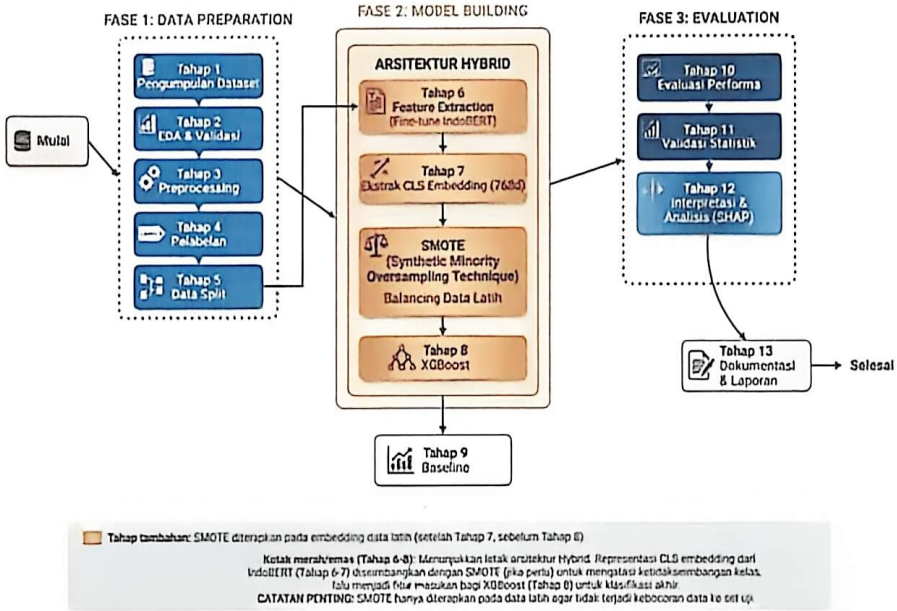
BAB III
METODOLOGI PENELITIAN

3.1 Jenis Penelitian

Penelitian ini merupakan penelitian kuantitatif dengan pendekatan eksperimental. Metode eksperimental digunakan untuk menguji efektivitas model hybrid IndoBERT-XGBoost dalam klasifikasi sentimen melalui pengukuran metrik performa yang objektif. Penelitian ini menggunakan paradigma positivisme dimana hasil dapat diukur dan diverifikasi secara empiris. Gambar 1 adalah desain penelitian.

Gambar 1. Desain Penelitian (Arsitektur Hybrid IndoBERT + XGBoost)



Desain penelitian pada Gambar 1 secara eksplisit menampilkan tahap penanganan ketidakseimbangan kelas menggunakan teknik SMOTE (Synthetic Minority Oversampling Technique), yang di tandai dengan kotak berbarwa kuning pada diagram. Tahap SMOTE diposisikan setelah proses ekstraksi CLS embedding

dari IndoBERT (Tahap 7) dan sebelum pelatihan model XGBoost (Tahap 8). Penempatan ini memastikan SMOTE hanya di terapkan pada representasi fitur data latih, sehingga distribusi data validasi dan data uji tetap alami serta risiko kebocoran data (data leakage) dapat dihindari.

3.2 Desain Sistem Penelitian

Penelitian ini mengimplementasikan arsitektur hybrid yang terdiri dari dua tahap utama:

Tahap 1: Feature Extraction dengan IndoBERT

Model IndoBERT pre-trained (IndoBERT-base-p2) akan di- fine-tune menggunakan dataset review Tokopedia. Output dari layer [CLS] token pada lapisan terakhir transformer akan digunakan sebagai representasi vektor untuk setiap review. Representasi ini menangkap konteks semantik dan sintaktik dari teks bahasa Indonesia.

Tahap 2: Klasifikasi dengan XGBoost

Vektor representasi dari IndoBERT akan menjadi input untuk classifier XGBoost. XGBoost akan dilatih untuk memetakan representasi tersebut ke dalam tiga kelas sentimen (positif, negatif, netral). Hyperparameter XGBoost seperti learning rate, max depth, n_estimators, dan subsample dioptimasi menggunakan RandomizedSearchCV. RandomizedSearchCV dipilih dibandingkan GridSearchCV karena pertimbangan efisiensi komputasi: dengan ruang pencarian hyperparameter yang luas pada data embedding berdimensi 768, pencarian acak sebanyak n_iter=20 kombinasi

mampu memberikan hasil yang kompetitif dengan waktu komputasi yang jauh lebih singkat.

1. Penjelasan Detail Setiap Tahapan Penelitian

Struktur kerja dalam diagram tersebut dibagi menjadi tiga fase utama yang sistematis, mulai dari pengolahan data mentah hingga pengujian hasil akhir.

