BAB 2
LANDASAN TEORI

2.1 Kesalahan Penulisan

Kesalahan penulisan adalah kesalahan yang terjadi saat proses pengetikan teks dan dapat mengubah makna suatu kata bahkan keseluruhan kalimat [17]. Terjadinya kesalahan ketik dapat menyebabkan informasi tidak tersampaikan dengan baik kepada pembaca serta menimbulkan kesalahpahaman terhadap informasi yang disampaikan.

Penggunaan ejaan yang benar diperlukan untuk menjaga kejelasan makna, mencegah kesalahpahaman, serta meningkatkan kualitas tulisan. Hal ini menjadi semakin relevan dalam penulisan karya ilmiah, laporan, makalah, hingga berita [21].

2.2 Aturan Penulisan Klitik

Pada konteks penulisan klitik, terdapat beberapa aturan yang perlu diperhatikan. Aturan penulisan klitik dapat mengikuti sesuai kaidah Ejaan yang Disempurnakan (EYD) atau Pedoman Umum Ejaan Bahasa Indonesia (PUEBI).

2.2.1 Ejaan yang Disempurnakan (EYD)

Ejaan Bahasa Indonesia yang Disempurnakan (EYD) [28] merupakan pedoman resmi yang dapat digunakan oleh instansi pemerintah, lembaga swasta, maupun masyarakat umum dalam menerapkan bahasa Indonesia secara tepat dan benar. Peraturan penulisan klitik seperti *-ku*, *-mu*, *-nya*, *ku-*, dan *kau-* berdasarkan peraturan yang tertulis di EYD sebagai berikut.

1. Kata ganti *ku-* dan *kau-* ditulis serangkai dengan kata yang mengikutinya, sedangkan *-ku*, *-mu*, dan *-nya* ditulis serangkai dengan kata yang mendahuluinya. Sebagai contoh.
    - Rumah itu telah *ku*jual
    - Majalah ini boleh *kau*baca
    - Buku*ku*, buku*mu*, dan buku*nya* tersimpan di perpustakaan

2. Kata ganti *kau* yang bukan bentuk terikat ditulis terpisah dengan kata yang lain.

    - Aku ingin *kau* bersungguh-sungguh dengan apa yang kaukatakan.
    - *Kau* masih muda, Bung.
    - Sebaiknya, *kau* mengurus adikmu saja

2.2.2 Pedoman Umum Ejaan Bahasa Indonesia (PUEBI)

PUEBI Daring merupakan aplikasi atau laman berbasis gawai dari Pedoman Umum Ejaan Bahasa Indonesia sesuai Permendikbud 50/2015 [29]. Versi ini dilengkapi dengan sejumlah catatan tambahan yang belum tercantum secara eksplisit dalam dokumen asli Permendikbud 50/2015. Aturan penulisan klitik berdasarkan PUEBI sebagai berikut.

1. Kata ganti *ku-* dan *kau-*, ditulis serangkai dengan kata yang mengikutinya atau kata setelahnya. Berikut beberapa contohnya dalam kalimat.

    - Baju yang *ku*pakai kemarin sudah terjual di *marketplace*. (kupakai merupakan contoh kata ganti ku-).
    - Kemarin *kau*bawa titipan itu dalam tas. (kaubawa merupakan contoh kata ganti kau-).

2. Kata ganti *-ku*, *-mu*, dan *-nya*, ditulis serangkai dengan kata yang mendahuluinya atau kata sebelumnya. Berikut beberapa contohnya dalam kalimat.

    - Motor*ku* kemarin malam terparkir di garasi. (motorku contoh kata ganti *-ku*).
    - Buku*mu*, dan Buku*nya* tersimpan di perpustakaan (bukumu contoh kata ganti *-mu*. Bukunya contoh kata ganti *-nya*)

3. Khusus untuk kata ganti *-ku*, *-mu*, dan *-nya*, dapat diberi tanda hubung (-), jika penulisannya digabung dengan singkatan atau yang diawali huruf kapital. Berikut beberapa contohnya dalam kalimat.

