Klasifikasi Halusinasi pada LLM menggunakan XGBoost dengan Ekstraksi Fitur FastText dan *Optimizer* PSO

Anindha Arsya Dhaniswara[1], Yuni Yamasari[2]

[1,2] Teknik Informatika/Teknik Informatika, Universitas Negeri Surabaya
[1]anindha.22109@mhs.unesa.ac.id
[2]yuniyamasari@unesa.ac.id

***Abstrak*— Halusinasi pada Large Language Models (LLM) merupakan kondisi ketika model menghasilkan informasi yang tidak sesuai dengan fakta. Penelitian ini bertujuan menganalisis kinerja XGBoost dengan representasi teks FastText untuk mendeteksi halusinasi, serta mengevaluasi pengaruh teknik balancing dan optimasi hyperparameter menggunakan Particle Swarm Optimization (PSO). Metode balancing yang digunakan meliputi Random Oversampling, SMOTE, dan Random Undersampling. Hasil pengujian menunjukkan bahwa model baseline memperoleh akurasi terbaik sebesar 91,48% pada rasio pembagian data 70:30. Penerapan balancing belum memberikan peningkatan yang berarti, sedangkan optimasi menggunakan PSO mampu meningkatkan akurasi menjadi 92,71% dengan nilai AUC sebesar 97,63%. Temuan ini menunjukkan bahwa optimasi hyperparameter menggunakan PSO lebih efektif dalam meningkatkan performa model dibandingkan teknik balancing pada penelitian ini.**

Kata Kunci*— *Deteksi Halusinasi, Large Language Model, FastText, Teknik Balancing, XGBoost, PSO

I. PENDAHULUAN

Perkembangan Large Language Models (LLM) seperti GPT, PaLM, dan LLaMA telah membawa perubahan besar dalam berbagai tugas pemrosesan bahasa alami. Kemampuan model-model tersebut dalam menghasilkan teks yang menyerupai tulisan manusia menjadikannya banyak digunakan dalam berbagai bidang. Namun, di balik kemampuannya tersebut, LLM masih memiliki keterbatasan berupa fenomena halusinasi, yaitu kondisi ketika model menghasilkan informasi yang terdengar meyakinkan tetapi tidak sesuai dengan fakta yang sebenarnya [1[1]]. Keberadaan halusinasi dapat menurunkan kepercayaan terhadap keluaran model, terutama ketika digunakan pada domain yang membutuhkan tingkat akurasi informasi yang tinggi.

Berbagai penelitian menunjukkan bahwa halusinasi masih menjadi permasalahan yang belum sepenuhnya teratasi pada LLM modern. Hasil evaluasi pada benchmark HaluEval dan TruthfulQA memperlihatkan bahwa model-model terbaru masih menghasilkan respons yang mengandung halusinasi dengan tingkat yang bervariasi [2]. Kondisi tersebut mendorong perlunya pengembangan metode yang mampu mengidentifikasi keluaran halusinatif secara otomatis agar risiko penyebaran informasi yang tidak akurat dapat diminimalkan.

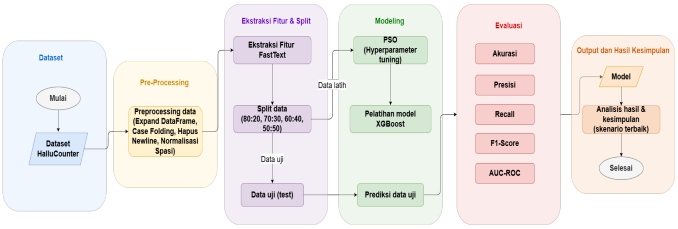
Pendekatan machine learning telah banyak digunakan untuk mendeteksi halusinasi karena mampu mempelajari pola yang membedakan teks halusinatif dan non-halusinatif Seperti yang ditunjukkan oleh penelitian terdahulu dengan judul ”Detecting Hallucinations in Large Language Models Using Machine Learning Classifiers: A Comparative Study” oleh Al-Najjar dkk, model seperti Random Forest dapat mencapai akurasi yang sangat tinggi yaitu sekitar 94.77% dalam mendeteksi halusinasi, meskipun terdapat tantangan pada evaluasi recall yang hanya mencapai 4.85%. Keunggulan utama pendekatan Machine Learning adalah kemampuannya untuk belajar dari data yang berlabel, sehingga setelah model terlatih, proses deteksi dapat berjalan sangat cepat dan otomatis tanpa memerlukan human validator atau komputasi generatif yang berulang-ulang. Keunggulan tersebut menjadikan pendekatan ini sebagai fondasi yang ideal untuk membangun sistem deteksi yang efisien, robust, dan scalable [3]. Salah satu algoritma yang banyak digunakan adalah XGBoost karena memiliki kemampuan klasifikasi yang baik dan efisien dalam menangani data berdimensi tinggi . Namun, performa model tidak hanya dipengaruhi oleh algoritma yang digunakan, tetapi juga oleh representasi fitur, distribusi kelas data, serta konfigurasi hyperparameter yang diterapkan. FastText menjadi salah satu metode representasi teks yang mampu menghasilkan representasi kata yang lebih informatif melalui pemanfaatan karakter n-gram sehingga dapat menangkap hubungan antar kata dengan lebih baik [4].

Selain representasi fitur, ketidakseimbangan distribusi kelas juga dapat memengaruhi hasil klasifikasi. Kondisi ini berpotensi membuat model lebih cenderung mempelajari kelas yang jumlahnya lebih dominan dibandingkan kelas lainnya. Oleh karena itu, diperlukan teknik balancing untuk membantu model memperoleh distribusi data yang lebih seimbang. Di sisi lain, pemilihan hyperparameter yang kurang optimal juga dapat membatasi kemampuan model dalam mencapai performa terbaik. Salah satu metode optimasi yang banyak digunakan adalah Particle Swarm Optimization (PSO), yang bekerja dengan mencari kombinasi hyperparameter terbaik melalui mekanisme pencarian berbasis populasi [5].

Berdasarkan permasalahan tersebut, penelitian ini mengusulkan pendekatan deteksi halusinasi menggunakan kombinasi FastText dan XGBoost dengan mengevaluasi pengaruh teknik balancing berupa Random Oversampling, SMOTE, dan Random Undersampling, serta optimasi hyperparameter menggunakan Particle Swarm Optimization (PSO). Pengujian dilakukan pada beberapa skenario pembagian data untuk mengetahui konfigurasi yang mampu menghasilkan performa klasifikasi terbaik dalam mendeteksi halusinasi pada keluaran LLM.

