BAB II
KAJIAN LITERATUR

2.1 *Artificial Intelligence*

Artificial Intelligence atau yang dikenal dengan kecerdasan buatan merupakan sistem komputer yang dikembangkan untuk meniru kemampuan intelektual manusia, seperti proses pengambilan keputusan, penalaran logis, serta berbagai aspek kecerdasan lainnya. AI termasuk dalam cabang ilmu komputer yang berorientasi pada perancangan dan pengembangan sistem yang mampu menjalankan tugas sebagaimana yang dilakukan manusia. Tujuan utama AI adalah menciptakan mesin yang dapat belajar, memahami informasi, merencanakan tindakan, serta beradaptasi dengan lingkungan sehingga mampu menyelesaikan berbagai pekerjaan secara mandiri [13].

Dalam AI terdapat tiga konsep dasar yang perlu diketahui [14]:

1. Pembelajaran Mesin (*Machine Learning*)
   Teknik ini memungkinkan mesin belajar dari data dan pengalaman tanpa diprogram kembali, sehingga mampu mengenali pola dan melakukan analisis secara mandiri.
2. Pembelajaran Mendalam (*Deep Learning*)
   Teknik ini mengajarkan mesin untuk melakukan tugas seperti manusia dengan cara belajar dari data, menggunakan model jaringan saraf tiruan yang memiliki banyak lapisan.
3. Jaringan Saraf Tiruan (*Artificial Neural Network)*
   Sistem komputasi yang meniru cara kerja sistem saraf biologis manusia dalam memproses informasi, dengan memanfaatkan kumpulan neuron yang saling terhubung untuk mengenali pola dan melakukan pembelajaran dari data.

Penerapan AI dalam Sistem Informasi mencakup berbagai aplikasi yang digunakan untuk meningkatkan efisiensi, analisis data, dan kualitas pengambilan keputusan. Berikut beberapa penerapannya yaitu [15]:

1. Analisis Data dan Prediksi: AI dimanfaatkan untuk mengolah data dalam jumlah besar serta memprediksi tren di masa depan. Penerapan ini membantu berbagai sektor, seperti perbankan dan bisnis, dalam mengenali peluang dan mengantisipasi risiko.
2. Sistem Rekomendasi: Seperti yang diterapkan pada Netflix dan Amazon, AI menganalisis perilaku serta preferensi pengguna untuk memberikan saran produk atau konten yang relevan.

3. *Natural Language Processing*: AI memungkinkan komputer memahami dan memproses bahasa manusia. Contohnya adalah chatbot, analisis sentimen, dan sistem penerjemahan otomatis.
4. Pengenalan Pola dan Citra: AI digunakan untuk mengenali wajah, menganalisis citra medis, serta mendukung sistem pengawasan keamanan berbasis video.
5. Penelitian Ilmiah: Dalam bidang riset, AI membantu mengolah data eksperimen, menjalankan simulasi kompleks, dan menemukan pola tersembunyi dalam data ilmiah.

2.1.1 *Machine Learning*

Machine Learning adalah metode dalam kecerdasan buatan (AI) yang dikembangkan untuk meniru cara manusia dalam berpikir dan mengambil keputusan guna menyelesaikan suatu permasalahan [16]. *Machine learning* sering diterapkan dalam tugas klasifikasi dan *clustering*, terutama pada pengolahan data dalam jumlah besar atau *big data* [17]. *Machine learning* terbagi menjadi empat jenis, yaitu [18]:

1. *Supervised Learning*

   Supervised learning merupakan pendekatan dalam *machine learning* yang menggunakan data berlabel, yaitu dataset yang telah diketahui kelas atau targetnya. Metode ini bertujuan agar model dapat mempelajari pola dari data tersebut sehingga mampu memprediksi atau mengklasifikasikan data baru secara tepat. Adapun jenis dari *supervised learning* yaitu regresi dan klasifikasi.
2. *Unsupervised Learning*

   Unsupervised learning mampu mengidentifikasi berbagai pola tersembunyi dalam data tanpa menggunakan label sebelumnya, serta membantu menemukan fitur yang relevan untuk proses pengelompokan. Proses *clustering* ini dapat dilakukan secara *real-time* sehingga data masukan dapat dianalisis dan dikelompokkan secara otomatis. Pada umumnya unsupervised learning digunakan untuk *clustering*, *association rule*, dan *dimensionality reduction*.
3. *Semi-supervised Learning*

   Semi-supervised learning merupakan pendekatan yang menggabungkan metode *supervised* dan *unsupervised* untuk membangun suatu fungsi atau model. Metode ini menggunakan kombinasi data berlabel dalam jumlah terbatas dan data tanpa label dalam jumlah lebih besar untuk meningkatkan kinerja pembelajaran. Berikut beberapa contoh penerapan *semi-supervised learning* yaitu *speech recognition*, *web content classification*, dan *the document classification*.

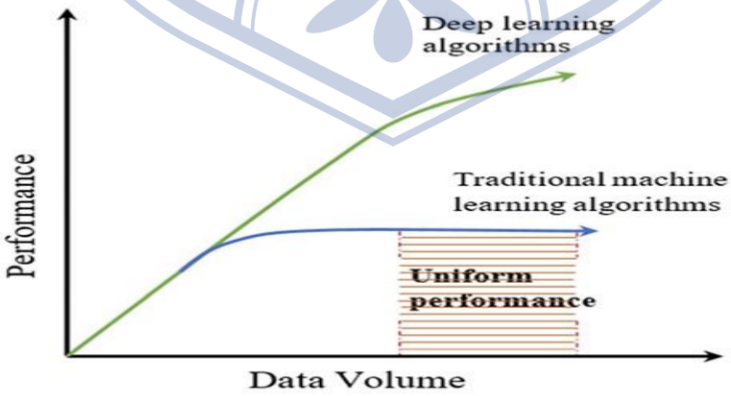
4. *Reinforcement Learning*

   Reinforcement learning merupakan jenis algoritma *machine learning* yang memungkinkan agen *software* atau sistem untuk belajar secara otomatis dalam menentukan tindakan terbaik guna memaksimalkan kinerjanya. Cara kerja nya adalah dengan melatih sistem untuk mengambil tindakan berdasarkan kondisi yang dihadapi. Karena tidak terdapat jawaban benar yang pasti, sistem belajar dari pengalaman sebelumnya guna meminimalkan kesalahan dan meningkatkan kinerja.

Ada beberapa algoritma yang ada di dalam *machine learning* yaitu *Random Forest*, *Support Vector Machine*, *K-Nearest Neighbors*, *Decision Trees*, *Naive Bayes* dan sebagainya. Cara kerja *machine learning* dimulai dengan proses pelatihan menggunakan data, di mana algoritma membentuk pola atau aturan tertentu. Setelah itu, kinerja model diuji menggunakan data pengujian untuk menghasilkan prediksi atau hasil secara otomatis tanpa intervensi manusia [19].

2.1.2 *Deep Learning*

Deep learning merupakan bagian dari *machine learning* yang menggunakan algoritma untuk mempelajari pola dari data sehingga dapat menyelesaikan berbagai tugas secara otomatis tanpa perlu diprogram secara eksplisit. Pendekatan *machine learning* tradisional umumnya kurang efektif dibandingkan *deep learning*, karena sangat bergantung pada jumlah data yang besar serta fitur yang dirancang secara manual. Proses pembuatan fitur tersebut membutuhkan keahlian manusia dan pemahaman mendalam terhadap domain yang digunakan. Berikut Gambar 2.1 yang menunjukkan perbandingan antara *deep learning* dan *machine learning* tradisional [20].



