BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1. Tinjauan Pustaka

Perkembangan *machine learning* telah mendorong penerapan model *Hybrid* dalam analisis sentimen guna mengatasi berbagai tantangan seperti ketidakseimbangan kelas, ambiguitas makna, serta kompleksitas bahasa dalam data ulasan. Dalam konteks aplikasi SIAP UNDIP Mahasiswa yang tersedia di Android, ulasan pengguna dikumpulkan melalui *website Google Play Store*. Penelitian ini mengusulkan pendekatan *Hybrid* berupa kombinasi BWELM dan *Random Forest* dengan menggunakan teknik *stacking*. BWELM menghasilkan *output* berupa probabilitas kelas, kemudian dianalisis oleh *Random Forest* sebagai *meta-classifier* untuk menghasilkan prediksi akhir. Tabel 2.1 merangkum studi-studi terdahulu yang menjadi dasar pemilihan dan pengembangan metode ini.

Tabel 2.1 Penelitian terdahulu

No	Penelitian	Objek	Metode	Hasil	*Gap Research*
1	Li dkk. (2014)	Klasifikasi *imbalance* data dari *dataset* biner & data *multiclass* KEEL *repository*	*Boosting Weighted* ELM dengan *AdaBoost*	BWELM mengatasi masalah data *imbalance* dengan lebih seimbang dibanding WELM	Belum diterapkan untuk domain analisis sentimen
2	Liu dkk. (2017)	Klasifikasi *multi-class imbalanced* data ekspresi gen	*Objective Cost-Sensitive Boosting*-WELM	OCS-BWELM mengatasi bias kelas minoritas dan mayoritas data	Belum diuji untuk analisis sentimen sosial media atau aplikasi
3	Karthika dkk. (2019)	Klasifikasi sentimen Produk di *Flipkart*	*Random Forest* vs SVM	*Random Forest* unggul dalam akurasi	Belum mengeksplorasi *boosting* dan *feature selection*
4	Feng (2019)	Analisis sentimen pada Weibo	*AdaBoost* + *Naïve Bayes*	*AdaBoost* meningkatkan performa akurasi *Naive Bayes*	Rentan *overfitting* dan tidak stabil untuk data *noisy*

Tabel 2.2 Penelitian terdahulu (lanjutan)

No	Penelitian	Objek	Metode	Hasil	*Gap Research*
5	Prabhakar dkk. (2019)	Analisis sentimen Twitter maskapai AS	*Modified AdaBoost Ensemble*	*Boosting* efektif untuk klasifikasi *tweet*	Belum dikombinasikan dengan *Random Forest*
6	Cheng dkk. (2020)	Klasifikasi *Multi-label imbalanced* teks	BLWELM (*Boosting Label* WELM)	Akurat dan efisien untuk data tidak seimbang	Masih lemah pada data dengan label kompleks dan fitur tinggi
7	van de Kamp (2022)	Prediksi GDP AS	*Hybrid XGBoost + Random Forest* via *Stacking*	*Hybrid model (stacking)* mengurangi MSE lebih baik dari model tunggal	Belum diterapkan pada data teks atau analisis sentimen
8	Natras dkk. (2022)	Prediksi ionosfer (VTEC)	*Random Forest, AdaBoost, XGBoost* via *Voting*	Kombinasi *ensemble* hasil terbaik untuk prediksi regresi	Belum diterapkan untuk analisis teks atau sentimen
9	Ganesan (2024)	Klasifikasi citra SAR untuk deteksi jejak kapal	*AdaBoost-Weighted* ELM	Meningkatkan akurasi deteksi data *imbalance* pada citra	Fokus di domain citra, bukan teks atau sentimen
10	Padhy dkk. (2024)	Analisis sentimen Twitter	*Random Forest, Naïve Bayes, Decision Tree,* KNN, *Logistic Regression*	*Random Forest* terbaik untuk data berdimensi tinggi	Belum integrasikan model *Hybrid/ensemble* lanjutan

Li dkk. (2014) merupakan pengembang utama dari model *Boosting Weighted* ELM (BWELM) yang mengintegrasikan bobot pelatihan dari *AdaBoost* ke dalam struktur ELM. Model ini dirancang untuk menangani data tidak seimbang dengan menyesuaikan kontibusi masing-masing sampel berdasarkan distribusi bobot iteratif. Hasil eksperimen menunjukkan bahwa BWELM mampu meningkatkan akurasi kelas minoritas secara signifikan dibandingkan ELM standar. Studi ini menjadikan pondasi utama dalam pengembangan model *Hybrid* yang

terdiri dari BWELM ditujukan untuk klasifikasi sentimen dengan ketidakseimbangan distribusi kelas. Pengembangan pendekatan tersebut, Liu dkk. (2017) mengusulkan *Objective Cost-Sensitive* BWELM (OCS-BWELM) yang mengoptimalkan fungsi *cost* berdasarkan distribusi aktual data. Tidak hanya memperkuat akurasi, model ini juga mengurangi ketergantungan terhadap parameter subjektif menjadi lebih adaptif dalam menangani variasi distribusi kelas. Ganesan (2024) menerapkan konsep *AdaBoost*-WELM dalam domain pengenalan citra SAR, dan membuktikan bahwa kombinasi ini tetap efektif dalam menghadapi ketidakseimbangan data antara objek *wake* dan *sea clutter*. Meskipun berbasis visual, pendekatan ini menunjukkan bahwa BWELM fleksibel untuk berbagai jenis data termasuk data teks seperti ulasan aplikasi. Kemudian, Cheng dkk. (2020) mengembangkan *Boosting Label Weighted* ELM (BLW-ELM) untuk menangani data *multi-label* dengan ketidakseimbangan ekstrim. Model ini menyesuaikan bobot label secara iteratif melalui *boosting* dan menunjukkan peningkatan generalisasi tanpa eksplorasi eksplisit distribusi data. Keempat studi tersebut menegaskan bahwa BWLEM dan variannya sangat relevan digunakan dalam kasus klasifikasi, terutama saat menghadapi data dengan disrtibusi kelas yang tidak seimbang.

Sementara itu, penggunaan algoritma *Random Forest* juga dominan dalam penelitian analisis sentimen. Studi Kartika dkk. (2019) membandingkan performa *Random Forest* dan SVM dalam klasifikasi ulasan produk dari *Flipkart* dan menemukan bahwa *Random Forest* memberikan akurasi tertinggi sebesar 97%, sekaligus menunjukkan ketahanan terhadap *overfitting*. Padhy dkk. (2024) meneliti klasifikasi sentimen pada Twitter menggunakan pendekatan *word count vectorization* dan *majority voting*, menemukan bahwa *Random Forest* memiliki AUC 0,96 menjadikannya algoritma yang kuat dalam menangani data pendek dan *sparsity* tinggi. Dalam konteks ini, *Random Forest* terbukti unggul karena kemampuannya menangani data teks berdimensi tinggi serta kestabilannya pada variasi data. Di sisi lain, Feng (2019) mengeksplorasi integrasi *boosting* dalam klasifikasi sentimen dengan mengombinasikan *AdaBoost* dan *Multinomial Naïve Bayes* untuk data Weibo. Meskipun tidak menggunakan ELM, penelitian ini mengilustrasikan bagaimana *boosting* meningkatkan performa klasifikasi

sederhana, terutama dalam menangani sentimen biner dan multi-kelas. Demikian pula, Prabhakar dkk. (2019) menerapkan pendekatan modifikasi bahwa kombinasi *boosting* dan *bagging* dapat meningkatkan akurasi, walaupun masih bergantung pada fitur manual dan pengolahan dalam Bahasa Inggris.