    - Aku pergi ke Lamongan, tapi KTP-*ku* ketinggalan di rumah. (KTP-ku contoh kata ganti *-ku*).

- Jangan lupa SIM-*mu* harus dibawa agar tidak kena tilang. (SIM-mu contoh kata ganti -*mu*).
- Sampaikan ke Rani, STNK-*nya* ketinggalan di rumah paman. (STNK-nya contoh kata ganti -*nya*).

2.3 N-Gram

N-gram merupakan potongan *substring* sepanjang *n* karakter yang diambil dari suatu *string*. Secara sederhana, *n-gram* dapat diartikan sebagai urutan sejumlah *n* karakter yang bersebelahan dalam sebuah kata atau teks. Metode ini banyak diterapkan dalam bidang pemrosesan bahasa alami untuk menghasilkan, menganalisis, atau memodelkan kata maupun karakter [30].

Proses pembentukan *n-gram* dilakukan dengan cara memecah teks menjadi potongan-potongan karakter sebanyak *n* secara berurutan dari awal hingga akhir dokumen. Nilai *n* menentukan panjang potongan karakter yang dihasilkan. Untuk membantu proses ini, biasanya ditambahkan karakter kosong di awal dan akhir kata sebagai padding [31].

Sebagai ilustrasi, kata "Rumah nya" dapat diuraikan menjadi beberapa bentuk *n-gram* seperti berikut (tanda "_" menunjukkan blank):

- *Unigram* ($n = 1$): R, u, m, a, h, _, n, y, a
- *Bigram* ($n = 2$): _R, Ru, um, ma, ah, h_, _n, ny, ya, a_
- *Trigram* ($n = 3$): _Ru, Rum, uma, mah, ah_, h_n, _ny, nya, ya_
- *Quadgram* ($n = 4$): _Rum, Ruma, umah, mah_, ah_n, h_ny, _nya, nya_
- *Quintgram* ($n = 5$): _Ruma, Rumah, umah_, mah_n, ah_ny, h_nya, _nya_

2.4 Rule-based

Sistem berbasis aturan (*rule-based*) merupakan awal dari perkembangan AI. *Rule-based* beroperasi menggunakan logika dan seperangkat aturan yang telah ditentukan sebelumnya [32]. Pada awal perkembangan *machine learning*, pendekatan berbasis aturan menjadi fokus utama dalam penelitian, dengan akar yang telah ada sejak tahun 1960-an. Metode ini mencapai masa kejayaannya pada awal 2000, ketika pembelajaran berbasis aturan menjadi salah satu teknik yang populer dalam bidang *machine learning* [33].

Cara kerja *rule-based* bersifat transparan, setiap *instance* diklasifikasikan ke dalam suatu kelas setidaknya satu aturan terpenuhi. Aturan-aturan ini secara alami berfungsi sebagai penjelasan yang mudah dipahami terhadap hasil klasifikasi. Pengklasifikasi *rule-based* dapat direpresentasikan secara ringkas melalui formula *Disjunctive Normal Form* (DNF), yaitu kombinasi logika *OR* dari beberapa *term* konjungtif *AND* [33].

Sebagai contoh, representasi *rule-based classifier* dalam bentuk *Disjunctive Normal Form* (DNF) dapat dituliskan sebagai berikut:

$$(\mathrm{A} \wedge \mathrm{B}) \vee (\mathrm{C} \wedge \mathrm{D}) \tag{2.1}$$

Formula tersebut menunjukkan bahwa suatu *instance* akan diklasifikasikan ke dalam kelas tertentu apabila aturan pertama (*A* dan *B* terpenuhi) atau aturan kedua (*C* dan *D* terpenuhi). Sebagai ilustrasi, dalam konteks klasifikasi cuaca, aturan tersebut dapat direpresentasikan sebagai:

$$(\text{Hujan} \wedge \text{Berawan}) \vee (\text{Lembap} \wedge \text{Suhu Tinggi})$$

Artinya, kondisi akan dikategorikan sebagai *Cuaca Buruk* apabila terjadi Hujan dan Berawan, atau apabila kondisi Lembap dan Suhu Tinggi terpenuhi. Representasi dalam bentuk DNF ini menjadikan model *rule-based* mudah dipahami dan diinterpretasikan, karena setiap keputusan dapat ditelusuri langsung melalui kombinasi aturan logika yang digunakan.

