal of Computational and Experimental Science and Engineering (IJCESEN), 2025*	*Hybrid: Isolation Forest, LOF, One-Class SVM → XGBoost & Logistic Regression (supervised)*	Klaim meningkatkan *Precision*, *recall*, F1, ROC-AUC dan menurunkan FP/FN dibanding metode tunggal; konsisten di berbagai benchmark;

ID (Sitasi)	Penulis	Jurnal	Metode	Hasil
				skalabel untuk data berdimensi tinggi
	Chen et al.	*Procedia Engineering (Elsevier)*	*Enhanced DBSCAN*	Enhanced DBSCAN: TDR 93.7%, FAR 2.41%; K-Means: TDR 84.1%, FAR 3.65%
[10]	Deng et al.	*2020 5th International Conference on Information Science, Computer Technology and Transportation (ISCTT), IEEE*	*DBSCAN, K-Means*	Hasil menunjukkan bahwa DBSCAN lebih efektif daripada K-Means dalam mendeteksi anomali pada data serangan jaringan karena DBSCAN mampu mengenali cluster dengan bentuk tidak beraturan dan memisahkan noise. Metode ini meningkatkan akurasi dan mengurangi *false positive*.
[11]	Nalini, M.	*International Journal of Machine Learning and Cybernetics*	*HDBIF-CO (Hybrid Density-Based Isolation Forest with Coati Optimization*	Metode HDBIF-CO menunjukkan performa tinggi dengan akurasi 98.9%, *Precision* 97.9%, *Recall* 98.5%, dan *F1-score* 98.6%, menandakan efektivitas tinggi dalam mendeteksi anomali dan serangan siber.
[18]	Lu, H.	*ITM Web of Conferences*	*SVM, Isolation Forest, DBSCAN*	SVM unggul untuk point anomalies; DBSCAN paling efektif bila pre-processed (data sudah terstruktur/series); Isolation Forest paling efisien untuk collective anomalies. (Cuplikan menyertakan tabel referensi angka dari studi lain; angka eksperimen utamanya tidak tercantum di cuplikan)
[12]	Hairach, M.	*2023 7th IEEE Congress on Information Science and Technology (CiSt)*	*DBSCAN, k-means, Isolation Forest, and LOF*	K-Means & DBSCAN unggul saat dituning, algoritma lain potensial dengan optimasi lanjutan

Dalam beberapa tahun terakhir, penerapan *Machine Learning* untuk deteksi anomali menunjukkan perkembangan pesat, terutama pada sektor industri dan keamanan siber. Berbagai penelitian menegaskan bahwa keberhasilan sistem deteksi anomali sangat bergantung pada pemilihan algoritma yang tepat, optimasi parameter, serta strategi integrasi antar model.

Penelitian menunjukkan bahwa Support Vector Machine (SVM) mampu mendeteksi anomali secara real-time pada parameter kelistrikan mesin industri dengan akurasi 96,5%, *Precision* 94,8%, dan *Recall* 92,3% [14] . Hasil ini menegaskan efektivitas SVM dalam mengelola data non-linear berlabel serta kestabilannya sebagai model baseline untuk sistem berbasis sensor. Sementara itu, pendekatan Transformer-Based Autoencoder yang dikombinasikan dengan Isolation Forest dan XGBoost, menghasilkan akurasi 95% dan *Recall* sempurna (100%) dalam mendeteksi intrusion [13] terbukti meningkatkan ketahanan model terhadap dinamika sensor dan menurunkan *false positive*. Kontribusi serupa ditunjukkan oleh kombinasi iForest–DBSCAN + LOESS, yang mencapai akurasi hampir sempurna (99,8% pada offshore-1 dan 99,9% pada onshore) dengan presisi 0,98. Keunggulan metode ini tidak hanya pada akurasi tinggi, tetapi juga efisiensi komputasi dibanding model berbasis neural network [15]. Dalam konteks data sensor industri, ditermukan bahwa *Random Forest* dan LSTM sama-sama memiliki *F1-score* di atas 90%, menandakan bahwa ensemble learning unggul dalam data tabular, sementara recurrent network efektif untuk data time-series dengan pola temporal yang kompleks [18].

Pendekatan hibrida juga terbukti menjanjikan dalam penelitian yang menggabungkan Isolation Forest, LOF, dan One-Class SVM dengan XGBoost dan Logistic Regression, menghasilkan peningkatan signifikan pada *Precision*, *recall*, dan *F1-score* , serta penurunan *false positive* dan false negative. Hal ini memperlihatkan bahwa kombinasi *unsupervised–supervised* mampu meningkatkan adaptabilitas dan keandalan sistem deteksi anomali secara keseluruhan [17] .

Dari sisi algoritma berbasis hasil menunjukkan bahwa Enhanced DBSCAN dan DBSCAN klasik memiliki kemampuan lebih baik dibanding K-Means dalam mengenali pola klaster tidak beraturan serta memisahkan noise, dengan True Detection Rate mencapai 93,7% dan *False alarm* Rate hanya 2,41% [10][19]. Pendekatan *Hybrid* Density-Based Isolation Forest with Coati Optimization (HDBIF-CO) yang diperkenalkan oleh Nalini (2025) juga menunjukkan performa tinggi dengan akurasi 98,9%, *Precision* 97,9%, *Recall* 98,5%, dan *F1-score* 98,6%, membuktikan efektivitas optimasi metaheuristik dalam deteksi anomali kompleks. Selanjutnya, Lu (2025) menyoroti bahwa SVM unggul untuk point anomalies, DBSCAN lebih efektif untuk data yang telah dipraolah dengan baik, sedangkan Isolation Forest efisien untuk mendeteksi collective anomalies. Temuan ini sejalan dengan Hairach (2023) yang menegaskan pentingnya parameter tuning agar DBSCAN dan K-Means dapat mencapai performa optimal.

