

Perbandingan Machine Learning dan Deep Learning untuk Klasifikasi Spam dengan Class Balancing

Bayu Yanuargi[#], Fahmi Auliya Tsani[#]

[#] Program Studi Rekayasa Keamanan Siber, Politeknik AI Budi Mulia, Sleman, DIY, Indonesia
E-mail: salsahumayhira18[at]gmail.com, mr.ronalhadi[at]gmail.com, mutiarahmi[at]pnpp.ac.id

ABSTRACTS

SPAM/HAM classification faces two primary challenges: class imbalance and the need to select a model that is both accurate and efficient. This study compares The Conventional Machine Learning Algorithm Naive Bayes and Support Vector Machine with Recurrent Deep Learning Algorithm Long Short-Term Memory, and Gated Recurrent Unit using a class-balancing strategy. The data underwent cleaning and deduplication before being split into a development set (80%) and a final test set (20%). Balancing was applied exclusively to the training set by downsampling HAM and oversampling SPAM to reach 1,500 samples each, while the validation and final test sets retained their natural distributions. NB and SVM utilized TF-IDF, whereas LSTM and GRU employed tokenization, 128-dimensional embeddings, and padding masks. NB achieved an accuracy of 98.63% and a Macro-F1 score of 96.65%, the highest among the four models. SVM yielded a PR-AUC of 97.55%; GRU achieved the highest SPAM recall at 94.96%; and LSTM attained the highest ROC-AUC at 98.92%. NB's training time was merely 0.004 seconds, compared to 6.61 seconds for GRU and 12.95 seconds for LSTM. The results demonstrate that traditional models remain competitive, offering a superior performance-efficiency trade-off for short-message data following class balancing.

Manuscript received Sep 17, 2026;
revised Sep 22 2026; accepted Sept
23, 2026 Date of publication Sep
30, 2026. International Journal,
JITSI : Jurnal Ilmiah Teknologi
Sistem Informasi licensed under a
Creative Commons Attribution-
Share Alike 4.0 International
License



ABSTRAK

Klasifikasi SPAM/HAM menghadapi dua tantangan utama, yaitu ketidakseimbangan kelas dan kebutuhan memilih model yang akurat sekaligus efisien. Penelitian ini membandingkan Algoritma Konvensional Naive Bayes, Support Vector Machine, dan Recurrent Deep Learning yaitu Long Short-Term Memor, dan Gated Recurrent Unit dengan strategi class balancing. Data dibersihkan dan dideduplikasi, kemudian dipisahkan menjadi development 80% dan final test 20%. Balancing hanya diterapkan pada training set melalui downsampling HAM dan oversampling SPAM hingga masing-masing 1.500 sampel; validation dan final test tetap mengikuti distribusi alami. NB dan SVM menggunakan TF-IDF, sedangkan LSTM dan GRU menggunakan tokenisasi, embedding 128 dimensi, dan masking padding. NB memberikan accuracy 98,63% dan Macro-F1 96,65%, tertinggi dari empat model. SVM menghasilkan PR-AUC 97,55%, GRU memperoleh recall SPAM tertinggi 94,96%, dan LSTM memperoleh ROC-AUC tertinggi 98,92%. Waktu training NB hanya 0,004 detik, dibandingkan GRU 6,61 detik dan LSTM 12,95 detik. Hasil menunjukkan bahwa model tradisional tetap kompetitif dan memberikan trade-off performa-efisiensi yang lebih baik pada data pesan pendek setelah class balancing

Keywords / Kata Kunci — *Classification; SPAM; Machine Learning; Deep Learning; Email.*

CORRESPONDING AUTHOR

Bayu Yanuargi
Program Studi Rekayasa Keamanan Siber, Politeknik AI Budi Mulia, Sleman, DIY, Indonesia
Email: bayuyanuargi[at]plai.ac.id

1. PENDAHULUAN

Pesan spam adalah komunikasi digital yang dikirim tanpa permintaan dari penerima dan telah berevolusi dari sekadar iklan menjadi salah satu sarana yang sering digunakan untuk phishing, penipuan, penyebaran tautan berbahaya, serta manipulasi melalui rekayasa sosial. Keberadaan spam tidak hanya menurunkan kenyamanan pengguna, tetapi juga dapat membebani sumber daya komunikasi dan meningkatkan potensi risiko keamanan informasi. Studi terkini mengindikasikan bahwa metode spam dan smishing semakin fleksibel, sehingga memerlukan pendekatan deteksi yang berlandaskan Natural Language Processing (NLP) dan Machine Learning (ML) untuk mengenali pola teks dengan lebih efisien [1][2]. Oleh karena itu, efektivitas sistem penyaring spam seharusnya dievaluasi tidak hanya berdasarkan akurasi, tetapi juga melalui presisi, recall, F1-score, serta analisis *false positive* dan *false negative*.

Berbagai penelitian mutakhir menunjukkan bahwa algoritma machine learning klasik tetap kompetitif untuk klasifikasi SPAM dan HAM, terutama ketika dikombinasikan dengan representasi tekstual yang tepat. Pada penelitian yang membandingkan Naïve Bayes, Support Vector Machine, Decision Tree, Random Forest, dan Neural Network, serta menunjukkan bahwa metode klasik mampu menghasilkan performa tinggi pada SMS spam classification [3]. Penelitian yang lebih baru juga memperlihatkan bahwa SVM dengan kombinasi fitur tekstual dan metadata mampu mencapai performa yang kuat sekaligus mempertahankan efisiensi implementasi. Temuan serupa dilaporkan pada penelitian berbasis TF-IDF, yang menunjukkan bahwa SVM dan Naïve Bayes tetap menjadi pendekatan relevan untuk pemrosesan teks berdimensi tinggi [4]. Hal ini menegaskan bahwa kualitas preprocessing dan representasi fitur dapat memiliki pengaruh yang sama pentingnya dengan kompleksitas algoritma.

Di sisi lain, pendekatan deep learning menawarkan kemampuan untuk mempelajari representasi dan dependensi sekuensial secara otomatis sehingga tidak sepenuhnya bergantung pada representasi *sparse* seperti Bag-of-Words atau TF-IDF. Penelitian lainnya mengembangkan arsitektur Multi-Channel CNN dengan kombinasi *word embeddings* dan melaporkan performa yang kompetitif pada klasifikasi SMS spam. Pengembangan selanjutnya juga memanfaatkan LSTM, BiLSTM, GRU, CNN, dan mekanisme *attention* untuk menangkap informasi kontekstual serta hubungan antarkata dalam pesan [5]. Studi pada model hybrid CNN-GRU bahkan menunjukkan kemampuan klasifikasi yang sangat tinggi, tetapi sekaligus mengidentifikasi kebutuhan memori dan sumber daya komputasi yang lebih besar sebagai salah satu keterbatasannya [6]. Dengan demikian, peningkatan kompleksitas deep learning belum tentu memberikan keuntungan proporsional, khususnya pada dataset yang relatif kecil.

