BAB 2
LANDASAN TEORI

Pada bagian ini dijabarkan teori-teori yang mendasari penelitian secara lengkap dan menyeluruh.

2.1 Ejaan Yang Disempurnakan (EYD)

Ejaan Yang Disempurnakan (EYD) adalah pedoman resmi yang digunakan untuk menulis dalam bahasa Indonesia, sebagaimana diatur oleh pemerintah melalui aturan-aturan yang tertuang dalam Pedoman Umum Ejaan Bahasa Indonesia (PUEBI) [18]. EYD terus berkembang seiring dengan perkembangan bahasa dan kebutuhan masyarakat, yang kemudian diperbarui menjadi PUEBI, termasuk dalam edisi EYD 5 yang menjadi acuan penelitian ini. Ejaan Yang Disempurnakan adalah sistem ejaan bahasa Indonesia yang berlaku sejak tahun 1972, menggantikan sistem ejaan Van Ophuijsen dan Ejaan Republik (Soewandi). Tujuan utamanya adalah untuk menyeragamkan penulisan bahasa Indonesia agar lebih mudah dipahami, efektif, dan efisien [19].

EYD mencakup aturan-aturan penting dalam penulisan bahasa Indonesia, salah satunya yaitu penulisan huruf kapital. Huruf kapital digunakan untuk menandai hal-hal tertentu sesuai konteksnya. Berikut adalah beberapa aturan dasar yang digunakan:

1. Huruf kapital digunakan sebagai huruf pertama awal kalimat.
2. Huruf kapital tidak digunakan sebagai huruf pertama yang digunakan sebagai satuan ukuran.
3. Huruf kapital digunakan pada nama teori, hukum, dan rumus.
4. Huruf kapital tidak digunakan untuk menuliskan huruf pertama kata yang bermakna 'anak dari', seperti bin, binti, boru, dan van.
5. Huruf kapital digunakan pada awal kalimat dalam petikan langsung.
6. Huruf kapital digunakan sebagai huruf pertama yang berkaitan dengan nama agama, kitab suci, dan Tuhan, termasuk sebutan dan kata ganti Tuhan serta singkatan nama Tuhan.

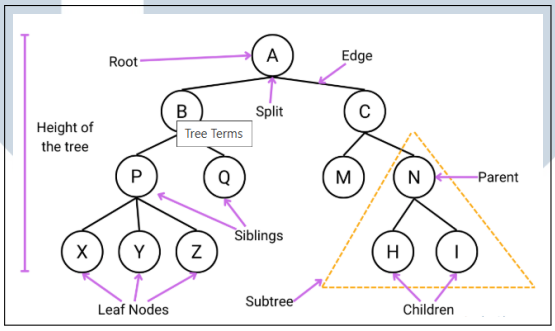
7. Huruf kapital digunakan sebagai huruf pertama unsur nama kebangsawanan, keturunan, keagamaan, atau akademik yang diikuti nama orang dan gelar akademik yang mengikuti nama orang.
8. Huruf kapital digunakan sebagai huruf pertama unsur nama jabatan dan pangkat yang diikuti nama orang atau nama instansi.
9. Huruf kapital digunakan sebagai huruf pertama seperti pada nama bangsa, suku, bahasa.
10. Huruf kapital tidak digunakan pada nama bangsa, suku, bahasa yang berupa bentuk dasar kata turunan.
11. Huruf kapital digunakan pada huruf pertama, seperti pada nama tahun, bulan, hari, dan hari besar atau hari raya.
12. Huruf kapital digunakan pada huruf pertama unsur nama peristiwa sejarah.
13. Huruf pertama peristiwa sejarah yang tidak digunakan sebagai nama ditulis dengan huruf nonkapital.
14. Huruf kapital digunakan sebagai huruf pertama nama geografi.
15. Huruf pertama unsur geografi yang tidak diikuti nama diri ditulis dengan huruf nonkapital.
16. Huruf pertama nama diri geografi yang digunakan sebagai nama jenis ditulis dengan huruf nonkapital.
17. Huruf kapital digunakan sebagai huruf pertama kata penunjuk hubungan kekerabatan, seperti bapak, ibu, kakak, dan adik serta kata atau ungkapan lain (termasuk unsur bentuk ulang utuh) yang digunakan sebagai sapaan.