Secara keseluruhan, literatur-literatur tersebut menunjukkan bahwa tidak ada satu algoritma yang unggul secara universal untuk semua jenis anomali. Namun, kombinasi pendekatan *supervised*, *unsupervised*, dan *hybrid*, disertai dengan optimasi parameter yang tepat, terbukti mampu meningkatkan akurasi, *recall*, dan *F1-score* secara signifikan. Berdasarkan hasil studi tersebut, penelitian ini mengambil arah komparatif terhadap berbagai kategori algoritma dengan tujuan merancang model rekomendasi deteksi anomali yang optimal dan aplikatif untuk sistem pemantauan mesin produksi di PT Saka Farma. Penelitian yang dilakukan akan menggabungkan beberapa algoritma individual dari penelitian terdahulu dan di uji coba konfigurasi *hybrid unsupervised – supervised* untuk kasus *anomaly detection.* Evaluasi performa difokuskan pada metrik *F1-score*, *Precision*, dan *recall*, karena ketiganya dinilai paling representatif dalam mengukur kemampuan model terhadap data sensor yang bersifat tidak seimbang (imbalanced).

2.2 Teori tentang Topik Skripsi

Berikut adalah kajian teori yang relevan dengan topik penelitian ini. Kajian ini disusun untuk memberikan landasan teoritis yang kuat guna mendukung

proses analisis, pemahaman, dan pengembangan penelitian. Dengan mengacu pada teori-teori yang telah teruji secara akademis, kajian ini bertujuan untuk menjelaskan konsep utama yang menjadi inti dari penelitian, serta menunjukkan bagaimana teori tersebut diterapkan dalam konteks yang sesuai dengan topik skripsi.

2.2.1 *Machine Learning*

Machine Learning merupakan cabang dari kecerdasan buatan (Artificial Intelligence) yang berfokus pada pengembangan algoritma yang memungkinkan komputer untuk belajar dari data tanpa diprogram secara eksplisit[20]. Tujuan utamanya adalah untuk menemukan pola atau hubungan tersembunyi dalam dataset, yang kemudian dapat digunakan untuk membuat prediksi atau keputusan. Dalam konteks industri manufaktur, *Machine Learning* menjadi pilar utama dalam implementasi smart factory, di mana model dapat dilatih untuk mengidentifikasi kondisi optimal mesin, memprediksi permintaan pasar, atau mengoptimalkan proses produksi[21]. Terdapat tiga kategori utama *Machine Learning*: *supervised* learning (belajar dari data berlabel), *unsupervised* learning (mencari pola di data tanpa label), dan reinforcement learning (belajar dari interaksi dengan lingkungan)[20].

Penerapan *Machine Learning* dalam perawatan prediktif (*Predictive Maintenance*) merupakan salah satu aplikasinya yang paling krusial[22]. Model *Machine Learning* dapat dilatih menggunakan data historis dari sensor-sensor mesin, seperti suhu, getaran, tekanan, dan arus listrik. Model tersebut belajar memetakan hubungan antara kondisi operasional mesin dengan potensi kegagalannya. Pendekatan ini memungkinkan perusahaan untuk beralih dari jadwal perawatan yang tetap (time-based) atau reaktif menjadi perawatan yang berdasarkan kondisi aktual mesin (condition-based) [23]. Dengan memprediksi kapan suatu komponen mesin akan gagal, perusahaan dapat menjadwalkan

perbaikan atau penggantian secara proaktif, sehingga meminimalkan downtime produksi dan mengoptimalkan biaya perawatan[24].

2.2.2 *Anomaly* Detection

Anomaly Detection merupakan bidang penting dalam analisis data yang berfokus pada pengidentifikasian instans data yang menunjukkan penyimpangan signifikan dari pola atau perilaku yang dianggap normal[25]. Instans data yang menyimpang ini dikenal sebagai pencilan atau outlier, yang keberadaannya seringkali mengindikasikan peristiwa kritis yang memerlukan perhatian khusus, seperti penipuan, kerusakan peralatan, atau potensi serangan siber. Secara fundamental, proses deteksi anomali bertujuan untuk membangun profil atau model yang akurat mengenai normalitas data, dan kemudian mengukur sejauh mana setiap titik data baru menyimpang dari profil tersebut[26]. Titik dengan tingkat penyimpangan yang tinggi inilah yang diklasifikasikan sebagai anomali.

Dalam konteks deteksi anomali pada data time-series mesin, masalah ini seringkali ditangani menggunakan pendekatan *Unsupervised* Learning karena data time-series dari mesin operasional secara alamiah jarang memiliki label kegagalan yang memadai[27]. Secara umum, metode deteksi anomali dapat dikelompokkan berdasarkan ketersediaan label data—yaitu *Supervised*, Semi-*Supervised*, dan *Unsupervised*—dengan pendekatan *Unsupervised* menjadi pilihan utama karena kelangkaan anomali yang membuat pelabelan data sulit dan mahal. Berbagai algoritma telah terbukti efektif dalam skenario *unsupervised* ini. Salah satunya adalah Isolation Forest (IF), suatu algoritma berbasis pohon yang bekerja dengan mengisolasi anomali sebagai titik data yang memerlukan lebih sedikit partisi untuk dipisahkan dalam struktur pohon acak[28]. Kontras dengan metode yang

memprofilkan normalitas, IF memanfaatkan karakteristik minoritas anomali untuk mempermudah isolasinya. Selain itu, metode berbasis kernel seperti One-Class Support Vector Machine (OCSVM) juga digunakan; algoritma ini mengidentifikasi batas-batas kondisi operasional normal, dan setiap titik data yang terletak di luar batas tersebut diklasifikasikan sebagai anomali[29]. Terakhir, pendekatan berbasis deep learning seperti Autoencoder semakin populer, di mana model dilatih untuk merekonstruksi data input normal[30]. Titik data yang menghasilkan reconstruction error yang tinggi yaitu, model tidak dapat merekonstruksinya dengan baik dianggap sebagai anomali yang potensial.

2.2.3 Data Mining

Data Mining didefinisikan sebagai suatu proses sistematis yang bertujuan untuk mengekstraksi pola, tren, dan wawasan yang valid, baru, dan bermanfaat dari himpunan data yang besar. Secara konseptual, Data Mining didasarkan pada perpaduan disiplin ilmu, di mana prinsip-prinsip statistika digunakan untuk memvalidasi pola, dan algoritma *Machine Learning* diimplementasikan untuk membangun model prediktif (melalui *Supervised* Learning) atau untuk mengidentifikasi struktur data yang tersembunyi (melalui *Unsupervised* Learning). Dengan demikian, Data Mining bertindak sebagai jembatan antara data mentah dan pengetahuan yang dapat ditindaklanjuti[31].