Permasalahan penting lainnya adalah *class imbalance*, karena ketidakseimbangan distribusi kelas dapat menyebabkan model lebih berorientasi pada kelas mayoritas dan mengabaikan pola kelas minoritas [7]. Salah satu pendekatan yang umum digunakan adalah *oversampling* pada kelas SPAM dan *downsampling* pada kelas HAM. Namun, proses resampling harus dilakukan secara hati-hati karena penerapan *oversampling* sebelum pemisahan data berpotensi menyebabkan *data leakage*, terutama ketika sampel hasil duplikasi muncul pada data training dan testing. Studi mengenai klasifikasi tidak seimbang menunjukkan bahwa *oversampling* dapat meningkatkan *recall* kelas minoritas, tetapi dalam beberapa kondisi juga dapat menurunkan *precision* apabila batas keputusan bergeser terlalu jauh menuju kelas mayoritas [8]. Berdasarkan pertimbangan tersebut, penelitian ini melakukan deduplikasi sebelum *splitting*, mempertahankan validation dan final test tanpa resampling, serta menerapkan balancing hanya pada training set.

Penelitian sebelumnya mengenai klasifikasi SPAM pad email umumnya berfokus pada pengembangan arsitektur [3] [4] [5], penerapan algoritma *machine learning* maupun *deep learning*, serta peningkatan performa model berdasarkan metrik evaluasi tertentu. Namun, aspek keseimbangan distribusi data antara kelas SPAM dan HAM masih relatif kurang mendapat perhatian, padahal ketidakseimbangan kelas dapat menyebabkan model lebih dominan mempelajari pola kelas mayoritas dan menghasilkan evaluasi yang kurang representatif. Oleh karena itu, penelitian ini berupaya mengisi celah tersebut dengan menerapkan pendekatan penyeimbangan dataset sebelum proses pelatihan. Strategi ini diharapkan dapat mengurangi bias terhadap kelas mayoritas, meminimalkan potensi *data leakage*, serta menghasilkan evaluasi klasifikasi SPAM dan HAM yang lebih adil dan reliabel.

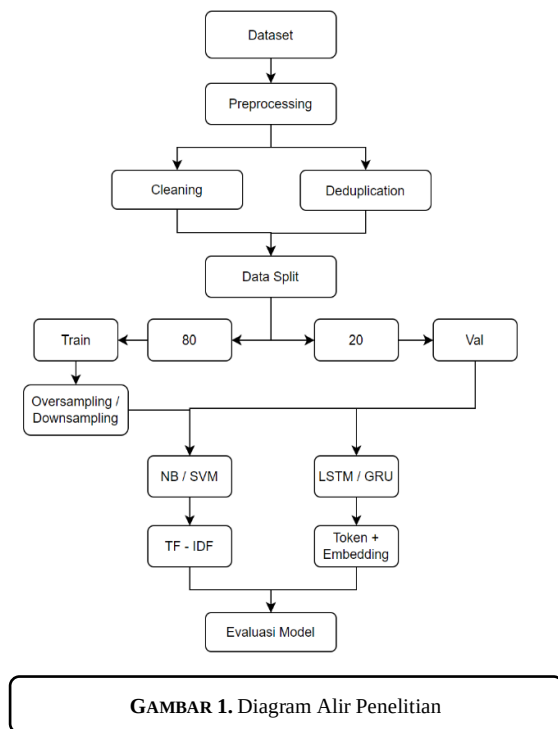
Berdasarkan kajian tersebut, masih terdapat peluang penelitian dalam mengevaluasi secara langsung model machine learning tradisional dan deep learning menggunakan prosedur eksperimen yang konsisten, khususnya ketika data training diseimbangkan tetapi model tetap diuji menggunakan data yang tidak dimodifikasi. Penelitian ini bertujuan untuk: (1) membandingkan performa Naïve Bayes, SVM, LSTM, dan GRU dalam klasifikasi

SPAM/HAM; (2) menganalisis pengaruh *class balancing* terhadap kemampuan mendeteksi kelas SPAM; (3) mengevaluasi karakteristik *false positive* dan *false negative* masing-masing model; serta (4) membandingkan *predictive performance* dengan kebutuhan komputasi selama proses training. Kebaruan penelitian terletak pada evaluasi terpadu antara classification performance, error characteristics, class-balancing strategy, dan computational efficiency, dengan balancing hanya diterapkan pada training set sehingga validation dan final test tetap independen dan lebih merepresentasikan kemampuan generalisasi model

2. METODOLOGI PENELITIAN

2.1. Desain Penelitian dan Dataset

Alur penelitian ditunjukkan pada Gambar 1. Dataset awal dibersihkan melalui lowercasing, penggantian URL dan alamat email dengan token generik, normalisasi angka, penghapusan tag HTML dan karakter nonalfabetik, serta normalisasi spasi. Duplikasi dihapus berdasarkan *clean_text* untuk mencegah pesan identik masuk ke subset yang berbeda. Dataset yang digunakan dalam penelitian ini berasal dari keluarga Email SPAM Classification dari kaggle, yaitu korpus publik berbahasa Inggris yang membedakan pesan menjadi dua kelas utama, yaitu HAM sebagai pesan legitimate dan SPAM sebagai pesan yang tidak diinginkan. Dataset tersebut masih banyak digunakan dalam penelitian terkini karena menyediakan dasar eksperimental yang relatif standar untuk membandingkan berbagai metode klasifikasi spam [9], [10]. Namun, distribusi kelasnya bersifat tidak seimbang karena jumlah HAM secara signifikan lebih besar dibandingkan SPAM. Kondisi ini dapat menyebabkan model cenderung mempelajari kelas mayoritas dan tetap memperoleh *accuracy* tinggi meskipun kemampuan mendeteksi SPAM rendah. Oleh sebab itu, evaluasi perlu memperhatikan Macro-F1, *recall* SPAM, *precision*, PR-AUC, FPR, dan FNR agar performa terhadap kelas minoritas dapat dianalisis secara lebih komprehensif.



Berdasarkan pada Gambar 1, data unik kemudian dibagi secara stratified menjadi development set sebesar 80% dan final test sebesar 20% dengan random seed 42. Development set selanjutnya dipisahkan menjadi raw training dan validation. Akibat desain dua tahap ini, proporsi efektif terhadap seluruh data adalah sekitar 64% raw training, 16% validation, dan 20% final test. Validation dan final test tidak mengalami resampling. Hal ini menjaga evaluasi pada distribusi data yang lebih mendekati kondisi alami dan mencegah duplikasi hasil oversampling masuk ke data evaluasi. *Balancing* hanya diterapkan pada raw training set. Kelas HAM yang berjumlah 2.878 instance di-downsample tanpa replacement menjadi 1.500 instance, sedangkan kelas SPAM yang berjumlah 380 instance di-oversample dengan replacement menjadi 1.500 instance. Dengan demikian training set akhir berjumlah 3.000 instance dengan rasio kelas 1:1. Pendekatan ini sengaja tidak diterapkan pada validation dan final test. Secara metodologis, strategi tersebut mengurangi bias kelas mayoritas tanpa mengorbankan independensi data evaluasi.

Kelemahannya adalah oversampling berbasis replacement mengulang contoh SPAM yang sama, sehingga potensi overfitting akan dipantau melalui validation loss dan early stopping.