Tren penggabungan beberapa model dalam pendekatan *Hybrid* melalui teknik *stacking* menunjukkan potensi peningkatan akurasi dan stabilitas prediksi. Studi van de Kamp (2022) mengembangkan model *Hybrid* dengan menggunakan *XGBoost* untuk seleksi fitur dan *Random Forest* sebagai model prediksi akhir dalam konteks *nowcasting* pertumbuhan ekonomi AS. Kombinasi ini menunjukkan performa yang konsisten dengan MSE yang rendah, meskipun cukup sensitif terhadap parameter *tuning*. Dalam pendekatan yang serupa, Natras dkk. (2022) menggunakan *voting regressor* berbasis *Random Forest*, *AdaBoost*, dan *XGBoost* untuk memprediksi gangguan ionosfer, yang berhasil meningkatkan akurasi jangka pendek meski masih terkendala interpretabilitas dan sensitivitas *input*. Kedua studi tersebut memperlihatkan bahwa penggabungan model dalam kerangka *stacking* maupun *voting* mampu mengatasi kekurangan dari model individual dan memberikan prediksi yang lebih stabil serta akurat.

Dengan mempertimbangkan hasil dari berbagai studi terkait, penggunaan BWELM terkait studi oleh Li dkk. (2014), Liu dkk. (2017), Ganesan (2024), dan Cheng dkk. (2020), menunjukkan potensi mekanisme pembobotan dan penguatan adaptif melalui *boosting.* Di sisi lain, algoritma *Random Forest* telah menunjukkan performa solid dalam berbagai konteks klasifikasi sentimen, sebagaimana dalam studi oleh Karthika dkk. (2019), Feng (2019), Prabhakar dkk. (2019), dan Padhy dkk. (2024). Penggabungan kedua pendekatan ini dalam kerangka *Hybrid stacking* yang diilustrasikan oleh van de Kamp (2022) dan Natras dkk. (2022) menjadi startegi potensial yang mampu mengatasi keterbatasan masing-masing model sekaligus meningkatkan akurasi dan stabilitas prediksi. Oleh karena itu, penelitian yang mengembangkan model *Hybrid* dari BWELM dan *Random Forest* untuk analisis sentimen ulasan aplikasi dianggap sebagai pendekatan strategis dalam menghasilkan sistem klasifikasi yang tangguh dan adaptif terhadap kompleksitas distribusi data serta dinamika bahasa dalam teks ulasan.

2.2. Dasar Teori

2.2.1. Analisis Sentimen

Analisis sentimen merupakan proses komputasi untuk mengidentifikasi, mengekstrak, dan mengklasifikasikan opini dalam teks menjadi positif, negatif, atau netral (Awajan dkk., 2021). Metode ini juga dikenal sebagai *opinion mining* dan termasuk dalam *text mining*, yang mengubah teks menjadi representasi numerik agar dapat diklasifikasikan berdasarkan polaritas sentimen (Gao dkk., 2024). Tujuannya adalah mengenali kata atau frasa bermuatan emosional sebagai dasar penilaian sikap, namun tantangan muncul karena makna kata dapat bervariasi sesuai konteks (Ng dan Law, 2020). Oleh karena itu, analisis sentimen tidak hanya menilai polaritas, tetapi juga memperhatikan konteks emosional, opini, dan evaluasi dalam penyampaian pendapat.

2.2.2. *Web scraping*

Web scraping merupakan teknik otomatisasi pengambilan data dari situs *web* secara langsung melalui pemrosesan HTML tanpa API resmi (Kumar dan Sachdeva, 2020). Data seperti teks, gambar, dan tabel diekstrak lalu dikonversi ke format terstruktur seperti CSV atau JSON. Teknik ini banyak dimanfaatkan dalam *e-commerce* dan bisnis untuk mengumpulkan data harga, ulasan, dan tren pasar. Namun, perubahan struktur HTML situs dapat menyebabkan proses *scraping* gagal (Singrodia dkk., 2019). Dalam analisis sentimen, *web scraping* berperan penting dalam mengekstraksi teks dari web untuk diolah dengan *text mining* dan NLP (Anisha dkk., 2021).

2.2.3. *Pre-Processing*

Pre-processing merupakan tahap pengolahan data untuk menghilangkan variasi atau artefak yang tidak diinginkan (Mishra dkk., 2020), khususnya dalam analisis sentimen untuk membersihkan dan menormalisasi data teks agar hasil lebih akurat (Palomino dan Aider, 2022). Teks ulasan aplikasi seringkali tidak terstruktur dan memuat elemen tidak relevan seperti emotikon, *URL*, tanda baca, atau kata tidak baku (Karthika dkk., 2019). Pada penelitian ini, *pre-processing* dilakukan

untuk meningkatkan kualitas data melalui tahapan *labeling, text cleansing, case folding, normalization, tokenization, stopword removal,* dan *stemming*.

1. *Labeling Data* adalah proses memberikan label pada data teks, seperti kategori positif/negatif, agar dapat digunakan dalam pelatihan model *machine learning*. Pelabelan dapat dilakukan secara manual berdasarkan interpretasi manusia, terutama jika *dataset* tidak memiliki label bawaan (Özmen & Gündüz, 2025).
2. *Text Cleansing* merupakan tahap awal dalam *pre-processing* teks untuk membersihkan elemen-elemen yang tidak relevan seperti karakter khusus, URL, *hashtag*, emoji, dan *tag* HTML, sehingga teks menjadi lebih rapi dan siap dianalisis (Padhy dkk., 2024).
3. *Case Folding* adalah proses mengubah seluruh huruf menjadi huruf kecil agar kata-kata yang sama tidak dianggap berbeda hanya karena perbedaan kapitalisasi, sehingga meningkatkan konsistensi dan akurasi analisis (Elistiana dkk., 2023).
4. *Normalization* bertujuan menyederhanakan format teks dengan menghapus karakter khusus, tanda baca, dan angka, sehingga hanya menyisakan huruf dan spasi untuk menjaga konsistensi data (Zheng, 2024).
5. *Tokenization* adalah proses memecah teks menjadi bagian-bagian kecil yang disebut token. Ini mempermudah pemahaman struktur kalimat dan pelabelan kata seperti kata benda atau kata sifat dalam analisis sentimen (Zheng, 2024).
6. *Stopword Removal* dilakukan untuk menghapus kata-kata umum yang tidak memiliki makna penting, seperti “itu”, “dan”, atau “sebuah” menggunakan pustaka seperti NLTK agar fokus analisis hanya pada kata-kata bermakna (Zheng, 2024).
7. *Stemming* adalah proses mengubah kata berimbuhan menjadi bentuk dasarnya, seperti “menggunakan” menjadi “guna”, untuk menyederhanakan variasi kata yang memiliki makna sama dan meningkatkan efektivitas analisis (Salman dan Al-Jawher, 2024).

2.2.4. *Feature Extraction*

Feature extraction dalam analisis sentimen penting untuk mengubah teks mentah menjadi representasi numerik yang dapat diproses oleh model *machine*

learning (Ahuja dkk., 2019). Proses ini membantu mengurangi kompleksitas data sambil mempertahankan informasi relevan, sehingga meningkatkan efisiensi dan akurasi (Gumaei dkk., 2019). Karena teks mentah berukuran besar sulit diproses secara langsung, TF-IDF menjadi salah satu metode populer. TF-IDF memungkinkan model mengidentifikasi kata-kata yang paling relevan terhadap sentimen (Salman dan Al-Jawher, 2024).