Fase 1 : Data Preparation (Tahap 1-5)

Tahap 1: Pengumpulan Dataset. Dataset ulasan pengguna aplikasi Tokopedia dikumpulkan secara langsung melalui teknik web scraping dari Google Play Store menggunakan library google-play-scraper berbasis Python. Proses scraping dilakukan secara mandiri oleh peneliti untuk mendapatkan data yang aktual dan relevan. Dataset yang berhasil dikumpulkan memuat teks ulasan, rating bintang (1-5), dan Tahsebelum preprocessing. Data dimuat ke lingkungan pengolahan menggunakan library pandas dalam Google Colab. Pemilihan metode scraping langsung didasarkan pada kebutuhan data primer yang aktual dan tidak bergantung pada pihak ketiga.

Tahap 2: EDA & Validasi. Exploratory Data Analysis (EDA) dilakukan untuk memahami karakteristik dan distribusi data sebelum pemrosesan. Analisis mencakup: (1) distribusi rating bintang 1–5 dan proporsi tiap kelas sentimen; (2) distribusi panjang teks review dalam jumlah kata dan

karakter; (3) deteksi class imbalance antar kelas positif, negatif, dan netral; serta (4) pengecekan missing values dan duplikasi data. Validasi kualitas dataset memastikan konsistensi antara teks review dan rating yang diberikan pengguna agar label proxy dapat diandalkan.

Tahap 3: Preprocessing. Preprocessing membersihkan dan mentransformasi teks mentah menjadi format yang siap diproses model. Tahapan yang dilakukan meliputi: (a) Case folding, yaitu mengubah seluruh karakter menjadi huruf kecil; (b) Cleaning, menghapus URL, mention (@), hashtag (#), emoji, angka, dan karakter khusus; (c) Normalisasi, mengubah kata tidak baku, slang, dan singkatan menggunakan kamus normalisasi bahasa Indonesia; (d) Tokenisasi menggunakan IndoBERT Tokenizer; Adapun penanganan ketidakseimbangan kelas menggunakan teknik SMOTE tidak dilakukan pada tahap preprocessing ini, melainkan diterapkan pada Tahap 8 (Train XGBoost) terhadap data embedding hasil ekstraksi IndoBERT.

Tahap 4: Pelabelan. Pelabelan sentimen dilakukan secara otomatis (proxy labeling) berdasarkan rating bintang pengguna dengan skema: rating 4-5 = Positif, rating 3 = Netral, rating 1-2 = Negatif. Untuk mengantisipasi potensi mismatch antara teks dan rating numerik, validasi manual dilakukan

pada sampel acak 5-10% dari total data guna mengestimasi tingkat akurasi proxy labeling ini.

Tahap 5: Data Split. Dataset yang telah bersih dan berlabel dibagi menggunakan stratified sampling untuk mempertahankan proporsi kelas di setiap subset. Rasio pembagian adalah 70% data latih (training set), 15% data validasi (validation set), dan 15% data uji (testing set). Data uji bersifat eksklusif dan tidak pernah dilihat oleh model selama proses pelatihan untuk menjamin objektivitas evaluasi akhir.

1. Fase 2: Model Building(Tahap 6-9)

Tahap 6: Fine-tune IndoBERT. Model IndoBERT-base-p2 dimuat dari Hugging Face Model Hub dan di-fine-tune menggunakan data latih ulasan Tokopedia. Proses fine-tuning dilakukan dengan menambahkan classification head berupa lapisan dense dengan 3 unit output (positif, netral, negatif). Parameter fine-tuning: learning rate 2e-5, batch size 16, 3 epoch, dan warmup steps 500. Tujuannya agar IndoBERT menghasilkan representasi vektor yang spesifik dan relevan untuk domain sentimen e-commerce.

Tahap 7: Extract Embedding. Setelah fine-tuning, IndoBERT digunakan sebagai feature extractor. Setiap teks review diproses oleh IndoBERT dan output dari token [CLS] pada lapisan

transformer terakhir diekstrak sebagai representasi vektor berdimensi 768. Vektor [CLS] merangkum informasi semantik dan kontekstual dari keseluruhan teks secara bidirectional dan menjadi input fitur untuk XGBoost pada tahap berikutnya.

Tahap 8: Train XGBoost. Vektor embedding 768-dimensi dari IndoBERT menjadi input untuk melatih XGBoost Classifier. Sebelum pelatihan, ketidakseimbangan kelas pada data latih ditangani menggunakan teknik SMOTE (Synthetic Minority Oversampling Technique) yang diterapkan khusus pada data embedding hasil ekstraksi IndoBERT agar tidak menimbulkan kebocoran data (data leakage) ke data uji. Hyperparameter dioptimasi menggunakan RandomizedSearchCV (n_iter=20, 5-fold cross validation) dengan ruang pencarian: learning rate [0.01, 0.1, 0.3], max_depth [3, 5, 7, 9], n_estimators [100, 200, 300], subsample [0.6, 0.8, 1.0], colsample_bytree [0.6, 0.8, 1.0]. Fungsi objektif yang digunakan adalah multi:softmax untuk klasifikasi multi-kelas. Early stopping dengan patience=2 diterapkan untuk mencegah overfitting.

