

Komparasi Algoritma Naive Bayes dan Random Forest untuk Identifikasi Kata Berpotensi Spam

Muhammad Fajrin Aswad Ad-Duali¹, Dwika Putra Adinata²

^{1,2}Teknik Informatika, Fakultas Teknik dan Ilmu Komputer, Universitas Nusantara PGRI Kediri

E-mail: ¹fajrinadduali@gmail.com, ²dwikaputraadinata@gmail.com

Abstrak – Penelitian ini mengeksplorasi penerapan *machine learning* untuk mendeteksi kata berpotensi spam pada *Ling-Spam Dataset*, yang menghadapi tantangan distribusi kelas tidak seimbang (*imbalanced*) dalam penyaringan pesan elektronik. Metode *Naive Bayes* dan *Random Forest* digunakan untuk mengatasi masalah inefisiensi filter berbasis aturan statis yang sering gagal mengenali pola promosi dinamis. *Naive Bayes* bekerja sebagai baseline probabilistik, sementara *Random Forest* menerapkan mekanisme ensemble untuk meningkatkan stabilitas prediksi pada fitur leksikal yang kompleks. Dataset terdiri dari 2.893 email yang diproses melalui ekstraksi fitur *TF-IDF*, dengan skema pembagian data terstratifikasi (80% pelatihan, 20% pengujian) guna menjaga proporsi kelas. Hasil eksperimen menunjukkan *Naive Bayes* mencatatkan akurasi global tertinggi sebesar 99,47%, sedangkan *Random Forest* menghasilkan akurasi 98,42% dengan keunggulan nilai presisi sempurna mencapai 100%. Temuan ini membuktikan bahwa pendekatan ensemble lebih efektif dalam meminimalisir kesalahan positif palsu (*false positive*), memberikan solusi filtrasi yang objektif dan andal untuk meningkatkan keamanan email tanpa memblokir pesan yang sah

Kata Kunci — NLP, Email spam, Klasifikasi teks, Machine learning, Naive Bayes, Random Forest

1. PENDAHULUAN

Email spam merupakan masalah yang signifikan dalam komunikasi digital [1]. Pada dasarnya, *email spam* adalah pesan yang tidak diinginkan oleh penerimanya [2]. Kehadiran *email* ini tidak hanya mengganggu aktivitas sehari-hari pengguna [3], tetapi juga sering kali membawa risiko keamanan yang serius [4]. Banyak *spam* dirancang sebagai sarana untuk melakukan pencurian data [5]. Oleh karena itu, kemampuan untuk membedakan secara akurat antara *email* tidak sah dan *email* sah secara efektif sangat penting untuk menjaga keamanan dan kenyamanan pengguna [6].

Tantangan utama yang diangkat dalam penelitian ini adalah proses klasifikasi itu sendiri, bagaimana sebuah sistem dapat membedakan secara efektif antara *email* tidak sah dan *email* sah [7]. Metode berbasis aturan statis yang seringkali gagal karena tidak mampu beradaptasi dengan taktik *spammer* yang terus berkembang. Sebagai solusi untuk masalah ini, pendekatan *Machine Learning* (ML) telah banyak diterapkan dan menjadi pendekatan utama dalam penelitian deteksi spam [8]. Keunggulan utama ML adalah kemampuannya untuk melatih model agar dapat belajar mengenali pola-pola tekstual kompleks yang spesifik pada *email spam*, sesuatu yang sulit dilakukan oleh metode berbasis aturan [9].