TF-IDF *(Term Frequency-Inverse Document Frequency)* adalah metode penting dalam ekstraksi fitur untuk analisis sentimen, yang mengukur bobot kata berdasarkan frekuensi kemunculannya di suatu dokumen (TF) dan kelangkaannya di seluruh korpus (IDF) (Almuayqil dkk., 2022). TF dihitung menggunakan Persamaan 2.1 dengan membagi jumlah kemunculan kata dalam dokumen dengan total kata dalam dokumen.

$$TF(t,d) = \frac{f_{t,d}}{\Sigma_k f_{k,d}} \tag{2.1}$$

Dengan:

t = Term atau kata yang akan dihitung frekuensinya

d = Dokumen

$f_{t,d}$ = Jumlah kemunculan term t dalam dokumen d

$\Sigma_k f_{k,d}$ = Jumlah seluruh term dalam dokumen d (total kata)

Sedangkan IDF dihitung menggunakan Persamaan 2.2, dengan logaritma dari rasio total dokumen terhadap jumlah dokumen yang mengandung kata tersebut.

$$IDF(t) = log\left(\frac{N}{df(t)}\right) \tag{2.2}$$

Dengan:

N = Jumlah total dokumen dalam korpus

$df(t)$ = Jumlah dokumen yang mengandung term t

Gabungan keduanya menghasilkan nilai TF-IDF dengan menggunakan Persamaan 2.3, yang menunjukkan seberapa penting sebuah kata dalam konteks tertentu dan jarangnya kata itu muncul di dokumen lain.

$$TF - IDF(t,d) = Tf(t,d) \times IDF(t) \tag{2.3}$$

Dengan:

$TF(t,d)$ = Frekuensi relatif kemunculan kata dalam dokumen

$df(t)$ = Bobot pembeda kata berdasarkan kelangkaan antar dokumen

Setiap dokumen direpresentasikan sebagai vektor numerik TF-IDF, misalnya berdimensi 5.000 jika terdapat 5.000 kata unik. Matriks ini menjadi *input* bagi BWELM dan *Random Forest*, dimana *Random Forest* menangani pola *non-linear*, sementara BWELM mempercepat pelatihan dan menyeimbangkan bobot antar kelas untuk membedakan sentimen.

2.2.5. *Hyperparameter Tuning*

Hyperparameter tuning merupakan proses penting untuk menentukan kombinasi parameter optimal guna memaksimalkan performa model, meskipun tidak dipelajari langsung dari data (Ilemobayo dkk., 2024). Pada model *Hybrid* BWELM dan *Random Forest*, *tuning* merupakan bagian penting karena masing-masing memiliki parameter sensitif seperti jumlah neuron pada BWELM serta jumlah pohon dan kedalaman maksimum *Random Forest* (Almuayqil dkk., 2022; van de Kamp, 2022). *Tuning* ini membantu BWELM menyesuaikan kompleksitas opini (Liu dkk., 2017), dan meningkatkan stabilitas prediksi *Random Forest* (Karthika dkk., 2019). Adapun *hyperparameter tuning* yang digunakan pada penelitian ini yaitu:

2.2.5.1. *Optuna Optimization*

Optuna adalah kerangka kerja *open-source* untuk optimasi *hyperparameter* berbasis *Bayesian Optimization* yang dirancang dengan prinsip *define-by-run*. Pendekatan ini memungkinkan ruang pencarian *hyperparameter* dibentuk secara dinamis selama proses eksekusi fungsi objektif, berbeda dengan pendekatan *define-and-run* yang lebih kaku dan statis (Akiba dkk., 2019). *Optuna* menggunakan algoritma *Tree-structured Parzen Estimator* (TPE) sebagai strategi pencarian utama. TPE bekerja dengan memodelkan distribusi probabilitas dari fungsi objektif berdasarkan hasil-hasil percobaan sebelumnya, kemudian memaksimalkan *Expected Improvement* (EI) untuk memiliki kombinasi parameter berikutnya yang paling menjanjikan (Shekhar dkk., 2021). Secara sistematis, TPE menggantikan proses eksplorasi *brute-force* dengan model probabilistik menggunakan formulasi berdasarkan Persamaan 2.4 berikut (Akiba dkk., 2019):

$$P(f|D_t) \propto P(D_t|f) \cdot P(f) \quad (2.4)$$

Dengan:

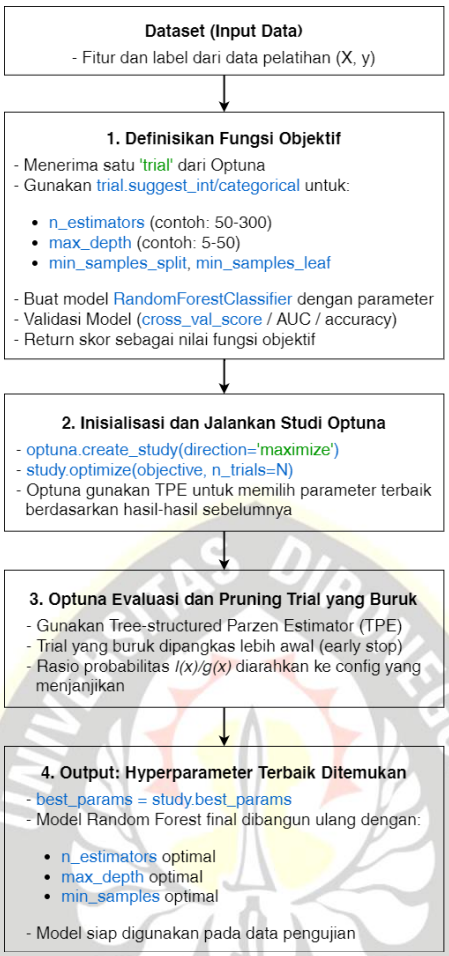
$D_t = \{(x_1, f(x_1)), \dots, (x_t, f(x_t))\}$ = Data dari hasil percobaan (*trial)* sebelumnya

$f(x)$ = Fungsi objektif yang dievaluasi terhadap kombinasi *hyperparameter* x

$P(f)$ = Distribusi prior dari nilai fungsi

TPE selanjutnya memisahkan pencarian ke dalam dua distribusi: (1) $l(x)$ untuk parameter dengan skor terbaik, dan $g(x)$ untuk parameter lainnya, (2) lalu memilih x yang emmaksimalkan rasio $l(x)/g(x)$ yang secara efektif mengarahkan pencarian ke area parameter yang menjanjikan.

Dalam penerapannya pada model *Random Forest, Optuna* dimanfaatkan untuk menyusun dan mengevaluasi berbagai konfigurasi *hyperparameter* secara otomatis dan efisien. *Hyperparameter* seperti *`n_estimators`, `max_depth`, `min_samples_split`,* dan *`min_samples_leaf`* disarankan nilainya oleh *Optuna* melalui fungsi *`trial.suggest_int()`*, kemudian digunakan untuk membangun model *Random Forest* (Xiao dkk., 2024). Model ini dievaluasi menggunakan teknik validasi silang *(cross-validation)* dengan metrik AUC, yang menjadi nilai fungsi objektif. Nilai tersebut digunakan oleh TPE untuk memperbarui distribusi probabilistik dan menentukan strategi pencarian di percobaan selanjutnya (Suryadi dkk., 2024). Adapun skema penggunaan *Optuna Optimization* pada *Random Forest* diilustrasikan oleh Gambar 2.1.



Gambar 2.1 Skema Penggunaan *Optuna* pada *Random Forest*

Berdasarkan Gambar 2.1, TPE *Algorithm* digunakan sebagai pendekatan probabilistik untuk menentukan nilai parameter terbaik berdasarkan distribusi probabilitas hasil *trial* sebelumnya (Akiba dkk., 2019). *Cross-validation* (CV) digunakan untuk menilai performa model selama proses *tuning*, misalnya dengan skor AUC atau akurasi (Xiao dkk., 2024; Suryadi dkk., 2024). *Purning strategy* secara otomatis menghentikan percobaan yang memiliki skor rendah dan meningkatkan efisiensi *tuning* (Joy dan Selvan, 2022). Kemudian, model final dengan *hyperparameter* optimal dilatih ulang pada seluruh data pelatihan sebelum diujikan (Rahmi dkk., 2024).