Gambar 2. 1 Perbandingan *Deep Learning* dan *Machine Learning* Tradisional [20]

Teknik *deep learning* memungkinkan model komputasi mempelajari representasi fitur melalui banyak lapisan pemrosesan dan tingkat abstraksi. ANN menjadi fondasi utama pengembangannya dan terbukti efektif di berbagai bidang. Namun, ANN memiliki keterbatasan seperti risiko *overfitting* dan tidak selalu menjamin tercapainya solusi optimal, sehingga dikembangkan arsitektur yang lebih dalam untuk mengatasinya. Istilah "*deep*" pada *deep learning* mengacu pada banyaknya lapisan dalam jaringan yang memproses data secara bertahap. Model ini terdiri dari *input layer*, beberapa *hidden layer*, dan *output layer*. Data masuk melalui *input layer*, diproses menggunakan bobot dan fungsi aktivasi pada *hidden layer* untuk menangkap hubungan yang kompleks, lalu menghasilkan prediksi pada *output layer* sesuai jumlah kelas yang diinginkan. Kesalahan prediksi dihitung dengan *backpropagation* dan digunakan untuk memperbarui bobot secara berulang hingga diperoleh hasil yang cukup akurat [20].

Teknik *deep learning* dikembangkan dan diterapkan pada banyak bidang, seperti visi komputer, analisis sentimen teks, pengenalan suara, hingga penerjemahan mesin [20] [21]. Algoritma *deep learning* seperti *Convolutional Neural Network* (CNN) digunakan dalam berbagai tugas visi computer, sedangkan *Recurrent Neural Network* (RNN) umumnya diterapkan pada pengolahan data sekuensial seperti pemrosesan bahasa alami. Selain itu, arsitektur transformer semakin populer untuk menangani tugas yang memanfaatkan mekanisme *attention,* seperti penerjemahan mesin dan generasi teks [21]. Pengembangan model tersebut melibatkan tahap prapemrosesan data, pemilihan fitur, penentuan parameter optimal, serta evaluasi akurasi dan kecepatan konvergensi [20]. Dengan tahapan tersebut, model *deep learning* diharapkan mampu menghasilkan kinerja yang optimal dalam berbagai tugas yang kompleks.

2.1.3 *Natural Language Processing*

Natural Language Processing (NLP) merupakan teknologi berbasis *machine learning* yang dirancang agar komputer mampu menginterpretasikan, mengolah, serta memahami bahasa yang digunakan oleh manusia [22]. Dalam penerapannya, NLP berfokus pada analisis sintaksis dan semantik untuk memahami makna bahasa serta menangani ambiguitas yang muncul dalam bahasa manusia. Oleh karena itu, NLP memanfaatkan metode komputasional untuk menganalisis dan mengolah bahasa manusia [23]. Berikut adalah beberapa teori dan pendekatan yang digunakan dalam *Natural Language Processing* [24]:

1. Pemrosesan Bahasa Alami Berbasis Aturan (*Rule-Based* NLP)

Metode ini menggunakan aturan linguistik atau tata bahasa yang telah ditetapkan sebelumnya untuk memahami dan mengolah teks, mencakup aspek sintaksis dan semantik. Teknik yang digunakan antara lain pengurai sintaksis (parser) dan generator bahasa alami.

2. Pembelajaran Mesin dalam Pemrosesan Bahasa Alami (*Machine Learning in* NLP)
   Pendekatan ini memanfaatkan algoritma pembelajaran mesin untuk mengenali pola dari data bahasa alami. Teknik ini digunakan dalam berbagai tugas seperti klasifikasi teks, ekstraksi informasi, pemodelan topik, dan penerjemahan mesin, dengan metode yang umum digunakan antara lain *Naive Bayes*, *Support Vector Machine* (SVM), Jaringan Saraf, dan Model Bahasa.
3. Representasi Bahasa Alami (*Natural Language Representation*)
   Representasi bahasa alami merupakan proses mengubah teks ke dalam bentuk yang dapat diproses oleh komputer. Representasi ini mencakup tingkat kata, frasa, dan dokumen, serta struktur grafis untuk analisis sintaksis dan semantik. Beberapa metode yang umum digunakan meliputi *word embedding*, model bahasa berbasis Transformer, dan grafik dependensi.

2.1.4 Analisis Sentimen

Analisis sentimen merupakan komponen penting dalam *natural language processing* (NLP) yang berfungsi untuk secara sistematis mengidentifikasi dan mengkategorikan sentimen dalam data teks sehingga memungkinkan penilaian terhadap nada emosional, pendapat, dan sikap yang disampaikan melalui bahasa tertulis maupun lisan [25]. Ada tiga jenis utama dalam analisis sentimen, yaitu [26]:

1. *Multimodal sentiment analysis*
   Multimodal sentiment analysis adalah jenis analisis sentimen yang memanfaatkan data teks, audio, dan visual secara bersamaan untuk mengidentifikasi emosi melalui isyarat seperti ekspresi wajah dan intonasi suara secara lebih akurat.
2. *Aspect-based sentiment analysis*
   Analisis berbasis aspek menggunakan teknik NLP untuk mengidentifikasi dan menilai emosi serta opini yang berkaitan dengan fitur atau aspek tertentu dari produk dan layanan, seperti menilai sentimen terhadap makanan, pelayanan, dan suasana dalam ulasan restoran.
3. *Multilingual sentiment analysis*

Setiap bahasa memiliki struktur tata bahasa, sintaksis, dan kosakata yang berbeda, sehingga cara mengekspresikan sentimen pun tidak sama. Oleh karena itu, dalam analisis sentimen multibahasa, tiap bahasa dilatih secara khusus untuk mengekstraksi emosi dari teks yang dianalisis.

Tujuan analisis sentimen adalah untuk menentukan apakah suatu kalimat mengandung sentimen positif, negatif, atau netral [27]. Informasi sentimen pada pariwisata berguna bagi wisatawan sebagai referensi dalam memilih destinasi, dan bagi pengelola wisata sebagai bahan evaluasi serta peningkatan kualitas pelayanan. Analisis sentimen dapat dikategorikan menjadi tiga kelas sentimen, yaitu [28]:

1. Sentimen Positif

   Sentimen positif merupakan sikap atau ungkapan yang menunjukkan dukungan, kesukaan, atau penilaian baik terhadap suatu hal. Sebagai contoh, ulasan yang memuji kualitas suatu produk termasuk dalam kategori sentimen positif. Hal ini umumnya berkaitan dengan pengalaman yang menyenangkan atau rasa puas.
2. Sentimen Negatif

   Sentimen negatif adalah sikap atau ungkapan yang menunjukkan penolakan, ketidaksukaan, atau kritik terhadap suatu hal. Contohnya dapat berupa ulasan yang mengungkapkan ketidakpuasan terhadap pelayanan atau mutu produk. Sentimen ini biasanya berkaitan dengan pengalaman yang tidak menyenangkan atau rasa kecewa.
3. Sentimen Netral

   Sentimen netral merupakan kategori yang diberikan pada teks yang tidak menunjukkan kecenderungan emosional yang jelas ke arah positif maupun negatif, atau ketika terjadi ambiguitas pelabelan sehingga diperlukan satu label penengah.

2.1.5 *Large Language Model* (LLM)

Large Language Model (LLM) merupakan model kecerdasan buatan yang berbasis *Natural Language Processing* (NLP), yang mampu memahami, menafsirkan, serta menghasilkan teks dengan kualitas yang menyerupai kemampuan pemahaman manusia [29]. LLM biasanya digunakan dalam pembuatan teks, analisis teks, penerjemahan bahasa, analisis sentimen, menjawab pertanyaan, serta berbagai fungsi terkait lainnya [30]. Terdapat model LLM yang telah dikembangkan, diantaranya GPT-4.1, GPT-4o mini, GPT-3 mini, Claude 3.7 Sonnet, DeepSeek R1, DeepSeek V3, Gemini 2.0 Flash, Gemini 2.5 Pro, Llama 3.1, Llama 4, Perplexity, Grok 2, Grok 3, Mistral, serta Sabiá 3 [31].