2.5 Hypertuning Parameter

Hyperparameter tuning dilakukan dengan mencari kombinasi *hyperparameter* yang paling optimal. *Hyperparameter* merupakan parameter yang mengatur proses pelatihan serta struktur dari model *machine learning*. Berbeda dengan parameter model yang dipelajari selama proses *training*, *hyperparameter* ditentukan terlebih dahulu sebelum *training* dimulai. *Hyperparameter* memiliki peran penting dalam menentukan kinerja model. Contoh *hyperparameter* antara lain jumlah pohon pada *random forest*, kedalaman pada *decision tree*, dan sebagainya [34].

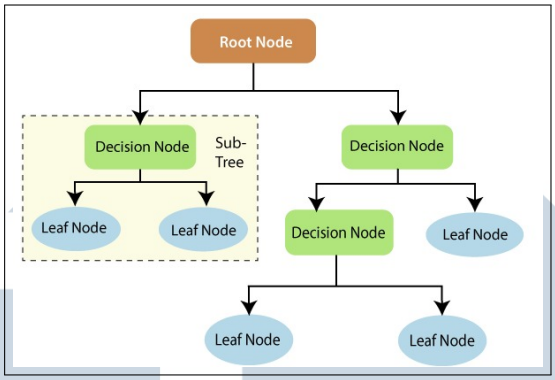
Hyperparameter tuning memiliki beberapa pendekatan, di antaranya *Grid Search*, *Random Search*, *Bayesian Optimization*, *Meta-learning*, dan *Automated*

Machine Learning (AutoML). Pada penelitian ini, metode *hyperparameter tuning* yang digunakan adalah *Grid Search*. *Grid Search* merupakan teknik *brute-force* yang secara menyeluruh menelusuri kombinasi *hyperparameter* yang telah ditentukan sebelumnya. Proses ini membutuhkan sumber daya komputasi yang besar, terutama ketika ruang pencarian *hyperparameter* sangat luas [35].

Hyperparameter tuning pada *Random Forest* mengacu pada parameter yang digunakan, diantaranya *n_estimators, max_depth, min_samples_split, min_samples_leaf, max_features, max_samples*, dan sebagainya. Parameter *n_estimators* merupakan parameter untuk menentukan jumlah *decision tree* yang akan dibangun dalam model untuk proses prediksi [36], *max_depth* merupakan paramater untuk menentukan jumlah maksimal untuk kedalaman pohon, *max_features* digunakan untuk jumlah maksimal fitur yang dipertimbangkan setiap kali model melakukan pemilihan *split*, *min_samples_split* digunakan untuk menentukan jumlah minimum sampel yang diperlukan untuk membagi sebuah node, dan *min_samples_leaf* untuk menentukan jumlah minimum sampel yang harus dimiliki oleh sebuah *leaf node* [37].

2.6 Decision Tree

Decision tree merupakan metode klasifikasi yang menggunakan struktur pohon, di mana setiap *node* merepresentasikan sebuah atribut atau fitur, setiap cabang menunjukkan nilai dari atribut tersebut, dan setiap *leaf* atau daun merepresentasikan kelas hasil klasifikasi [17]. *Decision tree* secara eksplisit menampilkan semua kemungkinan alternatif dan menelusuri setiap alternatif hingga pada kesimpulannya dalam satu tampilan, sehingga memudahkan proses perbandingan di antara berbagai alternatif seperti pada Gambar 2.1. Algoritma *decision tree* digunakan untuk membagi atribut atau fitur yang akan diuji pada setiap *node* guna menentukan apakah pemisahan tersebut merupakan yang "terbaik" untuk setiap kelas [38].