II. METODOLOGI PENELITIAN

Penelitian mengenai klasifikasi halusinasi pada LLM ini memiliki tahapan awal yaitu *data collection* atau persiapan dataset, dilanjutkan dengan *pre-processing text* untuk memastikan bahwa data yang telah dikumpulkan sudah bersih untuk selanjutnya digunakan sebagai inputan model. Setelah dilakukan *pre-processing text,* tahapan selanjutnya adalah ekstraksi fitur menggunakan FastText untuk mengolah informasi atau data berupa teks menjadi representasi vektor, selanjutnya, dataset dibagi menjadi data *training* dan *test.* Pada tahap eksperimen, dilakukan dua skenario pengujian yaitu menggunakan dataset asli dan dataset yang melalui proses *balancing* untuk mengatahui apakah memengaruhi performa atau kinerja model. Setelah itu, diterapkan hyperparameter tuning menggunakan *Particle Optimization Swarms (PSO)* untuk mencari kombinasi parameter dari model *XGBoost* yang optimal, pemodelan menggunakan algoritma *XGBoost*, Langkah terakhir yaitu tahap evaluasi model klasifikasi. Alur tahapan penelitian yang digunakan dalam penelitian ini ditunjukkan pada Gambar 1.



Gbr. 1 Alur Penelitian

A. *Deskripsi Dataset*

Penelitian ini menggunakan subset Jeopardy dari Dataset HalluCounterEval yang dipublikasikan oleh Urlana dkk. (2025) melalui *platform Hugging Face* [6]. HalluCounterEval merupakan dataset yang dibangun untuk mengidentifikasi halusinasi pada respons yang dihasilkan oleh LLM. Subset Jeopardy terdiri atas 1000 entri data yang terdiri dari kolom Question. Gold_Answer, LLM_Responses, Human_Labels, LLM_Used, Sub_Category, dan Main Category.

Penelitian ini menggunakan tiga kolom utama, yaitu Question, LLM_Responses, dan Human_Labels. Atribut Question berfungsi sebagai teks masukan, LLM_Responses sebagai output model bahasa, sedangkan Human_Labels digunakan sebagai label acuan dalam proses klasifikasi dengan label 1 yang merupakan label halusinasi dan 0 untuk label non-halusinasi. Proses ekspansi data tersebut menghasilkan 10.010 entri yang terdiri dari 6.757 data berlabel halusinasi dan 3.253 data berlabel non-halusinasi. Berikut pada Tabel I merupakan sampel dataset dengan menggunakan tiga kolom utama.

TABEL I
SAMPEL DATASET TIGA KOLOM UTAMA

Question	LLM_Responses	Human_Labels
Types of jumps in this sport include the lutz, the salchow & the axel	['Figure skating.', 'Figure skating.', 'Figure skating.', 'Figure Skating.', 'Figure skating.', 'Figure skating.', 'Figure skating.', 'Figure skating.', 'Figure skating.', 'Figure skating.']	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Camille Winbush, seen here, is Vanessa, the teen who's always questioning her uncle's authority on this comedy	["The Parent 'Hood.", 'The Baby-Sitters Club.', 'The Steve Harvey Show', 'The Steve Harvey Show.', 'The Bernie Mac Show', 'The Bernie Mac Show.', 'The Steve Harvey Show.', "The Parent 'Hood.", 'The Fresh Prince of Bel-Air.', 'The Steve Harvey Show']	[1, 1, 1, 1, 0, 0, 1, 1, 1, 1]

B. *Data Pre-Processing*

Data Pre-Processing merupakan suatu tahapan yang melibatkan serangkaian teknik yang digunakan untuk mengubah data mentah dan tidak terstruktur menjadi format yang mudah dipahami dan dianalisis. Tujuannya agar data teks yang akan digunakan ke dalam model bersih, konsisten, dan memiliki makna yang jelas [7]. Pada penelitian ini, *pre-processing data* tidak terlalu banyak diterapkan karena dataset asli sudah cukup bersih. Berikut detail tahapan *pre-processing data*:

1) Ekspansi *Dataset*

Dataset asli berisi masing-masing 10 respons dalam bentuk list pada setiap baris data. Untuk memungkinkan analisis independen terhadap setiap respons yang dihasilkan oleh model, dataset diperluas sehingga setiap pasangan answer-label menjadi satu entri tersendiri.

2) Case Folding

Tujuan dari langkah ini adalah untuk mengubah teks menjadi huruf kecil (*lowercase*). Proses ini membantu mengatasi variasi kata akibat perbedaan kapitalisasi. Case Folding dapat meningkatkan konsistensi dalam dokumen, mengurangi *data sparsity*, serta mempercepat proses pencarian dan analisis teks [8].

3) Text Cleaning

Langkah ini dilakukan dengan menghapus elemen yang tidak relevan dalam teks sehingga data menjadi lebih terstruktur. Data teks mentah mengandung berbagai elemen yang tidak diinginkan seperti tanda baca, angka, simbol khusus, baris baru (\n), tab (\t), dan spasi yang berlebih. [8].

C. *Ekstraksi Fitur*

Pada penelitian ini, FastText digunakan sebagai metode ekstraksi fitur untuk mengubah data teks menjadi representasi vektor numerik yang dapat diproses oleh model klasifikasi. FastText merepresentasikan kata berdasarkan karakter n-gram sehingga mampu menangkap informasi morfologis serta menghasilkan representasi yang lebih baik untuk kata yang jarang muncul maupun out-of-vocabulary (OOV) [4]. metode ini, merepresentasikan suatu kata yang dibentuk dari kombinasi vektor seluruh subword yang menyusun kata tersebut sehingga hubungan semantik dan sintaktik antarkata dapat dipelajari lebih baik [9]. Representasi vektor suatu kata pada FastText diperoleh dari penjumlahan seluruh vektor n-gram penyusunnya yang dirumuskan sebagai berikut:

$$\left[h_w = \sum_{g \in G_w} z_g \right] \quad (1)$$

dengan (h_w) merupakan representasi vektor kata (w), (G_w) adalah himpunan karakter n-gram yang membentuk kata (w), dan (z_g) adalah vektor dari n-gram (g) [4].

D. *Data Splitting*

Pada tahap pembangian data datau *data splitting*, dataset dibagi menjadi dua subset utama, yaitu *training set* dan data uji atau *testing set*. Pembagian data ini bertujuan untuk memastikan bahwa model dapat dilatih dan dievaluasi secara adil. Penelitian sebelumnya umumnya menggunakan satu rasio pembagian data, yaitu 70:30 atau 80:20 [3]. Namun, dalam penelitian ini dilakukan eksperimen dengan beberapa variasi rasio pembagian data, yaitu 80:20, 70:30, 60:40. Dan 50:50. Variasi ini diterapkan untuk mengevaluasi konsistensi performa model terhadap perbedaan proporsi *training set* dan *test set* sehingga dapat diketahu rasion pembagian yang menghasilkan kinerja optimal pada proses klasifikasi.