EYD berperan penting dalam menjaga kredibilitas tulisan, terutama dalam konteks jurnalistik. Kesalahan dalam penggunaan EYD, khususnya huruf kapital, dapat menurunkan kualitas tulisan dan membingungkan , pembaca[20, 21]. Oleh karena itu, pemahaman dan penerapan EYD menjadi keharusan bagi penulis, terutama di era digital dengan produksi konten yang masif. Dengan adanya sistem otomatis yang dapat mendeteksi kesalahan kapitalisasi sesuai EYD 5, diharapkan penulis dan jurnalis dapat lebih mudah menghasilkan karya yang sesuai dengan standar bahasa Indonesia [22].

2.2 Decision Tree

Decision Tree (Pohon Keputusan) adalah algoritme pembelajaran mesin yang digunakan untuk menyelesaikan masalah klasifikasi dan regresi [23]. Model ini berbentuk pohon dengan struktur hierarkis yang terdiri dari simpul (node) dan cabang (branch). Setiap simpul merepresentasikan pengambilan keputusan atau pengujian terhadap atribut tertentu, sementara cabangnya menunjukkan hasil dari pengujian tersebut [23]. Daun (leaf) pada pohon menggambarkan kelas atau nilai prediksi yang diperoleh. Decision Tree dikenal karena kemampuannya membangun model yang mudah dipahami dan intuitif, serta membagi dataset menjadi subset menggunakan aturan sederhana yang mudah diinterpretasikan.

Arsitektur Decision Tree dapat digambarkan sebagai berikut:



Gambar 2.1. Arsitektur Decision Tree [1]

Pada Gambar 2.1, terdapat *root node*, yaitu node awal dalam Decision Tree yang mewakili seluruh dataset. Dari *root node*, data kemudian dibagi berdasarkan atribut-atribut keputusan pada *internal nodes*. Internal nodes ini berfungsi untuk membuat keputusan dalam membagi dataset menjadi subset-subset yang lebih kecil. Setelah beberapa pembagian, pohon mencapai *leaf nodes*, yang merupakan node akhir dan menyimpan keputusan final atau prediksi yang dihasilkan berdasarkan data yang ada. Cabang-cabang pohon (*branches*) menghubungkan *root node* dengan *internal nodes* atau *leaf nodes* dan mencerminkan keputusan yang diambil berdasarkan atribut tertentu.

Berikut adalah rumus utama dalam Decision Tree [24]:

$$H(S) = -\sum_{i=1}^{k} p_i \log_2(p_i) \tag{2.1}$$

Dimana:

- S: Dataset
- k: Jumlah kelas dalam dataset
- p_i: Proporsi data dari kelas ke-i

A Information Gain (ID3 Algorithm)

Information Gain mengukur pengurangan ketidakpastian (*entropy*) ketika data dibagi berdasarkan suatu atribut. Rumus untuk Information Gain adalah sebagai berikut [24]:

$$IG(A) = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v) \tag{2.2}$$

Dimana:

- A: Atribut yang dipertimbangkan
- $\text{Values}(A)$: Nilai-nilai unik dari atribut A
- S_v: Subset dari S di mana atribut A memiliki nilai v
- $|S|$: Jumlah data dalam dataset

B Gini Index (CART Algorithm)

Gini Index mengukur impuritas sebuah node. Rumus Gini Index adalah sebagai berikut [25]:

$$Gini(S) = 1 - \sum_{i=1}^{k} p_i^2 \tag{2.3}$$

Dimana:

- p_i: Proporsi data dari kelas ke-i

Gini Index untuk atribut dihitung dengan rumus [25]:

$$Gini(A) = \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Gini(S_v) \tag{2.4}$$

Dengan struktur pohon yang sederhana dan intuitif, Decision Tree menjadi algoritma yang banyak digunakan dalam berbagai aplikasi [26]. Namun, perlu diperhatikan potensi masalah *overfitting* pada model yang terlalu dalam, serta

kebutuhan untuk melakukan *pruning* (pemangkasan) untuk menghindari *overfitting* dan meningkatkan generalisasi model.