Gambar 2.1. Diagram skematik *decision tree* [2]

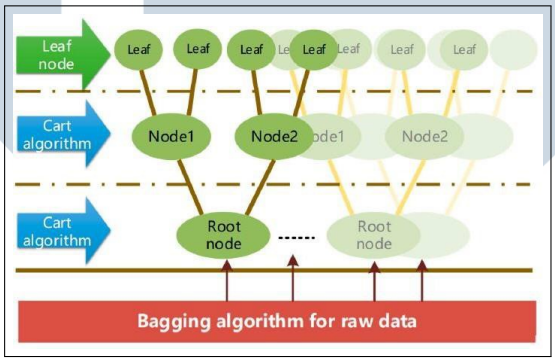
2.7 Random Forest

Algoritma *Random Forest* ini pertama kali diperkenalkan oleh Breiman dan sejak itu banyak digunakan dalam berbagai bidang, baik untuk klasifikasi maupun prediksi nilai numerik [4]. *Random Forest* (RF) merupakan salah satu algoritma *machine learning* yang termasuk dalam kategori *ensemble learning*, yaitu teknik yang menggabungkan banyak *decision tree* untuk meningkatkan akurasi prediksi serta mengurangi risiko *overfitting*, kondisi ketika model memiliki akurasi tinggi pada *training*, tapi menghasilkan hasil kurang akurat pada data baru. *Ensemble learning* merupakan algoritma dalam *machine learning* yang membentuk kumpulan pengklasifikasi. Sebuah *ensemble of classifiers* adalah himpunan dari beberapa pengklasifikasi yang keputusannya masing-masing (biasanya melalui proses pemungutan suara berbobot maupun tanpa bobot) digabungkan dengan cara tertentu untuk mengidentifikasi *instance* baru [39].

Gambar 2.2 dan 2.3 menggambarkan bahwa *Random Forest* bekerja dengan membangun sejumlah *decision tree* menggunakan data latih yang diambil secara acak melalui teknik *bootstrap sampling*. Pada setiap pohon, fitur yang digunakan untuk melakukan pemisahan *node* juga dipilih secara acak. Proses randomisasi ini mengurangi korelasi antar pohon dan menghasilkan model yang lebih kuat dibandingkan metode tunggal seperti *Decision Tree* atau bahkan metode *bagging* [4] [40]. Metode *bagging* merupakan teknik *ensemble* yang menggunakan beberapa *decision tree* yang dilatih dari data pelatihan hasil *resampling*. Setiap pohon dibuat dari subset data yang diambil secara acak dengan pengembalian, kemudian hasil prediksi dari seluruh pohon digabung menggunakan voting mayoritas untuk memperoleh hasil akhir yang lebih akurat dan stabil [41].

Input: *N - Quantitative amount of bootstrap samples*
M - Total number of features
m - Sample size
k - Next node
Output: A Random Forest (RF)
Steps:
*1. **Creates** N bootstrap samples from the dataset.*
2. Every node (sample) takings a feature randomly of size m where m<M.
3. Builds a split for the m features selected in Step 2 and detects the k node by using the best split point.
4. Split the tree iteratively until one leaf node is attained and the tree remains completed.
5. The algorithm is trained on each bootstrapped independently.
6. Using trees classification voting predicted data is collected from the trained trees (n).
7. The final RF model is build using the peak voted features.
*8. **return** RF*
End.