2.2. Representasi Fitur dan Konfigurasi Model

Pada model machine learning tradisional, yaitu Naive Bayes (NB) dan Support Vector Machine (SVM), pesan teks terlebih dahulu dikonversi menjadi representasi numerik menggunakan Term Frequency–Inverse Document Frequency (TF-IDF). Metode ini memberikan bobot yang lebih tinggi kepada kata yang sering muncul pada suatu dokumen tetapi relatif jarang ditemukan pada keseluruhan korpus. Dengan demikian, kata-kata yang bersifat diskriminatif terhadap kelas SPAM atau HAM akan memperoleh bobot lebih besar dibandingkan kata-kata umum. Pada penelitian ini, jumlah fitur dibatasi maksimum 20.000 fitur, dengan kombinasi unigram dan bigram untuk menangkap informasi kata individual maupun pasangan kata. Parameter *min_df* = 2 digunakan untuk menghilangkan term yang terlalu jarang, sedangkan *max_df* = 0.95 mengurangi pengaruh term yang muncul hampir di seluruh dokumen. Secara umum, bobot TF-IDF untuk sebuah term *t* pada dokumen *d* dirumuskan sebagai Persamaan (1), di mana "*tf*" (*t*,*d*) menunjukkan frekuensi kemunculan term pada dokumen, *N* merupakan

jumlah keseluruhan dokumen, dan "df" (t) merupakan jumlah dokumen yang mengandung term tersebut. Penelitian ini juga menggunakan sublinear term frequency, sehingga frekuensi kata yang sangat tinggi tidak meningkatkan bobot secara linear. Pendekatan ini penting pada klasifikasi spam karena kata tertentu seperti free, winner, claim, atau call dapat muncul berulang kali dalam satu pesan, tetapi pengulangan tersebut tidak selalu memberikan informasi tambahan secara proporsional.

$$TF-IDF(t, d) = tf(t, d) \times \log\left(\frac{N}{df(t)}\right) \quad (1)$$

Multinomial Naive Bayes (MNB) digunakan sebagai salah satu model baseline karena merupakan algoritma probabilistik yang sederhana, cepat, dan secara tradisional memiliki performa yang baik pada klasifikasi dokumen berbasis fitur frekuensi. Model ini menggunakan Teorema Bayes untuk menghitung probabilitas suatu pesan termasuk ke dalam kelas SPAM atau HAM berdasarkan distribusi fitur yang terdapat di dalam pesan tersebut. Secara konseptual, model memilih kelas c yang memiliki probabilitas posterior terbesar setelah mempertimbangkan fitur x yang mewakili pesan. Hubungan tersebut dapat dinyatakan sebagai:

$$P(c | x) = \frac{P(x|c)P(c)}{P(x)} \quad (2)$$

Pada klasifikasi teks, Multinomial Naive Bayes mengasumsikan bahwa setiap term memberikan kontribusi secara independen terhadap keputusan klasifikasi setelah kelas diketahui. Walaupun asumsi independensi tersebut tidak sepenuhnya sesuai dengan karakteristik bahasa alami, pendekatan ini sering bekerja dengan baik pada data teks berdimensi tinggi dan *sparse*. Dalam penelitian ini, Naive Bayes menerima vektor TF-IDF yang sama dengan SVM sehingga perbandingan kedua algoritma tidak dipengaruhi oleh perbedaan representasi fitur. MNB berfungsi sebagai *probabilistic baseline* untuk mengukur sejauh mana metode yang relatif sederhana dapat membedakan pesan legitimate dan pesan spam. Kelebihan utama model ini adalah waktu pelatihan yang rendah, kebutuhan memori yang relatif kecil, serta proses inferensi yang cepat. Karakteristik tersebut membuat NB relevan apabila sistem *spam filtering* akan diimplementasikan pada lingkungan dengan sumber daya terbatas. Namun demikian, asumsi independensi antarfitur dapat menjadi keterbatasan ketika keputusan klasifikasi bergantung pada konteks atau urutan kata. Oleh karena itu, performanya dibandingkan dengan SVM serta model sekuensial LSTM dan GRU yang mempunyai mekanisme pemodelan hubungan antar-token yang lebih kompleks.

Support Vector Machine (SVM) digunakan sebagai model machine learning diskriminatif yang bekerja dengan mencari *decision boundary* atau *hyperplane* yang menghasilkan margin maksimum antara kelas SPAM dan HAM. Dalam ruang fitur TF-IDF berdimensi tinggi, setiap pesan direpresentasikan sebagai sebuah titik berdasarkan bobot term yang terkandung di dalamnya. SVM kemudian mencari *hyperplane* yang dapat memisahkan kedua *kelompok* data dengan jarak maksimum terhadap sampel yang berada paling dekat dengan batas klasifikasi, yang disebut *support vectors*. Secara umum, fungsi keputusan linear dapat direpresentasikan sebagai:

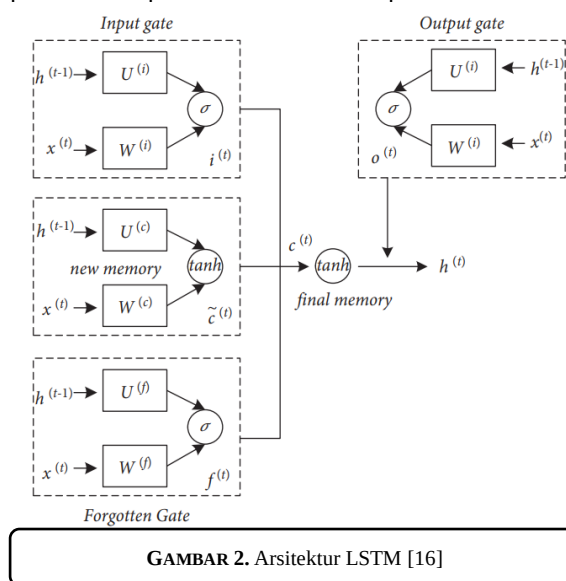
$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b \quad (3)$$

Di mana $\mathbf{x}$ merupakan vektor TF-IDF dari sebuah pesan, $\mathbf{w}$ adalah bobot yang dipelajari oleh model, dan b merupakan bias. Nilai fungsi tersebut menentukan sisi *hyperplane* tempat suatu sampel berada dan selanjutnya menentukan apakah pesan diklasifikasikan sebagai SPAM atau HAM. Penelitian ini menggunakan implementasi LinearSVC dengan parameter regularisasi $C = 1.0$. Pemilihan kernel linear didasarkan pada karakteristik representasi teks TF-IDF yang memiliki dimensi besar tetapi sebagian besar elemennya bernilai nol. Pada kondisi tersebut, LinearSVC umumnya lebih efisien dibandingkan penggunaan kernel nonlinear. Parameter C mengatur kompromi antara lebar margin dan penalti terhadap kesalahan klasifikasi. Nilai $C = 1.0$ digunakan sebagai konfigurasi moderat untuk mempertahankan keseimbangan antara generalisasi dan kemampuan memisahkan data *training*. Karena NB dan SVM menggunakan proses *preprocessing* serta matriks TF-IDF yang identik, perbedaan performa keduanya dapat lebih tepat diinterpretasikan sebagai konsekuensi dari mekanisme pembelajaran model, bukan karena perbedaan *input*. SVM juga menjadi pembanding penting bagi LSTM dan GRU untuk menentukan apakah pemodelan sekuensial memberikan keuntungan yang signifikan dibandingkan representasi tekstual tradisional.