2.3 Random Forests

Random Forest adalah algoritma *ensemble learning* yang digunakan untuk klasifikasi dan regresi. Algoritma ini menggabungkan beberapa *decision tree* untuk meningkatkan akurasi dan mengurangi risiko *overfitting* yang sering terjadi pada model *decision tree* tunggal. *Random Forest* dikembangkan oleh Leo Breiman dan Adele Cutler pada awal 2000-an dan telah menjadi salah satu metode yang paling populer dalam *machine learning* [27].

Random Forest bekerja dengan membangun sejumlah *decision tree* selama pelatihan dan menggabungkan hasilnya untuk menghasilkan prediksi yang lebih stabil dan akurat. Proses ini melibatkan dua langkah utama:

1. **Bootstrap Aggregating (Bagging):** *Random Forest* menggunakan teknik *bagging*, di mana beberapa subset data diambil secara acak dengan pengembalian (*bootstrapping*) untuk melatih setiap *decision tree*. Dengan cara ini, setiap pohon dilatih pada data yang berbeda, sehingga mengurangi varians model [27].

2. **Random Feature Selection:** Saat membangun setiap *decision tree*, *Random Forest* memilih subset acak dari fitur yang tersedia untuk menentukan split pada setiap node. Ini membantu dalam mengurangi korelasi antara *tree* dan meningkatkan keragaman model, yang pada gilirannya meningkatkan akurasi prediksi [28].

Beberapa turunan dari *Random Forest* yang telah dikembangkan untuk meningkatkan kinerja dan efisiensi algoritma ini meliputi:

1. **Extremely Randomized Trees (ExtraTrees):** Metode ini mirip dengan *Random Forest*, tetapi pada setiap split, *ExtraTrees* memilih *threshold* secara acak untuk setiap fitur, bukan hanya memilih fitur terbaik. Ini dapat meningkatkan kecepatan pelatihan dan mengurangi varians [29].
2. **Random Forest untuk Regresi:** Meskipun *Random Forest* awalnya dikembangkan untuk klasifikasi, algoritma ini juga dapat digunakan untuk regresi dengan menghitung rata-rata dari prediksi yang dihasilkan oleh setiap *tree*.

3. **Random Forest dengan Penyesuaian:** Beberapa penelitian telah mengembangkan variasi *Random Forest* yang mengadaptasi algoritma untuk menangani data tidak seimbang, seperti menggunakan bobot untuk kelas minoritas.

4. **Random Forest Berbasis Jaringan Saraf:** Kombinasi *Random Forest* dengan jaringan saraf untuk meningkatkan akurasi prediksi, di mana *Random Forest* digunakan untuk memilih fitur yang paling relevan sebelum pelatihan jaringan saraf.

2.4 Ensemble Learning

Ensemble learning adalah pendekatan dalam *machine learning* yang menggabungkan beberapa model untuk meningkatkan akurasi dan stabilitas prediksi. Konsep dasar dari *ensemble learning* adalah bahwa kombinasi dari beberapa model yang berbeda dapat menghasilkan hasil yang lebih baik dibandingkan dengan model tunggal. Pendekatan ini didasarkan pada prinsip bahwa kesalahan yang dibuat oleh model-model individu dapat saling mengimbangi, sehingga meningkatkan kinerja keseluruhan.

Ada beberapa metode *ensemble learning* yang umum digunakan, di antaranya adalah *bagging*, *boosting*, dan *stacking*.

1. **Bagging (Bootstrap Aggregating)** Metode ini melibatkan pelatihan beberapa model independen pada subset data yang berbeda yang diambil secara acak dengan pengembalian (*bootstrapping*). Hasil dari model-model ini kemudian digabungkan, biasanya dengan cara voting untuk klasifikasi atau averaging untuk regresi. *Random Forest* adalah salah satu contoh algoritma yang menggunakan teknik *bagging* [27].