Tahap 9: Train Baselines. Sebagai pembanding, beberapa model baseline dilatih menggunakan data yang sama: (1) IndoBERT Standalone – IndoBERT dengan classification head linier tanpa XGBoost; (2) XGBoost Standalone - XGBoost dengan

fitur TF-IDF tanpa embedding IndoBERT; (3) Naive Bayes dan SVM dengan fitur TF-IDF. Perbandingan performa antara model hybrid dan seluruh baseline menjadi dasar evaluasi keunggulan pendekatan yang diusulkan.

2. Fase 3: Evaluation (Tahap 10-13)

Tahap 10: Evaluasi Performa. Semua model dievaluasi menggunakan data uji (testing set) yang tidak digunakan selama pelatihan. Evaluasi menggunakan metrik Accuracy, Precision, Recall, F1-Score (macro-average dan weighted-average), serta Confusion Matrix untuk membandingkan seluruh model secara objektif dan terukur.

Tahap 11:Validasi Statistik. Validasi statistik menggunakan K-Fold Cross Validation (k=5) dan uji signifikansi berupa Paired T-Test. K-Fold memastikan robustness model dengan merotasi setiap fold sebagai data uji secara bergantian. Uji T-Test mengkonfirmasi apakah perbedaan performa antar model secara statistik signifikan dengan tingkat kepercayaan 95% (p-value < 0.05).

Tahap 12: Interpretasi & Analisis. Interpretabilitas model dilakukan menggunakan SHAP values untuk mengidentifikasi kontribusi masing-masing fitur terhadap prediksi akhir, serta Feature Importance dari XGBoost untuk mengetahui fitur paling berpengaruh. Error analysis kualitatif juga dilakukan terhadap

kasus misclassification untuk memahami pola kesalahan dan memberikan rekomendasi perbaikan.

Tahap 13: Dokumentasi & Laporan. Seluruh proses penelitian dari latar belakang, metodologi, hasil eksperimen, analisis, hingga kesimpulan dan rekomendasi didokumentasikan dalam bentuk laporan skripsi formal sesuai panduan penulisan karya ilmiah Universitas An Nuur Purwodadi.

Intinya: Strategi ini memanfaatkan kekuatan IndoBERT untuk memahami konteks bahasa dan XGBoost untuk akurasi klasifikasi yang tinggi.

3.3 Fokus Penelitian

Fokus penelitian ini meliputi:

1. Implementasi model hybrid IndoBERT-XGBoost untuk klasifikasi sentimen review aplikasi Tokopedia
2. Komparasi performa model hybrid dengan baseline models (IndoBERT standalone, XGBoost standalone)
3. Optimasi hyperparameter untuk meningkatkan akurasi model
4. Analisis kesalahan klasifikasi dan interpretasi hasil model

3.4 Sumber Data

Data yang digunakan dalam penelitian ini adalah ulasan pengguna aplikasi Tokopedia yang dikumpulkan secara langsung melalui teknik web scraping dari Google Play Store menggunakan library google-play-scraper berbasis Python.

Proses pengumpulan data dilakukan secara mandiri oleh peneliti sehingga data yang diperoleh bersifat primer dan aktual. Dataset yang berhasil dikumpulkan berjumlah 35.038 ulasan dan setelah proses preprocessing menghasilkan 27.018 ulasan bersih dengan distribusi Positif (30,4%), Negatif (59,2%), dan Netral (10,4%). Review dilabeli ke dalam tiga kategori berdasarkan rating: positif (rating 4-5), netral (rating 3), dan negatif (rating 1-2).

3.5 Pengumpulan data

Data dikumpulkan menggunakan library google-play-scraper pada Python yang memungkinkan pengambilan ulasan langsung dari halaman aplikasi Tokopedia di Google Play Store secara otomatis. Tahap awal meliputi

1. Loading Data: Memuat dataset ke dalam lingkungan penelitian (seperti Google Colab atau Jupyter Notebook) menggunakan library pandas.
2. Exploratory Data Analysis (EDA): Melakukan analisis eksplorasi untuk memahami karakteristik dataset, meliputi:
    a. Distribusi rating (bintang 1-5).
    b. Distribusi panjang teks review (jumlah kata/karakter).
    c. Analisis kelas sentimen (positif, netral, negatif) untuk mendeteksi ketidakseimbangan data (*class imbalance*).
    d. Pengecekan nilai yang hilang (*missing values*) dan duplikasi data.
3. Validasi Kualitas Dataset: Memastikan integritas data hasil scraping, termasuk konsistensi antara teks review dan rating yang diberikan pengguna, serta pengecekan kelengkapan kolom yang diperlukan.