E. *Data Balancing*

Tahap penyeimbangan data dilakukan untuk mengatasi ketidakseimbangan distribusi kelas pada dataset yang dapat menyebabkan model cenderung bias terhadap kelas dengan jumlah data yang lebih besar [10]. Untuk melihat bagaimana penerapan *data balancing* terhadap performa model, penelitian ini menerapkan tiga teknik *balancing data*, yaitu *SMOTE, Random Oversampling,* dan *Random Undersampling*.

1) *SMOTE*

SMOTE (Synthetic Minority Over-Sampling Technique) adalah salah satu metode yang banyak digunakan untuk mengatasi permasalahan ketidakseimbangan kelas pada data. Metode yang diperkenalkan oleh Chawla dkk. pada tahun 2002 ini bekerja dengan menambahkan data baru pada kelas minoritas secara sintetis, sehingga tidak hanya mengandalkan penggandaan data yang sudah ada [11]. Data sintetis tersebut dibentuk berdasarkan kedekatan antar sampel pada kelas minoritas menggunakan algoritma *k-nearest neighbors (KNN)*. Dengan cara ini, jumlah data minoritas dapat bertambah tanpa menghasilkan salinan data yang sama persis. Namun, pada beberapa kondisi, sampel sintetis yang dihasilkan dapat kurang merepresentasikan karakteristik data sebenarnya [11].

2) *Random Oversampling*

Random Oversampling adalah pendekatan pembelajaran mesin yang dapat mengatasi masalah yang terkait dengan ketidakseimbangan kelas. Ketika jumlah sampel yang digunakan oleh berbagai kelas sangat berbeda, terjadi ketidakseimbangan kelas. Ketidakseimbangan ini membuat kelompok minoritas kehilangan peran model yang positif. Untuk mengatasi masalah ini, *Random Oversampling* bertujuan meningkatkan representasi kelompok minoritas dalam sampel [10], [11]. Meskipun sederhana dan mudah diterapkan, metode ini memiliki keterbatasan karena data yang ditambahkan tidak mengandung informasi baru, sehingga pada beberapa kasus dapat meningkatkan risiko *overfitting* dan tidak selalu menghasilkan peningkatan performa model [11].

3) *Random Undersampling*

Random Undersampling adalah teknik penyeimbangan data yang melibatkan pengurangan secara acak sebagian sampel dari kelas mayoritas hingga distribusi antar kelas menjadi lebih seimbang [11]. Metode ini relatif sederhana, karena tidak memerlukan proses atau parameter yang rumit, sehingga mudah diterapkan. Namun, penghapusan data secara acak dapat menyebabkan hilangnya informasi penting yang dalam beberapa kasus dapat memengaruhi kemampuan model untuk mempelajari pola dalam data secara optimal [12].

F. *Hyperparameter Tuning PSO*

Untuk memperoleh kombinasi hyperparameter yang optimal, penelitian ini menerapkan *hyperparameter tuning* pada model *XGBoost* menggunakan *Particle Swarm Optimization (PSO)*. Parameter yang dioptimalkan meliputi *max_depth*, *n_estimators*, dan *learning_rate* [48]. Dalam metode ini, setiap partikel merepresentasikan satu kombinasi nilai hyperparameter yang akan dievaluasi. Pergerakan partikel ditentukan oleh posisi dan kecepatannya yang diperbarui pada setiap iterasi menggunakan Persamaan (2) dan Persamaan (3).

$$v_i(t+1) = \omega \cdot v_i(t) + c_1 \cdot r_1 \cdot \left(p_i - x_i(t)\right) + \cdot \left(g - x_i(t)\right) \quad (2)$$

$$x\_i(t+1) = x\_i(t) + v\_i(t+1) \quad (3)$$

Pada persamaan tersebut, $v_i(t)$ menunjukkan kecepatan partikel ke-i pada iterasi ke-t, sedangkan $x_i(t)$ menyatakan posisi partikel yang merepresentasikan kandidat solusi. Nilai (p_i) merupakan posisi terbaik yang pernah dicapai partikel (*personal best*), sementara g adalah posisi terbaik yang diperoleh dari seluruh partikel (*global best*). Parameter ω digunakan untuk mempertahankan pengaruh kecepatan sebelumnya, sedangkan c_1 dan c_2 mengatur pengaruh pengalaman individu dan kelompok. Adapun r_1 dan r_2 merupakan bilangan acak yang membantu memperluas proses pencarian solusi. Melalui mekanisme tersebut, partikel bergerak menuju kombinasi hyperparameter yang menghasilkan performa model terbaik [48].

G. *Pengembangan Model*

Pada penelitian ini, digunakan metode algoritma *XGBoost*. Metode ini merupakan pengembangan dari metode *Gradient Boosting* yang diperkenalkan oleh Chen dan Guestrin pada tahun 2016 [13]. Algoritma ini membangun pohon keputusan secara bertahap, di mana setiap pohon baru bertujuan mengurangi kesalahan prediksi yang masih tersisa dari pohon sebelumnya. Fungsi objektif *XGBoost* dapat dinyatakan sebagai berikut:

$$Obj^{(t)} = \sum_{i=1}^{n} l\left(y_i, \widehat{y_i^{(t-1)}} + f_t(x_i)\right) + \Omega(f_t \quad (4)$$

Dengan fungsi regularisasi:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \quad (5)$$

Pada Persamaan (4), l menyatakan fungsi *loss* yang mengukur selisih antara nilai aktual dan hasil prediksi model. Sementara itu, f_t menunjukkan pohon keputusan yang dibangun pada iterasi ke- t. Pada persamaan (5), T merupakan jumlah *leaf node* dalam pohon, sedangkan w_j adalah bobot pada *leaf* ke-j. Parameter Ω dan λ digunakan sebagai komponen regularisasi untuk mengendalikan kompleksitas model sehingga dapat membantu mengurangi risiko *overfitting* [13]. Dalam penelitian ini, model XGBoost dibangun menggunakan nilai *max_depth* sebesar 6, *n_estimators* sebanyak 200, *learning_rate* sebesar 0,1, *subsample* sebesar 0,8, dan *colsample_bytree* sebesar 0,8.

