BAB 2
LANDASAN TEORI

Bagian ini akan menjelaskan teori-teori yang mendasari penelitian ini, yaitu: *English Premiere League* (EPL), *deep learning*, *Long Short -Term Memory* (LSTM) dan metrik evaluasi.

2.1 *English Premiere League* (EPL)

English Premier League (EPL) merupakan kompetisi liga sepak bola profesional tingkat tertinggi di Inggris yang diakui secara global sebagai salah satu liga paling populer, kompetitif, dan bernilai secara finansial di dunia. Dengan jangkauan siaran ke ratusan negara dan basis penggemar miliaran orang, EPL tidak hanya menjadi tontonan utama bagi komunitas sepak bola global tetapi juga entitas industri olahraga bernilai besar, di mana klub-klubnya secara konsisten mendominasi daftar pendapatan tertinggi di Eropa berkat hak siar masif dan daya tarik komersialnya [13, 14]. Tingginya intensitas permainan, kehadiran pemain-pemain bintang internasional, dan persaingan ketat antar klub menjadikan EPL sebagai subjek analisis dan diskusi yang tak pernah habis di kalangan penggemar, media, maupun akademisi yang tertarik pada dinamika dan prediksi dalam olahraga.

Secara teknis, kompetisi EPL diikuti oleh 20 klub yang masing-masing memainkan total 38 pertandingan dalam satu musim dengan format *round-robin* (kandang dan tandang). Perolehan poin diatur dengan memberikan tiga poin untuk kemenangan, satu poin untuk hasil imbang, dan nol poin untuk kekalahan. Peringkat akhir klub dalam klasemen ditentukan berdasarkan total poin yang dikumpulkan. Jika terdapat poin yang sama, penentuan peringkat dilakukan berdasarkan selisih gol, kemudian jumlah gol yang dicetak, dan jika masih sama, akan merujuk pada kriteria lebih lanjut seperti hasil pertemuan langsung atau bahkan play-off dalam kondisi tertentu. Pada akhir musim, tiga tim dengan peringkat terbawah akan terdegradasi ke *English Football League* (EFL), sementara dua tim teratas dari EFL bersama satu tim pemenang EFL akan mendapatkan promosi ke EPL, memastikan adanya sirkulasi dan dinamika kompetitif antar musim [15].

2.2 Artificial Intelligence, Machine Learning dan Deep Learning

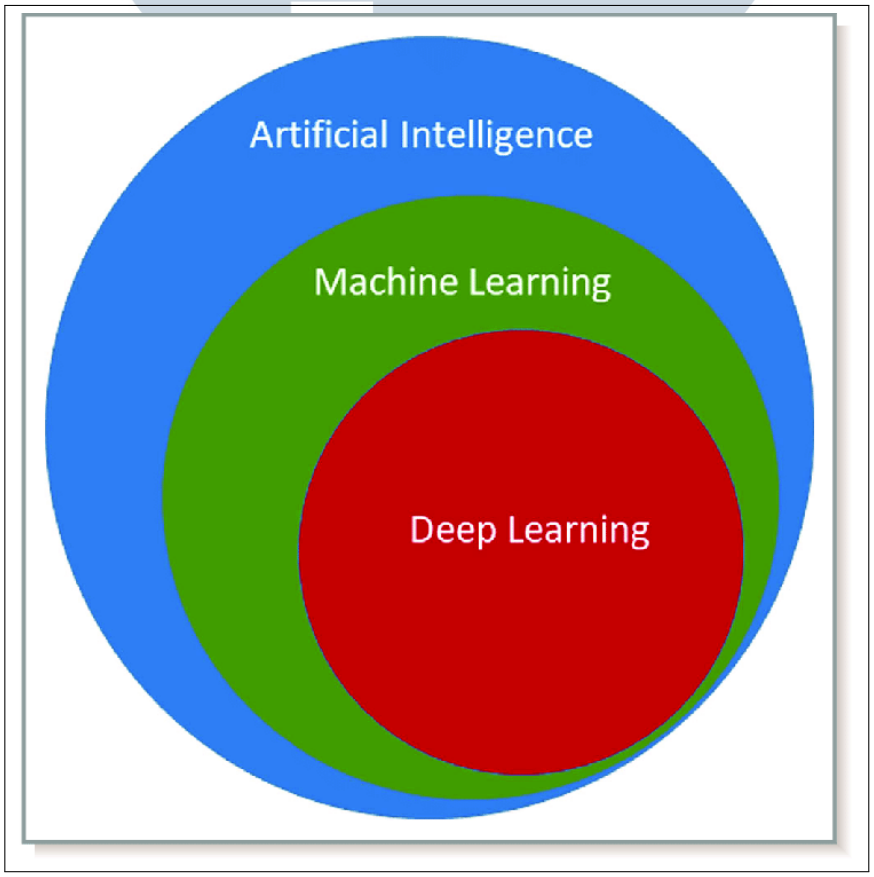
Bidang teknologi informasi terus berkembang pesat, ditandai dengan munculnya berbagai konsep canggih seperti *Artificial Intelligence* (AI), *Machine Learning* (ML), dan *Deep Learning* (DL). Ketiga konsep ini saling terkait secara hierarkis dan merupakan pilar utama dalam menciptakan sistem komputasi yang cerdas [16]. *Artificial Intelligence* (AI), atau Kecerdasan Buatan, merupakan bidang ilmu komputer yang berfokus pada pengembangan sistem yang mampu melakukan tugas-tugas yang umumnya membutuhkan kecerdasan manusia [17]. AI bertujuan untuk menciptakan mesin yang dapat berpikir, belajar, membuat keputusan, dan menyelesaikan tugas-tugas kompleks layaknya manusia [18]. Konsep-konsep dalam AI sangat luas, meliputi *machine learning*, *deep learning*, robotika, *computer vision*, *natural language processing* (NLP), dan sistem rekomendasi. AI memungkinkan efisiensi tinggi karena sebagian besar operasinya tidak bergantung pada intervensi manusia.