Large Language Model (LLM) dapat dimanfaatkan untuk mendukung proses pelabelan data secara otomatis [32]. LLM mampu memproses data teks dalam jumlah besar dengan efisien serta memahami makna semantik dari suatu teks. Selain itu, LLM juga dapat mengidentifikasi kesamaan antar dokumen sehingga mampu menghasilkan label topik yang relevan dan bermakna [33]. Kemampuan LLM dalam memahami bahasa alami dan menjalankan instruksi dari pengguna memungkinkan model ini digunakan dalam berbagai tugas berbasis teks. Instruksi yang diberikan kepada model disebut sebagai *prompt*, sedangkan proses merancang dan menyempurnakan *prompt* agar menghasilkan *output* yang optimal dikenal dengan istilah *prompt engineering* [34].

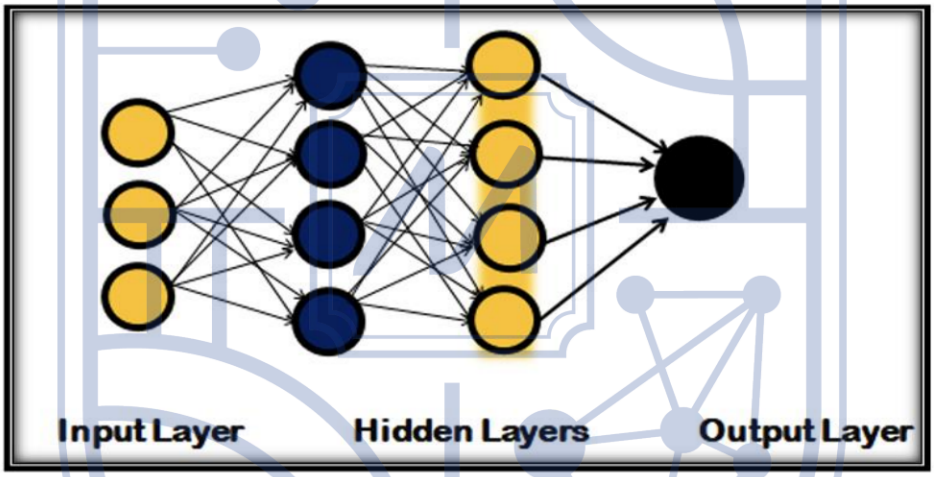
Prompt engineering adalah kemampuan dalam merancang serta mengoptimalkan instruksi yang diberikan kepada LLM agar model mampu menghasilkan *output* yang lebih baik dan sesuai kebutuhan pengguna. Teknik ini membantu pengguna memanfaatkan kemampuan *generative* AI secara lebih efektif untuk menyelesaikan berbagai permasalahan [35]. Dalam *prompt engineering*, terdapat beberapa teknik yang umum digunakan, seperti *few-shot prompting*, *chain-of-thought prompting*, dan *active prompting*. Teknik *few-shot prompting* memanfaatkan sejumlah kecil contoh sebagai panduan agar model dapat memahami cara menyelesaikan suatu tugas berdasarkan konteks yang diberikan. Untuk meningkatkan kemampuan penalaran model pada tugas yang lebih kompleks, digunakan *chain-of-thought prompting* yang memberikan contoh langkah-langkah berpikir dalam menyelesaikan masalah. Sementara itu, *active prompting* dikembangkan untuk mengatasi keterbatasan contoh yang diberikan di awal dengan melibatkan manusia dalam proses evaluasi dan anotasi ulang terhadap jawaban model yang dianggap paling tidak pasti, sehingga proses penalaran model dapat terus diperbaiki secara bertahap [36].

Dalam penelitian ini, digunakan salah satu *Large Language Model* (LLM), yaitu Gemini 2.5 Flash, untuk melakukan pelabelan aspek dan sentimen pada dataset. Pemilihan Gemini 2.5 Flash didasarkan pada hasil penelitian [37] yang menunjukkan bahwa model ini memberikan kinerja terbaik dibandingkan GPT-4.1-mini pada berbagai konfigurasi *prompting*, dengan menghasilkan anotasi yang lebih akurat, lebih stabil, dan lebih konsisten, terutama pada tugas klasifikasi serta ekstraksi informasi berbasis teks. Proses pelabelan dilakukan dengan memanfaatkan API Gemini 2.5 Flash, dengan menyusun dan mengirimkan *prompt* secara sistematis agar model dapat menghasilkan label yang sesuai dengan konteks data. *Prompt* dirancang secara terstruktur dengan memberikan instruksi yang jelas terkait klasifikasi aspek dan sentimen. Penggunaan model ini bertujuan untuk

meningkatkan efisiensi serta menghasilkan pelabelan yang lebih konsisten, dengan *output* berupa label aspek dan kategori sentimen dari setiap data teks.

2.2 *Artificial Neural Networks*

Artificial Neural Networks merupakan sistem pengolahan informasi yang dirancang dengan karakteristik kerja yang memiliki kemiripan dengan jaringan saraf biologis [38]. ANN banyak digunakan untuk menyelesaikan berbagai permasalahan prediksi, seperti tugas klasifikasi maupun regresi. Cara kerja *Artificial Neural Networks* yaitu data pertama-tama dimasukkan melalui *input layer*, kemudian diolah pada *hidden layer*, dan selanjutnya diteruskan ke dalam *output layer* [39]. Berikut Gambar 2.2 yang menunjukkan arsitektur *Artificial Neural Networks.*



Gambar 2. 2 Arsitektur *Artificial Neural Networks* [40]

Dalam arsitektur *Artificial Neural Networks* terdapat tiga lapisan utama yaitu [40]:

1. *Input Layer*

   Input layer merupakan bagian awal dari jaringan saraf yang terdiri atas sejumlah neuron *input* buatan. Neuron-neuron pada lapisan ini berfungsi menerima data atau informasi dari luar sistem untuk kemudian diteruskan ke lapisan berikutnya. Dengan demikian, *input layer* menjadi titik awal dalam proses pengolahan data pada jaringan saraf.

2. *Hidden Layer*

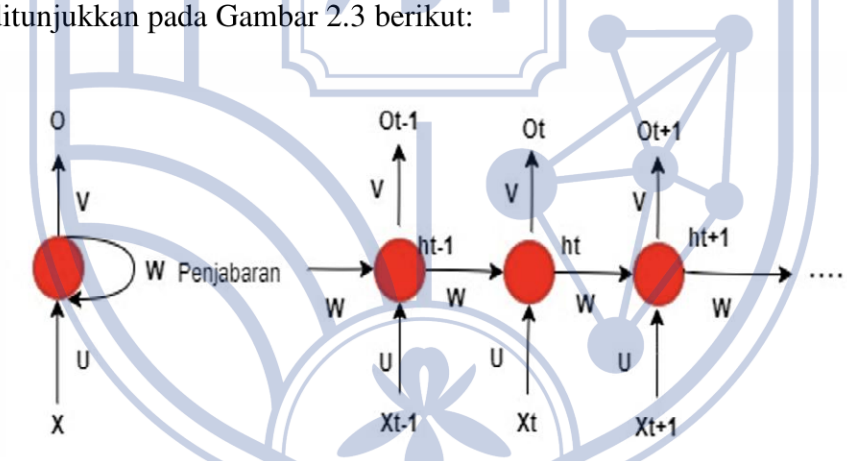
   Hidden layer terletak di antara *input layer* dan *output layer*. Pada lapisan ini, setiap neuron menerima *input* dari lapisan sebelumnya, mengalikannya dengan bobot tertentu, lalu memproses hasilnya untuk diteruskan ke lapisan berikutnya.

3. *Output Layer*

Output layer merupakan lapisan terakhir dalam struktur *artificial neural network* yang bertugas menghasilkan *output* akhir sesuai dengan tujuan perancangan sistem. Sebagai simpul akhir dalam jaringan, neuron pada *output layer* dapat dirancang dan diatur secara khusus sesuai dengan kebutuhan, misalnya untuk menghasilkan nilai klasifikasi atau prediksi tertentu.