Optuna digunakan sebagai metode *hyperparameter optimization* karena lebih efisien dan adaptif dibandingkan dengan *Grid Search* maupun *Random Search*. Bebeda dengan *Grid Search* atau *Random Search* yang bersifat statis,

Optuna menggunakan algoritma TPE berbasis *Bayesian* untuk mengarahkan pencarian ke kombinasi parameter yang paling menjanjikan, berdasarkan hasil tiral sebelumnya (Akiba dkk., 2019). Selain itu, *Optuna* dilengkapi fitur *pruning* yang dapat menghentikan percobaan yang tidak potensial lebih awal, sehingga menghemat waktu dan sumber daya komputasi (Joy dan Selvan, 2022). Dengan pendekatan *define-by-run* yang fleksibel, *Optuna* memungkinkan *tuning* dilakukan secara dinamis, sangat cocok untuk ruang parameter yang kompleks dan saling bergantung. Secara eksplisit, penggunaan *Optuna* pada *Random Forest* bertujuan untuk menemukan kombinasi *hyperparameter* yang memberikan performa terbaik pada *dataset* tertentu. Proses ini tidak hanya meningkatkan akurasi atau AUC model, tetapi juga menghindari praktif *tuning* manual yang memakan waktu dan rawan *overfitting*. Dalam studi oleh Joy dan Selvan (2022) serta Rahmi dkk. (2024), penerapan *Optuna* pada *Random Forest* terbukti mampu meningkatkan kinerja klasifikasi secara signifikan dibandingkan penggunaan parameter *default*, bahkan mengungguli pendekatan *tuning* tradisional seperti *Grid Search* dan *Random Search.*

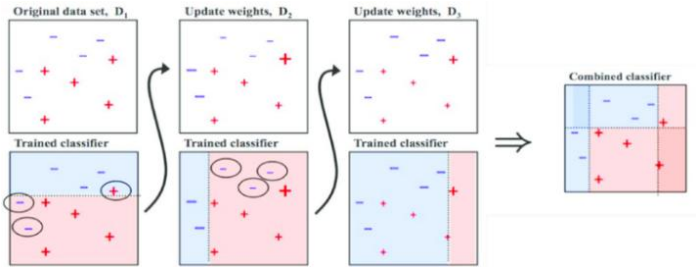
2.2.6. *Boosting Weighted Extreme Learning Machine* (BWELM)

BWELM merupakan pengembangan dari *Weighted* ELM (WELM) yang memanfaatkan pendekatan *AdaBoost* dengan WELM sebagai pengklasifikasi utama (Raghuwanshi dan Shukla, 2019). Meskipun waktu pelatihan dan pengujian BWELM lebih lama, performanya lebih baik karena tidak bergantung pada pembobotan tetap. Namun, menurut Jiang dkk. (2020), BWELM masih memiliki keterbatasan dalam menghadapi data dengan variabilitas tinggi atau pola kompleks, terutama pada data teks, meskipun tetap unggul dalam kecepatan komputasi dan generalisasi yang baik.

2.2.6.1. Algoritma *AdaBoost*

AdaBoost (Adaptive Boosting) merupakan algoritma *ensemble learning* yang dirancang untuk meningkatkan akurasi klasifikasi dengan menggabungkan sejumlah *weak classifier* berakurasi rendah menjadi satu *strong classifier* yang lebih

andal (Azizah dkk., 2023; Zulfikri dkk., 2019). Ilustrasi alur kerja algoritma *AdaBoost* dapat dilihat pada Gambar 2.2.



Gambar 2.2 Alur Kerja *AdaBoost*

Proses pelatihan algoritma *AdaBoost* dilakukan secara iteratif, dimana setiap *weak learner* dilatih berdasarkan distribusi bobot yang disesuaikan dari iterasi sebelumnya (Li dkk., 2014). Sampel yang salah klasifikasi akan diberi bobot lebih tinggi agar model selanjutnya lebih fokus memperbaikinya (Feng, 2019). Pendekatan ini meningkatkan akurasi secara bertahap dengan menekankan pada data yang sulit diklasifikasikan (Prabhakar dkk., 2019). Proses diawali dengan inisialisasi bobot yang sama untuk seluruh sampel pelatihan yang ditunjukkan pada Persamaan 2.5:

$$D_1(x_i) = \frac{1}{N} \tag{2.5}$$

Iterasi *boosting* untuk $t = 1, 2, \ldots, T$ diawali dengan melatih *weak learner* $h_t(x)$ menggunakan distribusi bobot D_t sebagai penyesuaian pada data pelatihan (Feng, 2019; Natras dkk., 2022). Kemudian, menghitung tingkat kesalahan pada model dengan menggunakan Persamaan 2.6:

$$\varepsilon_t = \sum_{i=1}^{N} D_t(x_i) \cdot \| \, (h_t(x_i) \neq y_i) \tag{2.6}$$

Dimana ‖ adalah fungsi indikator yang bernilai 1 jika prediksi salah dan 0 jika benar (Feng, 2019; Salaman dan Al-Jawher, 2024). Lalu menghitung bobot model dalam *ensemble* ditunjukkan oleh Persamaan 2.7:

$$\alpha_t = \frac{1}{2} ln \left(\frac{1-\varepsilon_t}{\varepsilon_t} \right) \tag{2.7}$$

Bobot α_t menunjukkan seberapa kuat model h_t dalam *ensemble* akhir (Feng, 2019; Natras dkk., 2022). Langkah selanjutnya adalah memperbarui bobot sampel dengan menggunakan Persamaan 2.8:

$$D_{t+1}(x_i) = \frac{D_t(x_i) \cdot exp(-\alpha_t \cdot y_i \cdot h_t(x_i))}{Z_t} \quad (2.8)$$

Z_t merupakan konstanta normalisasi agar jumlah semua bobot tetap 1 (Feng, 2019). Adapun model akhir prediksi dilakukan melalui voting tertimbang, dimana fungsi ini menggabungkan semua *weak learners* dengan bobotnya masing-masing (Prabhakar dkk., 2019; Natras dkk. 2022), dengan menggunakan Persamaan 2.9:

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x)) \quad (2.9)$$

Dengan:

$D_t(x_i)$ = Bobot distribusi untuk data ke-i pada iterasi ke-t

ε_t = Tingkat kesalahan klasifikasi model h_t

α_t = Bobot dari model ke-t dalam *ensemble*

$h_t(x)$ = Prediksi dari *weak learner* ke-t

y_i = Label sebenarnya dari sampel x_i

Z_t = Faktor normalisasi distribusi bobot

$H(x)$ = Prediksi akhir dari *AdaBoost*

2.2.6.2. Model BWELM

BWELM merupakan pengembangan lanjutan dari metode ELM, yang dirancang untuk mengatasi permasalahan ketidakseimbangan kelas *(class imbalance)* dalam tugas klasifikasi, termasuk pada domain analisis sentimen. ELM sendiri adalah metode pembelajaran untuk jaringan saraf dengan satu lapisan tersembunyi (*Single-hidden Layer Feedforward Neural Network* / SLFN), yang memiliki keunggulan utama pada kecepatan pelatihan dan kemampuan generalisasi yang baik. ELM menetapkan bobot *input* dan bias secara acak, kemudian menghitung bobot *output* secara analitik melalui solusi *least-squares* menggunakan *invers Moore-Penrose* (Li dkk., 2014). Model dasar ELM dirumuskan dalam Persamaan 2.10 berikut:

$$\beta = H^{\dagger}T \quad (2.10)$$

Dengan:

H = Matriks *output* dari lapisan tersembunyi

T = Vektor target

β = Bobot *output* dari jaringan

$H^{\dagger}$ = *Invers Moore-Penrose* dari H

Namun, pada data tidak seimbang, model ELM standar cenderung bias terhadap kelas mayoritas. Untuk itu, dikembangkan model *Weighted* ELM (WELM) yang memperkenalkan matriks bobot diagonal W untuk memberikan bobot lebih besar pada sampel dari kelas minoritas dan mengurangi pengaruh dari kelas mayoritas (Li dkk., 2014; Wang dkk., 2018). Formula WELM ditunjukkan oleh Persamaan 2.11:

$$\beta = (H^T W H + CI)^{-1} H^T W T \tag{2.11}$$

Dengan:

W = Matriks diagonal bobot sampel pelatihan

C = Parameter regularisasi

Untuk lebih meningkatkan performa pada data yang sangat tidak seimbang, WELM kemudian digabungkan dengan metode *boosting* dengan membentuk BWELM. Dalam pendekatan ini, WELM dijadikan sebagai *weak learner* dalam kerangka kerja *AdaBoost*. Beberapa WELM dilatih secara iteratif, dimana setiap iterasi menyesuaikan bobot sampel berdasarkan kesalahan klasifikasi pada iterasi sebelumnya (Li dkk., 2014). Inisialisasi bobot distribusi dilakukan secara asimetris sebagaimana ditunjukkan pada Persamaan 2.12:

$$D_1(x_i) = \frac{1}{m.\#t_i} \tag{2.12}$$

Dengan:

m = Jumlah kelas

$\#t_i$ = Jumlah sampel pada kelas t_i

Setiap iterasi t dalam *boosting* melibatkan pelatihan model Ω_t menggunakan bobot $W_t = diag(D_t(x_i))$. Kesalahan model dihitung menggunakan Persamaan 2.13:

$$\varepsilon_t = \sum_{i=1}^{N} D_t(x_i) \cdot \|(\Omega_t(x_i) \neq y_i) \tag{2.13}$$

Bobot keputusan model ke-t dihitung berdasarkan Persamaan 2.14:

$$\alpha_t = \frac{1}{2} ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \quad (2.14)$$

Distribusi bobot diperbarui menggunakan Persamaan 2.15:

$$D_{t+1}(x_i) = \frac{D_t(x_i) \cdot exp(-\alpha_t \cdot \|(\Omega_t(x_i)=y_i))}{Z_t} \quad (2.15)$$

Dengan:

Z_t = Faktor normalisasi agar total distribusi tetap 1

Prediksi akhir ditentukan melalui *voting* berbobot dari seluruh model WELM yang telah dilatih, seperti yang ditunjukkan pada Persamaan 2.16:

$$f(x) = arg \max_k \sum_{t=1}^{T} \alpha_t \cdot \| (F_t(x) = k) \quad (2.16)$$

Dengan:

$f(x)$ = Label hasil klasifikasi akhir

$F_t(x)$ = Prediksi dari model ke-t

α_t = Bobot model berdasarkan akurasi

Dalam model BWELM, peran algoritma *AdaBoost* sangat penting sebagai mekanisme penguat adaptif yang secara dinamis memodifikasi distribusi bobot sampel berdasarkan performa klasifikasi pada iterasi sebelumnya (Liu dkk., 2017). Dengan strategi ini, BWELM secara konsisten memfokuskan pelatihan pada sampel-sampel yang paling sulit diklasifikasikan yang umumnya berasal dari kelas minoritas. Hal ini menjadikan BWELM sangat efektif dalam menghadapi tantangan ketidakseimbangan kelas, terutama dalam konteks analisis sentimen dimana opini negatif atau netral sering kali menjadi kelas dengan representasi data yang rendah dan lebih sulit diprediksi (Wang dkk., 2018).

Penggunaan *AdaBoost* dalam BWELM memberikan beberapa keuntungan dan manfaat utama, yaitu:

1. Memperkuat fokus pada kelas minoritas dengan memberikan bobot lebih besar kepada sampel dari kelas minoritas atau sampel yang sebelumnya salah klasifikasi, BWELM secara adaptif mengarahkan model untuk memperbaiki kelemahan dalam prediksi (Li dkk., 2014; Liu dkk., 2017).

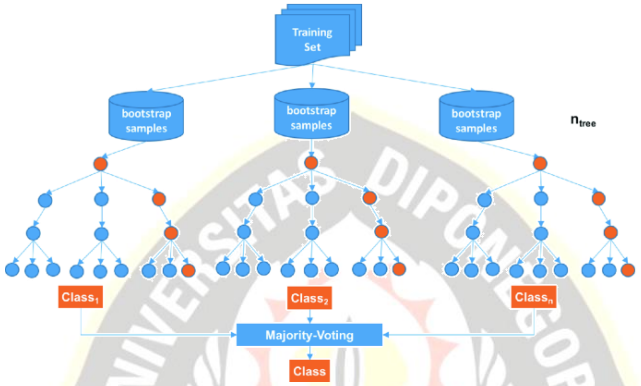
2. Menjaga keseimbangan klasifikasi dengan pembaruan bobot yang terkontrol dan selektif, tidak seperti pendekatan yang terlalu menekankan pada kelas minoritas hingga mengorbankan performa mayoritas (Liu dkk., 2017).
3. Fleksibel untuk *multi-label* dan *multi-class* karena mampu menyesuaikan pembobotan secara otomatis tanpa harus menghitung distribusi kelas yang kompleks secara eksplisit, menjadikannya algoritma yang *robust* dan adaptif (Cheng dkk., 2020).
4. Efisiensi dan ketepatan klasifikasi dalam implementasi praktis seperti klasifikasi citra, BWELM menunjukkan performa unggul dalam hal akurasi dan penurunan tingkat *false alarm* jika dibandingkan dengan metode klasik, terutama berkat peran *AdaBoost* yang terus menyempurnakan fokus pelatihan pada iterasi berikutnya (Ganesan, 2024).

Secara keseluruhan, keunggulan BWELM dibandingkan ELM dan WELM standar terletak pada kemampuannya untuk menghasilkan klasifikasi yang lebih seimbang, tanpa mengorbankan akurasi pada kelas mayoritas. BWELM juga menunjukkan performa yang kompetitif, bahkan sering kali lebih unggul dibandingkan metode konvensional ketika dikombinasikan dengan teknik representasi fitur seperti TF-IDF (Raghuwanshi dan Shukla, 2019). Dalam berbagai studi, BWELM berhasil meningkatkan metrik penting seperti *F1-score* dan *G-mean*, khususunya pada prediksi kelas minoritas menjadikannya sebagai salah satu model yang sangat layak digunakan dalam analisis sentimen dengan distribusi data yang tidak merata.

2.2.7. *Random Forest*

Random Forest merupakan algoritma *ensemble* berbasis pohon keputusan yang efektif untuk klasifikasi dan regresi, termasuk analisis sentimen (Iranzad dan Liu, 2024). Algoritma ini membangun banyak pohon melalui *bootstrap sampling* dari subset data acak (Karthika dkk., 2019), dengan pemilihan sebagian fitur saat pemisahan *node* untuk mencegah *overfitting* (Setiawan dkk., 2022). Hasil prediksi diperoleh melalui *voting* mayoritas atau rata-rata (Padhy dkk., 2024), dan akurasi dapat ditingkatkan dengan *tuning* parameter serta *feature engineering* (Zheng, 2024). Fitur penting dalam *Random Forest* mencakup: (1) *Bagging*, yaitu

penggabungan prediksi dari pohon-pohon berbeda (Almuayqil dkk., 2022); (2) *Voting*, baik *hard* (berdasarkan suara terbanyak) maupun *soft* (berdasarkan probabilitas) untuk penentuan hasil akhir (Padhy dkk., 2024); (3) *Feature randomness* untuk menghindari dominasi fitur tertentu (Karthika dkk., 2019); serta (4) Kemampuan integrasi dengan teknik *stacking* dalam model *Hybrid* meski bukan fitur *default* (Zheng, 2024). Gambar 2.3 merupakan alur kerja dari algoritma *Random Forest*.