Long Short-Term Memory (LSTM) digunakan untuk merepresentasikan pendekatan deep learning yang mampu mempelajari dependensi antar-token secara berurutan. Berbeda dengan NB dan SVM yang menggunakan vektor TF-IDF tanpa mempertahankan urutan token secara eksplisit, LSTM memproses pesan sebagai sebuah *sequence*. Tokenizer hanya dibangun dari training set untuk menghindari *data leakage* dari validation maupun test set. Jumlah kosakata dibatasi hingga 20.000 token, sedangkan setiap pesan dinormalisasi menjadi *sequence* dengan panjang maksimum 200 token. Pesan yang lebih pendek ditambahkan padding, sedangkan pesan yang melebihi batas maksimum dipotong. Setiap token kemudian dipetakan menjadi representasi *embedding* berdimensi 128, sehingga model dapat mempelajari hubungan semantik token secara langsung selama proses *training* [11].

Lapisan LSTM menggunakan 128 recurrent units yang bertugas memproses sequence dan mempertahankan informasi yang relevan melalui mekanisme *input gate*, *forget gate*, dan *output gate*. Mekanisme tersebut memungkinkan model mempertahankan informasi penting dalam rentang sequence yang relatif panjang sekaligus membuang informasi yang kurang relevan. Output LSTM diteruskan ke Dense layer 64 neuron dengan aktivasi ReLU, diikuti Dropout sebesar 0,3 untuk mengurangi risiko overfitting. Lapisan akhir menggunakan satu neuron dengan aktivasi sigmoid, sehingga menghasilkan probabilitas klasifikasi biner SPAM/HAM. Embedding dikonfigurasi dengan *mask_zero=True* agar token padding tidak ikut memengaruhi recurrent state. Konfigurasi masking ini penting karena sebagian besar email memiliki panjang yang jauh lebih pendek dari 200 token, sehingga tanpa masking representasi padding dapat memberikan noise terhadap representasi akhir sequence.

Gated Recurrent Unit (GRU) digunakan sebagai model recurrent alternatif untuk mengevaluasi apakah arsitektur yang lebih sederhana dapat memberikan performa yang sebanding dengan LSTM. Seperti LSTM, GRU memproses pesan sebagai sequence token sehingga mampu mempertahankan informasi mengenai urutan kata dan dependensi kontekstual. GRU menggunakan preprocessing yang identik dengan LSTM, yaitu tokenizer yang dibangun secara eksklusif pada training set, vocabulary maksimal 20.000 token, sequence maksimum 200 token, serta embedding berdimensi 128 dengan *mask_zero=True*. Konsistensi konfigurasi tersebut bertujuan agar perbedaan performa antara LSTM dan GRU terutama berasal dari perbedaan mekanisme recurrent, bukan akibat perbedaan representasi data atau kapasitas embedding.



GAMBAR 2. Arsitektur LSTM [16]

Perbedaan utama GRU dengan LSTM terletak pada struktur gating yang lebih sederhana. GRU menggunakan dua mekanisme utama, yaitu update gate dan reset gate, sedangkan LSTM menggunakan tiga gate dan cell state yang terpisah. Penyederhanaan tersebut menyebabkan GRU memiliki jumlah parameter lebih sedikit dan secara teoritis dapat memiliki waktu training lebih rendah. Pada penelitian ini, GRU menggunakan 128 recurrent units, kemudian diikuti Dense 64 neuron dengan aktivasi ReLU dan Dropout 0,3. Output akhir menggunakan aktivasi sigmoid untuk menghasilkan probabilitas SPAM/HAM. Dengan kapasitas layer yang dibuat setara dengan LSTM, eksperimen dapat menguji apakah pengurangan kompleksitas arsitektur GRU menghasilkan efisiensi komputasi yang lebih baik tanpa menyebabkan penurunan performa klasifikasi yang signifikan.

Untuk menjaga konsistensi eksperimen, LSTM dan GRU menggunakan konfigurasi training yang sama. Kedua model dioptimasi menggunakan Adam optimizer dengan *learning rate* awal sebesar 0,001 dan *batch size* 32. Maksimum training ditetapkan sebanyak 20 epoch, tetapi proses dapat dihentikan sebelum mencapai jumlah tersebut melalui mekanisme EarlyStopping berdasarkan *validation_loss* dengan *patience*=3. Dengan mekanisme ini, training dihentikan apabila tidak terjadi perbaikan *validation loss* selama tiga epoch berturut-turut, sehingga risiko overfitting dapat dikurangi sekaligus menghindari proses training yang tidak diperlukan.

Selain EarlyStopping, digunakan ReduceLROnPlateau dengan *patience*=2. Mekanisme ini secara otomatis menurunkan *learning rate* ketika *validation loss* tidak mengalami peningkatan selama periode tertentu. Kombinasi EarlyStopping dan adaptive learning rate digunakan agar proses optimasi lebih stabil dan model dapat mendekati minimum loss secara lebih bertahap. Konfigurasi training yang identik antara LSTM dan GRU juga menjadi bagian penting dari desain eksperimen karena memungkinkan perbandingan performa, convergence behavior, serta kebutuhan waktu komputasi secara lebih adil.

TABEL 1. Konfigurasi Utama Model

Model	Representasi / Arsitektur	Konfigurasi Utama
Naive Bayes	TF-IDF $\rightarrow$ MultinomialNB	Max features = 20.000; unigram-bigram; min_df = 2; max_df = 0,95; sublinear TF
SVM	TF-IDF $\rightarrow$ LinearSVC	Max features = 20.000; unigram-bigram; min_df = 2; max_df = 0,95; C = 1,0
LSTM	Embedding $\rightarrow$ LSTM $\rightarrow$ Dense $\rightarrow$ Dropout $\rightarrow$ Sigmoid	Vocabulary = 20.000; max sequence = 200; embedding = 128; LSTM = 128; Dense = 64; dropout = 0,3
GRU	Embedding $\rightarrow$ GRU $\rightarrow$ Dense $\rightarrow$ Dropout $\rightarrow$ Sigmoid	Vocabulary = 20.000; max sequence = 200; embedding = 128; GRU = 128; Dense = 64; dropout = 0,3

Pada Tabel 1, penggunaan konfigurasi yang relatif setara memungkinkan penelitian ini membandingkan dua paradigma klasifikasi teks yang berbeda. NB dan SVM merepresentasikan pendekatan feature-based machine learning, di mana informasi tekstual terlebih dahulu direkayasa menjadi fitur TF-IDF sebelum diberikan kepada classifier. Sebaliknya, LSTM dan GRU merepresentasikan pendekatan representation learning, di mana representasi embedding dan pola sekuensial dipelajari secara langsung selama training. Dengan desain tersebut, perbandingan tidak hanya menunjukkan model yang menghasilkan nilai accuracy atau F1-score tertinggi, tetapi juga memperlihatkan apakah tambahan kompleksitas arsitektur recurrent memberikan keuntungan yang sebanding dengan kebutuhan waktu training dan sumber daya komputasinya.