2.2.1 *Recurrent Neural Networks*

Recurrent Neural Networks adalah teknik yang digunakan untuk mengolah data berbentuk runtutan atau deret waktu dan banyak diterapkan dalam bidang *neural network* [41]. Model *Recurrent Neural Network* (RNN) memiliki karakteristik khusus karena mampu mempertahankan informasi di dalam arsitektur jaringannya melalui adanya minimal satu mekanisme *feedback loop* [42]. *Recurrent Neural Networks* memiliki tiga kompenen utama, yaitu: lapisan *input* yang mengubah setiap kata menjadi representasi vektor, lapisan tersembunyi yang bekerja secara berulang untuk memperbarui *hidden state* selama pemrosesan urutan kata, serta lapisan *output* yang bertugas memperkirakan probabilitas kata selanjutnya berdasarkan *hidden state* yang sedang aktif [43]. Arsitektur *Recurrent Neural Networks* ditunjukkan pada Gambar 2.3 berikut:



Gambar 2. 3 Arsitektur *Recurrent Neural Networks* [43]

Perhitungan pada *hidden state* dinyatakan melalui persamaan berikut [44]:

$$h_t = F(W_{hh}h_t - 1 + W_{xh}\mathrm{x_t} + b_h) \quad \text{.....(1)}$$

Dimana:

h_t = *output* pada waktu ke-t

$\mathrm{x_t}$ = *input* pada waktu ke-t

F = fungsi aktivasi

$h_t - 1$ = *hidden state* pada waktu sebelumnya.

h_t = *hidden state* pada waktu ke-t

$W_{hh}, W_x h$ = matriks bobot

b_h = bias

Sedangkan *output* jaringan pada waktu ke-t dirumuskan sesuai dengan persamaan berikut:

$$y_t = \sigma\left(W_{hy} h_t + b_y\right) \quad \text{.....(2)}$$

Dimana:

y_t = *output* pada waktu ke-t

σ = fungsi aktivasi *sigmoid*

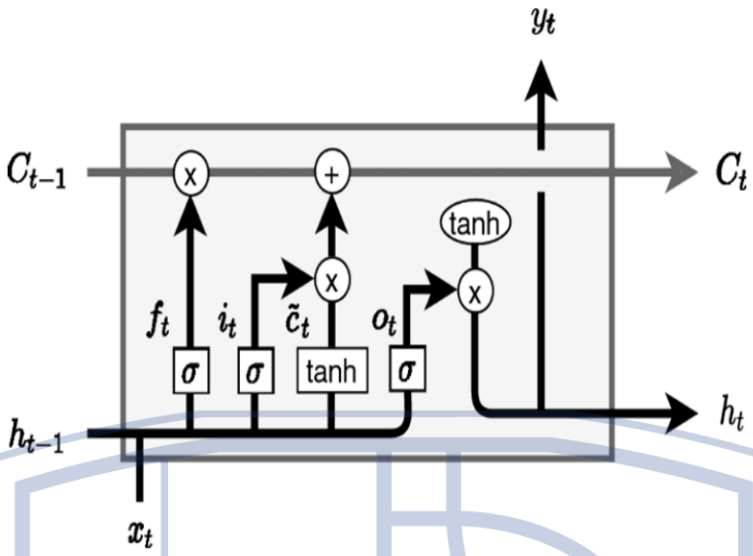
b_y = bias

W_{hy} = matriks bobot

Pada model jaringan RNN, *hidden state* pada waktu ke-t (h_t) berperan sebagai penyimpan informasi dari langkah sebelumnya sehingga model mampu memahami keterkaitan dalam suatu urutan data. Nilai *hidden state* ini diperbarui berdasarkan *input* saat ini (x_t) dengan menerapkan fungsi aktivasi nonlinier F, seperti *tanh* atau *ReLU*, yang memungkinkan jaringan menangkap pola yang lebih kompleks. Sementara itu, *output* pada waktu ke-t (y_t) diperoleh melalui fungsi aktivasi σ, misalnya *softmax* atau *sigmoid*, yang menyesuaikan bentuk keluaran sesuai dengan kebutuhan tugas pemodelan. Seluruh proses komputasi ini dipengaruhi oleh matriks bobot W_{hh}, W_{xh}, dan W_{hy} yang menghubungkan antar lapisan, serta parameter bias b_h dan $b_y$yang memberikan fleksibilitas tambahan dalam proses pembelajaran jaringan.

2.2.2 *Long Short-Term Memory* (LSTM)

Long Short-Term Memory (LSTM) merupakan salah satu model jaringan saraf yang dikembangkan untuk mempelajari dan mempertahankan hubungan konteks dalam data teks yang bersifat sekuensial [45]. LSTM dapat dipandang sebagai pengembangan dari RNN yang dilengkapi dengan mekanisme memori serta dua keadaan utama, yaitu keadaan jangka pendek untuk menyimpan keluaran sebelumnya dan keadaan jangka panjang untuk mempertahankan informasi yang relevan dalam jangka waktu lebih lama [46].



Gambar 2. 4 Arsitektur *Long Short-Term Memory* [47]

Pada Gambar 2.4 terdapat arsitektur dari LSTM yang terdiri dari tiga gerbang utama, yaitu *forget gate* (f_t), *input gate* (i_t), dan *output gate* (o_t). Pada arsitektur LSTM, tahap awal dimulai dengan menentukan informasi mana yang perlu dilupakan dari *cell state*. Proses ini dikendalikan oleh lapisan *sigmoid* yang dikenal sebagai *forget gate*. Pada tahap ini, sel LSTM menerima dua input utama, yaitu $ht{-}1$ (*hidden state* sebelumnya) dan xt (*input*). Selanjutnya, pada tahap kedua, model menentukan informasi apa yang akan disimpan dalam *cell state*. Tahapan ini terdiri dari dua bagian, yaitu lapisan *sigmoid* yang berfungsi untuk memilih nilai yang akan diperbarui, serta lapisan *tanh* yang menghasilkan vektor kandidat baru. Kedua hasil tersebut kemudian digabungkan untuk memperbarui *cell state*. *Cell state* berperan sebagai jalur utama dalam membawa informasi dari satu waktu ke waktu berikutnya dalam jaringan LSTM. Pada setiap *timestep*, nilai *cell state* diperbarui dengan memanfaatkan *forget gate* dan *input gate*, sehingga dapat mengatur informasi mana yang perlu dihapus maupun ditambahkan. Sementara itu, *output gate* digunakan untuk menentukan keluaran dari *cell state* saat ini. Prosesnya diawali dengan lapisan *sigmoid* yang memilih bagian dari *cell state* yang akan dijadikan output, kemudian lapisan tanh mengubah nilainya ke dalam rentang -1 hingga 1. Hasil dari kedua lapisan tersebut selanjutnya dikalikan untuk menghasilkan *output* akhir. Berikut persamaan-persamaan yang digunakan pada LSTM [47]:

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \quad (3)$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (4)$$

$$\tilde{C}_t = tan\, h(W_c x_t + U_c h_{t-1} + b_c) \quad (5)$$

$$C_t = i_t \circ \tilde{C}_t + f_t \circ C_{t-1} \quad (6)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (7)$$

$$h_t = o_t \circ tan\, h(C_t) \quad (8)$$

Dimana:

f_t = *forget gate*

i_t = *input gate*

$\tilde{C}_t$ = *cell state candidate*

C_t = *cell state*

o_t = *output gate*

h_t = *hidden state*

W = *matrik bobot*

h_{t-1} = *hidden state* sebelumnya

x_t = data *input*

b = bias

σ = fungsi aktivasi *sigmoid*

$tanh$: fungsi aktivasi *tanh*

Sel LSTM memanfaatkan dua jenis fungsi aktivasi, yaitu *sigmoid* dan *tanh*. Fungsi *sigmoid* digunakan untuk mentransformasikan nilai *input* ke dalam rentang [0, 1], sedangkan fungsi *tanh* mengubah nilai *input* ke dalam rentang [-1, 1]. Berikut adalah persamaan fungsi aktivasi *sigmoid* dan fungsi *tanh*.