H. *Evaluasi Model*

Evaluasi kinerja model merupakan tahap penting untuk menilai seberapa baik model mampu melakukan prediksi terhadap data uju secara akurat. Dalam penelitian ini, evaluasi dilakukan untuk memastikan bahwa model yang dihasilkan tidak hanya memiliki performa tinggi pada data latih, tetapi juga mampu melakukan generalisasi terhadap data yang belum pernah dilihat sebelumnya. Proses evaluasi dilakukan dengan memanfaatkan Confusion Matrix yang menggambarkan hubungan antara hasil prediksi model dan kondisi aktual pada data uji. *Confusion Matrix* terdiri dari empat komponen utama, yaitu True Positive (TP) yang menunjukkan jumlah data positif yang diprediksi benar oleh model, *True Negative (TN)* menunjukkan jumlah data negatif yang diprediksi benar, *False Positive (FP)* menunjukkan data negatif yang alah diprediksi benar, dan *False Negative (FN)* menunjukkan jumlah data positif yang salah diprediksi sebagai negatif [14]. Dari nilai-nilai tersebut dilakukan beberapa metrik performa yaitu:

1) Akurasi

Akurasi digunakan untuk mengukur proporsi jumlah prediksi yang benar terhadap keseluruhan data uji. Metrik ini memberikan gambaran umum mengenai kemampuan model dalam mengklasifikasikan data dengan benar [14]

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \quad (6)$$

2) Presisi

Presisi mengukur seberapa besar proporsi prediksi positif yang benar dari seluruh prediksi positif yang dibuat oleh model. Metrik ini penting ketika *false positive* memiliki konsekuensi signifikan [14]

$$Presisi = \frac{TP}{TP + FP} \quad (7)$$

3) Recall

Recall atau sesitivitas, menunjukkan kemampuan model dalam mendeteksi seluruh data yang benar-benar positif. Nilai *recall* yang tinggi menunjukkan bahwa model berhasil mengenali besar data positif yang ada [14].

$$Recall = \frac{TP}{TP + TN} \quad (8)$$

4) F1-Score

F1-score merupakan rata-rata harmonis antara presisi dan recall yang digunakan untuk menilai keseimbangan antara kedua metrik tersebut terutama pada data yang tidak seimbang [15].

$$F1 - score = 2 \times \frac{TP}{TP + FP + FN} \quad (9)$$

5) ROC-AUC

ROC-AUC digunakan untuk mengukur kemampuan model dalam membedakan antara kelas positif dan negatif pada berbagai ambang batas klasifikasi. Nilai *AUC (Area Under Curve)* menunjukkan seberapa besar area dibawah kurva *ROC*, semakin mendekati 1 makan semakin baik kemampuan model dalam mebedakan kedua kelas [14].
Berikut adalah persamaan ROC :

$$TPR = \frac{TP}{TP + FN} \quad (10)$$

$$FPR = \frac{FP}{FP + TN} \quad (11)$$

Berikut adalah persamaan AUC :

$$\int_0^1 TPR(FPR), dFPR \quad (12)$$

III. *ANALISIS DAN PEMBAHASAN*

Setelah penjelasan terkait tahapan dalam melakukan klasifikasi pada bab sebelumnya, berikut disajikan terkait hasil dari setiap proses yang dilakukan mulai dari *pre-processing data* hingga evaluasi model untuk selanjutnya dianalisis.

A. Hasil *Pre-processing Data*

Seperti yang sudah dipaparkan sebelumnya, *pre-processing data* pada penelitian ini terdiri dari tahapan yaitu ekspansi dataset yaitu memecah 10 respons dalam bentuk list pada setiap baris data, *case folding,* dan *cleaning text* dengan proses *remove newline*, dan normalisasi spasi atau menghapus spasi

berlebih. Hasil dari penerapan *pre-processing data* dapat dilihat pada tabel II, III, dan IV berikut:

TABEL III
SAMPEL DATASET SETELAH DIEKSPANSI

No	Question	LLM_Responses	Label
1	Native to just one U.S. state, this language is of Polynesian origin	Hawaiian language. Native to Hawaii, it is of Polynesian origin.	0
2	Native to just one U.S. state, this language is of Polynesian origin	Hawaiian language	0
3	Native to just one U.S. state, this language is of Polynesian origin	Hawaiian	0
…	…	…	…
10	Native to just one U.S. state, this language is of Polynesian origin	Hawaiian language. Native to Hawaii, it is of Polynesian origin.	0

Setelah proses ekspansi dataset, kemudian dilakukan pembersihan dengan menormalisasi teks pada masing-masih kolom *Question* dan *LLM_Responses* sebelum dilanjutkan ke tahapan ekstraksi fitur.

TABEL IIII
PRE-PROCESSING DATA KOLOM *QUESTION*

Step	Result
Original	This hesitation word Americans spell U-H is spelled this way in Britain
Case Folding	this hesitation word americans spell u-h is spelled this way in britain
Remove Newline	this hesitation word americans spell u-h is spelled this way in britain
Normalisasi Spasi	this hesitation word americans spell u-h is spelled this way in britain
Final	this hesitation word americans spell u-h is spelled this way in britain

TABEL IVII
PRE-PROCESSING DATA KOLOM *LLM_RESPONSES*

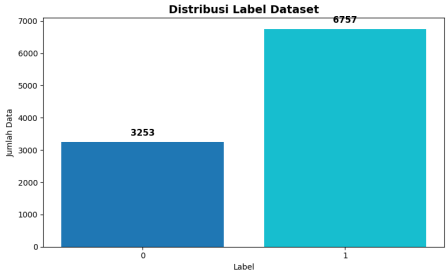
Step	Result
Original	"you have".
Case Folding	"you have".
Remove Newline	"you have".
Normalisasi Spasi	"you have".
Final	"you have".

B. Hasil Ekstraksi Fitur

Pada tahap ekstraksi fitur, masing-masing teks Question dan *LLM_Responses* diolah menggunakan FastText sehingga menghasilkan representasi vektor berdimensi 300 untuk setiap kolomnya. Kedua vektor tersebut kemudian digabungkan (*concatenate*) menjadi satu matrix fitur, sehingga diperoleh matrix berukuran 10010 × 600 yang merupakan gabungan dari 10010 sampel data dengan total 600 dimensi fitur, terdiri dari 300 dimensi dari Question dan 300 dimensi dari *LLM_Responses*.

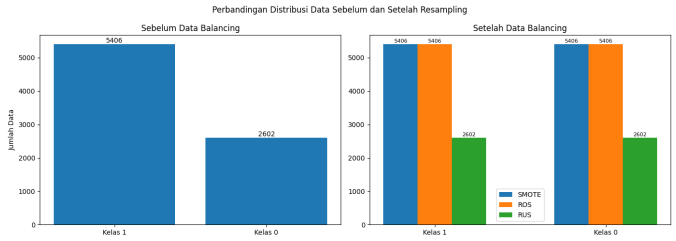
C. Hasil *Data Balancing*

Penanganan *imbalance data* dilakukan untuk mengatasi ketidakseimbangan distribusi kelas yang berpotensi memengaruhi performa model klasifikasi. Berdasarkan analisis awal terhadap dataset yang digunakan terdapat perbedaan jumlah data yang cukup signifikan antara kelas mayoritas dan kelas minoritas. Kondisi ini ditunjukkan pada Gambar 2 dimana kelas mayoritas (kelas 1) memiliki jumlah data yang lebih dominan sebanyak 6757 dibandingkan kelas minoritas (kelas 0) yang berjumlah 3253.