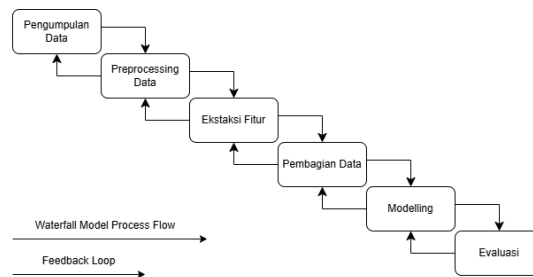
Penelitian terdahulu telah menguji dan memvalidasi performa pendekatan *machine learning*. Sebuah studi komparatif [10], menganalisis beberapa algoritma dan menemukan bahwa *Support Vector Machine* (SVM) mampu memberikan akurasi terbaik hingga 99% dalam mendeteksi *email spam*, mengungguli *Naive Bayes* dan *Logistic Regression*. Penelitian lain [11] menerapkan algoritma *Logistic Regression* yang dikombinasikan dengan metode ekstraksi fitur *TF-IDF* (*Term Frequency-Inverse Document Frequency*) berhasil mencapai akurasi, presisi, dan *recall* sebesar 98%. Penelitian ini menunjukkan bahwa algoritma ML mampu mencapai performa yang sangat tinggi untuk klasifikasi pada *dataset spam* umum.

Penelitian ini akan berfokus pada tantangan tersebut dengan menggunakan *Ling-Spam Dataset* yang bersumber dari *kaggle*, sebuah kumpulan data *email* yang dikurasi dari milis untuk para akademisi dan peminat linguistik. Dataset ini memiliki komposisi yang tidak seimbang, terdiri dari 2.412 pesan sah (*legitimate*) dan 481 pesan *spam*, yang menjadi tantangan tersendiri untuk dipecahkan. Dengan menerapkan pendekatan *machine learning* pada dataset ini, sistem akan dilatih untuk mengenali pola-pola spesifik yang tidak dapat dilakukan oleh filter biasa. Penelitian ini diharapkan mampu membangun model klasifikasi yang akurat dan relevan secara kontekstual.

Penelitian ini dibatasi pada penggunaan *Ling-Spam Dataset*, yang seluruh isi emailnya berbahasa Inggris, sehingga proses analisis, pra-pemrosesan teks, dan pemodelan klasifikasi hanya dilakukan pada dokumen

berbahasa Inggris. Penelitian ini tidak mencakup *email* dengan bahasa lain, termasuk bahasa Indonesia, karena perbedaan karakteristik linguistik dapat memengaruhi proses tokenisasi, *stopword removal*, serta pembentukan fitur.

2. METODE PENELITIAN



Gambar 1. Diagram *Waterfall Model*

2.1 Pengumpulan Data

Dalam penelitian ini, data dikumpulkan dari *Ling Spam Dataset* yang tersedia melalui *Platform Kaggle*. *Dataset* tersebut berisi total 2.893 *email*, terdiri dari 2.412 *email* kategori *ham* (*legitimate*) dan 481 *email* kategori *spam*.

2.2 Preprocessing Data

Agar data yang telah dikumpulkan dapat diolah secara optimal pada tahap pelatihan dan evaluasi model, dilakukan proses pra-pemrosesan yang mencakup *lowercasing* [12] untuk menyamakan teks, *cleaning* [13] untuk menghilangkan karakter yang tidak relevan, serta *stopword removal* agar hanya informasi penting yang dipertahankan [14].

2.3 Ekstraksi Fitur

Setelah melalui tahap pra-pemrosesan, teks yang telah dibersihkan diubah menjadi representasi numerik menggunakan teknik *TF-IDF* [15].

2.4 Pembagian Data

Data yang telah diekstraksi kemudian dibagi menjadi dua bagian, yaitu 80% sebagai data latih dan 20% sebagai data uji. Pembagian dilakukan secara terstratifikasi agar proporsi kelas *spam* dan *ham* tetap seimbang pada kedua subset, sehingga evaluasi model dapat dilakukan secara lebih akurat.

2.5 Modelling

Pada tahap ini, model machine learning dilatih menggunakan fitur yang telah dihasilkan dari proses TF-IDF. Penelitian ini menggunakan dua algoritma klasifikasi, yaitu *Naïve Bayes* [16] sebagai model dasar dan *Random Forest* [17] sebagai model pembanding. Kedua model dilatih menggunakan data latih yang telah dipersiapkan, kemudian digunakan untuk melakukan prediksi pada data uji.