Gambar 2.2. *Pseudocode random forest* [3]



Gambar 2.3. Diagram skematik algoritma *random forest* [4]

2.8 Metrik Evaluasi

Evaluasi suatu model pada dasarnya didasarkan pada pemahaman mengenai kriteria yang menentukan apakah sebuah model dapat dianggap baik [42]. Penilaian ini direpresentasikan melalui metrik evaluasi, yang berperan penting dalam mengukur kemampuan generalisasi model klasifikasi yang telah dilatih. Metrik tersebut digunakan untuk menilai dan meringkas kualitas performa model ketika diuji menggunakan data baru yang belum pernah digunakan dalam proses pelatihan. Salah satu metrik yang paling umum digunakan adalah *accuracy*, yang mengukur proporsi prediksi benar terhadap keseluruhan data uji. Performa model klasifikasi seperti *accuracy* dianalisis menggunakan *confusion matrix*, yang memberikan gambaran mengenai distribusi hasil prediksi model terhadap kondisi aktual.

Tabel 2.1 menunjukkan contoh *confusion matrix*, di mana baris merepresentasikan kondisi aktual, dan kolom menunjukkan kondisi yang

diprediksi. Matriks ini terdiri dari empat komponen utama yaitu *True Positive* (TP), yaitu ketika model memprediksi positif dan kondisi sebenarnya juga positif, *False Negative* (FN), yaitu ketika model memprediksi negatif padahal kondisi sebenarnya positif, *False Positive* (FP), yaitu ketika model memprediksi positif padahal kondisi sebenarnya negatif, dan *True Negative* (TN), yaitu ketika model memprediksi negatif dan kondisi sebenarnya juga negatif [1].

Tabel 2.1. Tabel *Confusion Matrix* [1]

		Prediksi	
		Positif	Negatif
Aktual	Positif	*True Positive* (TP)	*False Negative* (FN)
	Negatif	*False Positive* (FP)	*True Negative* (TN)

2.8.1 Accuracy

Accuracy merupakan ukuran yang menunjukkan seberapa efektif sebuah algoritma dalam mengklasifikasikan data dengan benar dibandingkan dengan keseluruhan data yang tersedia. Perhitungan *accuracy* dijabarkan dalam persamaan sebagai berikut [43].

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{2.2}$$

Keterangan:
TP = *True Positive* (Benar Positif)
TN = *True Negative* (Benar Negatif)
FP = *False Positive* (Salah Positif)
FN = *False Negative* (Salah Negatif)

Accuracy dihitung sebagai persentase data yang diklasifikasikan secara benar oleh algoritma, mencakup seluruh data yang dievaluasi. Secara matematis, *Accuracy* dapat ditentukan melalui perbandingan antara jumlah prediksi yang benar (baik prediksi positif maupun negatif) dengan total seluruh prediksi yang dilakukan

2.8.2 Precision

Precision mengukur proporsi prediksi positif yang benar-benar tepat. Metode ini penting pada situasi di mana *false positive* sangat tidak diinginkan. Secara matematis, *precision* didefinisikan sebagai persamaan berikut [43].

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

Precision dihitung sebagai rasio antara jumlah data yang diprediksi benar pada kategori positif dengan total data yang diprediksi sebagai positif. *Precision* menggambarkan proporsi data yang benar-benar relevan (positif) dari seluruh data yang dianggap relevan oleh algoritma.

2.8.3 Recall

Recall mengukur proporsi data positif sebenarnya yang berhasil diidentifikasi dengan benar. Metode ini penting pada kasus di mana kesalahan *false negative* memiliki dampak besar. Secara matematis, *recall* dihitung sebagai [43].

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

Recall mengukur proporsi data dalam kelas tertentu yang berhasil diklasifikasikan dengan benar sebagai anggota kelas tersebut, sehingga memberikan gambaran seberapa baik algoritma dalam menangkap semua data yang relevan.

2.8.4 F1-score

F1-score adalah ukuran yang digunakan untuk menilai kinerja keseluruhan dari suatu model dengan mempertimbangkan keseimbangan antara *precision* dan *recall*. Metode ini sangat berguna terutama pada kasus dengan ketidakseimbangan kelas dalam *dataset*. Secara matematis, *F1-score* dapat dihitung dengan persamaan berikut, yang diadaptasi dari [43].

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.5}$$

F1-score, yaitu rata-rata dari *precision* dan *recall*, memberikan ukuran yang seimbang terhadap kinerja model.