Penetapan hyperparameter dalam penelitian ini menggunakan konfigurasi statis yang ditentukan berdasarkan praktik yang umum digunakan pada klasifikasi teks serta eksperimen pendahuluan (*preliminary trials*). Pendekatan TF-IDF yang dikombinasikan dengan model linear seperti SVM masih banyak digunakan pada penelitian klasifikasi teks dan deteksi spam terkini karena sesuai untuk representasi teks berdimensi tinggi dan bersifat sparse [12]. Sementara itu, arsitektur recurrent seperti LSTM dan GRU tetap digunakan dalam penelitian spam detection karena kemampuannya memodelkan dependensi sekuensial dan representasi kontekstual pada teks. Berdasarkan pertimbangan tersebut, penelitian ini menetapkan maksimum 20.000 fitur/vocabulary untuk membatasi kompleksitas representasi, $C=1,0$ sebagai konfigurasi regularisasi moderat pada LinearSVC [13], serta embedding 128 dimensi dan 128 recurrent units sebagai kapasitas menengah pada LSTM dan GRU. Nilai-nilai tersebut juga dievaluasi melalui eksperimen pendahuluan untuk memastikan proses training stabil dan kebutuhan komputasi tetap terkendali. Penelitian ini tidak melakukan optimasi hyperparameter secara ekstensif menggunakan Grid Search atau Random Search karena tujuan utama eksperimen adalah membandingkan karakteristik algoritma pada prosedur preprocessing dan class balancing yang konsisten, bukan memperoleh konfigurasi optimum individual untuk setiap model.

2.3. Evaluasi Kinerja

Evaluasi seluruh model dilakukan menggunakan final *test set* yang telah dipisahkan sebelum proses balancing, sehingga distribusi data uji tetap alami dan tidak dipengaruhi oleh *oversampling* maupun *downsampling*. Dalam penelitian ini, SPAM ditetapkan sebagai kelas positif dan HAM sebagai kelas negatif. Kinerja model diukur menggunakan *accuracy*, *precision* SPAM, *recall* SPAM, *specificity* HAM, *F1-score* SPAM, Macro-F1, Weighted-F1, ROC-AUC, PR-AUC, Matthews Correlation Coefficient (MCC), Cohen's Kappa, False Positive Rate (FPR), dan False Negative Rate (FNR). Penggunaan beberapa metrik diperlukan karena *accuracy* saja dapat memberikan gambaran yang terlalu optimistis pada dataset tidak seimbang. Oleh karena itu, metrik berbasis *confusion matrix* dan ukuran agregat digunakan secara bersamaan untuk memperoleh evaluasi yang lebih representatif terhadap kelas mayoritas maupun minoritas [14].

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (5)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (6)$$

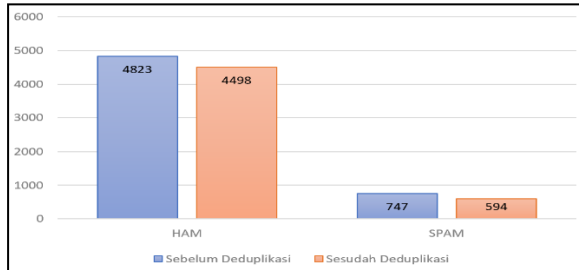
$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (7)$$

Macro-F1 digunakan sebagai metrik komparasi utama karena memberikan bobot yang sama terhadap kelas HAM dan SPAM, sehingga performa kelas minoritas tidak tertutupi oleh dominasi HAM. PR-AUC digunakan sebagai metrik tambahan karena lebih sensitif dalam mengevaluasi kemampuan model mengenali kelas positif pada data tidak seimbang, sedangkan ROC-AUC digunakan untuk menilai kemampuan diskriminasi model pada berbagai *threshold*. *Specificity*, FPR, dan FNR digunakan untuk menganalisis karakteristik kesalahan, khususnya risiko HAM salah diklasifikasikan sebagai SPAM dan SPAM yang lolos sebagai HAM. MCC dan Cohen's Kappa memberikan ukuran tambahan terhadap konsistensi prediksi secara keseluruhan. Selain performa prediktif, penelitian ini juga membandingkan *training time* dan *inference time* untuk mengevaluasi *trade-off* antara performa klasifikasi dan efisiensi komputasi. Eksperimen dijalankan menggunakan Python pada Google Colab, dengan scikit-learn untuk NB dan SVM serta TensorFlow / Keras untuk LSTM dan GRU [15].

3. HASIL DAN PEMBAHASAN

Berdasarkan Dataset yang digunakan, meskipun data yang digunakan merupakan dataset publik yang tersedia melalui Kaggle, distribusi kelas awal menunjukkan ketidakseimbangan yang cukup tinggi antara HAM dan SPAM. Berdasarkan gambar 3, distribusi dataset, sebelum proses deduplikasi terdapat 4.823 data HAM dan 747 data SPAM. Setelah data duplikat dihapus, jumlah tersebut berkurang menjadi 4.498 HAM dan 594 SPAM. Hasil

ini menunjukkan bahwa meskipun dataset yang digunakan merupakan data publik dari Kaggle, distribusi kelas masih sangat tidak seimbang. Setelah deduplikasi, data SPAM hanya sekitar 11,7% dari keseluruhan 5.092 instance, sedangkan HAM mencapai sekitar 88,3%. Ketimpangan tersebut berpotensi menyebabkan model lebih dominan mempelajari karakteristik HAM sehingga kemampuan dalam mengenali kelas SPAM sebagai kelas minoritas dapat menjadi kurang optimal. Oleh karena itu, ketidakseimbangan kelas perlu ditangani sebelum proses pelatihan model.



GAMBAR 3. Komposisi Jumlah Data pada kedua Kelas

TABEL 2. Komposisi Data Splitting dan Proses Balancing Data

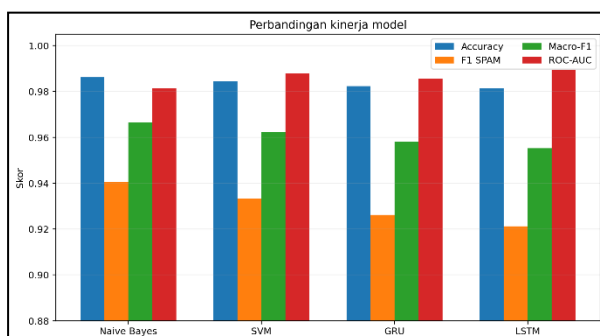
Status	Kelas	TRAIN	VAL	TEST
Tanpa Balancing	HAM	2878	720	900
Tanpa Balancing	SPAM	380	95	119
Dengan Balancing	HAM	1500	720	900
Dengan Balancing	SPAM	1500	95	119

Berdasarkan Tabel 2, balancing diterapkan secara eksklusif pada training set. Sebelum balancing, training set terdiri atas 2.878 HAM dan 380 SPAM. Selanjutnya, kelas HAM di-*downsampling* menjadi 1.500 instance, sedangkan kelas SPAM di-*oversampling* menjadi 1.500 instance sehingga diperoleh rasio training 1:1. Sebaliknya, validation set tetap mempertahankan distribusi asli sebanyak 720 HAM dan 95 SPAM, sedangkan test set terdiri atas 900 HAM dan 119 SPAM. Strategi ini memungkinkan model memperoleh paparan yang seimbang terhadap kedua kelas selama proses training, sementara data validation dan test tetap merepresentasikan distribusi alami. Balancing yang hanya dilakukan pada training set juga bertujuan mencegah sampel hasil *resampling* masuk ke data evaluasi, sehingga risiko *data leakage* dapat diminimalkan dan hasil evaluasi lebih mencerminkan kemampuan generalisasi model.