$$\sigma(x) = \frac{1}{1+e^{-x}}, -\infty < x < \quad (9)$$

$$tanh(x) = \frac{2}{1+e^{-2x}} - 1, -\infty < x < \quad (10)$$

Loss function adalah metode yang digunakan untuk mengukur seberapa baik suatu algoritma dalam memodelkan data. Fungsi ini digambarkan dalam bentuk kurva yang memberikan arahan dalam menyesuaikan parameter agar model menjadi lebih akurat. Salah satu jenis *loss function* yang umum digunakan pada permasalahan klasifikasi adalah *Binary Cross Entropy* (BCE), yang menghasilkan *output* berupa nilai probabilitas dalam rentang 0 hingga 1 [48].

$$BCE = -[y.Log(p) - (1-y).Log\,(1-(p)) \quad (11)$$

Dimana:

y = nilai target

p = nilai prediksi

Namun pada kasus dataset yang memiliki distribusi kelas tidak seimbang (*imbalanced dataset*), penggunaan *Binary Cross Entropy* (BCE) standar dapat menyebabkan model lebih cenderung mempelajari kelas mayoritas dibandingkan kelas minoritas. Untuk mengatasi permasalahan tersebut, digunakan *Weighted Binary Cross Entropy* (*Weighted* BCE), yaitu pengembangan dari BCE dengan menambahkan bobot pada masing-masing kelas. Pemberian bobot ini bertujuan untuk meningkatkan kontribusi kelas minoritas dalam proses perhitungan *loss* sehingga model dapat memberikan perhatian yang lebih besar terhadap kelas yang jumlah datanya lebih sedikit. Dengan demikian, *Weighted* BCE dapat membantu meningkatkan kemampuan model dalam mengklasifikasikan kedua kelas secara lebih seimbang [48].

$$Weighted\ BCE = -[W_{Pos}\ .y.Log(p) + W_{Neg}.(1-y).Log\ (1-(p))\ldots\ldots\ldots(12)$$

Dimana:

W_{Pos} = bobot kelas positif (kelas minoritas)

W_{Neg} = bobot kelas negatif (kelas mayoritas)

LSTM memiliki beberapa *hyperparameter* sebagai berikut [49][50]:

1. *Batch Size* yaitu jumlah sampel data dalam satu kali iterasi pelatihan model.
2. *Epoch* yaitu jumlah pengulangan proses pelatihan terhadap seluruh data latih hingga model mencapai performa yang diharapkan.
3. *Dropout* yaitu teknik regularisasi yang digunakan untuk mengurangi risiko *overfitting* dengan menonaktifkan sebagian neuron acak selama pelatihan.
4. *Unit* yaitu menunjukkan jumlah neuron pada *hidden layer* yang berperan dalam mempelajari pola pada data.
5. *Optimizer* digunakan untuk memperbarui bobot model guna meminimalkan nilai *loss* selama proses pelatihan.
6. *Learning Rate* digunakan untuk mengatur besar langkah pembaruan bobot dan bias selama proses pelatihan model agar menghasilkan performa model yang optimal

2.3 *Word Embedding*

Word Embedding merupakan metode representasi kata dalam bentuk vektor numerik, di mana setiap kata dalam suatu kosakata dipetakan ke dalam representasi matematis yang memungkinkan pemrosesan teks secara komputasional [51]. *Word embedding* juga dikenal sebagai teknik dalam *natural language processing* (NLP) yang bertujuan merepresentasikan dan menangkap makna semantik suatu kata. Teknik ini menghasilkan representasi numerik

yang menggambarkan suatu kata berdasarkan konteks kemunculannya. Setiap kata dipetakan menjadi vektor padat berdimensi tertentu untuk mengukur kemiripan antar kata. Karena efisien, *word embedding* banyak digunakan dalam berbagai tugas NLP seperti klasifikasi teks, pengelompokan dokumen, penandaan kelas kata, pengenalan entitas bernama, dan analisis sentimen [52].

2.3.1 *Bag of Words*

Model *Bag of Words* (BoW) merupakan salah satu metode embedding paling dasar yang digunakan untuk mengubah teks menjadi data numerik. Model *Bag of Words* (BoW) berlandaskan pada asumsi bahwa setiap kata bersifat saling terpisah dan tidak memiliki ketergantungan dengan kata lainnya.. Dalam konteks analisis teks, asumsi ini menunjukkan bahwa satu kata tidak dipengaruhi oleh kata yang ada di sekitarnya. Walaupun didasarkan pada asumsi independensi kata, pendekatan BoW tetap efektif karena mampu menghitung frekuensi kemunculan setiap kata dalam tiap dokumen dan menyusunnya dalam bentuk *document-term matrix*. Pada matriks tersebut, baris merepresentasikan dokumen, sedangkan kolom menunjukkan seluruh kata unik dalam kosakata V (*vocabulary*) dari suatu korpus. Nilai pada setiap sel menunjukkan seberapa sering kata tertentu muncul dalam dokumen tertentu . Bentuk representasi ini sangat praktis untuk keperluan klasifikasi, karena setiap kata unik dapat dijadikan sebagai fitur atau variabel prediktor dalam proses analisis [53].

Model *Continuous Bag of Words* (CBOW) yang dikembangkan oleh Mikolov dalam penelitian *Word2Vec* merupakan pengembangan dari metode *bag of words* tradisional. Model ini bekerja dengan memprediksi kata pusat (wt) berdasarkan kata-kata konteks yang berada dalam rentang jendela tertentu di sekelilingnya. Tidak seperti pendekatan yang mempertimbangkan keseluruhan konteks, CBOW hanya menggunakan kata-kata terdekat sebelum dan sesudah kata target. Model ini berlandaskan pada asumsi bahwa arti sebuah kata dapat dipahami melalui hubungan dengan kata-kata di sekitarnya, sehingga mampu merepresentasikan makna semantik secara lebih efektif [54].

Secara matematis, diberikan konteks kata $w_{t-C}, \dots, w_{t+C}$, di mana Cadalah ukuran jendela konteks dan $w_t$adalah kata target, maka probabilitas untuk memprediksi w_t dihitung berdasarkan kata-kata konteks yang didefinisikan sebagai berikut [54]:

$$P(w_t|w_{t-c}, \dots, w_{t+c}) = \prod_{\substack{-C \le j \le C \\ j \neq 0}} P(w_t|w_{t+j}) \quad \text{.....(13)}$$

Dimana:

P($w_t \mid w_{t+j}$) = menunjukkan peluang munculnya kata target w_t jika diketahui kata konteks w_{t+j}.

C = ukuran jendela konteks.

w_t = kata target yang akan diprediksi.

w_{t+j} = kata-kata yang berada di sekitar w_t sebagai konteks.

Model ini bertujuan meningkatkan peluang prediksi kata target w_t dengan mempertimbangkan kata-kata konteksnya. Proses ini dilakukan dengan memperbarui representasi vektor kata (*embedding*) melalui minimisasi fungsi *loss*, yaitu nilai negatif dari *cross-entropy*.

$$L = -\frac{1}{T}\sum_{t=1}^{T} \log P(w_t \mid w_{t-C}, \ldots, w_{t+C}) \quad (14)$$

Dimana:

L = fungsi *loss* yang dioptimalkan (diperkecil nilainya) selama training.

T = jumlah kata yang terdapat dalam korpus.

$\log P(w_t \mid w_{t-C}, \ldots, w_{t+C})$ = nilai logaritma dari probabilitas bersyarat kemunculan kata target w_t berdasarkan kata-kata konteksnya.

2.3.2 *Skip-Gram*

Skip-gram adalah metode dalam pembentukan *word embedding* yang berfokus pada proses memprediksi kata-kata di sekitar suatu kata tertentu. Model tersebut sangat efektif dalam mengidentifikasi relasi semantik yang rumit dan jarang ditemui, yang umumnya terdapat pada teks-teks teknis seperti dokumen hukum [54].

Model Skip-gram bertujuan meningkatkan peluang dalam memprediksi kata-kata konteks $w_{t-c}, \ldots, w_{t+c}$, yang berada di sekitar suatu kata target w_t. Probabilitas kondisional tersebut didefinisikan sebagai berikut [54]:

$$P(w_{t-C}, \ldots, |w_t) = \prod_{\substack{-C \le j \le C \\ j \neq 0}} P(w_{t+j}|w_t) \quad (15)$$

Dimana:

P($w_{t+j} \mid w_t$) = menyatakan peluang munculnya kata konteks w_{t+j} dengan diketahui kata target w_t.