Gambar 2.3 Alur Algoritma *Random Forest*

Pada klasifikasi, *Random Forest* memberikan prediksi akhir berdasarkan voting mayoritas dari semua pohon keputusan dengan menggunakan Persamaan 2.17 berikut (Zheng, 2024):

$$\hat{y} = mode\{h_1(x), h_2(x), \dots, h_T(x)\} \quad (2.17)$$

Dengan:

$\hat{y}$ = Prediksi akhir

$h_t(x)$ = Prediksi dari pohon ke-t

T = Jumlah total pohon

Penggunaan *Random Forest* efektif untuk klasifikasi sentimen pada teks seperti ulasan dan komentar (Almuayqil dkk., 2022), dengan terlebih dahulu mengubah teks menjadi representasi numerik seperti TF-IDF (Setiawan dkk., 2022). Menggunakan banyak pohon dan fitur acak (Karthika dkk., 2019), algoritma ini mampu mengenali pola sentimen secara akurat (Zheng, 2024), mengatasi ketidakseimbangan kelas (Padhy dkk., 2024), serta tahan terhadap *overfitting* dan *noise* berkat teknik *bagging* dan *feature randomness* (van de Kamp, 2022).

2.2.8. Model *Hybrid*

Model *Hybrid* dalam *machine learning* menggabungkan dua atau lebih algoritma untuk meningkatkan akurasi, stabilitas, dan generalisasi, terutama dalam menangani data teks yang tidak terstruktur dan tidak seimbang seperti pada analisis sentimen (Alawi dan Bozkurt, 2024). Pendekatan ini memungkinkan kelemahan satu metode ditutupi oleh kelebihan metode lain (Lilhore dkk., 2021). Penelitian ini mengusulkan kombinasi *Boosting Weighted Extreme Learning Machine* (BWELM) dan *Random Forest*, dimana BWELM menggunakan mekanisme *boosting* untuk menyesuaikan bobot pelatihan secara adaptif (Feng, 2019; Liu dkk., 2017), sementara *Random Forest* efektif menangani data berdimensi tinggi dan mencegah *overfitting* melalui *voting* dan pemilihan fitur acak (Karthika dkk., 2019; Zheng, 2024).

Penelitian sebelumnya menunjukkan efektivitas pendekatan *Hybrid*, seperti keberhasilan *AdaBoost* dalam meningkatkan akurasi pada data Weibo (Feng, 2019), serta peningkatan *F1-score* dalam analisis sentimen Twitter (Prabhakar dkk., 2019). Model *XGBoost-Random Forest* (van de Kamp, 2022) dan *voting regressor* gabungan *Random Forest-AdaBoost*-XGBoost (Natras dkk., 2022) juga menunjukkan kinerja unggul pada data kompleks. Berdasarkan temuan ini, kombinasi BWELM dan *Random Forest* digunakan untuk meningkatkan akurasi klasifikasi sentimen dalam ulasan aplikasi.

2.2.8.1. Pendekatan *Stacking*

Pendekatan *stacking* atau *stacked generalization* merupakan metode penggabungan prediksi dari beberapa model *(base learners)* pada level pertama (level-0) dan memanfaatkan model *meta (meta-classifier)* pada level kedua (level-1) untuk membuat prediksi akhir yang lebih akurat (Sharma dkk., 2023). Secara umum, *stacking* bertujuan untuk mengurangi kesalahan generalisasi *(generalization error)* dari model dasar, dan memanfaatkan keunggulan individual dari setiap *base learner* untuk memperkuat prediksi kolektif (Mahendran dkk., 2020).

Tahapan proses dalam pendekatan *stacking* menurut Sharma dkk. (2023) dan Mahendran dkk. (2020) adalah:

1. Pelatihan *Base Learners* (Level-0): Beberapa model dasar dilatih pada data pelatihan. Model-model ini menghasilkan prediksi probabilistik terhadap kelas target (dalam kasus klasifikasi).
2. Konstruksi *Meta-Dataset*: *Output* dari seluruh *base learner* dijadikan sebagai fitur baru *(meta-fitur)* untuk membenruk *dataset* baru *(meta-dataset)*. Label kelas dari *dataset* asli tetap digunakan.
3. Pelatihan *Meta-Classifier* (Level-1): Model *meta (meta-classifier)* dilatih menggunakan *meta-dataset* untuk mempelajari pola dari prediksi yang dihasilkan oleh *base learners*.
4. Prediksi Final: Saat melakukan prediksi terhadap data baru, data tersebut diperoses oleh *base learners*, kemudian *output*-nya menjadi *input* bagi *meta-classifier* untuk menghasilkan keputusan akhir.

Perhitungan prediksi dalam pendekatan *stacking* dapat dituliskan secara matematis sebagaimana Persamaan 2.18, 2.19, dan 2.20 berikut:

$$P_{base}(k) = h_{base}(x) \quad (2.18)$$

$$P_{final}(k) = f_{meta}\left(P_{base}(k)\right) \quad (2.19)$$

$$y_{final} = arg\max_{k} P_{final}(k) \quad (2.20)$$

Dengan keterangan:

x = Data *input* (misalnya hasil vektorisasi ulasan pengguna)

$h_{base}(x)$ = Fungsi prediksi dari *base learner*

$P_{base}(k)$ = Probabilitas bahwa instance x termasuk kelas k menurut *base learner*

f_{meta} = Fungsi prediksi dari *meta-classifier*

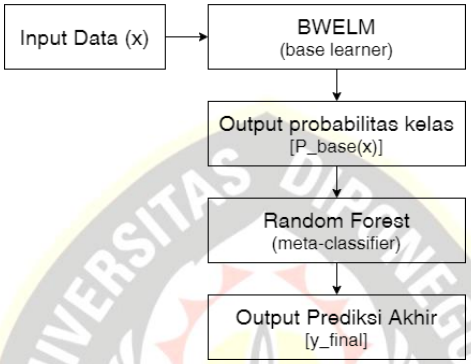
$P_{final}(k)$ = Probabilitas akhir untuk kelas k yang diprediksi oleh *meta-classifier*

y_{final} = Kelas dengan probabilitas tertinggi yang dipilih sebagai hasil akhir

Kelas dengan nilai $P_{final}(k)$ tertinggi akan dipilih sebagai *output* akhir dari model yang digabungkan menggunakan teknik *stacking*.

Pada penelitian ini, teknik *stacking* digunakan untuk membentuk model *Hybrid* dengan menggabungkan dua algoritma yaitu BWELM sebagai *base learner* dan *Random Forest* sebagai *meta-classifier*. Pemilihan kedua model ini didasarkan pada kelebihan masing-masing: (1) BWELM unggul dalam menangani masalah

ketidakseimbangan kelas, dengan pendekatan *boosting* dan penyesuaian bobot pada setiap *instance* pelatihan. BWELM menerapkan prinsip ELM yang cepat dan *generalizable*, ditambah dengan adaptasi bobot dari *boosting* seperti pada *AdaBoost* (Li dkk., 2014); (2) *Random Forest* dikenal memiliki *robustness* terhadap *noise* dan fitur berdimensi tinggi, serta memberikan performa yang konsisten pada berbagai jenis *dataset* (van de Kamp, 2022). Visualisasi dari alur *stacking* pada model *Hybrid* BWELM dan *Random Forest* dapat dilihat pada Gambar 2.4.