Untuk memastikan peslaksanaan proyek Data Mining yang efisien dan terstruktur, digunakanlah kerangka kerja standar seperti CRISP-DM (Cross-Industry Standard Process for Data Mining)[32][33]. Metodologi ini menyediakan panduan siklus hidup yang terperinci, dimulai dari Pemahaman Bisnis untuk mendefinisikan masalah, diikuti oleh Pemahaman Data dan

Persiapan Data yang sangat krusial dalam pembersihan dan transformasi data mentah[34]. Tahap selanjutnya melibatkan Pemodelan, di mana teknik penambangan data diaplikasikan, dan Evaluasi model yang dilakukan tidak hanya berdasarkan metrik teknis tetapi juga kelayakan bisnis. Melalui pendekatan yang sistematis ini, Data Mining dapat diterapkan secara efektif di berbagai sektor.

2.3 Teori tentang *Framework*/Algoritma yang digunakan

2.3.1 CRISP-DM (Cross-Industry Standard Process for Data Mining)

CRISP-DM (Cross-Industry Standard Process for Data Mining) adalah sebuah pendekatan terstruktur yang dirancang untuk memandu proses data mining secara sistematis[35]CRISP-DM menyediakan kerangka kerja iteratif yang terdiri dari enam tahapan utama, yang masing-masing saling berhubungan dan mendukung keberhasilan proyek bagai metodologi yang generik, CRISP-DM dapat diterapkan pada berbagai domain, termasuk analitik bisnis, ilmu kesehatan, pemasaran, dan teknologi informasi. Selain memberikan struktur yang sistematis, CRISP-DM juga mempromosikan dokumentasi yang lengkap di setiap tahap, sehingga hasilnya dapat dipertanggungjawabkan dan direplikasi.

2.3.1.1 Business Understanding

Tahap pertama, *Business Understanding*, merupakan fondasi awal dalam metodologi CRISP-DM. Pada tahap ini, tujuan bisnis yang ingin dicapai melalui penelitian didefinisikan dengan jelas. Peneliti melakukan identifikasi terhadap kebutuhan utama organisasi atau proyek, mendeskripsikan masalah secara rinci, dan merumuskan pertanyaan penelitian yang spesifik. Selain itu, indikator keberhasilan proyek juga ditetapkan untuk mengukur sejauh

mana hasil yang diperoleh dapat memenuhi tujuan yang telah dirumuskan. Pemahaman yang mendalam pada tahap ini sangat penting, karena akan menjadi panduan bagi seluruh langkah berikutnya dalam proses *data mining.*

2.3.1.2 Data Understanding

Setelah tujuan bisnis terdefinisi, tahap *Data Understanding* dilakukan untuk mengumpulkan dan mengeksplorasi data yang akan digunakan dalam penelitian. Peneliti menganalisis sumber data, mempelajari atribut-atribut penting dalam dataset, dan mengidentifikasi potensi masalah seperti nilai yang hilang, *outlier*, atau ketidakseimbangan kelas. Tahap ini bertujuan untuk memastikan bahwa data yang tersedia relevan dan cukup untuk mendukung proses analisis lebih lanjut. Selain itu, analisis eksplorasi data awal dilakukan untuk mengidentifikasi pola-pola awal atau wawasan yang dapat membantu dalam membangun model di tahap berikutnya.

2.3.1.3 Data Preparation

Tahap *Data Preparation* berfokus pada pembersihan dan transformasi data agar siap digunakan dalam proses modeling. Aktivitas yang dilakukan mencakup penghapusan nilai duplikat, penanganan nilai yang hilang, normalisasi atau standardisasi data, penghapusan *outlier*, serta pengkodean variabel kategorikal menjadi format numerik yang dapat digunakan oleh algoritma. Data yang telah diproses kemudian dibagi menjadi subset data pelatihan dan pengujian untuk memastikan model yang dibangun dapat diuji dan divalidasi dengan baik. Kualitas data yang disiapkan pada tahap ini akan sangat memengaruhi keberhasilan langkah *modeling*.

2.3.1.4 Modelling

Pada tahap *Modeling*, peneliti memilih dan menerapkan algoritma *data mining* yang sesuai untuk menjawab pertanyaan penelitian. Algoritma seperti *clustering, classification*, atau *regresi* digunakan tergantung pada tujuan yang ingin dicapai. Parameter algoritma juga dioptimalkan untuk menghasilkan model dengan performa terbaik. Selain itu, eksperimen dilakukan untuk membandingkan berbagai pendekatan algoritmik, guna memastikan metode yang paling efektif diterapkan dalam penelitian ini. Hasil dari tahap ini berupa model yang dihasilkan dari analisis dataset yang telah disiapkan sebelumnya.

2.3.1.5 Evaluation

Model yang telah dibangun dievaluasi pada tahap *Evaluation* untuk memastikan bahwa hasil yang diperoleh memenuhi tujuan yang telah dirumuskan pada tahap *Business Understanding*. Penilaian dilakukan dengan menggunakan metrik performa, seperti akurasi, presisi, tergantung pada jenis masalah yang dianalisis. Evaluasi ini juga mencakup analisis terhadap relevansi model dengan konteks bisnis atau penelitian yang sedang dilakukan. Jika model dinilai belum optimal, tahap ini memberikan ruang untuk iterasi dan perbaikan pada langkah sebelumnya.

2.3.1.6 Deployment

Tahap terakhir adalah *Deployment*, di mana model yang telah dievaluasi diterapkan ke dalam konteks bisnis atau operasional yang relevan. Hasil model dapat digunakan untuk mendukung pengambilan keputusan strategis, mengotomasi proses, atau memberikan prediksi yang berguna bagi organisasi. Selain itu, tahap ini juga mencakup

dokumentasi hasil dan penyusunan laporan formal, sehingga seluruh proses dan temuan dapat dipahami dan digunakan oleh pemangku kepentingan. Tahap ini memastikan bahwa manfaat dari penelitian dapat dirasakan secara praktis dan berkelanjutan.

2.3.2 DBSCAN

DBSCAN adalah algoritma clustering berbasis kepadatan yang mampu mengelompokkan data tanpa perlu menentukan jumlah cluster di awal[36]. Algoritma ini efektif untuk data berdimensi tinggi, tidak beraturan, dan mengandung noise, seperti data sensor industri atau data siber. DBSCAN menggunakan dua parameter utama: epsilon (ε), yaitu radius maksimum untuk menentukan tetangga suatu titik, dan MinPts, yaitu jumlah minimum titik dalam radius tersebut agar titik dianggap sebagai titik inti (core point). Titik-titik yang tidak memenuhi syarat akan dikategorikan sebagai noise[37].