3.6 Pra-pemrosesan Data

Tahapan preprocessing teks bahasa Indonesia:

1. Case folding: konversi semua teks menjadi huruf kecil
2. Cleaning: penghapusan URL, mention, hashtag, emoji, angka, dan karakter khusus
3. Normalisasi: standardisasi kata tidak baku, slang, dan singkatan menggunakan kamus normalisasi bahasa Indonesia
4. Tokenisasi: menggunakan IndoBERT tokenizer yang sudah dilatih pada korpus bahasa Indonesia
5. Penanganan ketidakseimbangan kelas: SMOTE diterapkan pada tahap pelatihan XGBoost (lihat subbab 3.7), bukan pada tahap pra-pemrosesan
6. Data splitting: 70% training, 15% validation, 15% testing dengan stratified sampling

3.7 Pembentukan Model

1. Fine-tuning IndoBERT
    a. Load pre-trained IndoBERT-base-p2 dari Hugging Face model hub
    b. Tambahkan classification head (dense layer) untuk 3 kelas output
    c. Fine-tune model dengan parameter: learning rate 2e-5, batch size 16, 3 epoch, warmup steps 500
    d. Ekstrak embedding dari layer [CLS] untuk setiap instance
2. Training XGBoost Classifier
    a. Input: embedding vectors dari IndoBERT (768 dimensi untuk base model)

b. Hyperparameter tuning menggunakan RandomizedSearchCV (n_iter=20, 5-fold CV): learning_rate [0.01, 0.1, 0.3], max_depth [3, 5, 7, 9], n_estimators [100, 200, 300], subsample [0.6, 0.8, 1.0], colsample_bytree [0.6, 0.8, 1.0]

c. Objective function: multi:softmax untuk klasifikasi multi-class

d. Early stopping dengan patience=2 untuk mencegah overfitting

3. Baseline Models

Untuk komparasi, akan dibangun baseline models:

a. IndoBERT standalone: fine-tuned IndoBERT dengan classification head

b. XGBoost standalone: XGBoost dengan TF-IDF features

3.8 Evaluasi dan Validasi Model

1. Metrik Evaluasi

Model akan dievaluasi menggunakan metrik berikut:

a. Accuracy: proporsi prediksi benar dari total prediksi

b. Precision: TP / (TP + FP) untuk setiap kelas

c. Recall: TP / (TP + FN) untuk setiap kelas

d. F1-Score: 2 × (Precision × Recall) / (Precision + Recall)

e. Macro-average dan weighted-average untuk metrik multi-class

f. Confusion Matrix untuk analisis kesalahan klasifikasi

Berikut adalah penjelasan matematis lengkap beserta makna dan contoh dari setiap metrik evaluasi yang digunakan dalam penelitian ini.

Sebelum memahami rumus, perlu dipahami terlebih dahulu empat komponen dasar hasil klasifikasi yang disebut confusion matrix.

Komponennya terdiri :

1). TP (True Positive) = jumlah data yang diprediksi sebagai kelas X dan memang benar kelas X;

2). TN (True Negative) = jumlah data yang diprediksi bukan kelas X dan memang bukan kelas X;

3). FP (False Positive) = jumlah data yang diprediksi sebagai kelas X padahal sebenarnya bukan kelas X (kesalahan tipe I); dan

4). FN (False Negative) = jumlah data yang diprediksi bukan kelas X padahal sebenarnya adalah kelas X (kesalahan tipe II). Contoh konkret: jika model memprediksi sebuah ulasan sebagai "Positif" padahal sebenarnya "Negatif", maka ini dihitung sebagai FP untuk kelas Positif dan FN untuk kelas Negatif.

a. Accuracy (Akurasi)

Accuracy merupakan proporsi total prediksi yang benar dari keseluruhan data uji. Rumus Accuracy adalah sebagai berikut:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (3.1)$$

Nilai Accuracy berkisar antara 0 hingga 1 (atau 0% hingga 100%). Accuracy yang tinggi tidak selalu berarti model baik, terutama pada dataset tidak seimbang. Contoh: jika 90% ulasan berlabel Positif dan model selalu memprediksi Positif, Accuracy mencapai 90% meskipun model gagal total mengenali kelas Negatif dan Netral. Oleh karena itu,

pada penelitian ini Accuracy digunakan bersama metrik Precision, Recall, dan F1-Score agar evaluasi lebih komprehensif dan tidak menyesatkan.