2.6 Evaluasi

Evaluasi dilakukan dengan membandingkan hasil prediksi model terhadap label asli pada data uji. Kinerja setiap model diukur menggunakan beberapa metrik, yaitu akurasi, presisi, *recall*, dan *F1-score*. Selain itu, *confusion matrix* digunakan untuk melihat jumlah prediksi benar dan salah pada masing-masing kelas, sehingga efektivitas model dalam membedakan *spam* dan *ham* dapat dinilai dengan lebih mendalam.

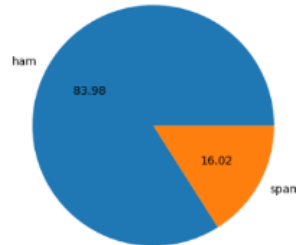
3. HASIL DAN PEMBAHASAN

3.1 Pengumpulan Data

Data yang digunakan dalam penelitian ini berasal dari *Ling-Spam Dataset* yang terdiri dari 2.893 *email*, dengan rincian 2.412 *email ham* dan 481 *email spam*. *Dataset* ini diperoleh dari *Kaggle* dan dipilih karena memiliki karakteristik khusus berupa pesan yang berasal dari milis linguistik, sehingga relevan untuk menguji performa model pada domain teks yang bersifat kontekstual. Distribusi kelas yang tidak seimbang pada *dataset* ini menjadi temuan awal penting, karena kondisi tersebut dapat memengaruhi hasil pelatihan model dan tingkat akurasi yang dicapai. Oleh sebab itu, seluruh proses evaluasi mempertimbangkan karakteristik dataset agar analisis performa model lebih representatif.

3.2 Preprocessing Data

Proses pra-pemrosesan berhasil mengubah data mentah menjadi teks yang lebih bersih dan terstruktur. Tahapan *lowercasing*, *cleaning*, dan *stopword removal* mengurangi banyak elemen yang tidak relevan seperti simbol, angka, format *email*, dan kata-kata umum. Penggabungan *subject* dan *message* juga meningkatkan kedalaman konteks pada setiap *email*. Hasil pra-pemrosesan ini memberikan korpus teks yang lebih representatif, sehingga fitur yang diekstraksi dari data menjadi lebih informatif dan siap digunakan pada tahap pemodelan.



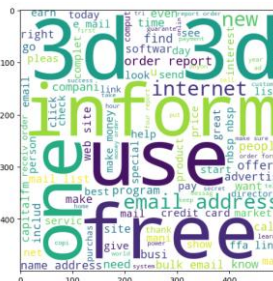
Gambar 2. Pie chart distribusi kelas

Pie chart menunjukkan distribusi kelas pada dataset didominasi oleh kelas *ham* sebesar 83.98%, sementara kelas *spam* hanya mencakup 16.02%. Perbedaan proporsi ini memperlihatkan bahwa *dataset* tidak seimbang, dengan jumlah pesan *ham* jauh lebih banyak dibandingkan *spam*. Ketidakseimbangan ini penting diperhatikan karena dapat memengaruhi performa model dalam membedakan kedua kelas tersebut.

```
1 df['transformed_text'] = df['text'].apply(transform_text)
```

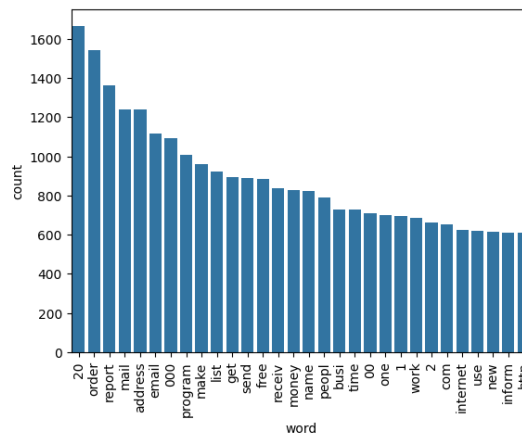
Gambar 3. Tangkapan layar kode program transformasi data

Fungsi *preprocessing transform_text* pada setiap baris kolom *text* di dalam *dataframe*. Hasil transformasi kemudian disimpan ke kolom baru bernama *transformed_text*. Dengan cara ini, setiap teks asli diproses menjadi versi yang telah dibersihkan dan dinormalisasi sehingga siap digunakan pada tahap analisis atau pemodelan.