Gambar 2.4 Alur Implementasi Stacking pada Model

Alur implementasi teknik *stacking* pada model berdasarkan pada Gambar 2.4 adalah sebagai berikut:

1) Input Data dan *Preprocessing*

   Dataset terstruktur hasil *preprocessing* dan diolah menjadi fitur numerik dengan label kelas asli, kemudian dibagi menjadi data *training* dan *testing*.

2) Level-0: BWELM *(Base Learner)*

   BWELM dilatih pada data *training* yang menghasilkan *output* probabilistik untuk setiap *instance* x_i, kemudian model menghasilkan vektor probabilitas $P_{base}(x_i) = [p_1, p_2, \dots, p_K]$ dimana p_K adalah probabilitas bahwa x_i termasuk kelas ke-k dan $\sum_k p_k = 1$.

3) Level-1: *Random Forest (Meta-Classifier)*

   Input ke *Random Forest* merupakan *output* dari BWELM (probabilitas kelas) untuk seluruh *instance training* digunakan sebagai fitur baru *(meta-features)*, dengan target label tetap menggunakan label asli dari *dataset*. Kemudian, RF dilatih menggunakan *meta-dataset* yang terdiri dari fitur (vektor probabilitas

dari BWELM) dan label (label kelas asli). Proses belajar RF dengan mempelajari pola distribusi dari probabilitas yang merefleksikan keunikan keputusan BWELM. Karena RF adalah *non-linear classifier*, ia dapat mengenali pola-pola kompleks seperti probabilitas tinggi pada dua kelas sekaligus *(ambiguity)*, korelasi antar dimensi probabilitas yang tidak linier, dan pola probabilitas yang sering muncul dalam kelas tertentu *(overconfidence, underconfidence)*.

4) Prediksi Final

BWELM memproses dan memberikan vektor probabilitas kelas $P_{base}(x_{test})$, dimana vektor ini menjadi *input* ke RF. RF mengklasifikasikan *instance* berdasarkan aturan-aturan *decision tree* yang telah dipelajari dari pola-pola probabilitas sebelumnya. Kemudian menghasilkan *output* akhir berupa prediksi kelas y_{final}.

Manfaat dari arsitektur ini ialah BWELM menangani ketidakseimbangan dan memberikan probabilitas yang peka terhadap kelas minoritas, kemudian *Random Forest* belajar dari pola prediksi probabilistik dan menangani ketidakteraturan atau ambiguitas prediksi dari BWELM.

Keunggulan dari kombinasi BWELM + *Random Forest*, berdasarkan studi oleh Mahendran dkk. (2020) dan Sharma dkk. (2023) melalui pendekatan *stacking* dapat secara signifikan meningkatkan akurasi prediksi pada data yang kompleks dan berdimensi tinggi, termasuk data dengan ketidakseimbangan kelas dan kompeksitas linguistik tinggi seperti data teks atau ulasan pengguna. Kelebihan dari kombinasi ini antara lain; menggabungkan kecepatan dan efisiensi BWELM dalam mempelajari pola minoritas, memperkuat hasil prediksi melalui *ensemble Random Forest*, menurunkan risiko *overfitting* yang umum terjadi pada model tunggal, meningkatkan akurasi dan kestabilan sistem klasifikasi (Li dkk., 2014; van de Kamp, 2022).

2.2.9. Evaluasi Model

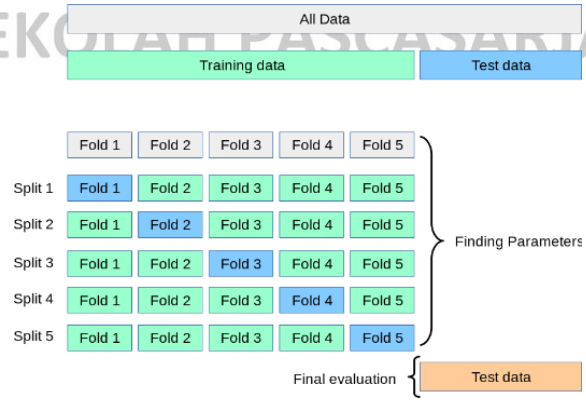
Evaluasi model dalam analisis sentimen bertujuan menentukan sejauh mana model dapat memberikan hasil prediksi yang akurat dan andal (Qiu, 2024), melalui

teknik validasi yaitu *cross-validation*, serta metrik evaluasi seperti akurasi, presisi, *recall*, *F1-score*, dan ROC AUC (Ghatora dkk., 2024).

2.2.9.1. Validasi

Validasi merupakan proses penting dalam *machine learning* yang digunakan untuk mengukur kemampuan dalam memprediksi data yang belum pernah dilihat sebelumnya (Qiu, 2024). Validasi bertujuan untuk memastikan bahwa model tidak hanya menghafal data pelatihan *(overfitting)*, tetapi juga dapat melakukan generalisasi yang baik terhadap data baru (Ilemobayo dkk., 2024). Validasi juga berperan krusial dalam menilai kualitas dan kestabilan model gabungan seperti pada model *Hybrid* BWELM dan RF, terutama dalam menangani variasi karakteristik data seperti ketidakseimbangan label dan kompleksitas sentimen pada ulasan aplikasi (van de Kamp 2022; Natras dkk., 2022).

Salah satu metode validasi yang umum digunakan adalah *k-fold cross-validation*. Penggunaan *k-fold cross-validation* juga telah diterapkan dalam berbagai penelitian, baik untuk mengukur performa model maupun untuk membentuk *ensemble* dan estimasi ketidakpastian prediksi (Dutschmann dkk., 2023). Metode ini membagi *dataset* menjadi *k* bagian atau *fold* dengan ukuran yang kurang lebih sama. Model kemudian dilatih pada *k-1* bagian data dan diuji pada bagian yang tersisa. Proses ini diulang sebanyak *k* kali, dimana setiap bagian data secara bergiliran menjadi data uji. Hasil evaluasi dari tiap iterasi kemudian dirata-ratakan untuk memperoleh estimasi performa model yang stabil dan representatif (Qiu, 2024; van de Kamp, 2022). Metode *k-fold* divisualisasikan pada Gambar 2.5.



Gambar 2.5 *K-Fold Cross-Validation*

Adapun perhitungan dari *k-fold cross-validation* berdasarkan pada Persamaan 2.21 berikut:

$$CV_{error} = \frac{1}{k}\sum_{i=1}^{k} Error_i \quad (2.21)$$

dengan:

k = Jumlah lipatan data

$Error_i$ = Nilai kesalahan pada *fold* ke-i

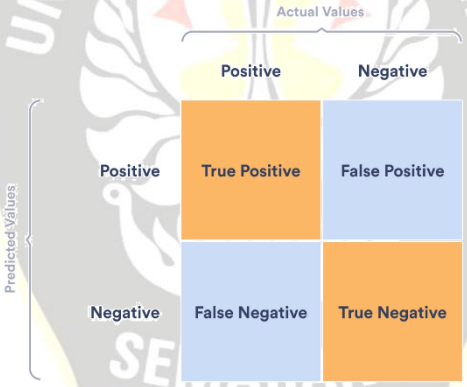
Metode *k-fold cross-validation* memberikan keunggulan dalam hal efisiensi penggunaan data, karena seluruh data akan digunakan sebagai data pelatihan dan juga sebagai data pengujian. Hal ini sangat membantu dalam konteks BWELM yang memerlukan adaptabilitas terhadap data yang tidak seimbang, dan juga pada *Random Forest* yang mengandalkan kekuatan *ensemble* untuk meningkatkan ketahanan terhadap variasi subset data (Lilhore dkk., 2023; Li dkk., 2014). Berikut merupakan tahapan pelaksanaan *k-fold cross-validation* secara sistematis:

1) Membagi *dataset* menjadi *k* bagian *(fold)* yang seimbang dan acak.
2) Melatih model pada *k-1* bagian data sebagai data pelatihan.
3) Menguji model pada satu *fold* yang tersisa sebagai data pengujian.
4) Mengulangi proses pelatihan dan pengujian sebanyak *k* kali sehingga setiap *fold* digunakan sekali sebagai data uji.
5) Menghitung rata-rata performa dari seluruh iterasi untuk memperoleh skor akhir, seperti akurasi dan *F1-score*.
6) Menggunakan hasil validasi untuk mengatur bobot pada model-model dalam *ensemble* atau menyesuaikan parameter model, misalnya pada model *Hybrid* BWELM + RF.