Evaluasi pada *final test set* dilakukan untuk memperoleh gambaran objektif mengenai kemampuan generalisasi setiap model setelah seluruh proses pelatihan dan optimasi selesai. Perbandingan mencakup dua algoritma *machine learning*, yaitu Naive Bayes dan SVM, serta dua arsitektur *deep learning*, yaitu LSTM dan GRU. Analisis difokuskan pada metrik *accuracy* dan Macro-F1 agar evaluasi tidak hanya mencerminkan ketepatan klasifikasi secara keseluruhan, tetapi juga kemampuan model dalam mempertahankan performa yang seimbang pada kelas HAM dan SPAM.. Tabel 3 dan Gambar 4 menunjukkan kinerja keempat model pada final test set berjumlah 1.019 instance. Naive Bayes memperoleh *accuracy* tertinggi sebesar 0,9863 dan Macro-F1 tertinggi sebesar 0,9665. SVM berada sangat dekat dengan *accuracy* 0,9843 dan Macro-F1 0,9622. Di antara model *deep learning*, GRU memperoleh Macro-F1 0,9581 dan LSTM 0,9553. Selisih antar-model relatif kecil, tetapi hasil tersebut menunjukkan bahwa peningkatan kompleksitas arsitektur tidak otomatis menghasilkan performa klasifikasi yang lebih tinggi pada dataset ini.

TABEL 3. Perbandingan performa model pada final test set

Model	Acc.	Prec.	Recall	F1 SPAM	Macro-F1	ROC-AUC	PR-AUC
Naive Bayes	0.9863	0.9487	0.9328	0.9407	0.9665	0.9815	0.9659
SVM	0.9843	0.9256	0.9412	0.9333	0.9622	0.9879	0.9755
GRU	0.9823	0.9040	0.9496	0.9262	0.9581	0.9856	0.9705
LSTM	0.9814	0.9098	0.9328	0.9212	0.9553	0.9892	0.9735



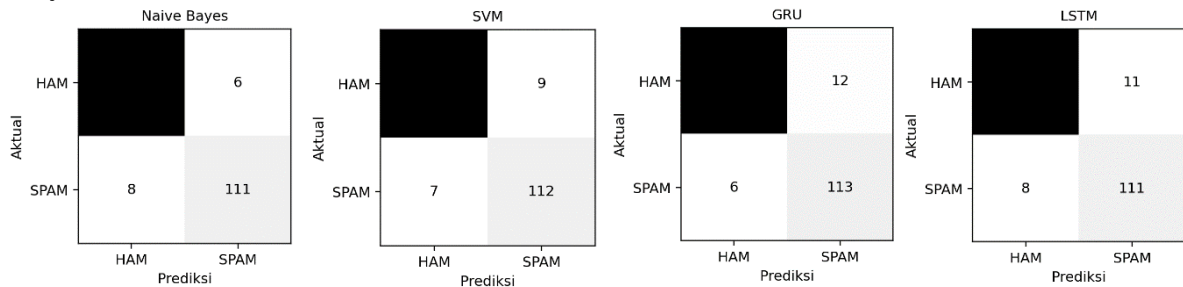
GAMBAR 4. Perbandingan metrik utama keempat model

Pada kelas SPAM, GRU menghasilkan recall tertinggi sebesar 0,9496, diikuti SVM sebesar 0,9412, sedangkan Naive Bayes (NB) dan LSTM masing-masing mencapai 0,9328. Hasil ini menunjukkan bahwa GRU memiliki sensitivitas paling tinggi dalam mendeteksi pesan SPAM dan hanya gagal mengenali enam instance SPAM pada final test set. Meskipun demikian, peningkatan recall tersebut diikuti oleh precision GRU yang lebih rendah, yaitu 0,9040, dibandingkan NB sebesar 0,9487 dan SVM sebesar 0,9256. Kondisi ini menunjukkan bahwa GRU lebih agresif dalam menetapkan pesan sebagai SPAM sehingga

menghasilkan jumlah false positive yang lebih besar. Sebaliknya, NB mampu mempertahankan precision tinggi sekaligus recall yang tetap kompetitif. Kombinasi kedua metrik tersebut menghasilkan F1-score SPAM tertinggi pada NB sebesar 0,9407, sehingga menunjukkan keseimbangan terbaik antara kemampuan mendeteksi SPAM dan menghindari kesalahan klasifikasi pesan HAM.

Dari perspektif kemampuan diskriminasi model pada berbagai threshold, SVM memperoleh PR-AUC tertinggi sebesar 0,9755, sedangkan LSTM menghasilkan ROC-AUC tertinggi sebesar 0,9892. PR-AUC SVM yang tinggi menunjukkan bahwa model mampu mempertahankan trade-off precision dan recall yang sangat baik dalam mengidentifikasi kelas SPAM sebagai kelas positif, terutama pada distribusi data yang tidak seimbang. Sementara itu, ROC-AUC LSTM yang tinggi menunjukkan kemampuan model dalam membedakan SPAM dan HAM secara kontinu meskipun nilai F1-score dan Macro-F1 pada threshold keputusan yang digunakan masih sedikit di bawah beberapa model tradisional. Temuan tersebut menunjukkan bahwa performa LSTM berpotensi ditingkatkan melalui threshold calibration tanpa harus mengubah representasi maupun arsitektur model. Secara keseluruhan, hasil ini juga memperlihatkan bahwa model tradisional seperti NB dan SVM tetap sangat kompetitif ketika representasi TF-IDF mampu menangkap pola diskriminatif utama dalam pesan.

Confusion matrix pada Gambar 5 menunjukkan bahwa setiap model memiliki karakteristik kesalahan yang berbeda dalam membedakan pesan HAM dan SPAM. Naive Bayes menghasilkan 894 true negative (TN), 6 false positive (FP), 8 false negative (FN), dan 111 true positive (TP). SVM memperoleh 891 TN, 9 FP, 7 FN, dan 112 TP, sedangkan GRU mencatat 888 TN, 12 FP, 6 FN, dan 113 TP. Sementara itu, LSTM menghasilkan 889 TN, 11 FP, 8 FN, dan 111 TP. Hasil tersebut menunjukkan bahwa Naive Bayes merupakan model paling konservatif dalam menjaga pesan HAM agar tidak salah diklasifikasikan sebagai SPAM. Sebaliknya, GRU memiliki sensitivitas tertinggi terhadap SPAM, tetapi menghasilkan lebih banyak false positive dibandingkan model lainnya.



GAMBAR 5. Confusion Matrix Pada Empat Model

False positive memiliki implikasi penting dalam sistem penyaringan spam karena pesan legitimate dapat keliru diklasifikasikan sebagai SPAM dan berpotensi tidak terbaca oleh pengguna. Naive Bayes menghasilkan False Positive Rate (FPR) terendah sebesar 0,0067, diikuti SVM sebesar 0,0100, LSTM sebesar 0,0122, dan GRU sebesar 0,0133. Sebaliknya, jika ditinjau dari False Negative Rate (FNR), GRU memberikan hasil terbaik dengan nilai 0,0504 karena hanya enam pesan SPAM yang salah diklasifikasikan sebagai HAM. Perbedaan tersebut menunjukkan adanya trade-off antara kemampuan menjaga pesan HAM dan sensitivitas terhadap SPAM. Oleh karena itu, pemilihan model sebaiknya mempertimbangkan biaya kesalahan operasional: NB lebih sesuai ketika prioritas utama adalah meminimalkan salah blokir pesan sah, sedangkan GRU lebih tepat ketika kegagalan mendeteksi SPAM dianggap memiliki risiko yang lebih besar.