Machine Learning (ML) adalah sub-bidang dari *Artificial Intelligence* yang memungkinkan sistem komputer untuk belajar dari data dan pola tanpa pemrograman eksplisit [19]. Definisi ML dapat dijelaskan sebagai penerapan algoritma matematika dan teknologi komputasi yang memproses data melalui tahap latihan (*training*) dan pengujian (*testing*) untuk menghasilkan prediksi di masa depan [17]. Proses pembelajaran ini bertujuan agar sistem mampu menjawab berbagai masalah secara cerdas. ML berkembang pesat dengan berbagai metode dan aplikasi menarik, termasuk interpretabilitas algoritma, ketahanan, privasi, keadilan, inferensi kausalitas, interaksi manusia-mesin, dan keamanan. Tujuan utama ML bukan untuk menghasilkan ”tebakan sempurna”, melainkan tebakan yang cukup akurat untuk memberikan manfaat nyata, terutama dalam domain di mana solusi sempurna tidak mungkin tercapai [17]. *Machine Learning* terbagi dalam beberapa jenis utama: *supervised learning* (pembelajaran terawasi), *unsupervised learning* (pembelajaran tanpa terawasi), dan *reinforcement learning* (pembelajaran penguatan) [20].

Deep Learning (DL) merupakan cabang dari *machine learning* yang didasarkan pada konsep jaringan saraf buatan (*artificial neural networks*) dengan banyak lapisan (*layers*) [21]. DL merupakan teknologi yang mampu belajar representasi data secara hierarkis dari data mentah. Model *deep learning* membangun abstraksi data melalui beberapa lapisan, di mana setiap konsep dihubungkan dengan konsep yang lebih sederhana, dan representasi yang lebih

abstrak dihitung berdasarkan representasi yang kurang abstrak. Meskipun sangat kuat dalam mengenali pola dan memiliki fleksibilitas luar biasa untuk berbagai aplikasi seperti perawatan kesehatan, pengenalan visual, analisis teks, dan keamanan siber, membangun model *deep learning* yang sesuai tetap merupakan tugas yang menantang karena sifat dinamis dan variasi data di dunia nyata.