Rumus neighborhood DBSCAN:.

$$N_\varepsilon(p) = \{q \in D \mid \text{dist}(p, q) \leq \varepsilon\}$$

Keterangan:

- $N_\varepsilon(p)$: himpunan titik-titik tetangga dari titik p
- D: himpunan seluruh data
- dist(p, q): fungsi jarak antara titik $p$dan q(umumnya Euclidean)
- ε: radius maksimum untuk menentukan tetangga

DBSCAN membentuk cluster dengan memperluas dari titik inti ke tetangga yang saling terjangkau. Titik-titik yang tidak termasuk dalam cluster manapun dianggap sebagai noise. Keunggulan utama DBSCAN adalah kemampuannya mengenali bentuk cluster arbitrer dan menangani outlier secara eksplisit, meskipun performanya sangat bergantung pada pemilihan parameter ε dan MinPts yang tepat[38].

2.3.3 Isolation Forest

Isolation Forest merupakan algoritma deteksi anomali berbasis pohon keputusan yang bekerja dengan prinsip isolasi terhadap titik data. Berbeda dengan pendekatan yang memodelkan normalitas, Isolation Forest justru berfokus pada pemisahan instans data secara acak[39]. Titik-titik yang menyimpang atau jarang muncul dalam distribusi data cenderung lebih mudah diisolasi, sehingga dapat dikenali sebagai anomali[40]. Algoritma ini sangat efisien untuk data berdimensi tinggi dan cocok digunakan dalam skenario *unsupervised* learning, seperti deteksi kerusakan mesin, fraud detection, dan monitoring sistem cyber.

Setiap pohon dalam Isolation Forest dibangun dengan memilih fitur dan nilai split secara acak, lalu membagi data hingga setiap instans terisolasi[41]. Kedalaman rata-rata dari jalur isolasi (path length) digunakan sebagai indikator apakah suatu titik merupakan anomali. Titik yang terisolasi dengan lebih sedikit pemisahan (jalur pendek) dianggap sebagai anomali, sedangkan titik yang membutuhkan lebih banyak pemisahan (jalur panjang) dianggap sebagai data normal.

Rumus Skor Anomali Isolation Forest

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

Keterangan:

- $s(x,n)$: skor anomali untuk titik $x$dengan total sampel n
- $E(h(x))$: rata-rata panjang jalur untuk mengisolasi x
- $c(n)$: nilai normalisasi, yaitu rata-rata panjang jalur dari data normal, dihitung sebagai:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}$$

Dengan $H(i)$adalah fungsi harmonik:

$$H(i) = \ln(i) + \gamma$$

dan $\gamma \approx 0.5772$ adalah konstanta Euler-Mascheroni.

Skor anomali $s(x, n)$mendekati 1 menunjukkan bahwa titik tersebut kemungkinan besar adalah anomali, sedangkan skor mendekati 0 menunjukkan bahwa titik tersebut normal.

.

2.3.4 AutoEncoder

Autoencoder merupakan arsitektur jaringan saraf tiruan yang digunakan dalam pembelajaran tidak terawasi (*unsupervised* learning) untuk merepresentasikan data secara efisien. Tujuan utama dari autoencoder adalah merekonstruksi inputnya sendiri melalui proses encoding dan decoding, sehingga model dapat mempelajari pola internal dari data tanpa memerlukan label[42]. Dalam konteks deteksi anomali, autoencoder sangat efektif karena data normal cenderung dapat direkonstruksi dengan baik, sedangkan data anomali menghasilkan error rekonstruksi yang tinggi[13].

Struktur dasar autoencoder terdiri atas tiga komponen utama, yaitu encoder, bottleneck (latent space), dan decoder. Encoder bertugas mengubah input berdimensi tinggi menjadi representasi berdimensi lebih rendah [43]. Bottleneck menyimpan informasi penting dalam bentuk kompresi, sedangkan decoder mencoba merekonstruksi kembali input asli dari representasi tersebut. Proses ini memungkinkan model untuk mengenali pola umum dalam data dan mendeteksi penyimpangan yang tidak sesuai dengan pola tersebut.

Rumus Rekonstruksi dan Loss Function

Misalkan $x \in \mathbb{R}^n$adalah input asli, maka:

- Encoding:

$$z = f(x) = \sigma(W_e x + b_e)$$

- Decoding:

$$\hat{x} = g(z) = \sigma(W_d z + b_d)$$

- Loss Function (Mean Squared Error):

$$L(x, \hat{x}) = \frac{1}{n}\sum_{i=1}^{n} \quad (x_i - \hat{x}_i)^2$$

Keterangan:

- W_e, W_d: bobot encoder dan decoder
- b_e, b_d: bias encoder dan decoder
- σ: fungsi aktivasi (misalnya ReLU atau sigmoid)
- z: representasi laten
- $\hat{x}$: hasil rekonstruksi

Semakin besar nilai loss, semakin besar kemungkinan bahwa input merupakan anomali.

Autoencoder memiliki keunggulan dalam menangani data berdimensi tinggi dan dapat disesuaikan dengan berbagai jenis data, termasuk data time-series dan citra. Namun, performa model sangat bergantung pada kualitas data latih dan pemilihan threshold yang tepat. Oleh karena itu, validasi dan tuning parameter tetap diperlukan untuk memastikan efektivitas deteksi anomali.

2.3.5 SVM

Support Vector Machine (SVM) merupakan algoritma pembelajaran mesin yang dirancang untuk menyelesaikan masalah klasifikasi dan regresi, dengan performa yang sangat baik terutama pada data berdimensi tinggi. Secara konseptual, SVM bekerja dengan mencari sebuah hyperplane optimal yang berfungsi sebagai batas pemisah antara dua kelas data[44]. Hyperplane ini dipilih sedemikian rupa agar memiliki margin maksimum, yaitu jarak terdekat antara hyperplane dengan titik-titik data dari masing-masing kelas. Titik-titik data yang berada tepat di batas margin disebut sebagai support vectors,

dan hanya titik-titik inilah yang berkontribusi langsung dalam proses pembentukan model klasifikasi[45].