b. Precision

Precision mengukur seberapa akurat model ketika ia membuat prediksi untuk suatu kelas tertentu, yaitu dari seluruh data yang diprediksi sebagai kelas X, berapa proporsi yang benar-benar kelas X. Dengan kata lain, Precision menjawab pertanyaan: "Dari semua ulasan yang diprediksi Positif oleh model, berapa persen yang memang benar-benar Positif?" Untuk klasifikasi tiga kelas (Positif, Netral, Negatif), Precision dihitung secara terpisah untuk setiap kelas. Rumusnya adalah:

$$\text{Precision} = TP / (TP + FP) \quad (3.2)$$

Precision yang tinggi berarti model jarang salah saat menyatakan sesuatu sebagai kelas tertentu (False Positive rendah). Contoh: Precision kelas Negatif = 0.85 berarti dari 100 ulasan yang diprediksi Negatif, 85 di antaranya memang benar Negatif dan 15 sebenarnya bukan Negatif. Metrik ini penting ketika biaya dari prediksi positif palsu (false alarm) sangat tinggi.

c. Recall (Sensitivitas)

Recall (juga disebut Sensitivitas atau True Positive Rate) mengukur seberapa lengkap model dalam mendeteksi suatu kelas, yaitu dari seluruh data yang sebenarnya kelas X, berapa proporsi yang berhasil ditemukan oleh model. Recall menjawab pertanyaan: "Dari

semua ulasan yang sebenarnya Negatif, berapa persen yang berhasil terdeteksi sebagai Negatif oleh model?" Rumusnya adalah:

$$\text{Recall} = TP / (TP + FN) \quad (3.3)$$

Recall yang tinggi berarti model mampu menemukan hampir semua data yang termasuk kelas tertentu (False Negative rendah). Contoh: Recall kelas Negatif = 0.78 berarti dari 100 ulasan yang sebenarnya Negatif, model berhasil mendeteksi 78 di antaranya dan melewatkan 22. Perlu diperhatikan bahwa Precision dan Recall sering kali berkonflik (trade-off): meningkatkan Recall cenderung menurunkan Precision dan sebaliknya. Keseimbangan antara keduanya diukur menggunakan F1-Score.

d. F1-Score

F1-Score adalah metrik yang menggabungkan Precision dan Recall menjadi satu angka tunggal menggunakan rata-rata harmonik. Rata-rata harmonik dipilih karena lebih sensitif terhadap nilai yang rendah dibandingkan rata-rata aritmetika biasa, sehingga F1-Score hanya akan tinggi apabila kedua Precision maupun Recall sama-sama tinggi. F1-Score sangat berguna dan direkomendasikan pada dataset yang tidak seimbang (class imbalance), seperti pada penelitian ini di mana jumlah ulasan Positif, Netral, dan Negatif kemungkinan tidak sama. Rumus F1-Score untuk satu kelas adalah:

$$F1\text{-}Score = 2 \times (Precision \times Recall) / (Precision + Recall) \quad (3.4)$$

Untuk klasifikasi tiga kelas (Positif, Netral, Negatif), digunakan dua varian rata-rata F1: (a) Macro-average F1, yaitu rata-rata F1-Score dari setiap kelas dengan bobot yang sama tanpa mempertimbangkan jumlah sampel per kelas – cocok digunakan ketika performa pada setiap kelas sama pentingnya meskipun jumlah datanya berbeda; dan (b) Weighted-average F1, yaitu rata-rata F1-Score yang ditimbang berdasarkan jumlah sampel (support) tiap kelas – cocok digunakan ketika ingin mempertimbangkan kontribusi kelas yang lebih besar. Contoh: jika F1 Positif = 0.90, F1 Netral = 0.70, F1 Negatif = 0.80, maka Macro-F1 = (0.90 + 0.70 + 0.80) / 3 = 0.80. Rumus Macro-average F1 adalah:

Macro-F1 = (1/N) x SUM(F1-Score_i) untuk i = 1 sampai N kelas (3.5)

e. Confusion Matrix

Confusion Matrix adalah tabel ringkasan kinerja klasifikasi yang menampilkan secara lengkap distribusi prediksi model dibandingkan label aktual. Untuk klasifikasi tiga kelas (Positif, Netral, Negatif), Confusion Matrix berbentuk matriks 3×3. Baris ke-i mewakili kelas aktual ke-i, sedangkan kolom ke-j mewakili kelas yang diprediksi oleh model. Elemen pada diagonal utama (baris = kolom) menunjukkan jumlah prediksi yang benar untuk setiap kelas. Elemen di luar diagonal menunjukkan kesalahan: misalnya elemen baris "Negatif" dan kolom "Positif" menunjukkan berapa ulasan yang sebenarnya Negatif namun diprediksi sebagai Positif oleh model. Dengan membaca Confusion Matrix, peneliti dapat mengidentifikasi pola kesalahan spesifik,

misalnya apakah model sering mengklasifikasikan Netral sebagai Positif, atau Negatif sebagai Netral, sehingga dapat diberikan analisis mendalam dan rekomendasi perbaikan yang tepat sasaran.