Pada Gambar 2.1 secara sederhana, *Artificial Intelligence* adalah konsep payung besar yang mencakup segala upaya untuk membuat mesin cerdas [22]. *Machine Learning* adalah bagian dari AI yang memungkinkan sistem belajar dari data. Sementara itu, Deep Learning adalah bagian dari*machine learning* yang menggunakan arsitektur jaringan saraf dengan banyak lapisan untuk pembelajaran yang lebih mendalam dan otomatis [22]. Hubungan ini sering digambarkan sebagai serangkaian lingkaran konsentris, di mana AI adalah lingkaran terluar, ML di dalamnya, dan DL di inti [20]. Pemahaman yang komprehensif tentang ketiganya sangat penting dalam konteks penelitian yang memanfaatkan teknologi ini.



Gambar 2.1. Diagram Ruang Lingkup AI, ML, dan DL

Sumber: [23]

2.3 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) merupakan arsitektur khusus dari Jaringan Saraf Berulang (*Recurrent Neural Networks*/RNN) yang dirancang untuk memproses urutan data yang panjang, seperti deret waktu atau urutan kata dalam kalimat [24]. Arsitektur ini dikembangkan untuk mengatasi permasalahan *vanishing gradient* pada model jaringan saraf tiruan tradisional, yang terjadi ketika gradien penyesuaian model menjadi sangat kecil sehingga menghambat kemampuan model dalam memperbarui parameter-parameter penting untuk pembelajaran yang efektif [25]. Keunggulan LSTM dibandingkan RNN konvensional terletak pada kemampuannya untuk mengingat informasi selama periode waktu yang arbitrer, bahkan ribuan *time-step* [26]. Kemampuan ini menjadikan LSTM sangat cocok untuk tugas-tugas yang melibatkan data berurutan seperti pemrosesan bahasa alami, pengenalan suara, dan prediksi deret waktu [27].

LSTM memiliki keunggulan signifikan dibandingkan RNN konvensional dalam memproses urutan panjang. RNN standar mengalami kesulitan dalam mempertahankan informasi dari langkah-langkah waktu yang jauh karena tidak memiliki mekanisme eksplisit untuk mengendalikan aliran informasi. Sebaliknya, LSTM dilengkapi dengan struktur gerbang—gerbang input, gerbang lupa, dan gerbang keluaran—yang memungkinkan jaringan secara selektif menyimpan, melupakan, atau mengeluarkan informasi sesuai kebutuhan. Dengan adanya *cell state* sebagai memori jangka panjang, LSTM mampu mempertahankan informasi penting selama ribuan langkah waktu tanpa mengalami degradasi performa [26].

Masalah *vanishing gradient* pada RNN terjadi karena sifat turunan dari fungsi aktivasi seperti sigmoid dan tanh yang nilainya kurang dari satu. Saat dilakukan propagasi balik melalui banyak langkah waktu, hasil perkalian turunan berantai ini menjadi sangat kecil, sehingga gradien yang mengalir ke lapisan awal mendekati nol. Akibatnya, parameter di lapisan awal sulit diperbarui secara efektif, menghambat proses pembelajaran [25, 28].

LSTM mengatasi masalah ini melalui arsitekturnya yang unik. Mekanisme *cell state* menyediakan jalur aliran informasi linier yang memungkinkan gradien mengalir lebih stabil selama propagasi balik. Selain itu, operasi penjumlahan dan fungsi gerbang sigmoid yang dikombinasikan dengan input terkontrol membantu mempertahankan informasi dan gradien dalam nilai yang tidak terlalu kecil, sehingga LSTM lebih tahan terhadap masalah *vanishing gradient* dibandingkan RNN biasa.

Setiap unit LSTM umumnya terdiri dari sebuah sel memori (*cell state*) dan tiga gerbang pengatur: gerbang input (*input gate*), gerbang lupa (*forget gate*), dan gerbang keluaran (*output gate*). Sel memori berfungsi untuk menyimpan informasi sepanjang interval waktu yang berbeda, sementara gerbang-gerbang tersebut secara selektif mengontrol aliran informasi masuk dan keluar dari sel memori tersebut [29]. Kemampuan untuk secara selektif mempertahankan atau membuang informasi ini memungkinkan jaringan LSTM untuk mempelajari dependensi jangka panjang secara efektif. Gerbang input berfungsi mengontrol aliran informasi baru yang masuk ke sel memori [30], sedangkan gerbang lupa mengatur informasi yang akan dihapus dari sel memori. Gerbang keluaran menentukan informasi yang akan digunakan sebagai output dari sel memori LSTM.