C = ukuran jendela konteks (*context window*).

w_t = kata target yang digunakan untuk memprediksi kata-kata dalam konteks.

w_{t+j} = kata-kata yang berada di dalam jendela konteks.

Model ini membentuk representasi vektor kata sehingga kata-kata dengan konteks yang mirip akan memiliki posisi yang dekat dalam ruang representasi. Pendekatan ini efektif dalam memahami makna kata, termasuk kata yang jarang digunakan. Proses pelatihan *Skip-gram* dilakukan dengan meminimalkan fungsi *loss negative cross-entropy*, yang mengukur perbedaan antara probabilitas yang diprediksi oleh model dan probabilitas aktual yang diamati dalam korpus [54]:

$$L = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-C \le j \le C \\ j \neq 0}} log\ P\left(w_{t+j}\middle|w_t\right) \quad \text{.....(16)}$$

Dimana:

L = fungsi *loss* yang diminimalkan selama proses pelatihan model

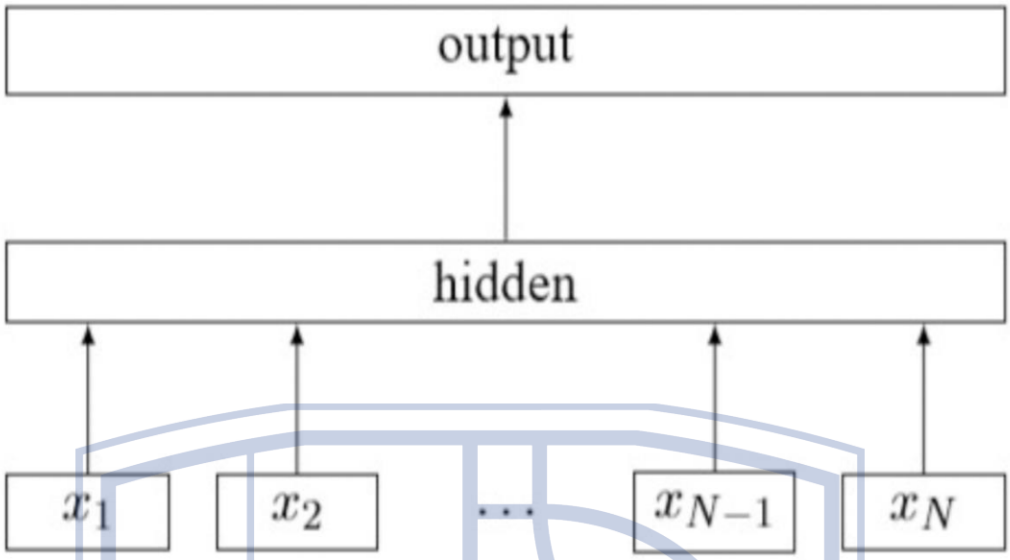
T = jumlah kata yang terdapat dalam korpus

log P(w_{t+j} | w_t) = nilai logaritma dari peluang memprediksi kata konteks w_{t+j} berdasarkan kata target w_t.

2.3.3 *FastText*

FastText merupakan salah satu metode *word embedding* yang dikembangkan oleh tim *Facebook AI Research* (FAIR) untuk merepresentasikan kata dalam bentuk vektor numerik [55]. Metode ini mengatasi berbagai keterbatasan dengan memanfaatkan *sub-string* karakter dalam kata sehingga efektif memproses teks pendek maupun panjang, serta lebih mampu mengenali kata langka, kata-kata yang tidak terdaftar dalam *vocabulary* (*out-of-vocabulary*), variasi ejaan, dan bentuk morfologis untuk menghasilkan pemahaman teks yang lebih mendalam [11].

FastText memiliki dua model yaitu model *CBOW* dan *Skip-gram*. Model CBOW merupakan model yang bertujuan memprediksi kata target berdasarkan konteks di sekitarnya, sementara skip-gram bertujuan untuk memprediksi kata-kata konteks di sekitar suatu kata tertentu [56].



Gambar 2. 5 Arsitektur *FastText* Linear Sederhana [56]

Pada Gambar 2.5 menampilkan arsitektur model linear sederhana pada *FastText* yang memanfaatkan fitur *N-gram*. Fitur – fitur tersebut kemudian diubah menjadi *embedding* dan dirata-ratakan untuk menghasilkan variabel tersembunyi. Arsitektur ini adalah *neural network* sederhana yang hanya terdiri dari satu lapisan utama. Representasi *bag-of-words* yaitu [56][57]:

1. Pertama-tama teks dimasukkan ke dalam *lookup layer* untuk mengambil nilai *embedding* dari setiap kata yang terdapat dalam teks.
2. Embedding kata-kata tersebut kemudian dirata-ratakan sehingga menghasilkan satu vektor *embedding* yang mewakili keseluruhan teks.

   $$v(w) = \frac{1}{|G_w|}\sum_{g \in G_w} z_g \quad \text{.....(17)}$$

   Dimana:

   $v(w)$ = vektor kata

   z_g = representasi vektor dari *n-gram*

   $|G_w|$ = himpunan *n-gram*
3. Pada *hidden layer*, akan didapatkan *n* kata *x* jumlah parameter, di mana menyatakan ukuran embedding dan *n* menunjukkan jumlah kata dalam kosakata.
4. Setelah proses rata-rata dilakukan, diperoleh satu vektor yang kemudian dimasukkan ke pengklasifikasi linear. Pada tahap ini diterapkan fungsi *softmax* terhadap hasil transformasi linear dari keluaran *inner layer*.

   Berikut adalah rumus fungsi log negatif yang digunakan dalam model *FastText* [56]:

   $$-\frac{1}{N} = \sum_{n=1}^{N} y_n log\left(f(BAx_n)\right) \quad \text{.....(18)}$$

Dimana:

x_n = merepresentasikan fitur *n-gram* dari kata

A = merepresentasikan matriks lookup dari word *embedding*

B = merepresentasikan transformasi keluaran linear model

f = merepresentasikan fungsi *softmax*

Fungsi *softmax* menghitung distribusi probabilitas suatu kejadian ke dalam n peristiwa yang berbeda. *Softmax* mengambil sekumpulan nilai kelas dan mengubahnya menjadi probabilitas dengan total bernilai 1, sehingga memetakan vektor berdimensi k dengan nilai rill sembarang menjadi vektor berdimensi k dengan nilai antara 0 hingga 1 [56].

Berikut adalah rumus fungsi *softmax* pada *FastText* [56]:

$$softmax(z) = \frac{exp(z)}{\sum_{k=1}^{K} exp(z)} \quad \text{.....(19)}$$

Dimana:

z = nilai skor logit untuk suatu kelas.

$\exp(z)$ = fungsi eksponensial dari z, digunakan untuk mengubah skor menjadi nilai positif.

$\sum_{k=1}^{K} exp(z)$ = menjumlahkan semua nilai eksponensial dari seluruh kelas.

FastText memiliki beberapa *hyperparameter* sebagai berikut [58]:

1. *Vector size* : jumlah dimensi pada vektor *embedding* yang digunakan dalam proses vektorisasi.
2. *Window* : jumlah kata di sekitar kata target yang dijadikan konteks saat pelatihan.
3. *Min Count* : frekuensi kemunculan minimum sebuah kata agar disertakan.
4. *Epoch* : jumlah perulangan yang dilakukan selama proses pembentukan vektor kata.