f. Rumus Fine-Tunning IndoBERT (Fungsi loss)

Selama proses fine-tuning IndoBERT, model dilatih dengan cara meminimalkan fungsi loss. Fungsi loss yang digunakan adalah Categorical Cross-Entropy Loss untuk klasifikasi multi-kelas. Fungsi ini mengukur seberapa jauh distribusi probabilitas prediksi model dari distribusi probabilitas label sebenarnya (one-hot encoding). Semakin kecil nilai Cross-Entropy Loss, semakin dekat prediksi model dengan label yang benar, yang artinya model semakin baik. Rumus Cross-Entropy Loss untuk satu batch berisi N sampel adalah:

$$L = -(1/N) \times SUM[y\_i \times \log(p\_i)] \text{ untuk } i = 1 \text{ sampai } N \qquad (3.6)$$

Penjelasan setiap simbol dalam rumus Cross-Entropy Loss:

1. N = jumlah sampel teks ulasan dalam satu batch training
2. y_i = vektor label sebenarnya dalam format one-hot encoding untuk ulasan ke-i, misalnya kelas Positif = [1,0,0], Netral = [0,1,0], Negatif = [0,0,1]
3. p_i = vektor probabilitas yang diprediksi model untuk ulasan ke-i, dihasilkan melalui fungsi Softmax dari lapisan output;
4. log = logaritma natural (ln). Cara membaca: jika model sangat yakin memprediksi kelas yang benar (p_i mendekati 1), maka loss mendekati 0 (bagus).

Sebaliknya jika model salah prediksi dengan keyakinan tinggi, loss menjadi sangat besar, sehingga model mendapat "hukuman" besar selama training. Agar skor logit mentah dari lapisan terakhir IndoBERT dapat diinterpretasikan sebagai probabilitas, digunakan fungsi Softmax yang mengubah vektor skor menjadi distribusi probabilitas yang menjumlah ke 1. Rumus Softmax untuk kelas ke-i dari K total kelas adalah:

Softmax(z_i) = exp(z_i) / SUM[exp(z_j)] untuk j = 1 sampai K kelas (3.7)

g. Rumus XGBoost (Objective Function)

XGBoost bekerja dengan cara membangun pohon keputusan (decision tree) satu per satu secara bertahap (boosting). Setiap pohon baru dibangun untuk memperbaiki kesalahan dari pohon-pohon sebelumnya. Untuk mengendalikan proses ini, XGBoost meminimasi sebuah fungsi objektif yang terdiri dari dua bagian:

1. fungsi loss L yang mengukur seberapa jauh prediksi model dari label sebenarnya; dan
2. term regularisasi Omega yang memberikan penalti pada pohon yang terlalu kompleks, sehingga mencegah overfitting. Rumus lengkap fungsi objektif XGBoost adalah:

Obj = SUM[l(y_i, y_hat_i)] + SUM[Omega(f_k)]

Omega(f) = gamma x T + (1/2) x lambda x ||w||^2 (3.8)

Penjelasan setiap simbol:

1. $l(y_i, \hat{y}_i)$ = fungsi loss yang mengukur selisih antara label sebenarnya y_i dengan prediksi $\hat{y}_i$ untuk sampel ke-i;
2. $Omega(f_k)$ = fungsi regularisasi untuk pohon ke-k yang didefinisikan sebagai $gamma \times T + (1/2) \times lambda \times \|w\|^2$;
3. T = jumlah daun (leaf nodes) dalam satu pohon – pohon yang lebih banyak daunnya lebih kompleks;
4. w = vektor bobot atau skor prediksi di setiap daun pohon;
5. gamma (γ) = parameter minimum information gain yang diperlukan agar sebuah percabangan (split) dilakukan – nilai gamma lebih besar membuat pohon lebih konservatif;
6. lambda (λ) = koefisien regularisasi L2 pada bobot daun – nilai besar berarti penalti lebih ketat terhadap bobot yang terlalu besar. Secara intuitif: XGBoost tidak hanya ingin model yang akurat (loss kecil), tetapi juga model yang sederhana (regularisasi kecil). Keseimbangan antara akurasi dan kesederhanaan ini yang membuat XGBoost sangat efektif menghindari overfitting pada data training.

h. Rumus TF-IDF(untuk Baseline)