Langkah pertama dalam arsitektur LSTM adalah menentukan informasi yang harus disimpan atau diabaikan dalam sel memori [30]. Bagian ini dilakukan oleh Forget Gate yang bertugas membuat keputusan tentang informasi apa yang harus diabaikan, seperti dirumuskan pada Persamaan (2.1).

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.1}$$

Tahap berikutnya adalah menentukan informasi baru yang akan disimpan dalam sel memori [30]. Proses ini terdiri dari dua tahapan: lapisan sigmoid yang disebut *input gate layer* yang menentukan nilai mana yang perlu diperbarui seperti pada Persamaan (2.2), dan lapisan tanh yang menghasilkan vektor sebagai nilai kandidat baru ($\tilde{C}_t$) untuk ditambahkan ke *cell state*, sebagaimana ditunjukkan dalam Persamaan (2.3).

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.2}$$

$$\tilde{C}t = \tanh(WC \cdot [h_{t-1}, x_t] + b_C) \tag{2.3}$$

Kedua tahapan ini kemudian digabungkan untuk memperbarui *cell state*, sebagaimana terlihat dalam Persamaan (2.4). *Cell state* lama (C_{t-1}) akan diperbarui menjadi *cell state* baru (C_t) dengan menggabungkan hasil dari gerbang lupa (f_t) dan hasil dari input gate serta kandidat nilai baru ($i_t \cdot \tilde{C}_t$).

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{2.4}$$

Tahapan akhir adalah menentukan hasil keluaran. Pertama, dilakukan perhitungan oleh *output gate* seperti dalam Persamaan (2.5) untuk menentukan informasi mana yang akan digunakan sebagai *output*. Kemudian, *cell state* yang telah diperbarui akan diteruskan melalui fungsi aktivasi tanh dan dikalikan dengan *output gate*, seperti yang ditunjukkan dalam Persamaan (2.6) [29].

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.5}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{2.6}$$

Model LSTM terdiri dari empat status utama yang masing-masing bertanggung jawab dalam mengatur aliran informasi: gerbang lupa, gerbang input, gerbang keluaran, dan sel memori baru [30]. Pada setiap langkah waktu t, input saat ini x_t adalah vektor berdimensi $1 \times m$ yang terdiri dari m fitur, dan keluaran memiliki dimensi $1 \times n$ [26]. Dalam struktur konvensional LSTM, bobot dan bias awal diinisialisasi secara acak dan tetap selama inisialisasi awal [30]. Parameter-parameter penting dalam model ini termasuk matriks bobot W_f, W_i, W_c, dan W_o yang menghubungkan input dengan masing-masing gerbang dan unit memori baru [30]. Bias awal dari masing-masing status (f, i, c, dan o) juga dimulai dengan nilai acak tetapi akan diperbarui secara independen selama pelatihan melalui proses *backpropagation* [31]. Oleh karena itu, vektor input keseluruhan adalah $[x_t, b]$ dengan dimensi $1 \times (m+1)$, dan matriks keluaran yang sesuai memiliki ukuran $k \times n$, di mana k adalah panjang urutan (*sequence length*) [26].

Fungsi sigmoid merupakan fungsi aktivasi non-linear yang umum digunakan dalam berbagai lapisan jaringan saraf [31]. Fungsi ini memiliki sifat kontinu, terdiferensiasi, dan nilainya dibatasi antara 0 dan 1. Fungsi ini dapat dinyatakan dalam Persamaan (2.7).

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.7}$$

Fungsi sigmoid banyak digunakan dalam klasifikasi biner, regresi logistik, serta pada bagian output jaringan saraf untuk menghasilkan probabilitas [31]. Namun, penggunaannya secara berlebihan pada jaringan yang dalam dapat menyebabkan masalah *vanishing gradient*, terutama karena turunan fungsi ini akan sangat kecil di ujung domain.

Fungsi tangen hiperbolik (*tanh*) adalah fungsi aktivasi yang bersifat zero-centered dan lebih halus dibandingkan sigmoid. Fungsi ini mengembalikan nilai dalam rentang -1 hingga 1 dan cocok untuk jaringan multi-layer karena mendukung konvergensi yang lebih baik [32]. Fungsi ini dapat dituliskan dalam Persamaan (2.8).