Dalam kasus data yang tidak dapat dipisahkan secara linear (non-linear separable), SVM menggunakan pendekatan yang disebut kernel trick[46]. Teknik ini memungkinkan data input diproyeksikan secara implisit ke ruang berdimensi lebih tinggi, di mana pemisahan antar kelas menjadi lebih mudah dilakukan secara linear. Fungsi kernel seperti Radial Basis Function (RBF), Polynomial, dan Sigmoid digunakan untuk mentransformasikan data tanpa perlu menghitung koordinat eksplisit di ruang baru tersebut. Arsitektur ini menjadikan SVM sangat cocok untuk data yang kompleks, seperti teks, citra, atau sinyal biomedis, di mana pola pemisahan antar kelas tidak selalu terlihat secara langsung.

Secara matematis, SVM memformulasikan pencarian hyperplane sebagai masalah optimasi terkendala[47]. Tujuan utamanya adalah meminimalkan norma vektor bobot w, yang secara tidak langsung akan memaksimalkan margin antar kelas. Fungsi objektif dan batasan klasifikasi dapat dituliskan sebagai berikut:

$$\min_{\mathrm{w},b} \frac{1}{2} || \mathrm{w} ||^2$$

dengan syarat: $y_i(\mathrm{w} \cdot \mathrm{x}_i + b) \geq 1$untuk semua i

Di mana $\mathrm{x}_i$adalah vektor fitur dari data ke-i, $y_i \in \{-1, +1\}$ adalah label kelas, w adalah vektor bobot, dan $b$adalah bias. Optimasi dilakukan untuk mencari w dan b yang menghasilkan margin terbesar sambil tetap memisahkan data dengan benar.

Untuk mempermudah pemahaman, proses kerja SVM dapat dijelaskan secara pseudocode sebagai berikut:

1. Transformasi Data: Jika data tidak linear, gunakan fungsi kernel $K(\mathrm{x}_i, \mathrm{x}_j)$untuk memetakan data ke ruang berdimensi tinggi.

2. Optimasi: Tentukan wdan $b$yang meminimalkan $||\ \mathrm{w}\ ||^2$sambil memenuhi batasan klasifikasi.

3. Klasifikasi: Untuk data baru x', tentukan kelasnya berdasarkan tanda dari fungsi keputusan:

$$\mathrm{sign}(\mathrm{w} \cdot \mathrm{x}' + b)$$

2.3.6 XGBoost

XGBoost (Extreme Gradient Boosting) merupakan algoritma *Machine Learning* berbasis pohon keputusan yang dirancang untuk menghasilkan model prediktif yang akurat dan efisien. Algoritma ini termasuk dalam keluarga boosting, yaitu teknik ensemble yang membangun model secara bertahap dengan menggabungkan banyak pohon lemah (weak learners) menjadi satu model kuat (strong learner)[48]. Setiap pohon baru yang dibangun bertujuan untuk memperbaiki kesalahan prediksi dari pohon sebelumnya, sehingga proses pelatihan bersifat iteratif dan adaptif terhadap error[49].

Keunggulan utama XGBoost terletak pada optimasi gradien yang ekstrem dan regularisasi tambahan yang membuatnya tahan terhadap overfitting[48]. Selain itu, XGBoost dirancang untuk mendukung komputasi paralel, penanganan data hilang secara otomatis, serta efisiensi memori yang tinggi. Arsitektur ini menjadikannya sangat cocok untuk data tabular, data klasifikasi multi-kelas, maupun regresi, terutama dalam kompetisi data science dan aplikasi industri yang membutuhkan performa tinggi.

Secara metodologis, XGBoost bekerja dengan meminimalkan fungsi loss secara bertahap menggunakan pendekatan gradient descent [50]. Pada setiap iterasi, model membangun pohon baru berdasarkan gradien dari fungsi loss terhadap prediksi sebelumnya. Untuk menghindari kompleksitas model yang berlebihan, XGBoost

menambahkan komponen regularisasi ke dalam fungsi objektifnya. Fungsi objektif XGBoost secara umum dapat dituliskan sebagai:

$$\mathcal{L}(t) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k)$$

Di mana:

- $l(y_i, \hat{y}_i^{(t)})$adalah fungsi loss (misalnya log loss atau squared error) antara label aktual dan prediksi pada iterasi ke-t.
- $\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$adalah fungsi regularisasi untuk pohon ke-k, dengan Tjumlah daun dan $w_j$bobot pada daun ke-j.

Untuk mempermudah pemahaman, proses kerja XGBoost dapat dijelaskan secara pseudocode sebagai berikut:

1. Inisialisasi: Prediksi awal diset sebagai nilai rata-rata (untuk regresi) atau log odds (untuk klasifikasi).
2. Iterasi Boosting:
    - Hitung gradien dan hessian dari fungsi loss terhadap prediksi saat ini.
    - Bangun pohon keputusan baru yang meminimalkan loss berdasarkan gradien.
    - Tambahkan pohon ke model ensemble.
3. Prediksi Akhir: Gabungkan semua pohon untuk menghasilkan prediksi akhir.

Dengan pendekatan ini, XGBoost mampu menghasilkan model yang akurat, stabil, dan scalable, serta sangat kompetitif dalam berbagai tugas klasifikasi dan regresi.

2.3.7 Random Forest

Random Forest dapat diadaptasi untuk deteksi anomali. Salah satu pendekatannya adalah dengan mengukur anomali berdasarkan kedekatan sebuah titik data dengan titik data lainnya di dalam setiap pohon. Anomali cenderung berada lebih jauh dari titik-titik data normal, sehingga memiliki skor anomali[51] yang lebih tinggi. Model ini juga dapat digunakan dengan pendekatan one-class, di mana model dilatih hanya pada data normal, dan setiap data baru yang berada di luar batas yang dipelajari akan diklasifikasikan sebagai *anomaly* [52].