Pada model baseline (XGBoost + TF-IDF, Naive Bayes, SVM), teks ulasan tidak dapat langsung diproses oleh algoritma machine learning karena berupa data tidak terstruktur. Oleh karena itu, teks perlu diubah menjadi representasi numerik terlebih dahulu. Metode yang digunakan adalah TF-IDF (Term Frequency-Inverse Document

Frequency), yaitu teknik pembobotan kata yang mempertimbangkan dua faktor sekaligus: seberapa sering suatu kata muncul dalam dokumen tertentu (TF), dan seberapa jarang kata tersebut muncul di seluruh korpus dokumen (IDF). Kata yang sangat umum seperti "yang", "dan", "di" akan mendapat skor TF-IDF rendah karena muncul di hampir semua dokumen, sehingga tidak membantu membedakan antar kelas sentimen. Sebaliknya, kata-kata khas seperti "lambat", "kecewa", atau "memuaskan" akan mendapat skor tinggi karena informatif untuk menentukan sentimen. Rumus TF-IDF adalah:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

$$\text{TF}(t, d) = (\text{Jumlah kemunculan t dalam d}) / (\text{Total kata dalam d})$$

$$\text{IDF}(t) = \log(N / \text{df}(t)) \quad (3.9)$$

Penjelasan setiap komponen:

1. TF(t, d) = Term Frequency, yaitu frekuensi relatif kata t dalam dokumen d, dihitung sebagai jumlah kemunculan kata t dibagi total kata dalam d;
2. IDF(t) = Inverse Document Frequency, yaitu log(N / df(t)) di mana N adalah total jumlah dokumen dan df(t) adalah jumlah dokumen yang mengandung kata t – semakin jarang kata muncul di berbagai dokumen, semakin besar nilai IDF-nya;
3. TF-IDF(t, d) = hasil perkalian TF dan IDF yang memberikan bobot akhir untuk kata t dalam dokumen d. Contoh konkret: kata "kecewa" muncul 3 kali dalam ulasan yang berisi 50 kata (TF =

3/50 = 0.06), dan hanya muncul di 100 dari 10.000 ulasan (IDF = log(10.000/100) = 2). Maka TF-IDF("kecewa") = 0.06 × 2 = 0.12. Setiap ulasan direpresentasikan sebagai vektor TF-IDF berukuran V (jumlah kosakata unik), yang kemudian menjadi input fitur bagi model baseline.

2. Validasi Statistik

a. K-fold cross validation (k=5) untuk memastikan robustness model

K-Fold Cross Validation adalah teknik evaluasi model yang lebih andal dibandingkan evaluasi satu kali pada satu set data uji, karena memberikan estimasi performa yang lebih representatif dari keseluruhan data. Cara kerjanya: seluruh dataset dibagi menjadi k bagian (fold) yang sama besar. Proses dilakukan sebanyak k iterasi: pada iterasi ke-i, satu fold digunakan sebagai data uji (test set), sedangkan k-1 fold sisanya digunakan sebagai data latih (training set). Pada penelitian ini digunakan k=5, artinya model dilatih dan diuji sebanyak 5 kali dengan kombinasi data yang berbeda. Hasilnya adalah 5 nilai performa yang kemudian dirata-ratakan. Keunggulan metode ini: setiap sampel data berkesempatan menjadi data uji tepat satu kali, sehingga hasil evaluasi tidak bergantung pada "keberuntungan" pembagian data tertentu. Standar deviasi dari 5 nilai juga dihitung untuk mengukur stabilitas (konsistensi) model. Rumus K-Fold CV Score adalah:

$$CVScore = (1/k) \times SUM[Performance(M\_i, F\_i)] \text{ untuk } i = 1 \text{ sampai } k \quad (3.10)$$

Keterangan: k = 5 (5-Fold Cross Validation), M_i adalah model yang dilatih tanpa fold ke-i, dan Performance(M_i, F_i) adalah performa (misalnya F1-Score) model M_i yang dievaluasi pada fold F_i. Standar deviasi dari k nilai performa juga dihitung untuk mengukur stabilitas model.

b. Statistical significance test (paired t-test) untuk membandingkan performa antar model

Paired t-test (uji t berpasangan) digunakan untuk membuktikan secara statistik bahwa perbedaan performa antara model hybrid IndoBERT-XGBoost dan model baseline bukan sekadar kebetulan (random chance), melainkan benar-benar signifikan. "Berpasangan" karena kedua model diuji pada data yang sama (5 fold yang sama), sehingga perbandingan lebih adil. Prosedur: dari 5 iterasi K-Fold, diperoleh 5 pasang nilai performa untuk model hybrid (X1_1,...,X1_5) dan baseline (X2_1,...,X2_5). Selisih d_i = X1_i − X2_i dihitung untuk setiap fold. Hipotesis nol (H0): rata-rata selisih $\bar{d} = 0$, artinya kedua model tidak berbeda secara signifikan. Hipotesis alternatif (H1): $\bar{d} \neq 0$. Statistik uji t dihitung sebagai:

$$t = d\_bar / (s\_d / sqrt(k))$$

$$d\_bar = (1/k) \times SUM(d\_i), \quad s\_d = sqrt[(1/(k-1)) \times SUM(d\_i - d\_bar)^2] \quad (3.11)$$