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.8}$$

Meski memiliki performa yang lebih baik dalam banyak kasus, fungsi tanh tetap tidak kebal terhadap masalah *vanishing gradient*, yang menjadi tantangan besar pada jaringan saraf berlapis dalam [32].

Masalah *vanishing gradient* terjadi ketika gradien yang dihitung untuk parameter lapisan awal menjadi sangat kecil sehingga menghambat pembelajaran efektif [28]. Hal ini terjadi ketika proses pelatihan menghitung turunan berantai melalui banyak lapisan aktivasi seperti sigmoid. Jika masing-masing turunan berada di bawah 1, maka hasil perkaliannya akan semakin kecil, bahkan mendekati nol [25]. Fenomena ini dijelaskan lebih lanjut dalam Persamaan (2.9).

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial a_n} \cdot \frac{\partial a_n}{\partial a_{n-1}} \cdot \ldots \cdot \frac{\partial a_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial W_1} \tag{2.9}$$

Ketika semua $\frac{\partial a_{i+1}}{\partial a_i} < 1$, maka semakin banyak lapisan yang dilalui, semakin kecil pula gradien pada W_1. Ini menyebabkan pembaruan parameter menjadi tidak efektif, terutama pada lapisan awal yang seharusnya menangkap pola dasar dari data [25].

Gradient dihitung menggunakan metode *backpropagation*, yaitu dengan menyusuri jaringan dari belakang ke depan dan menghitung turunan bertingkat menggunakan aturan rantai [24]. Proses ini sangat rentan terhadap fenomena *vanishing gradient* ketika digunakan bersama fungsi-fungsi aktivasi seperti sigmoid atau tanh yang memiliki turunan kecil dalam domain tertentu [28].

Selain arsitektur utamanya, LSTM memiliki berbagai *hyperparameter* yang

dapat dikonfigurasi secara fleksibel untuk mengoptimalkan kinerja [33]. Beberapa parameter penting tersebut meliputi:

$$\theta = W_f, W_i, W_c, W_o, b_f, b_i, b_c, b_o \tag{2.10}$$

Memilih kombinasi *hyperparameter* yang tepat sangat penting untuk performa akhir model, namun proses pencarian optimal ini sangat kompleks dan memerlukan teknik optimasi lanjutan agar tidak terjebak dalam pendekatan coba-coba [33].

Berikut adalah beberapa hyperparameter yang umumnya diuji dalam model LSTM:

1. **Jumlah *layer* LSTM**: Terdiri dari blok-blok memori yang saling terhubung secara berulang. Setiap blok memori terdiri dari satu sel memori yang mengandung tiga gerbang perkalian. Lapisan LSTM bertanggung jawab untuk menentukan informasi mana yang sebaiknya disimpan dalam status sel jaringan (c_t), yang melibatkan nilai kandidat memori dan aktivasi dari gerbang input (i_t). Menurut Dhake et al. (2023), konfigurasi jumlah layer yang optimal dapat dicapai melalui teknik optimasi berbasis genetika yang terbukti meningkatkan akurasi prediksi hingga 95% pada deret waktu [34].

2. **Jumlah unit LSTM**: Unit dalam arsitektur LSTM disebut sebagai sel. Setiap sel menerima kombinasi keadaan sebelumnya dan input saat ini sebagai masukan. Sel tersebut memainkan peran penting dalam memutuskan informasi yang disimpan dan dihapus dari memori. Studi oleh Albishi et al. (2024) menunjukkan bahwa peningkatan jumlah sel dapat memperbaiki akurasi hingga titik tertentu, tetapi berisiko menyebabkan *overfitting* bila jumlahnya terlalu besar [35].

3. **Jumlah *layer* Dense**: Lapisan ini memiliki koneksi penuh ke lapisan sebelumnya dan bertugas mengubah output menjadi prediksi. Penelitian terbaru oleh Nooraei et al. (2024) menunjukkan bahwa kedalaman Dense layer mempengaruhi konvergensi dan kompleksitas, terutama dalam kombinasi dengan LSTM pada sistem prediksi dinamis [36].