Gbr. 2 Distribusi Label pada Dataset

Pada tahap pemodelan, dataset kemudian dibagi menjadi data latih dan data uji. Pada salah satu skenario, distribusi data pada data latih masih menunjukkan kondisi yang tidak seimbang dengan jumlah data kelas mayoritas (1) sebanyak 5406 dan kelas minoritas (0) sebanyak 2602. Pada penelitian ini, penanganan ketidakseimbangan data dilakukan dengan menerapkan tiga teknik balancing, yaitu *SMOTE*, *Random Oversampling (ROS)*, dan *Random Undersampling (RUS)*. *SMOTE* dan *ROS* sama-sama menghasilkan jumlah data sebanyak 5406 pada masing-masing kelas dengan menambah jumlah data kelas minoritas, di mana SMOTE membentuk data sintetis, sedangkan ROS menduplikasi data yang sudah ada. Sebaliknya, *RUS* mengurangi jumlah data kelas mayoritas menjadi 2602 agar setara dengan kelas minoritas. Distribusi setelah diterapkan *data balancing* dapat dilihat pada Gambar 3 di bawah ini.



Gbr. 3 Distribusi Label Sebelum dan Sesudah Data Balancing

D. Hasil *Hyperparameter Tuning*

Berikut disajikan hasil pengujian model yang telah dioptimasi menggunakan algoritma *Particle Swarm Optimization* atau PSO. Proses Optimasi tersebut dilakukan untuk menentukan konfigurasi hyperparameter yang sesuai pada maisng-masing skenario pengujian yang meliputi variasi pembagian data latih serta data uji. Hasil konfigurasi parameter terbaik dari masing-masing rasio split ditampilkan pada Tabel IV di bawah ini:

TABEL VV
HASIL KONFIGURASI PARAMETER TERBAIK PSO

Split	*max_depth*	*n_estimators*	*learning_rate*
80:20	7	339	0.0475473
70:30	7	263	0.0887864
60:40	6	386	0.0466504
50:50	9	361	0.0976880

Pada tabel di atas, konfigurasi hyperparameter terbaik berbeda-beda untuk setiap rasio split data. Pada rasio 50:50, PSO menghasilkan nilai max_depth tertinggi 9 dibandingkan rasio lainnya, sedangkan rasio 60:40 menghasilkan nilai max_depth terendah 6 dengan jumlah n_estimators terbesar 386. Perbedaan ini menunjukkan bahwa karakteristik data pada masing-masing rasio split memengaruhi proses pencarian konfigurasi optimal oleh PSO, sehingga tidak terdapat satu kombinasi hyperparameter yang sama untuk seluruh skenario. Hasil optimasi juga menunjukkan adanya *trade-off* antara kompleksitas pohon *max_depth* dan jumlah iterasi *n_estimators*. Pada rasio 60:40, PSO memilih kombinasi *max_depth* rendah dengan *n_estimators* tinggi dan *learning_rate* kecil, yang mengindikasikan model belajar secara bertahap melalui banyak pohon sederhana. Sebaliknya, pada rasio 50:50, PSO memilih *max_depth* lebih tinggi dengan *learning_rate* lebih besar, yang menunjukkan model belajar lebih cepat dengan pohon yang lebih kompleks. Konfigurasi yang diperoleh dari masing-masing rasio split tersebut selanjutnya digunakan pada model XGBoost untuk dievaluasi.

E. Hasil Performa Model

Pada tahap ini dilakukan pengujian awal untuk mengetahui performa model *XGBoost* yang menggunakan representasi teks *FastText*. Pengujian dilakukan pada beberapa skenario pembagian data, yaitu 80:20, 70:30, 60:40, dan 50:50. Setiap skenario dievaluasi menggunakan metrik akurasi, presisi, *recall*, *F1-score,* dan nilai *AUC.* Perbandingan hasil pada masing-masing rasio pembagian data digunakan untuk mengetahui konfigurasi yang menghasilkan performa paling optimal. Selain itu, hasil pengujian ini menjadi dasar untuk menganalisis pengaruh penerapan teknik *balancing data* dan optimasi *hyperparameter* PSO pada tahap pengujian berikutnya.

1) Hasil *XGBoost Baseline*

Pada eksperimen skenario *XGBoost* tanpa perlakuan apapun (*baseline*), digunakan beberapa parameter yang disamakan untuk seluruh skenarionya yaitu *n_estimators* 100 , *learning_rate* 0.1, *max_depth* 5, *subsample* 0.8, dan *colsample_bytree* 0.8. Kemudian disajikan hasil performa setiap pembagian data yang telah dilakukan percobaan pada tabel V berikut:

TABEL V
HASIL PERFORMA MODEL BASELINE

Split	*Akurasi*	*F1-Score*	*Presisi*	*Recall*	*AUC*
80:20	91.36%	95.39%	93.79%	91.02%	96.74%
70:30	**91.48%**	**95.11%**	**93.89%**	**90.98%**	**96.99%**
60:40	91.36%	95.02%	93.78%	91.16%	96.56%
50:50	90.93%	94.30%	93.47%	90.91%	96.18%

Berdasarkan hasil pada tabel di atas, rasio pembagian data 70:30 menghasilkan akurasi tertinggi sebesar 91,48%, sedangkan rasio 50:50 memperoleh akurasi terendah sebesar 90,93%. Selisih akurasi antara kedua skenario tersebut sebesar 0,55%, yang menunjukkan bahwa perubahan proporsi data latih dan data uji memberikan pengaruh terhadap performa model, meskipun tidak terlalu besar. Hasil ini mengindikasikan bahwa model mampu mempertahankan kinerja yang relatif stabil pada berbagai skenario pembagian data.

Jika dilihat dari metrik lainnya, rasio 80:20 menghasilkan nilai F1-Score tertinggi sebesar 95,39%, sedangkan rasio 50:50 kembali menunjukkan performa terendah dengan nilai F1-Score sebesar 94,30%. Selisih sebesar 1,09% menunjukkan bahwa kemampuan model dalam menjaga keseimbangan antara precision dan recall cenderung lebih baik pada rasio 80:20 dibandingkan skenario lainnya. Selain itu, rasio 70:30 juga memperoleh nilai AUC tertinggi sebesar 96,99%, yang menunjukkan kemampuan diskriminasi kelas yang sangat baik.

Namun, karena akurasi digunakan sebagai metrik utama dalam penelitian ini, rasio pembagian data 70:30 dipilih sebagai konfigurasi terbaik untuk digunakan pada tahap pengujian berikutnya.