2.4 Pariwisata

Pariwisata dapat diartikan sebagai aktivitas perjalanan dari suatu tempat ke tempat lain yang pada akhirnya kembali ke titik awal, sehingga membentuk suatu siklus perjalanan [59]. Aktivitas pariwisata mencakup berbagai kegiatan perjalanan yang dilakukan seseorang atau kelompok ke suatu tempat di luar lingkungan sehari-hari untuk tujuan rekreasi, pendidikan, bisnis, maupun tujuan lainnya dalam jangka waktu tertentu. Terdapat berbagai jenis pariwisata yang telah dikenal secara umum yaitu [60]:

1. Wisata Budaya

   Wisata budaya merupakan kegiatan perjalanan yang dilakukan dengan tujuan memperluas wawasan dan pengalaman hidup, melalui kunjungan ke daerah lain atau ke luar negeri untuk mempelajari kehidupan masyarakat, kebiasaan, adat istiadat, budaya, serta seni yang mereka miliki.

2. Wisata Kesehatan

   Wisata kesehatan adalah perjalanan yang dilakukan seseorang dengan tujuan mengubah suasana dan lingkungan dari tempat tinggal sehari-hari demi memperoleh ketenangan serta pemulihan kondisi fisik dan mental.

3. Wisata Olahraga

   Wisata olahraga merupakan kegiatan perjalanan yang dilakukan dengan tujuan berolahraga atau secara khusus untuk berpartisipasi dalam acara atau kompetisi olahraga di suatu daerah atau negara.

4. Wisata Komersial

   Wisata komersial mencakup perjalanan yang bertujuan mengunjungi kegiatan berskala ekonomi, seperti pameran industri, pameran dagang, dan berbagai event komersial lainnya.

5. Wisata Industri

   Wisata industri adalah kegiatan perjalanan yang dilakukan oleh pelajar, mahasiswa, atau masyarakat umum ke kawasan industri dengan tujuan melakukan pengamatan, kunjungan, atau studi untuk menambah pengetahuan dan wawasan.

6. Wisata Bahari

   Wisata bahari merupakan kegiatan wisata yang berhubungan dengan perairan, seperti danau, pantai, dan laut.

7. Wisata Cagar Alam

   Wisata cagar alam adalah jenis wisata yang umumnya diselenggarakan oleh agen perjalanan khusus untuk mengunjungi kawasan konservasi, seperti cagar alam, taman lindung, hutan, dan daerah pegunungan yang dilindungi oleh peraturan perundang-undangan.

8. Wisata Bulan Madu

   Wisata bulan madu merupakan perjalan yang dirancang khusus bagi pasangan pengantin baru, dengan layanan dan fasilitas tertentu guna memberikan pengalaman yang nyaman dan berkesan.

Pariwisata berperan sebagai penggerak utama pembangunan ekonomi yang inklusif dengan membuka lapangan kerja, meningkatkan pendapatan masyarakat lokal, mendorong pembangunan infrastruktur, serta menjadi sumber devisa yang berkontribusi signifikan terhadap pertumbuhan ekonomi nasional [61]. Melalui pengelolaan yang optimal, pariwisata ini berpotensi terus tumbuh dan menjadi salah satu pilar utama perekonomian Indonesia di masa mendatang [62].

2.5 *Preprocessing*

Preprocessing adalah tahapan untuk membersihkan dan menyusun data mentah agar menjadi lebih terstruktur serta siap digunakan pada proses selanjutnya. Data yang diproses dapat berupa gambar, audio, video, maupun teks dengan berbagai bentuk dan karakteristiknya. Pada data teks, *preprocessing* diperlukan untuk menangani berbagai karakter dan elemen yang tidak relavan sehingga dapat meningkatkan kualitas dan akurasi model yang digunakan [63].

Preprocessing pada tahap ini dibagi menjadi dua yaitu *preprocessing* pra-pelabelan dan *preprocessing* setelah pra-pelabelan. Untuk *preprocessing* pra-pelabelan menggunakan fungsi *regular expression* (regex) dari python yaitu pola yang tersusun dari rangkaian karakter yang digunakan untuk mendeteksi kata maupun teks sesuai pola yang diinginkan [64]. Langkah-langkah dalam fungsi ini yaitu menghapus URL, tag HTML, *mentions*, karakter non-alfanumerik, dan spasi berlebih [65]. Sementara itu, *preprocessing* setelah pra-pelabelan meliputi *remove none* dan *duplicate* untuk menghapus nilai yang kosong dan data yang duplikat, *remove number* untuk menghapus angka pada teks, *case folding* dengan mengubah seluruh teks menjadi huruf kecil, *normalization* guna menyeragamkan bentuk teks agar lebih konsisten, *tokenization* yang memecah teks menjadi kumpulan kata, *remove stopword* dengan menghapus kata-kata yang tidak memiliki pengaruh penting, serta *stemming* untuk mengubah kata menjadi bentuk dasar [66][67].

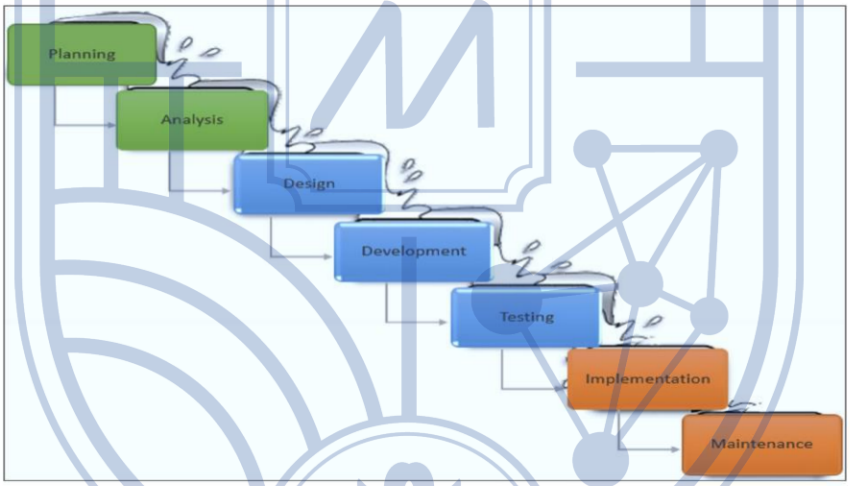
2.6 Metode *Data Splitting*

Data Splitting adalah metode yang digunakan untuk membagi dataset menjadi dua bagian yaitu *training set* dan *testing set* [68]. Untuk melakukan evaluasi model secara mendalam, dataset dapat dibagi menjadi tiga bagian, yaitu *training set*, *validation set*, dan *test set*. Tujuan nya adalah mencegah *overfitting* [69]. *Overfitting* terjadi ketika model terlalu menyesuaikan diri dengan data pelatihan, sehingga kurang mampu memberikan prediksi yang akurat pada data baru [70].

Training set adalah kumpulan data yang digunakan untuk melatih model dalam proses mempelajari pola dan hubungan dari suatu data. *Validation set* digunakan untuk memperkirakan kemampuan model selama proses pelatihan serta membantu dalam pemilihan *hyperparameter* yang optimal. Setelah pelatihan model selesai, *testing set* digunakan untuk menguji kemampuan model dalam melakukan prediksi pada data baru yang belum pernah dilihat sebelumnya, sehingga dapat menilai performa model [71].

2.7 Metode *Waterfall*

Model *waterfall* merupakan salah satu pendekatan dalam *Software Development Life Cycle* (SDLC) yang dikenal juga sebagai model air terjun. Model ini menggambarkan proses pengembangan perangkat lunak secara sistematis dan berurutan. Disebut sebagai model air terjun karena setiap tahap dikerjakan secara bertahap dari atas ke bawah. Keunggulan model ini terletak pada alur kerja yang jelas, serta efisiensi dalam hal waktu dan biaya [72]. Berikut Gambar 2.6 yang menujukkan tahapan dalam metode *waterfall* [73].



Gambar 2. 6 Tahapan Metode *Waterfall* [73]

1. Perencanaan (*Planning*)

   Tahap awal yang dilakukan adalah mendefinisikan desain, fungsi, permasalahan, serta tujuan dari proyek yang akan dikembangkan.

2. Analisis (*Analysis*)

   Pada tahap ini dilakukan identifikasi dan pendokumentasian seluruh kebutuhan bisnis, termasuk estimasi biaya serta penjadwalan proyek.