Gambar 4. WordCloud spam/ham

WordCloud tersebut menggambarkan kata-kata yang paling sering muncul dalam kategori *spam* atau *ham*. Ukuran kata menunjukkan frekuensi kemunculannya semakin besar kata, semakin sering kata tersebut muncul dalam teks. Dari visualisasi terlihat bahwa kata seperti *free*, *use*, *email*, *address*, dan *internet* muncul dengan frekuensi tinggi, yang merupakan ciri umum pesan *spam* yang biasanya menawarkan sesuatu secara gratis, meminta alamat email, atau mempromosikan layanan tertentu. *WordCloud* ini membantu memahami pola kata dominan yang membedakan pesan *spam* dari pesan normal (*ham*).



Gambar 5. Visualisasi Barplot

Barplot tersebut menampilkan kata-kata yang paling sering muncul dalam dataset beserta jumlah kemunculannya. Setiap batang mewakili satu kata, sementara tinggi batang menunjukkan frekuensi kata tersebut. Kata yang berada di sebelah kiri seperti *20*, *order*, *report*, dan *address* muncul dengan jumlah tertinggi, menandakan bahwa kata-kata ini memiliki peran dominan dalam teks, khususnya dalam pesan *spam*. Visualisasi ini membantu mengidentifikasi pola kata yang paling sering digunakan dan memberikan gambaran awal mengenai karakteristik umum dalam dataset.

```
1 X = tfidf.fit_transform(df['transformed_text']).toarray()
```

Gambar 6. Tangkapan layar kode program Transisi ke Ekstraksi Fitur

Fungsi melakukan proses ekstraksi fitur menggunakan *TF-IDF*. Fungsi *fit_transform()* mengubah teks yang sudah melalui tahap preprocessing menjadi representasi numerik berbasis bobot kata, sehingga setiap dokumen direpresentasikan sebagai vektor. Metode ini tidak hanya menghitung frekuensi kata, tetapi juga mempertimbangkan seberapa penting kata tersebut dalam keseluruhan corpus. Hasil akhirnya diubah menjadi *array* menggunakan *.toarray()*, sehingga dapat digunakan sebagai input untuk model *machine learning*.

3.3 Ekstraksi Fitur

Tahap ekstraksi fitur menggunakan *TF-IDF* menghasilkan matriks fitur berdimensi tinggi yang dapat merepresentasikan bobot masing-masing kata dalam *email*. Kata-kata yang sering muncul pada *email spam*, seperti istilah yang bersifat promosi atau ajakan, memiliki bobot lebih tinggi dibandingkan *email ham*. Representasi numerik ini berhasil menangkap perbedaan pola antara kedua kelas, yang terlihat dari sebaran bobot *TF-IDF* yang berbeda. Matriks fitur yang terbentuk kemudian digunakan sebagai input utama untuk proses pelatihan model.

```
1 from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
2 cv = CountVectorizer()
3 tfidf = TfidfVectorizer(max_features=3000)
```

Gambar 7. Tangkapan layar kode program Inisialisasi *TF-IDF*

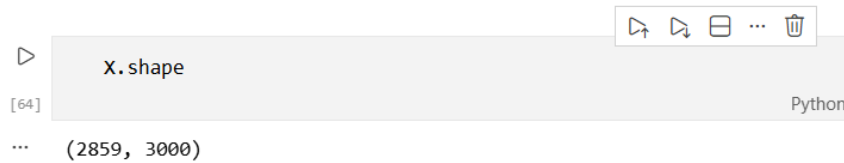
Fungsi menginisialisasi *CountVectorizer* dan *TfidfVectorizer* untuk mengubah teks menjadi fitur numerik. Pada *TfidfVectorizer*, parameter *max_features=3000* digunakan untuk mengambil 3.000 kata paling relevan sebagai fitur utama sebelum data diproses oleh model *machine learning*.

```
1 X = tfidf.fit_transform(df['transformed_text']).toarray()
```