2) Hasil *XGBoost* dengan Metode *Balancing*

Pengujian pada tahap ini difokuskan pada penerapan metode balancing untuk mengatasi ketidakseimbangan distribusi data. Tiga metode yang digunakan yaitu *SMOTE*, *Random Oversampling,* dan *Random Undersampling* . Masing-masing metode diterapkan pada data latih sebelum proses pelatihan model XGBoost dilakukan. Performa model yang dihasilkan dari setiap metode balancing kemudian dibandingkan dengan model baseline. Melalui perbandingan ini dapat dilihat sejauh mana penambahan sampel pada kelas minoritas memberikan pengaruh terhadap kemampuan model dalam

mengklasifikasikan data. Hasil performa dari masing-masing metode balancing tersaji dalam Tabel VI, VII, dan VIII berikut

TABEL VI
HASIL PERFORMA MODEL DENGAN RANDOM OVERSAMPLING

Split	*Akurasi*	*F1-Score*	*Presisi*	*Recall*	*AUC*
80:20	90.76%	95.36%	93.20%	92.62%	93.78%
70:30	90.91%	94.96%	93.33%	92.49%	94.18%
60:40	**91.03%**	**95.01%**	**93.41%**	**92.65%**	**94.19%**
50:50	90.43%	94.25%	93.01%	91.66%	94.40%

TABEL VII
HASIL PERFORMA MODEL DENGAN SMOTE

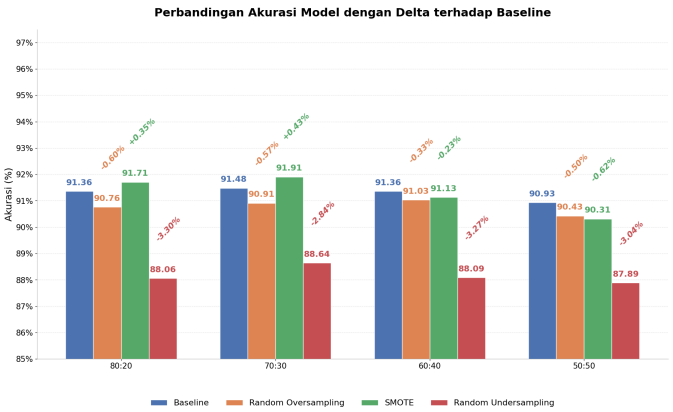
Split	*Akurasi*	*F1-Score*	*Presisi*	*Recall*	*AUC*
80:20	91.71%	95.48%	93.91%	93.09%	94.74%
70:30	**91.91%**	**95.33%**	**94.07%**	**93.05%**	**95.12%**
60:40	91.13%	95.01%	93.47%	92.97%	93.97%
50:50	90.31%	94.58%	92.83%	92.76%	92.90%

TABEL VIII
HASIL PERFORMA MODEL DENGAN RANDOM UNDERSAMPLING

Split	*Akurasi*	*F1-Score*	*Presisi*	*Recall*	*AUC*
80:20	88.06%	94.50%	90.97%	92.90%	89.12%
70:30	88.64%	94.78%	91.38%	93.68%	89.20%
60:40	88.09%	94.11%	90.97%	93.17%	88.86%
50:50	87.89%	94.09%	90.82%	93.02%	88.72%

Hasil percobaan menunjukkan bahwa Random Oversampling dan SMOTE mampu mempertahankan performa model pada berbagai rasio pembagian data. Pada metode Random Oversampling, nilai akurasi tertinggi diperoleh pada rasio 60:40 sebesar 91,03%, sedangkan SMOTE menghasilkan akurasi terbaik sebesar 91,91% pada rasio 70:30. Selain memberikan nilai akurasi tertinggi, SMOTE juga memperoleh nilai AUC tertinggi sebesar 95,12%, yang menunjukkan kemampuan klasifikasi yang lebih baik dibandingkan metode balancing lainnya.

Berbeda dengan kedua metode tersebut, Random Undersampling menghasilkan performa yang lebih rendah pada seluruh skenario pengujian dengan nilai akurasi berada pada rentang 87,89% hingga 88,64%. Penurunan ini diduga terjadi karena berkurangnya jumlah data pada kelas mayoritas sehingga informasi yang digunakan selama proses pelatihan menjadi lebih terbatas. Dari ketiga metode yang diuji, SMOTE memberikan hasil terbaik dan menjadi satu-satunya metode yang mampu meningkatkan nilai akurasi dibandingkan model baseline meskipun peningkatan yang diperoleh tidak terlalu besar. Untuk mengetahui seberapa besar kenaikan atau penurunan performa model apabila dibandingkan dengan baseline, maka disajikan visualisasi berupa *bar chart* menggunaakn *split* 70:30 sebagai representasi hasil pada Gambar 4 berikut:



Gbr. 4 Delta Akurasi Model Balancing terhadap Baseline

Gambar tersebut menunjukkan perubahan nilai akurasi setelah penerapan metode balancing pada setiap rasio pembagian data. Dibandingkan model baseline, SMOTE menjadi satu-satunya metode yang mampu memberikan peningkatan akurasi pada rasio 80:20 dan 70:30. Peningkatan terbesar terjadi pada rasio 70:30, yaitu sebesar +0,43%, dari 91,48% menjadi 91,91%. Pada rasio 60:40 dan 50:50, pengaruh SMOTE tidak terlalu terlihat karena perbedaan akurasi yang dihasilkan relatif kecil.

Random Oversampling menghasilkan nilai akurasi yang cukup dekat dengan model baseline, namun belum mampu memberikan peningkatan performa. Penurunan akurasi yang terjadi berada pada kisaran -0,33% hingga -0,60%. Sementara itu, Random Undersampling menunjukkan penurunan yang lebih jelas pada seluruh rasio pembagian data dengan selisih akurasi lebih dari 2% dibandingkan baseline. Hasil ini menunjukkan bahwa pengurangan jumlah data pada kelas mayoritas menyebabkan sebagian informasi penting tidak lagi digunakan selama proses pelatihan model. Dengan demikian, penerapan teknik balancing pada penelitian ini belum memberikan peningkatan akurasi yang signifikan. Bahkan, sebagian besar skenario pengujian menunjukkan penurunan performa dibandingkan model baseline, sehingga pengaruh *balancing* terhadap peningkatan akurasi model dapat dikatakan relatif terbatas.