Selain performa prediksi, berdasarkan pada Tabel 3, perbedaan terbesar antar-model terlihat pada efisiensi komputasi. Naive Bayes membutuhkan waktu training sekitar 0,0042 detik dan SVM sekitar 0,0192 detik, sedangkan GRU membutuhkan 6,61 detik dan LSTM 12,95 detik. Dengan konfigurasi eksperimen ini, training LSTM sekitar 3.100 kali lebih lambat daripada NB dan GRU sekitar 1.590 kali lebih lambat daripada NB. Pada inference, SVM menjadi yang tercepat, diikuti NB, sedangkan LSTM dan GRU memerlukan waktu ratusan kali lebih besar karena komputasi sekuensial pada recurrent layer.

TABEL 4. Efisiensi Komputasi

Model	Training time (s)	Inference time (s)	Macro-F1
Naive Bayes	0.0042	0.0017	0.9665
SVM	0.0192	0.0012	0.9622
GRU	6.6131	0.4331	0.9581
LSTM	12.9473	0.4697	0.9553

Hasil ini mendukung argumen bahwa kompleksitas deep learning perlu dievaluasi bersama keuntungan performanya. Pada penelitian ini, NB justru menghasilkan Macro-F1 tertinggi dengan waktu training paling rendah. SVM juga mempertahankan performa sangat tinggi dengan biaya komputasi kecil. Temuan ini relevan

dengan studi efisiensi spam detection yang membandingkan classifier tradisional serta literatur yang menekankan bahwa representasi fitur dan preprocessing dapat sangat menentukan kinerja model. Dengan kata lain, pada dataset teks pendek dengan pola lexical yang kuat, TF-IDF dapat menyediakan fitur diskriminatif yang cukup sehingga recurrent architecture tidak selalu memberi keuntungan bersih.

Strategi balancing berperan penting dalam desain eksperimen. Raw training set memiliki 2.878 HAM dan hanya 380 SPAM. Jika model dilatih langsung pada distribusi tersebut, objective function dapat lebih mudah dioptimalkan dengan memprioritaskan kelas mayoritas. Dengan downsampling HAM menjadi 1.500 dan oversampling SPAM menjadi 1.500, setiap model memperoleh paparan kelas yang seimbang selama training. Namun, karena validation dan final test tetap mempertahankan distribusi alami, peningkatan kemampuan mendeteksi SPAM tidak diperoleh melalui manipulasi data evaluasi. Hal ini membuat accuracy 98% yang diperoleh model lebih bermakna daripada evaluasi pada test set yang juga diseimbangkan secara artifisial.

Pendekatan ini sejalan dengan prinsip penanganan imbalanced data yang menempatkan resampling hanya pada training data [8]. Meskipun oversampling dengan replacement sederhana dan mudah direproduksi, teknik ini tidak menciptakan variasi semantik baru sehingga satu instance SPAM dapat muncul beberapa kali. Dalam penelitian lanjutan, metode text augmentation, focal loss, cost-sensitive learning, atau oversampling pada ruang representasi dapat dibandingkan untuk mengetahui apakah generalisasi kelas minoritas dapat ditingkatkan tanpa meningkatkan risiko overfitting.

Secara konseptual, NB unggul karena asumsi probabilitiknya sangat sesuai dengan sinyal lexical berulang yang umum pada spam, sedangkan SVM efektif memisahkan kelas pada ruang TF-IDF berdimensi tinggi. LSTM dan GRU memiliki kapasitas lebih besar untuk memodelkan dependensi urutan, tetapi keuntungan tersebut lebih nyata ketika urutan dan konteks jangka panjang benar-benar diperlukan. Pada korpus pesan pendek, banyak SPAM dapat dibedakan dari HAM melalui kata, frasa, pola ajakan, nomor, dan n-gram tertentu. Hal ini membantu menjelaskan mengapa model sparse yang lebih sederhana tetap sangat kompetitif. Penelitian terbaru juga menunjukkan bahwa model kompleks, termasuk transformer dan meta-learning, dapat mencapai performa tinggi, tetapi peningkatan tersebut perlu dibaca bersama ukuran data, generalisasi, dan computational complexity.

Terdapat dua keterbatasan utama dalam penelitian ini. Pertama, penelitian hanya menggunakan satu korpus email SPAM/HAM, sehingga karakteristik data yang digunakan belum tentu merepresentasikan seluruh variasi email pada lingkungan nyata. Meskipun dataset memuat pesan email yang dapat digunakan untuk mengevaluasi kemampuan klasifikasi, variasi struktur seperti panjang isi pesan, subject/header, attachment, metadata pengirim, serta pola bahasa yang lebih kompleks masih terbatas. Oleh karena itu, generalisasi hasil terhadap sistem penyaringan email pada skenario produksi perlu dilakukan secara hati-hati. Kedua, eksperimen dilakukan menggunakan satu dataset dan satu random seed, sehingga variasi performa akibat perbedaan pembagian data belum dianalisis secara menyeluruh. Penelitian selanjutnya dapat menggunakan korpus lain seperti Enron-Spam, SpamAssassin, atau TREC, menerapkan repeated stratified split atau cross-validation, serta menggunakan uji statistik seperti McNemar untuk mengevaluasi signifikansi perbedaan prediksi antar-model.

Jika model dipilih berdasarkan satu metrik saja, keputusan yang dihasilkan dapat berbeda. Berdasarkan pada Tabel 5, Naive Bayes merupakan pilihan terbaik ketika tujuan utama adalah kombinasi accuracy, Macro-F1, F1 kelas SPAM, dan minimisasi false positive. SVM lebih menarik ketika kualitas ranking kelas minoritas yang tercermin pada PR-AUC menjadi prioritas. GRU menjadi pilihan ketika recall SPAM diprioritaskan dan organisasi bersedia menerima peningkatan false positive, sedangkan LSTM memiliki ROC-AUC tertinggi sehingga layak dipertimbangkan untuk sistem yang akan melakukan threshold tuning sesuai biaya kesalahan operasional.

TABEL 5. Model terbaik berdasarkan tujuan operasional

<i>Tujuan</i>	<i>Model</i>	<i>Indikator utama</i>
Kinerja keseluruhan	Naive Bayes	Accuracy 0.9863; Macro-F1 0.9665
Menekan false positive	Naive Bayes	FPR 0.0067; specificity 0.9933
Menangkap SPAM maksimum	GRU	Recall SPAM 0.9496; FNR 0.0504
Ranking kelas minoritas	SVM	PR-AUC 0.9755
Kemampuan diskriminatif	LSTM	ROC-AUC 0.9892
Efisiensi training	Naive Bayes	0.0042 detik

Implikasinya, rekomendasi model tidak seharusnya hanya mengikuti model dengan arsitektur paling kompleks. Pada sistem produksi dengan volume pesan tinggi dan sumber daya terbatas, NB atau SVM memberikan keuntungan deployment yang jelas karena waktu training dan inference rendah. Pada sistem keamanan yang menempatkan biaya false negative lebih tinggi daripada false positive, GRU dapat dipilih atau threshold model lain dapat diturunkan. Kerangka keputusan berbasis objective ini juga membuat hasil eksperimen lebih operasional karena menghubungkan metrik statistik dengan konsekuensi penggunaan model.