Dalam konteks deteksi anomali, salah satu varian dari *Random Forest* adalah Isolation Forest, yang secara spesifik dirancang untuk mengisolasi anomali secara efisien. Anomali diisolasi dengan membuat partisi acak pada data, di mana anomali biasanya membutuhkan lebih sedikit partisi (memiliki path length yang lebih pendek) dibandingkan titik data normal. Skor anomali (s(x,n)) dari sebuah titik data (x) dalam Isolation Forest dapat dihitung menggunakan formula:

$$H(x) = mode\{h_1(x), h_2(x), \dots, h_T(x)\}$$

$keterangan:$

- $H(x)$*: prediksi akhir Random Forest untuk input* x
- $h_t(x)$*: prediksi dari pohon ke-*t
- T*: jumlah total pohon dalam hutan*

di mana h(x) adalah panjang jalur (path length) dari titik x dalam sebuah pohon, E(h(x)) adalah rata-rata h(x) dari semua pohon, dan c(n) adalah konstanta normalisasi yang bergantung pada ukuran dataset (n) [14]. Nilai skor anomali yang mendekati 1 menunjukkan anomali kuat, sedangkan nilai yang mendekati 0 menunjukkan data normal.

2.3.8 SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE (Synthetic Minority Over-sampling Technique) adalah metode yang digunakan untuk menangani permasalahan ketidakseimbangan kelas (class imbalance) dalam dataset, khususnya pada kasus klasifikasi di mana jumlah sampel dari satu kelas (biasanya kelas minoritas) jauh lebih sedikit dibandingkan kelas lainnya. Ketidakseimbangan ini dapat menyebabkan model *Machine Learning* underfit atau cenderung bias terhadap kelas mayoritas, sehingga performa klasifikasi terhadap kelas minoritas menjadi buruk.

Berbeda dengan metode oversampling konvensional yang hanya menggandakan data minoritas secara acak (random duplication), SMOTE bekerja dengan cara menghasilkan sampel sintetis baru berdasarkan interpolasi antara titik data minoritas yang ada[53]. Proses ini dilakukan dengan memilih secara acak salah satu tetangga terdekat dari setiap sampel minoritas, kemudian membuat data baru yang berada di antara keduanya dalam ruang fitur. Dengan demikian, SMOTE tidak hanya menambah jumlah data minoritas, tetapi juga memperluas distribusi data minoritas secara lebih alami di ruang fitur[54].

Secara umum, langkah kerja SMOTE dapat dijelaskan sebagai berikut:

1. Untuk setiap sampel minoritas x_i, cari $k$tetangga terdekatnya dalam kelas yang sama.
2. Pilih salah satu tetangga x_{zi}secara acak.
3. Buat sampel sintetis x_{new}dengan interpolasi linier:

$$x_{\text{new}} = x_i + \delta \cdot (x_{zi} - x_i)$$

Di mana $\delta \in [0,1]$adalah bilangan acak.

Dengan pendekatan ini, SMOTE mampu mengurangi risiko overfitting yang sering terjadi pada oversampling biasa, serta

membantu model untuk belajar representasi yang lebih baik dari kelas minoritas. SMOTE sangat cocok digunakan pada dataset klasifikasi biner atau multi-kelas yang memiliki distribusi tidak seimbang, terutama ketika data minoritas terlalu sedikit untuk dilatih secara efektif[55].

Namun, perlu diperhatikan bahwa SMOTE bekerja di ruang fitur, sehingga sensitivitas terhadap noise dan outlier tetap menjadi tantangan. Oleh karena itu, dalam praktiknya, SMOTE sering dikombinasikan dengan teknik lain seperti under-sampling kelas mayoritas atau algoritma klasifikasi yang robust terhadap ketidakseimbangan data seperti SVM, XGBoost, Random Forest, Logistic Regression, Naïve bayes dan algo lainnya [56].

2.3.9 Class Weighting

Class weighting adalah pendekatan statistik yang digunakan untuk mengatasi masalah ketidakseimbangan kelas (class imbalance) dalam algoritma klasifikasi. Ketika distribusi data antar kelas tidak seimbang misalnya, satu kelas jauh lebih dominan dibandingkan kelas lainnya maka model cenderung bias terhadap kelas mayoritas dan mengabaikan kelas minoritas[57]. Untuk mengurangi bias ini, class weighting memberikan bobot yang lebih besar pada kelas yang jarang muncul, sehingga kesalahan prediksi terhadap kelas tersebut akan dikenakan penalti yang lebih tinggi selama proses pelatihan[58].

Secara umum, bobot kelas dihitung secara invers terhadap frekuensi kemunculan kelas dalam data pelatihan. Semakin sedikit jumlah sampel dari suatu kelas, semakin besar bobot yang diberikan. Dalam implementasi praktis, banyak algoritma seperti Logistic Regression, Support Vector Machine (SVM), Random Forest, dan XGBoost menyediakan parameter class_weight atau scale_pos_weight yang dapat diatur secara manual atau otomatis (balanced) berdasarkan distribusi data.

Rumus umum untuk menghitung bobot kelas secara otomatis adalah:

$$w_j = \frac{n_{\text{samples}}}{n_{\text{classes}} \cdot n_j}$$

Di mana:

- w_j adalah bobot untuk kelas ke-j,
- n_{samples} adalah total jumlah sampel,
- n_{classes} adalah jumlah total kelas,
- n_j adalah jumlah sampel pada kelas ke-j.

Dengan menggunakan bobot ini, algoritma klasifikasi akan lebih sensitif terhadap kesalahan prediksi pada kelas minoritas, sehingga meningkatkan kemampuan model dalam mengenali pola dari kelas yang jarang muncul. Teknik ini sangat berguna ketika tidak memungkinkan untuk melakukan oversampling (seperti SMOTE) atau undersampling, terutama pada data yang terbatas atau bersifat sensitif terhadap modifikasi distribusi.

Class weighting dapat diterapkan secara langsung dalam parameter model atau melalui fungsi loss yang dimodifikasi[59]. Dalam konteks pembelajaran mesin modern, pendekatan ini sering dikombinasikan dengan teknik lain seperti resampling, threshold tuning, atau pemilihan metrik evaluasi yang lebih representatif seperti *F1-score* dan ROC-AUC.

2.3.10 Bootstrap Resampling

Bootstrap resampling adalah teknik statistik yang digunakan untuk memperkirakan distribusi suatu parameter atau meningkatkan jumlah data secara sintetis melalui proses pengambilan sampel ulang (resampling) dari data yang tersedia. Metode ini bekerja dengan cara mengambil sampel secara acak dari dataset asli, dengan pengembalian (with replacement), sehingga satu titik data dapat muncul lebih dari

sekali dalam satu sampel bootstrap[60]. Teknik ini sangat berguna ketika jumlah data terbatas, namun diperlukan estimasi yang stabil atau pelatihan model yang lebih robust.