4. **Fungsi aktivasi**: Pemilihan fungsi aktivasi yang tepat sangat mempengaruhi stabilitas dan efisiensi pembelajaran. Fungsi aktivasi seperti sigmoid, tanh,

atau ReLU masing-masing memiliki dampak yang berbeda. Berdasarkan studi oleh Rasheed et al. (2022), fungsi tanh lebih unggul pada masalah deret waktu karena pusat nol-nya mendukung konvergensi lebih baik dibandingkan sigmoid [37].

5. **Optimizer**: Berbagai algoritma optimasi seperti Adam, RMSprop, dan SGD memiliki karakteristik tersendiri. Sebuah eksperimen oleh Adepu et al. (2023) menyimpulkan bahwa Adam menghasilkan pelatihan lebih stabil dan cepat konvergen dibandingkan SGD dalam model LSTM untuk sistem siber-fisik [38].

6. ***Learning rate***: Parameter ini menentukan kecepatan adaptasi model terhadap data. Penelitian oleh Filatov et al. (2024) menemukan bahwa penyesuaian *learning rate* sangat berperan dalam mencegah konvergensi prematur dan meningkatkan akurasi generalisasi, terutama saat digunakan bersamaan dengan pengaturan *batch size* dinamis [39].

7. **Batch size**: Merupakan jumlah sampel data yang diproses dalam satu iterasi. Studi oleh Schäfer et al. (2024) pada bidang hidrologi menunjukkan bahwa ukuran batch yang terlalu kecil menyebabkan pelatihan lambat, sedangkan yang terlalu besar menyebabkan pelatihan tidak stabil. Penggunaan batch size moderat terbukti menghasilkan keseimbangan terbaik [40]. Penelitian sebelumnya oleh You et al. (2019) juga mendukung scaling batch size secara agresif menggunakan teknik warm-up [41].

8. **Epoch**: Jumlah epoch menunjukkan berapa kali dataset dilalui selama pelatihan. Sebuah studi oleh Kusuma et al. (2025) menyimpulkan bahwa pemilihan jumlah epoch yang optimal berbeda-beda tergantung dataset, dan teknik seperti *grid search* atau *Bayesian optimization* sangat membantu dalam menentukan jumlah epoch yang ideal [42].

2.4 Metrik Evaluasi

Untuk melatih model prediksi berbasis *deep learning*, diperlukan metrik evaluasi yang tepat untuk mengukur kemampuan model dalam menghasilkan prediksi yang mendekati nilai aslinya. Evaluasi dilakukan untuk memastikan bahwa model memiliki performa yang baik selain pada saat menggunakan data latih. Dua metrik evaluasi utama yang umum digunakan dalam regresi adalah *Mean Absolute*

Error (MAE) dan *Mean Squared Error* (MSE) [43]. Kemudian untuk mendukung evaluasi hasil prediksi peringkat, digunakan metrik evaluasi *Spearman Correlation*, dan *Kendall Tau*.

2.4.1 Mean Absolute Error

Mean Absolute Error (MAE) merupakan salah satu metrik evaluasi yang paling umum digunakan dalam permasalahan regresi. MAE mengukur rata-rata dari selisih absolut antara nilai prediksi model dan nilai aktual dari target. Sifatnya yang intuitif dan mudah diinterpretasikan membuat MAE menjadi pilihan populer dalam berbagai studi dan aplikasi *machine learning*. MAE memberikan estimasi rata-rata kesalahan yang terjadi dalam satuan asli variabel target, tanpa memberi penalti yang lebih besar terhadap *outlier* seperti yang dilakukan oleh *Mean Squared Error* (MSE) [44].

MAE dirumuskan sebagai rata-rata dari nilai absolut perbedaan antara nilai observasi aktual (y_i) dan nilai prediksi ($\hat{y}_i$) untuk seluruh data sebanyak n

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.11}$$

MAE bersifat linear terhadap besarnya kesalahan, sehingga setiap perbedaan memiliki kontribusi yang setara terhadap nilai akhir. Hal ini menjadikan MAE sebagai metrik yang cukup *robust* terhadap *noise* atau data ekstrem [43]. Dalam praktiknya, MAE sering digunakan sebagai tolok ukur awal dalam mengevaluasi performa model regresi, sebelum mempertimbangkan metrik lainnya seperti MSE atau *Root Mean Squared Error* (RMSE) yang lebih sensitif terhadap *outlier*.