4. KESIMPULAN

Penelitian ini mengevaluasi Naive Bayes, SVM, LSTM, dan GRU untuk klasifikasi HAM/SPAM dengan class balancing yang hanya diterapkan pada training set. Setelah cleaning dan deduplikasi, data dibagi menjadi development 80% dan final test 20%; raw training kemudian diseimbangkan menjadi 1.500 HAM dan 1.500 SPAM melalui downsampling dan oversampling. Validation dan final test tidak diubah. Hasil menunjukkan bahwa Naive Bayes memberikan performa keseluruhan terbaik dengan accuracy 98,63%, F1 SPAM 94,07%, dan Macro-F1 96,65%. SVM memperoleh PR-AUC tertinggi 97,55%, GRU memperoleh recall SPAM tertinggi 94,96%, dan LSTM menghasilkan ROC-AUC tertinggi 98,92%. Perbedaan tersebut menunjukkan bahwa tidak ada satu model yang unggul pada seluruh objective, tetapi model tradisional mampu memberikan predictive performance yang setara atau lebih tinggi pada beberapa metrik utama. Dari sisi efisiensi, NB dan SVM jauh lebih ringan dibandingkan recurrent deep learning. Waktu training NB hanya sekitar 0,004 detik, SVM 0,019 detik, GRU 6,61 detik, dan LSTM 12,95 detik. Dengan demikian, NB memberikan performance-efficiency trade-off terbaik pada eksperimen ini. Hasil tersebut mendukung hipotesis bahwa kompleksitas deep learning tidak selalu diperlukan untuk klasifikasi spam pada data teks pendek yang memiliki pola lexical kuat. Penelitian mendatang perlu menguji model pada korpus email yang benar-benar merepresentasikan struktur email, mengevaluasi beberapa random seed, melakukan uji signifikansi statistik, serta membandingkan class weighting, focal loss, text augmentation, dan threshold calibration untuk meningkatkan generalisasi kelas minoritas.

REFERENSI

- [1] T. A. Almeida, J. M. G. Hidalgo, and A. Yamakami, "Contributions to the study of SMS spam filtering: new collection and results," in *Proceedings of the 11th ACM Symposium on Document Engineering*, in DocEng '11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 259–262. doi: 10.1145/2034691.2034742.
- [2] U. Maqsood, S. Ur Rehman, T. Ali, K. Mahmood, T. Alsaedi, and M. Kundi, "An Intelligent Framework Based on Deep Learning for SMS and e-mail Spam Detection," *Applied Computational Intelligence and Soft Computing*, vol. 2023, no. 1, p. 6648970, Jan. 2023, doi: <https://doi.org/10.1155/2023/6648970>.
- [3] S. Zavrak and S. Yilmaz, "Email spam detection using hierarchical attention hybrid deep learning method," *Expert Syst. Appl.*, vol. 233, p. 120977, 2023, doi: <https://doi.org/10.1016/j.eswa.2023.120977>.
- [4] P. Wanda, "GRUSpam: robust e-mail spam detection using gated recurrent unit (GRU) algorithm," *International Journal of Information Technology*, vol. 15, no. 8, pp. 4315–4322, 2023, doi: 10.1007/s41870-023-01516-z.
- [5] M. Kshirsagar, V. Rathi, and C. Ryan, "Meta-learner-based frameworks for interpretable email spam detection," *Front. Artif. Intell.*, vol. Volume 8-2025, 2025, doi: 10.3389/frai.2025.1569804.
- [6] Y. P. Qiao, "Spam email classification based on SVM, Transformer and Naive Bayes," *Applied and Computational Engineering*, 2024, [Online]. Available: <https://api.semanticscholar.org/CorpusID:268585650>
- [7] 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, p. 87, 2024, doi: 10.1186/s40537-024-00943-4.
- [8] N. Ramoji Rao and G. A. F. Vinodhini, "Measuring the efficiency of random forest, naive bayes, multilayer perceptron and support vector machine in email spam detection," *AIP Conf. Proc.*, vol. 3270, no. 1, p. 020052, Apr. 2025, doi: 10.1063/5.0262529.
- [9] P. Charanarur, H. Jain, G. S. Rao, D. Samanta, S. S. Sengar, and C. T. Hewage, "Machine-Learning-Based Spam Mail Detector," *SN Comput. Sci.*, vol. 4, no. 6, p. 858, 2023, doi: 10.1007/s42979-023-02330-x.
- [10] H. Amnur, "Customer Relationship Management and Machine Learning Technology for Identifying the Customer," *JOIV: Int. J. Inform. Visualization*, vol. 1, no. 1, pp. 12–15, Mar. 2017, doi: 10.30630/joiv.1.1.10.
- [11] P. Ghogare, H. Dawoodi, and M. Patil, *Enhancing Spam Email Classification Using Effective Preprocessing Strategies and Optimal Machine Learning Algorithms*. 2023. doi: 10.21203/rs.3.rs-3417942/v1.

- [12] E. H. Tusher and M. A. Ismail, "FLSTM-ESD: A hybrid approach to fuzziness based long short-term memory network for email spam detection," *Advanced Engineering Informatics*, vol. 73, p. 104597, 2026, doi: <https://doi.org/10.1016/j.aei.2026.104597>.
- [13] G. Airlangga, "Spam Detection on YouTube Comments Using Advanced Machine Learning Models: A Comparative Study," *Brilliance: Research of Artificial Intelligence*, vol. 4, no. 2, pp. 500–508, Oct. 2024, doi: [10.47709/brilliance.v4i2.4670](https://doi.org/10.47709/brilliance.v4i2.4670).
- [14] R. Arumugam S, A. Deborah S, R. S, M. R S, and M. T T, "SSN_ARMM at SemEval-2024 Task 10: Emotion Detection in Multilingual Code-Mixed Conversations using LinearSVC and TF-IDF," in *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, A. Kr. Ojha, A. S. Doğruöz, H. Tayyar Madabushi, G. Da San Martino, S. Rosenthal, and A. Rosá, Eds., Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 730–736. doi: [10.18653/v1/2024.semeval-1.105](https://doi.org/10.18653/v1/2024.semeval-1.105).
- [15] R. Marriam, R. Mohamad, S. H. Moi, and H. Amnur, "A Comparative Study of Rumor Detection Domains: Machine Learning, Deep Learning, and Statistical Approaches," *JOIV : Int. J. Inform. Visualization*.
- [16] Y. Yang, H. A. Khorshidi, and U. Aickelin, "A review on over-sampling techniques in classification of multi-class imbalanced datasets: insights for medical problems," *Front. Digit. Health*, vol. Volume 6-2024, 2024, doi: [10.3389/fdgth.2024.1430245](https://doi.org/10.3389/fdgth.2024.1430245).
- [17] F. Ares-Robledo, H. Rifà-Pous, and R. Clarisó, "Graph neural networks for anomaly detection: a systematic review of dynamic temporal approaches," *Artif. Intell. Rev.*, vol. 59, no. 5, p. 129, 2026, doi: [10.1007/s10462-026-11532-7](https://doi.org/10.1007/s10462-026-11532-7).
- [18] X. Wen and W. Li, "Time Series Prediction Based on LSTM-Attention-LSTM Model," *IEEE Access*, vol. 11, pp. 48322–48331, 2023, doi: [10.1109/ACCESS.2023.3276628](https://doi.org/10.1109/ACCESS.2023.3276628)