Dalam konteks *Machine Learning* dan analisis data, bootstrap sering digunakan untuk dua tujuan utama yaitu memperkirakan variabilitas model atau metrik evaluasi, dan meningkatkan jumlah data pelatihan secara sintetis tanpa menambah data baru dari luar. Dengan membuat banyak versi dataset yang sedikit berbeda, bootstrap memungkinkan model untuk belajar dari variasi internal data, sehingga dapat meningkatkan generalisasi dan mengurangi overfitting[61].

Secara prosedural, langkah-langkah bootstrap resampling dapat dijelaskan sebagai berikut:

1. Diberikan dataset asli berukuran n, misalnya $D = \{x_1, x_2, \ldots, x_n\}$.
2. Buat satu sampel bootstrap D^*dengan mengambil $n$titik data dari D, secara acak dan dengan pengembalian.
3. Ulangi proses ini sebanyak Bkali untuk menghasilkan Bsampel bootstrap.
4. Gunakan setiap D^*untuk pelatihan model atau estimasi parameter, lalu analisis distribusi hasilnya.

Contoh sederhana rumus estimasi rata-rata bootstrap dari parameter θ adalah:

$$\hat{\theta}_{\text{bootstrap}} = \frac{1}{B}\sum_{b=1}^{B} \hat{\theta}^{(b)}$$

Di mana $\hat{\theta}^{(b)}$adalah estimasi dari sampel bootstrap ke-b, dan Badalah jumlah iterasi bootstrap.

Bootstrap sangat cocok digunakan pada dataset kecil, data yang sulit dikumpulkan ulang, atau ketika distribusi data tidak diketahui secara pasti. Teknik ini juga menjadi fondasi dari metode ensemble seperti bagging (Bootstrap Aggregating), yang digunakan dalam algoritma seperti *Random Forest* untuk meningkatkan stabilitas dan akurasi model[62].

2.3.11 Uji Ablasi

Uji Ablasi adalah metodologi eksperimental krusial yang digunakan untuk memvalidasi dan mengukur kontribusi spesifik dari setiap komponen atau fitur yang membentuk sebuah sistem komputasi kompleks. Konsep ini dipinjam dari ilmu biologi, di mana penghilangan bagian tertentu dilakukan untuk memahami fungsinya. Dalam konteks Machine Learning, uji ablasi dilakukan dengan cara menghilangkan (ablating) atau mengganti salah satu elemen utama dari model terlengkap (baseline atau kontrol) kemudian mengukur penurunan kinerja yang terjadi pada metrik evaluasi seperti F1-score atau PR AUC[63]. Jika penghilangan komponen tersebut menyebabkan penurunan kinerja yang signifikan, hal ini secara empiris membuktikan bahwa komponen tersebut esensial dan memberikan kontribusi unik terhadap hasil keseluruhan sistem.

Metodologi uji ablasi sangat penting diterapkan pada model *hybrid* karena keberhasilan model tersebut bergantung sepenuhnya pada sinergi antara dua arsitektur yang berbeda. Dalam skripsi ini, model *Hybrid* terdiri dari dua fase: fase pertama adalah Feature Extractor (unsupervised) dan fase kedua adalah Classifier (supervised).

2.3.12 Metrik Evaluasi

Untuk mengevaluasi performa model deteksi anomali, metrik yang umum digunakan adalah *Precision*, *Recall*, dan *F1-score* . Metrik ini sangat penting, terutama dalam kasus anomali yang jarang terjadi

(imbalanced data), karena akurasi saja tidak cukup untuk menilai kinerja model secara komprehensif [64].

2.3.5.1 Precision

Precision mengukur proporsi anomali yang diprediksi dengan benar dari seluruh prediksi anomali yang dilakukan model. Dengan kata lain, metrik ini menjawab pertanyaan: "Dari semua yang model prediksi sebagai anomali, seberapa banyak yang benar-benar merupakan anomali?" Nilai presisi yang tinggi menunjukkan bahwa model memiliki tingkat kesalahan prediksi positif yang rendah (*low false positive rate*), yang sangat penting dalam konteks perawatan prediktif. Prediksi positif palsu dapat menyebabkan *maintenance* yang tidak perlu dan membuang-buang sumber daya [64] . Rumus untuk *Precision* adalah sebagai berikut:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

.

Di mana:

- True Positives (TP) adalah jumlah anomali yang berhasil dideteksi dengan benar oleh model.
- *False positive*s (FP) adalah jumlah data normal yang salah diklasifikasikan sebagai anomali oleh model.

2.3.5.2 Recall

Recall mengukur proporsi anomali yang berhasil dideteksi oleh model dari seluruh anomali yang sebenarnya ada dalam dataset. Metrik ini menjawab pertanyaan: "Dari semua anomali yang benar-benar terjadi, seberapa banyak yang berhasil dideteksi oleh model?" Nilai *Recall* yang tinggi sangat krusial dalam perawatan prediktif karena kegagalan dalam mendeteksi anomali (false negative) dapat berakibat fatal, seperti kerusakan mesin yang tak terduga dan

downtime produksi yang signifikan[65]. Rumus untuk *Recall* adalah:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Di mana:

- True Positives (TP) adalah jumlah anomali yang berhasil dideteksi dengan benar oleh model.
- False Negatives (FN) adalah jumlah anomali yang gagal dideteksi oleh model (salah diklasifikasikan sebagai data normal).

2.3.5.3 F1-score

F1-score merupakan metrik yang menggabungkan *Precision* dan *Recall* ke dalam satu nilai tunggal. Metrik ini memberikan keseimbangan antara *Precision* dan *recall*, sehingga sangat berguna saat ada ketidakseimbangan antara biaya *false positive* dan *False Negative* [66]. *F1-score* sering digunakan untuk mengevaluasi kinerja model pada dataset yang tidak seimbang (imbalanced dataset), di mana jumlah anomali jauh lebih sedikit daripada data normal. Semakin tinggi nilai *F1-score* , semakin baik kinerja model secara keseluruhan dalam mengidentifikasi anomali. Rumusnya adalah:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.3.5.4 PR - AUC

Precision-Recall AUC merupakan metrik evaluasi yang mengukur kualitas model dalam membedakan kelas positif (anomali) dari kelas negatif (normal) dengan mempertimbangkan trade-off antara *Precision* dan *Recall* pada berbagai threshold[67]. Metrik ini sangat berguna pada dataset yang tidak seimbang (imbalanced dataset), karena lebih fokus pada performa deteksi kelas minoritas

dibandingkan ROC AUC yang bisa bias terhadap kelas mayoritas [66]. Semakin tinggi nilai PR AUC, semakin baik kemampuan model dalam menjaga keseimbangan antara *Precision* dan *Recall* di seluruh threshold.