2.4.2 Mean Squared Error

Mean Squared Error (MSE) merupakan salah satu metrik evaluasi yang paling umum digunakan dalam permasalahan regresi. MSE mengukur rata-rata dari kuadrat selisih antara nilai prediksi dan nilai aktual. Karakteristik utama dari MSE adalah memberikan penalti yang lebih besar terhadap kesalahan prediksi yang besar, karena penggunaan kuadrat dari selisih. Metrik ini sangat berguna ketika kesalahan besar dianggap lebih serius daripada kesalahan kecil, seperti dalam konteks yang sensitif terhadap deviasi ekstrem [44]. Metrik MSE dirumuskan seperti Persamaan (2.12)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.12}$$

Berbeda dengan *Mean Absolute Error* (MAE), MSE lebih sensitif terhadap *outlier*, karena nilai kesalahan yang lebih besar akan dikuadratkan dan berkontribusi secara signifikan terhadap hasil akhir [45]. Dalam praktiknya, MSE sering digunakan dalam proses pelatihan model *machine learning* karena bersifat terdiferensiasi, sehingga dapat digunakan sebagai *loss function* dalam algoritma *gradient-based optimization*.

2.4.3 Korelasi Spearman

Korelasi Spearman atau dikenal sebagai *Spearman's rho*, merupakan metode statistik non-parametrik yang digunakan untuk mengukur kekuatan dan arah hubungan monotonik antara dua variabel berjenjang. Berbeda dengan korelasi Pearson yang mengukur hubungan linier, *Spearman's correlation* mengukur derajat konsistensi urutan (*ranking*) antara dua set data. Oleh karena itu, metrik ini sangat cocok digunakan ketika data bersifat ordinal atau tidak memenuhi asumsi distribusi normal. Korelasi Spearman efektif dalam mengevaluasi model regresi peringkat, termasuk dalam konteks sistem rekomendasi dan prediksi klasemen olahraga. Rumus untuk menghitung koefisien korelasi Spearman (ρ) ditunjukkan dengan Persamaan (2.13).

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)} \tag{2.13}$$

di mana d_i adalah selisih antara peringkat aktual dan peringkat prediksi untuk observasi ke-i, dan n adalah jumlah observasi. Nilai ρ berada pada rentang -1 hingga $+1$, dengan nilai mendekati +1 menunjukkan hubungan monotonik positif yang kuat, sedangkan nilai mendekati -1 menunjukkan hubungan monotonik negatif. Korelasi Spearman banyak digunakan dalam penelitian ilmu sosial, biomedis, dan juga dalam evaluasi model *machine learning* yang memprediksi urutan atau prioritas [46].

2.4.4 Korelasi Kendall Tau

Kendall's tau correlation coefficient merupakan metode statistik non-parametrik yang digunakan untuk mengukur kekuatan dan arah hubungan antara dua variabel berdasarkan peringkat (ranking). Tidak seperti *Pearson correlation* yang mengandalkan hubungan linier, atau *Spearman's rho* yang mengukur konsistensi urutan, *Kendall's tau* mengevaluasi keselarasan pasangan-pasangan peringkat secara langsung. Metrik ini menghitung proporsi pasangan yang *concordant* (selaras) dan *discordant* (tidak selaras). Menurut M. G. Kendall, pendekatan ini lebih stabil untuk dataset kecil dan lebih peka terhadap perbedaan lokal dalam ranking [47]. Koefisien *Kendall's tau* (τ) dapat dihitung dengan Persaman :

$$\tau = \frac{C - D}{\frac{1}{2}n(n-1)} \tag{2.14}$$

di mana C adalah jumlah pasangan *concordant*, D adalah jumlah pasangan *discordant*, dan n adalah jumlah total observasi. Nilai τ berada dalam rentang -1 hingga $+1$, dengan nilai mendekati $+1$ menunjukkan keselarasan urutan yang kuat antara dua variabel. *Kendall's tau* banyak digunakan dalam bidang *social science*, *bioinformatics*, dan *information retrieval*, khususnya saat mengevaluasi sistem peringkat dan prediksi klasemen [48].