3) Hasil *XGBoost* dengan *PSO*

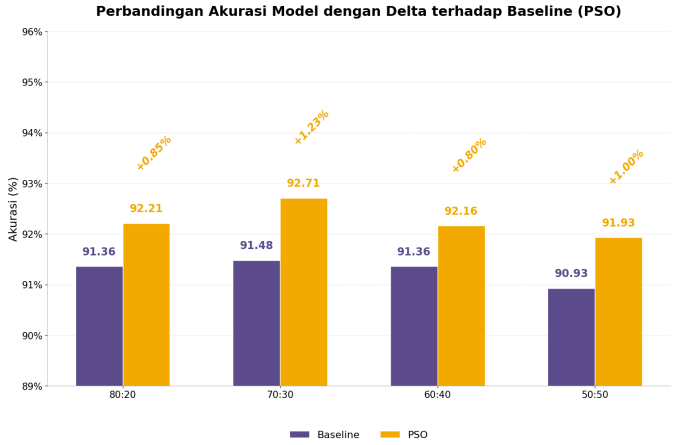
Setelah diperoleh kombinasi parameter terbaik pada masing-masing rasio pembagian data yang sudah dibahas sebelumnya dan tertampil pada Tabel IV kemudian dari parameter tersebut diterapkan pada model XGBoost untuk dilakukan pengujian kembali. Tahap ini bertujuan untuk melihat perubahan performa yang dihasilkan setelah proses optimasi menggunakan PSO. Hasil evaluasi kemudian dibandingkan dengan model baseline guna mengetahui sejauh mana optimasi hyperparameter mampu meningkatkan kinerja model dalam melakukan klasifikasi. Hasil performa model dengan penerapan PSO dapt dilihat dalam Tabel IX berikut:

TABEL IX
HASIL PERFORMA MODEL DENGAN PSO

Split	*Akurasi*	*F1-Score*	*Presisi*	*Recall*	*AUC*
80:20	92.21%	96.13%	94.41%	91.46%	97.56%
70:30	**92.71%**	**96.05%**	**94.76%**	**92.05%**	**97.63%**
60:40	92.16%	95.84%	94.34%	91.99%	96.82%
50:50	91.93%	95.39%	94.18%	91.68%	96.83%

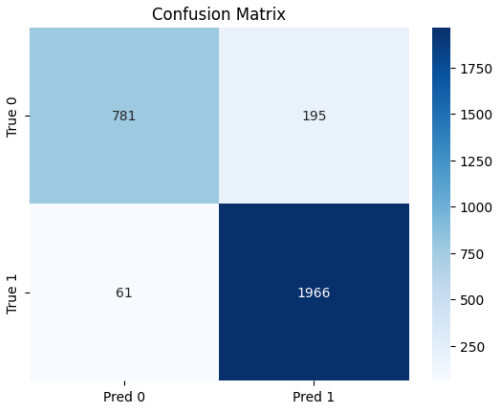
Pada rasio pembagian data 70:30, PSO menghasilkan kombinasi parameter berupa max_depth sebesar 7, n_estimators sebanyak 263, dan learning_rate sebesar 0,0887864. Konfigurasi tersebut menghasilkan performa terbaik dengan nilai akurasi sebesar 92,71%, F1-Score sebesar 96,05%, serta AUC sebesar 97,63%.

Dibandingkan dengan model baseline, penerapan PSO menghasilkan peningkatan akurasi pada seluruh rasio pembagian data yang diuji. Kenaikan terbesar diperoleh pada rasio 70:30, yaitu dari 91,48% menjadi 92,71% atau meningkat sebesar +1,23%. Pada rasio 80:20, akurasi meningkat sebesar +0,85%, sedangkan pada rasio 60:40 dan 50:50 masing-masing meningkat sebesar +0,80% dan +1,00%. Selain meningkatkan akurasi, optimasi menggunakan PSO juga menghasilkan peningkatan nilai AUC pada seluruh skenario pengujian. Hasil ini menunjukkan bahwa parameter yang diperoleh melalui proses optimasi mampu membantu model dalam membangun pola klasifikasi yang lebih baik dibandingkan penggunaan parameter awal. Berikut tertampil visualisasi mengenai peningkatan yang diperoleh, perbandingan akurasi antara model baseline dan model hasil PSO pada Gambar 5.

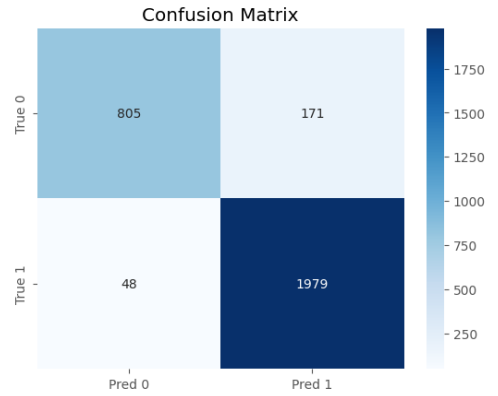


Gbr. 5 Delta Akurasi Model PSO terhadap Baseline

Perbedaan performa antara model baseline dan model hasil optimasi PSO dapat diamati lebih lanjut melalui confusion matrix pads Gambar 6 dan 7



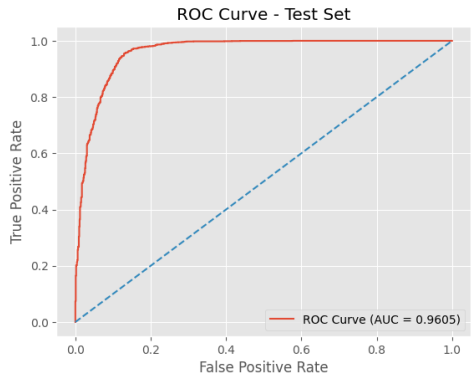
Gbr. 6 *Confusion Matrix* Model Baseline



Gbr. 7 *Confusion Matrix* Model PSO

Kedua gambar tersebut memperlihatkan perbedaan hasil klasifikasi antara model baseline dan model yang telah dioptimasi menggunakan PSO pada rasio pembagian data 70:30. Setelah optimasi dilakukan, jumlah prediksi benar pada kelas non-halusinasi meningkat dari 781 menjadi 805 data. Pada saat yang sama, jumlah kesalahan prediksi pada kelas tersebut berkurang dari 195 menjadi 171 data.

Perubahan juga terjadi pada kelas halusinasi. Jumlah data yang berhasil dikenali meningkat dari 1966 menjadi 1979 data, sedangkan data halusinasi yang tidak berhasil terdeteksi menurun dari 61 menjadi 48 data. Meskipun selisihnya tidak terlalu besar, perbaikan tersebut terjadi pada seluruh komponen confusion matrix. Hal ini menunjukkan bahwa peningkatan performa setelah optimasi PSO tidak hanya terlihat dari nilai akurasi, tetapi juga dari berkurangnya jumlah kesalahan klasifikasi yang dihasilkan model.