3. Perancangan (*Design*)

   Setelah kebutuhan ditentukan, dilakukan perancangan arsitektur sistem, model, serta prinsip desain yang akan digunakan dalam pengembangan.

4. Pengembangan (*Development*)

   Tahap ini melibatkan proses pembuatan sistem dengan memecah proyek menjadi beberapa komponen yang kemudian dikembangkan dan diintegrasikan hingga membentuk produk akhir.

5. Pengujian (*Testing*)

   Setelah proses pengembangan selesai, dilakukan pengujian oleh tim untuk menemukan kesalahan, celah keamanan, atau bug, yang kemudian diperbaiki oleh pengembang.

6. Implementasi (*Implementation*)

   Sistem yang telah selesai dikembangkan dan diuji kemudian siap untuk digunakan atau didistribusikan kepada pengguna.

7. Pemeliharaan (*Maintenance*)

   Setelah sistem digunakan, dilakukan proses pemeliharaan berupa perbaikan, penyesuaian, atau pengembangan lebih lanjut sesuai kebutuhan dan umpan balik pengguna.

Dalam metode *waterfall* tentunya memiliki kelebihan dan kekurangan dalam penerapan nya. Adapun kelebihan dari metode *waterfall* antara lain sebagai berikut [74]:

1. Sistem yang dihasilkan cenderung memiliki kualitas yang baik karena proses pengembangannya dilakukan secara bertahap dan terstruktur.
2. Tahapan pengembangan dilakukan secara berurutan, sehingga dapat mengurangi potensi kesalahan dalam setiap proses.
3. Dokumentasi sistem tersusun dengan rapi, karena setiap tahap harus diselesaikan terlebih dahulu sebelum melanjutkan ke tahap berikutnya.

Di sisi lain, metode *waterfall* juga memiliki beberapa kekurangan, diantaranya [74]:

1. Proses pengembangan memerlukan waktu yang relatif lama serta biaya yang cukup besar.
2. Membutuhkan pengelolaan yang baik karena tahapan yang telah dilalui sulit untuk diulang sebelum sistem selesai secara keseluruhan.
3. Kesalahan kecil yang terjadi pada tahap awal dapat berkembang menjadi masalah besar apabila tidak segera terdeteksi, sehingga berdampak pada tahap selanjutnya.
4. Dalam implementasinya, metode ini jarang berjalan sepenuhnya secara sekuensial sesuai teori, karena sering terjadi iterasi yang justru dapat menimbulkan permasalahan baru.

2.8 *Threshold Tuning*

Threshold tuning merupakan metode yang digunakan untuk menentukan nilai ambang batas yang paling optimal dalam proses klasifikasi, khususnya pada dataset yang memiliki distribusi kelas tidak seimbang [75]. Penentuan nilai *threshold* yang optimal bertujuan untuk meningkatkan performa model klasifikasi. Peningkatan performa tersebut dapat diukur melalui metrik seperti *accuracy*, *precision*, *recall*, dan *f-1 score* [76].

Dalam klasifikasi biner, model secara umum menggunakan *threshold* sebesar 0,5 sebagai nilai *default* [77]. Dengan menggunakan *threshold* sebesar 0,5, data yang memiliki probabilitas prediksi sama dengan atau lebih tinggi dari nilai tersebut akan ditetapkan sebagai kelas positif. Meskipun demikian, pada dataset yang tidak seimbang, pendekatan ini sering kali menghasilkan kecenderungan model untuk memprioritaskan prediksi pada kelas mayoritas daripada kelas minoritas. Untuk mengatasi hal tersebut, dilakukan dengan menguji berbegai nilai threshold probabilitas untuk meningkatkan nilai ambang yang mampu menghasilkan keseimbangan terbaik [78].

2.9 *Hyperband Optimization*

Hyperband adalah algoritma optimasi *hyperparameter* yang dikembangkan untuk mengatasi keterbatasan metode pencarian konvensional seperti *Grid Search* dan *Random Search* dengan menerapkan alokasi sumber daya yang lebih efisien sehingga proses pencarian kombinasi *hyperparameter* terbaik dapat dilakukan lebih cepat [79]. *Hyperband* bekerja dengan menjalankan eksperimen secara pararel dan menerapkan mekanisme pemberhentian dini (*early stopping*) untuk konfigurasi yang kurang optimal, sehingga sumber daya yang dapat difokuskan pada kandidat terbaik untuk meningkatkan efisiensi pencarian *hyperparameter* [80].

Beberapa jenis *hyperparameter* yang dioptimasi menggunakan *hyperband* seperti *learning rate* dan *batch size*. Agar dapat bekerja secara optimal, *hyperband* memerlukan perancangan dan penyesuaian yang tepat pada proses evaluasi dengan fidelitas rendah maupun tinggi. Kemampuannya dalam menyeimbangkan efisiensi penggunaan sumber daya dan kebutuhan evaluasi yang mendalam menjadikan metode ini sesuai untuk diterapkan pada permasalahan optimasi *hyperparameter* yang kompleks [81].

2.10 Evaluasi Model

Pada tahap ini, dilakukan evaluasi yang bertujuan untuk menilai model algoritma yang diterapkan. Alat evaluasi yang digunakan adalah *confusion matrix*. *Confusion matrix*

merupakan alat yang digunakan untuk menilai kinerja suatu model dengan cara membandingkan kelas hasil prediksi dengan kelas sebenarnya pada data [82]. Berikut Tabel 2.1 yang menunjukkan perhitungan *confusion matrix* [83]:

Tabel 2. 1 Perhitungan *Confusion Matrix* [83]

Nilai Prediksi	Nilai Aktual	
	Positif	Negatif
Positif	TP	FP
Negatif	FN	TN

Keterangan:

1. *True Positive* (TP)

   Hasil prediksi yang menunjukkan kelas positif dan sesuai dengan kondisi sebenarnya yang juga positif.

2. *True Negative* (TN)

   Hasil prediksi yang menunjukkan kelas negatif dan sesuai dengan kondisi sebenarnya yang juga negatif.

3. *False Positive* (FP)

   Hasil prediksi menunjukkan kelas positif, tetapi kondisi sebenarnya adalah negatif.

4. *False Negative* (FN)

   Hasil prediksi menunjukkan kelas negatif, tetapi kondisi sebenarnya adalah positif.

Dalam evaluasi model klasifikasi, kinerja model umumnya diukur menggunakan beberapa metrik seperti *accuracy*, *precision*, *recall*, dan *f-1 score*. Semakin tinggi nilai *accuracy*, *precision*, *recall*, dan *f-1 score* yang diperoleh, maka semakin baik pula performa model dalam melakukan klasifikasi [84]. Sebaliknya, jika nilai *accuracy*, *precision*, *recall*, dan *f-1 score* semakin rendah, maka semakin menurun pula kinerja model dalam melakukan klasifikasi. Berdasarkan tabel 2.1, dapat diperoleh persamaan untuk menghitung nilai *accuracy*, *precision*, *recall*, dan *f-1 score* pada rumus-rumus berikut [85]:

1. *Accuracy*

   Accuracy didefinisikan sebagai perbandingan antara jumlah prediksi yang benar dengan total seluruh prediksi, dan dihitung menggunakan rumus berikut:

   $$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{.....(20)}$$

2. *Precision*

Precision didefinisikan sebagai perbandingan antara jumlah prediksi positif yang benar dengan total prediksi positif, dan dihitung menggunakan rumus berikut:

$$Precision = \frac{TP}{TP+FP} \quad \text{.....(21)}$$

3. *Recall*

Recall didefinisikan sebagai perbandingan antara jumlah prediksi positif yang benar dengan jumlah seluruh data yang sebenarnya positif, dan dihitung menggunakan rumus berikut:

$$Recall = \frac{TP}{TP+FN} \quad \text{.....(22)}$$

4. *F1-Score*

F1-Score merupakan rata-rata harmonik dari nilai presisi dan recall, dan dihitung menggunakan rumus berikut:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad \text{.....(23)}$$