Rumus *PR AUC* secara umum adalah integral dari kurva *Precision-Recall*:

$$PR\_AUC = \int_0^1 Precision(Recall)\ d(Recall)$$

Dimana :

- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$

Kurva *Precision-Recall* dibentuk dengan memvariasikan threshold prediksi, dan area di bawah kurva (AUC) dihitung sebagai ukuran agregat performa model.

2.3.5.5 Confidence Interval

Confidence Interval (CI) digunakan untuk memberikan batas ketidakpastian pada estimasi metrik evaluasi [68]. CI 95% berarti bahwa jika eksperimen diulang berkali-kali, maka 95% dari interval yang dihitung akan mengandung nilai metrik sebenarnya. Dalam konteks evaluasi model, CI membantu memastikan bahwa perbedaan performa antar model bukan hanya akibat variasi sampel, tetapi benar-benar signifikan:

$$CI_{95\backslash\%} = \hat{\theta} \pm 1.96 \cdot SE(\hat{\theta})$$

di mana:

- $\hat{\theta}$= estimasi metrik (misalnya PR AUC)

- $SE(\hat{\theta})$= standard error dari estimasi
- $Z_{0.975} = 1.96$(nilai z untuk distribusi normal pada tingkat kepercayaan 95%)

2.4 Teori tentang Tools / Software yang digunakan

2.4.1 Jupyter Notebook

Jupyter Notebook adalah *platform open-source* yang dirancang untuk mendukung komputasi interaktif, memungkinkan pengguna menggabungkan kode, teks naratif, persamaan, dan visualisasi interaktif dalam satu dokumen[69]. Sejak dikembangkan dari proyek *IPython* pada 2014, *Jupyter* telah menjadi alat penting dalam data science dan komputasi ilmiah karena fleksibilitasnya untuk digunakan dalam lebih dari 40 bahasa pemrograman, termasuk *Python, R*, dan *Julia*[70]. Dengan struktur interaktif yang memungkinkan eksekusi kode secara blok dan analisis langsung, *Jupyter* mendukung proses iteratif "*write-eval-think loop*," yang mendukung eksplorasi dan pemahaman data secara bertahap. Fitur naratif komputasi *Jupyter* juga menjadikannya alat yang kuat untuk berbagi pengetahuan yang dapat direproduksi. Selain itu, *Jupyter* memiliki ekosistem luas, seperti *JupyterLab* sebagai antarmuka yang lebih canggih, *JupyterHub* untuk multi-pengguna, dan Binder untuk mengonversi repositori *Git* menjadi *server Jupyter* langsung[71]. Kemampuan *Jupyter* untuk mendukung standar terbuka dan integrasi dengan alat lain menjadikannya platform kolaboratif utama bagi komunitas ilmiah dalam berbagai disiplin ilmu.

2.4.2 Google Collab

Google Colaboratory (Google Colab) adalah sebuah layanan berbasis cloud yang dikembangkan oleh Google untuk menjalankan dan berbagi kode Python secara interaktif melalui notebook berbasis *Jupyter Notebook*. *Google Colab* menyediakan lingkungan pemrograman yang tidak memerlukan instalasi perangkat lunak tambahan pada komputer pengguna, karena seluruh proses komputasi dilakukan di server milik Google[72]. Layanan ini sangat

populer di kalangan peneliti, praktisi data, dan pelajar, karena mendukung berbagai pustaka populer seperti NumPy, Pandas, TensorFlow, PyTorch, Scikit-learn, dan Matplotlib. Dengan adanya dukungan GPU (Graphics Processing Unit) dan TPU (Tensor Processing Unit) gratis, Google Colab memungkinkan pemrosesan data dalam skala besar dan pelatihan model *Machine Learning* secara lebih cepat dan efisien dibandingkan hanya menggunakan CPU biasa.

Selain kemudahan akses dan ketersediaan sumber daya komputasi, Google Colab juga terintegrasi dengan Google Drive sehingga pengguna dapat menyimpan, mengakses, dan membagikan notebook secara mudah. Lingkungan ini mendukung eksekusi kode Python baris demi baris (cell execution), sehingga memudahkan proses eksplorasi data, pengembangan model, serta visualisasi hasil secara langsung[73]. Keunggulan lain adalah kemampuannya dalam kolaborasi waktu nyata (real-time collaboration), di mana beberapa pengguna dapat mengedit notebook yang sama secara bersamaan, mirip seperti fitur kolaborasi pada Google Docs. Dengan kombinasi fleksibilitas, kinerja, dan kemudahan akses, Google Colab menjadi salah satu platform favorit untuk eksperimen dan pengembangan proyek-proyek *Machine Learning*, analisis data, serta pembelajaran pemrograman Python secara interaktif.

2.4.3 Python

Python adalah bahasa pemrograman interpretatif yang pertama kali dikembangkan oleh *Guido van Rossum* pada 1991 dan telah menjadi salah satu bahasa terpopuler di dunia. Popularitasnya terutama didorong oleh sintaksis yang sederhana dan mirip bahasa Inggris, sehingga mudah dipelajari oleh pemula sekaligus efektif digunakan oleh profesional[74]. *Python* mendukung berbagai paradigma pemrograman, termasuk prosedural, berorientasi objek, dan fungsional, sehingga fleksibel dalam penggunaannya di berbagai bidang, seperti pengembangan *web*, aplikasi *desktop, Internet of Things*, dan *data science. Python* juga menyediakan

ekosistem *library* yang sangat luas, mulai dari analisis data dan *Machine Learning*, seperti *Pandas* dan *Scikit-Learn,* hingga alat visualisasi seperti *Matplotlib* dan *Seaborn*[75]. Kombinasi keunggulan ini membuat *Python* menjadi pilihan yang sangat populer bagi kalangan akademisi, peneliti, dan praktisi industri yang memerlukan bahasa pemrograman yang cepat, efektif, dan memiliki integrasi yang kuat dengan teknologi-teknologi terbaru.