Gambar 8. Tangkapan layar kode program Proses *fit_transform* tekstual ke numerik

Fungsi mengubah teks yang telah diproses pada kolom *transformed_text* menjadi representasi numerik menggunakan metode *TF-IDF*. Fungsi *fit_transform()* menghitung bobot *TF-IDF* untuk setiap kata dan mengubah

seluruh teks menjadi matriks fitur. Metode `.toarray()` kemudian mengonversi hasilnya menjadi *array* numerik yang siap digunakan sebagai *input* untuk model *machine learning*.



Gambar 9. Output *X.shape*

Hasil (2859, 3000) menunjukkan bahwa dataset telah dikonversi menjadi matriks fitur dengan 2.859 sampel dan 3.000 fitur. Artinya, setiap teks direpresentasikan sebagai vektor yang terdiri dari 3.000 nilai *TF-IDF*, dan total ada 2.859 data yang siap digunakan untuk pelatihan maupun pengujian model.

3.4 Pembagian Data

Pembagian dataset menjadi 80% data latih dan 20% data uji menghasilkan dua subset yang seimbang melalui metode stratifikasi. Hasil pembagian menunjukkan bahwa proporsi *spam* dan *ham* tetap terjaga pada kedua subset, sehingga model mendapatkan paparan data yang proporsional selama pelatihan dan evaluasi. Struktur pembagian ini memungkinkan hasil pengujian lebih adil dan representatif, terutama mengingat *dataset* awal memiliki ketidakseimbangan kelas.

```
1 X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

Gambar 10 Tangkapan layar kode program *train_test_split*

Fungsi *train_test_split* digunakan untuk membagi dataset menjadi dua bagian: data latih dan data uji. Data latih digunakan oleh model untuk belajar pola dari *dataset*, sedangkan data uji digunakan untuk menilai kemampuan model dalam melakukan prediksi pada data yang belum pernah dilihat sebelumnya. Parameter seperti *test_size* menentukan proporsi data uji, sementara *random_state* memastikan bahwa proses pembagian dilakukan secara konsisten dan dapat direproduksi.

```
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)

X_train shape: (2287, 3000)
X_test shape: (572, 3000)
```

Gambar 11. Tangkapan layar kode program menampilkan jumlah data *train* dan *test*

Hasil pembagian data menunjukkan bahwa data latih memiliki 2.287 sampel, sedangkan data uji terdiri dari 572 sampel. Kedua set data memiliki jumlah fitur yang sama, yaitu 3.000 fitur per sampel. Ini menunjukkan bahwa proporsi pemisahan data sudah sesuai dan model dilatih menggunakan sebagian besar data yang tersedia, sementara sisanya digunakan untuk menguji performa model secara objektif.

```
print("Distribusi kelas pada data latih:")
print(pd.Series(y_train).value_counts())

print("\nDistribusi kelas pada data uji:")
print(pd.Series(y_test).value_counts())