Melalu tahapan tersebut, *k-fold cross-validation* tidak hanya memberikan estimasi performa model yang akurat dan menyeluruh, tetapi juga memperkuat kemampuan model untuk melakukan generalisasi terhadap data baru dengan lebih baik (Dutschmann dkk., 2023; Ilemobayo dkk., 2024).

2.2.9.2. Metrik Evaluasi

Evaluasi menilai akurasi model dalam prediksi, khususnya pada tugas kompleks seperti analisis sentimen (Karthika dkk., 2019). Salah satu pendekatan utamanya adalah *Confusion Matrix*, merupakan tabel yang digunakan untuk mengevaluasi kinerja model klasifikasi dengan membandingkan hasil prediksi dengan nilai aktual. Tabel ini terdiri dari empat komponen utama, yaitu *True Positive* (TP) yang menunjukkan data positif yang diprediksi benar sebagai positif, *True Negative* (TN) yang menunjukkan data negatif yang diprediksi benar sebagai negatif, *False Positive* (FP) atau *Type I Error* ketika data negatif salah diprediksi sebagai positif, serta *False Negative* (FN) atau *Type II Error* ketika data positif salah diprediksi sebagai negatif. Dengan menghitung nilai TP, TN, FP, dan FN, berbagai metrik evaluasi seperti *accuracy*, *precision*, *recall*, dan *F1-score* dapat diturunkan untuk menilai seberapa baik model dalam mengklasifikasikan data (Rainio dkk., 2024).



Gambar 2.6 *Confusion Matrix*

Berdasarkan Gambar 2.6, performa model *Hybrid* BWELM dan *Random Forest* dievaluasi dengan lima metrik utama:

1. Akurasi mengukur proporsi prediksi yang benar dari keseluruhan data yang dihitung menggunakan Persamaan 2.22 dan 2.3 (Rainio dkk., 2024).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.22)$$

$$Accuracy = \frac{jumlah\ prediksi\ benar\ (positif\ dan\ negatif)}{jumlah\ keseluruhan\ data} \quad (2.23)$$

Dengan:

TP = *True Positive*

TN = *True Negative*

FP = *False Positive*

FN = *False Negative*

2. Presisi *(precision)* menilai ketepatan prediksi positif, dihitung menggunakan Persamaan 2.24 dan 2.5 (Padhy dkk., 2024).

$$Precision = \frac{TP}{TP + FP} \tag{2.24}$$

$$Precision = \frac{jumlah\ prediksi\ benar\ (positif)}{prediksi\ benar\ (positif) + prediksi\ salah\ (positif)} \tag{2.25}$$

3. *Recall* mengevaluasi kemampuan model dalam mengenali seluruh kasus positif, dihitung menggunakan Persamaan 2.26 dan 2.7 (Lilhore dkk., 2021).

$$Recall = \frac{TP}{TP + FN} \tag{2.26}$$

$$Recall = \frac{jumlah\ prediksi\ benar\ (positif)}{prediksi\ benar\ (positif) + prediksi\ salah\ (negatif)} \tag{2.27}$$

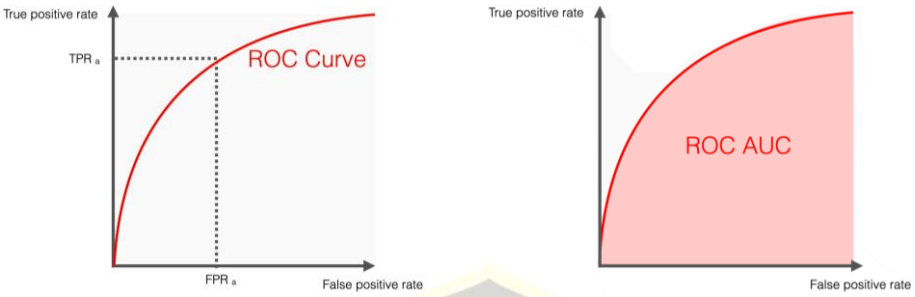
4. *F1-Score* sebagai rata-rata harmonis presisi dan *recall* sangat berguna pada data tidak seimbang, dihitung menggunakan Persamaan 2.28 (Natras dkk., 2022).

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.28}$$

Selain itu digunakan juga metrik ROC AUC (*Receiver Operating Characteristic-Area Under Curve*) adalah metrik evaluasi yang digunakan untuk mengukur kemampuan model klasifikasi dalam membedakan dua kelas. Grafik ROC memplot *True Positive Rate* (TPR) pada sumbu vertikal terhadap *False Positive Rate* (FPR) pada sumbu horizontal untuk berbagai ambang batas klasifikasi (*threshold*). TPR menggambarkan seberapa banyak data positif yang berhasil dikenali model (*recall*), sedangkan FPR menunjukkan seberapa banyak data negatif yang salah diklasifikasikan sebagai positif.

Gambar 2.7 menunjukkan kurva ROC di mana setiap titik mewakili pasangan nilai TPR dan FPR pada threshold tertentu. Semakin ke kiri atas kurva (TPR tinggi, FPR rendah), semakin baik kinerja model. Menampilkan ROC AUC, yaitu area di bawah kurva ROC yang menjadi ukuran tunggal performa model. Nilai AUC berkisar antara 0 hingga 1: (a) AUC = 0.5 menunjukkan model tidak lebih baik dari tebakan acak (garis diagonal), (b) AUC mendekati 1.0 menunjukkan model sangat baik dalam membedakan kelas positif dan negatif, (c) AUC < 0.5

menunjukkan kinerja model lebih buruk dari tebakan acak (Chicco dan Jurman, 2023). Dengan demikian, ROC AUC memberikan gambaran yang menyeluruh terhadap kemampuan model, tidak hanya pada satu *threshold*, tetapi pada semua kemungkinan *threshold* klasifikasi.



Gambar 2.7 ROC AUC *Matrix*

Kurva ROC dihitung berdasarkan Persamaan 2.26 dan 2.27 (Salman dan Al-Jawher, 2024):

$$True\ Positive\ Rate\ (TPR) = Sensitivity = Recall = \frac{TP}{TP+FN} \quad (2.26)$$

$$False\ Positive\ Rate\ (FPR) = \frac{FP}{FP+TN} \quad (2.27)$$

Kurva AUC dihitung menggunakan perhitungan diskret berdasarkan Persamaan 2.28 (Chicco dan Jurman, 2023):

$$AUC = \sum_{i=1}^{n-1}(FPR_{i+1} - FPR_i) \cdot \frac{TPR_{i+1}+TPR_i}{2} \quad (2.28)$$

Evaluasi ini penting karena BWELM efektif menangani data minoritas melalui boosting (Cheng dkk., 2020), sementara Random Forest stabil dalam klasifikasi data mayoritas dan berdimensi tinggi (Karthika dkk., 2019).