Gbr. 8 *ROC Curve* dengan *PSO*

Berkurangnya jumlah kesalahan klasifikasi pada confusion matrix juga tercermin pada kurva *ROC* yang dihasilkan oleh model hasil optimasi *PSO* pada pembagian data 70:30 karena menghasilkan kenaikan performa yang paling banyak dibandingkan rasio pembagian data yang lain. Kurva pada Gambar 8 berada jauh di atas garis diagonal yang menggambarkan prediksi acak, sehingga menunjukkan bahwa model memiliki kemampuan yang baik dalam membedakan data halusinasi dan non-halusinasi. Nilai *AUC* yang diperoleh mencapai 0,9763 atau 97,63%. Angka tersebut menunjukkan bahwa model mampu mempertahankan tingkat *True Positive Rate* yang tinggi meskipun *False Positive Rate* tetap berada pada tingkat yang rendah. Bentuk kurva yang cepat menanjak pada bagian awal kemudian mendekati sudut kiri atas juga memperlihatkan bahwa sebagian besar data positif berhasil dikenali tanpa menghasilkan terlalu banyak kesalahan prediksi pada kelas lainnya. Jika dikaitkan dengan hasil confusion matrix sebelumnya, peningkatan nilai *AUC* ini sejalan dengan bertambahnya jumlah prediksi benar dan berkurangnya jumlah kesalahan klasifikasi setelah optimasi dilakukan. Hal tersebut menunjukkan bahwa parameter yang diperoleh melalui *PSO* tidak hanya meningkatkan akurasi model, tetapi juga membantu model memisahkan karakteristik kedua kelas dengan lebih baik. Kemampuan ini penting karena model tidak hanya dituntut menghasilkan prediksi yang tepat pada satu nilai ambang tertentu, tetapi juga tetap mampu membedakan kedua kelas ketika nilai ambang klasifikasi berubah.

IV. KESIMPULAN

Eksperimen yang dilakukan menunjukkan bahwa kombinasi FastText dan XGBoost mampu menghasilkan performa yang konsisten pada berbagai rasio pembagian data, dengan hasil terbaik diperoleh pada rasio 70:30. Penerapan teknik balancing belum memberikan perubahan yang berarti terhadap akurasi model karena peningkatan yang dihasilkan relatif kecil, bahkan beberapa metode menunjukkan penurunan performa dibandingkan model awal. Perbaikan yang lebih terlihat diperoleh setelah optimasi hyperparameter menggunakan PSO, yang menghasilkan kenaikan akurasi sebesar 1,23% serta peningkatan nilai AUC. Temuan ini menunjukkan bahwa peningkatan kinerja model pada penelitian ini lebih dipengaruhi oleh proses optimasi hyperparameter dibandingkan penerapan teknik balancing data.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya sehingga penelitian ini dapat diselesaikan. Ucapan terima kasih juga disampaikan kepada kedua orang tua, dosen pembimbing, seluruh dosen dan sivitas akademika Program Studi Teknik Informatika, serta semua pihak yang telah memberikan dukungan, bantuan, dan motivasi hingga penelitian ini dapat diselesaikan.

REFERENSI

[1] L. Huang *et al.*, “A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions,” *ACM Trans. Inf. Syst.*, vol. 43, no. 2, Jan. 2025, doi: 10.1145/3703155.

[2] Y. Zhang *et al.*, “Siren’s Song in the AI Ocean: A Survey on Hallucination in Large Language Models,” Sep. 2025, [Online]. Available: http://arxiv.org/abs/2309.01219

[3] H. Al-Najjar, N. Al-Rousan, and D. Al-Rousan, “Detecting Hallucinations in Large Language Models Using Machine Learning Classifiers: A Comparative Study,” in *2025 1st International Conference on Computational Intelligence Approaches and Applications, ICCIAA 2025 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2025. doi: 10.1109/ICCIAA65327.2025.11013655.

[4] M. Umer *et al.*, “Impact of convolutional neural network and FastText embedding on text classification,” *Multimed. Tools Appl.*, vol. 82, no. 4, pp. 5569–5585, Feb. 2023, doi: 10.1007/s11042-022-13459-x.

[5] K. Langat, A. Waititu, and P. Ngare, “Modified XGBoost Hyper-Parameter Tuning Using Adaptive Particle Swarm Optimization for Credit Score Classification,” *Machine Learning Research*, vol. 9, no. 2, pp. 64–74, Oct. 2024, doi: 10.11648/j.mlr.20240902.15.

[6] A. Urlana, G. Kanumolu, C. V. Kumar, B. M. Garlapati, and R. Mishra, “HalluCounter: Reference-free LLM Hallucination Detection in the Wild!,” May 2025, [Online]. Available: http://arxiv.org/abs/2503.04615

[7] M. Lamba and M. Madhusudhan, *Text Mining for Information Professionals: An Uncharted Territory*. Springer International Publishing, 2022. doi: 10.1007/978-3-030-85085-2.

[8] M. Siino, I. Tinnirello, and M. La Cascia, “Is text preprocessing still worth the time? A comparative survey on the influence of popular preprocessing methods on Transformers and traditional classifiers,” *Inf. Syst.*, vol. 121, Mar. 2024, doi: 10.1016/j.is.2023.102342.

[9] V. Novotný, E. F. Ayetiran, D. Bacovský, D. Lupták, M. Štefánik, and P. Sojka, “One Size Does Not Fit All: Finding the Optimal Subword Sizes for FastText Models across Languages,” in *International Conference Recent Advances in Natural Language*

Processing, RANLP, Incoma Ltd, 2021, pp. 1068–1074. doi: 10.26615/978-954-452-072-4_120.

[10] M. Mujahid *et al.*, "Data oversampling and imbalanced datasets: an investigation of performance for machine learning and feature engineering," *J. Big Data*, vol. 11, no. 1, Dec. 2024, doi: 10.1186/s40537-024-00943-4.

[11] T. Wongvorachan, S. He, and O. Bulut, "A Comparison of Undersampling, Oversampling, and SMOTE Methods for Dealing with Imbalanced Classification in Educational Data Mining," *Information (Switzerland)*, vol. 14, no. 1, Jan. 2023, doi: 10.3390/info14010054.

[12] C. Yang, E. A. Fridgeirsson, J. A. Kors, J. M. Reps, and P. R. Rijnbeek, "Impact of random oversampling and random undersampling on the performance of prediction models developed using observational health data," *J. Big Data*, vol. 11, no. 1, Dec. 2024, doi: 10.1186/s40537-023-00857-7.

[13] P. Zhang, Y. Jia, and Y. Shang, "Research and application of XGBoost in imbalanced data," *Int. J. Distrib. Sens. Netw.*, vol. 18, no. 6, Jun. 2022, doi: 10.1177/15501329221106935.

[14] G. Naidu, T. Zuva, and E. M. Sibanda, "A Review of Evaluation Metrics in Machine Learning Algorithms," in *Lecture Notes in Networks and Systems*, Springer Science and Business Media Deutschland GmbH, 2023, pp. 15–25. doi: 10.1007/978-3-031-35314-7_2.

[15] S. Clara *et al.*, *Implementasi Seleksi Fitur Pada Algoritma Klasifikasi Machine Learning Untuk Prediksi Penghasilan Pada Adult Income Dataset*. 2021.

.