Keterangan: d_bar adalah rata-rata selisih performa, s_d adalah standar deviasi selisih, dan k adalah jumlah fold. Statistik t mengikuti distribusi t-Student dengan derajat kebebasan (df = k-1 = 4). Hipotesis nol (H0) adalah tidak ada perbedaan signifikan antara kedua model. Jika nilai p-value yang diperoleh kurang dari 0.05 (tingkat signifikansi 5%), maka H0 ditolak dan disimpulkan

bahwa model hybrid secara statistik lebih unggul dari baseline dengan tingkat kepercayaan 95%.

c. Analisis learning curves untuk mendeteksi overfitting/underfitting

sebagai alat bantu visual untuk melihat perkembangan kepintaran model dari waktu ke waktu selama proses pelatihan agar hasilnya tetap akurat dan stabil

3. Interpretabilitas Model

a. SHAP (SHapley Additive exPlanations) values untuk menginterpretasi kontribusi fitur
b. Feature importance dari XGBoost untuk identifikasi fitur paling berpengaruh
c. Error analysis: analisis kualitatif terhadap misclassification cases

Rumus SHAP (Shapley Additive exPlanations):

SHAP (SHapley Additive exPlanations) merupakan salah satu pendekatan untuk menjelaskan hasil prediksi model machine learning, yang dasar perhitungannya diambil dari konsep Shapley Value dalam teori permainan. Metode ini digunakan untuk mengukur seberapa besar pengaruh masing-masing fitur, misalnya satu dimensi tertentu dari vektor embedding IndoBERT terhadap hasil prediksi model pada sebuah ulasan tertentu.

Cara kerjanya dapat dibayangkan seperti ini: setiap fitur diperlakukan layaknya anggota tim yang berkontribusi untuk mencapai suatu hasil bersama. Untuk mengetahui seberapa besar peran masing-masing anggota (fitur), SHAP menghitung rata-rata kontribusi fitur tersebut dengan mempertimbangkan berbagai

kemungkinan kombinasi urutan kemunculannya bersama fitur lain. Apabila nilai SHAP suatu fitur bernilai positif, maka fitur tersebut cenderung memperkuat prediksi ke arah kelas tertentu. Sebaliknya, jika nilainya negatif, fitur tersebut justru menurunkan kecenderungan model untuk memilih kelas tersebut.

Secara matematis, nilai SHAP untuk fitur ke-j dirumuskan sebagai berikut:

$$\text{phi\_j} = \text{SUM}\{ [|S|! \times (|F|-|S|-1)! / |F|!] \times [f(S \cup \{j\}) - f(S)] \} \quad \text{untuk setiap S dalam } F \setminus \{j\} \tag{3.12}$$

Penjelasan simbol:

1. φ_j = nilai SHAP (kontribusi) fitur ke-j terhadap prediksi
2. F = himpunan semua fitur yang digunakan model (dalam konteks ini adalah 768 dimensi embedding dari IndoBERT)
3. S = semua subset dari F yang tidak mengandung fitur j
4. |S| = jumlah fitur dalam subset S, dan |F| = total jumlah fitur
5. $[|S|! \times (|F|-|S|-1)! / |F|!]$ = bobot yang memperhitungkan semua kemungkinan urutan fitur masuk ke model
6. $[f(S \cup \{j\}) - f(S)]$ = selisih prediksi model ketika fitur j ditambahkan ke subset S versus tanpa fitur j, yaitu "kontribusi marginal" fitur j.

SHAP memiliki tiga sifat penting yang membuatnya andal:

1. Additivity: jumlah semua nilai SHAP sama dengan selisih prediksi model dengan rata-rata prediksi global, yaitu $\text{SUM}(\varphi\_j) = f(x) - E[f(X)]$
2. Consistency: jika kontribusi suatu fitur meningkat di model baru, nilai SHAP-nya tidak akan turun

3. Dummy: fitur yang tidak berpengaruh pada prediksi mendapat nilai SHAP = 0. Dalam penelitian ini, SHAP digunakan untuk menganalisis fitur-fitur embedding mana dari IndoBERT yang paling berpengaruh terhadap klasifikasi sentimen Positif, Netral, dan Negatif, serta untuk memvalidasi bahwa keputusan model masuk akal secara semantik.