2. **Boosting** Berbeda dengan *bagging*, *boosting* melatih model secara berurutan, di mana setiap model baru berfokus pada kesalahan yang dibuat oleh model sebelumnya. Metode ini bertujuan untuk mengurangi bias dan meningkatkan akurasi. Contoh algoritma *boosting* yang populer adalah *AdaBoost* dan *Gradient Boosting* [30].

3. **Stacking** Metode ini melibatkan pelatihan beberapa model dasar dan kemudian menggunakan model lain (*meta-learner*) untuk menggabungkan prediksi dari model-model dasar tersebut. *Stacking* dapat menggabungkan

berbagai jenis model, sehingga memberikan fleksibilitas dalam pemilihan algoritma [31].

Keuntungan dari *ensemble learning* termasuk peningkatan akurasi, pengurangan varians, dan kemampuan untuk menangani data yang tidak seimbang. Namun, pendekatan ini juga memiliki kelemahan, seperti peningkatan kompleksitas model dan waktu komputasi yang lebih lama.

2.5 Confusion Matrix

Confusion Matrix adalah metode yang digunakan untuk mengevaluasi performa sebuah model klasifikasi dengan membandingkan hasil prediksi model terhadap nilai aktual dari data uji. Matriks ini membantu mengidentifikasi seberapa baik model bekerja, terutama dalam kasus-kasus di mana terdapat klasifikasi biner atau multi-kelas. Confusion Matrix terdiri dari empat elemen utama:

- **True Positive (TP)**: Jumlah kasus di mana model memprediksi kelas positif dan hasilnya benar.
- **False Negative (FN)**: Jumlah kasus di mana model salah tidak mendeteksi kelas positif (seharusnya positif tapi diprediksi negatif).
- **False Positive (FP)**: Jumlah kasus di mana model salah memprediksi kelas negatif sebagai positif.
- **True Negative (TN)**: Jumlah kasus di mana model memprediksi kelas negatif dengan benar.

Berikut ini adalah tabel Confusion Matrix:

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

[32]

Dengan Confusion Matrix, kita bisa menghitung beberapa metrik evaluasi untuk mengukur kualitas model.

- Akurasi mengukur seberapa sering model membuat prediksi yang benar secara keseluruhan. Akurasi adalah rasio dari jumlah prediksi yang benar terhadap total keseluruhan prediksi [32]:

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.5}$$

Akurasi adalah metrik yang paling umum digunakan, namun dapat menyesatkan ketika dataset yang digunakan tidak seimbang, yaitu ketika jumlah kelas positif dan negatif sangat berbeda.

- Presisi menunjukkan seberapa andal model dalam memprediksi kelas positif. Presisi dihitung sebagai rasio antara prediksi benar positif dengan total prediksi positif [32]:

$$\text{Presisi} = \frac{TP}{TP + FP} \tag{2.6}$$

Presisi berguna ketika kesalahan positif palsu (False Positive) memiliki dampak yang besar.

- Recall mengukur kemampuan model untuk menemukan semua contoh kelas positif yang sebenarnya. Recall dihitung sebagai rasio antara prediksi benar positif dengan total contoh positif sebenarnya [32]:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.7}$$

Recall berguna ketika kita ingin memastikan semua contoh positif terdeteksi, terutama ketika kesalahan negatif palsu (False Negative) sangat penting.

- F1-Score adalah metrik yang menggabungkan presisi dan recall, memberikan keseimbangan antara keduanya. F1-Score sangat berguna ketika terdapat ketidakseimbangan antara kelas positif dan negatif. Rumus F1-Score adalah [32]:

$$\text{F1-Score} = 2 \times \frac{\text{Presisi} \times \text{Recall}}{\text{Presisi} + \text{Recall}} \tag{2.8}$$

F1-Score sangat bermanfaat untuk menilai model pada dataset yang tidak seimbang atau ketika salah satu metrik (presisi atau recall) memiliki nilai yang jauh lebih rendah daripada yang lain.