✓ 0.0s

Distribusi kelas pada data latih:
0    1928
1     359
Name: count, dtype: int64

Distribusi kelas pada data uji:
0     473
1      99
Name: count, dtype: int64
```

Gambar 12. Distribusi kelas pada data latih dan data uji

Distribusi kelas pada data latih menunjukkan jumlah kelas 0 sebanyak 1.928 data dan kelas 1 sebanyak 359 data. Sementara itu, pada data uji terdapat 473 data untuk kelas 0 dan 99 data untuk kelas 1. Hasil ini memperlihatkan bahwa kedua *dataset* memiliki komposisi yang tidak seimbang, di mana kelas 0 jauh lebih dominan dibandingkan kelas 1. Meskipun demikian, proporsi antara data latih dan data uji tetap konsisten, sehingga model masih dapat belajar dengan pola distribusi yang serupa.

3.5 Modelling

Pada tahap pemodelan, kedua algoritma *Naive Bayes* dan *Random Forest* telah dilatih menggunakan fitur *TF-IDF*. Hasil pelatihan menunjukkan bahwa *Naive Bayes* mampu belajar dengan cepat namun cenderung kurang akurat dalam mengenali *email spam*, terutama ketika pola bahasa lebih kompleks. Sebaliknya, *Random Forest* menunjukkan performa lebih stabil karena kemampuannya menggabungkan banyak pohon keputusan. Model ini lebih baik dalam mengidentifikasi pola-pola *spam* yang tidak sederhana. Secara keseluruhan, hasil pelatihan mengindikasikan bahwa *Random Forest* memiliki potensi lebih besar dalam menangani dataset kontekstual seperti *Ling-Spam*.

```
1 def train_classifier(clf,X_train,y_train,X_test,y_test):  
2     clf.fit(X_train,y_train)
```

Gambar 13. Tangkapan layar kode program proses pelatihan model

Fungsi *train_classifier* berfungsi sebagai prosedur umum untuk melatih setiap model klasifikasi. Model yang diberikan sebagai parameter (*clf*) akan dilatih menggunakan data latih melalui perintah *clf.fit(X_train, y_train)*. Pendekatan ini memungkinkan semua model yang diuji, termasuk *Naive Bayes* dan *Random Forest*, diproses dengan cara yang konsisten.

3.6 Evaluasi

Evaluasi menggunakan data uji menunjukkan bahwa *Random Forest* memberikan nilai akurasi, presisi, *recall*, dan *F1-score* yang lebih tinggi dibandingkan *Naive Bayes*. *Confusion matrix* memperlihatkan bahwa *Random Forest* mampu mengurangi jumlah prediksi salah, terutama pada klasifikasi *email spam*. Hal ini sangat penting mengingat jumlah data *spam* yang lebih sedikit sehingga model lebih mudah mengalami bias. Hasil evaluasi ini menegaskan bahwa model berbasis *ensemble* lebih mampu mengenali karakteristik *spam* dalam *dataset* yang kompleks dan tidak seimbang.

```
mnb.fit(X_train,y_train)  
y_pred2 = mnb.predict(X_test)  
print(accuracy_score(y_test,y_pred2))  
print(confusion_matrix(y_test,y_pred2))  
print(precision_score(y_test,y_pred2))  
✓ 0.0s  
0.9947552447552448  
[[472  1]  
 [ 2 97]]  
0.9897959183673469
```

Gambar 14. Tangkapan layar kode program *Confusion Matrix Naive Bayes (NB)*

Fungsi evaluasi *Naive Bayes* yang menunjukkan performa model dalam memprediksi kelas *spam* dan *ham* melalui metrik akurasi, *precision*, serta *confusion matrix*. Tahap ini digunakan untuk melihat efektivitas *baseline* model sebelum dibandingkan dengan algoritma lain seperti *Random Forest*.

```
rfc.fit(X_train, y_train)
y_pred_rf = rfc.predict(X_test)

print("Accuracy RF:", accuracy_score(y_test, y_pred_rf))
print("Precision RF:", precision_score(y_test, y_pred_rf))
print("Confusion Matrix RF:")
print(confusion_matrix(y_test, y_pred_rf))
```

✓ 1.4s

Accuracy RF: 0.9842657342657343
Precision RF: 1.0
Confusion Matrix RF:
[[473 0]
 [9 90]]

Gambar 15. Tangkapan layar kode program *Confusion Matrix Random Forest (RF)*

Fungsi evaluasi *Random Forest* yang menghitung akurasi, *precision*, dan *confusion matrix*. Evaluasi ini menunjukkan bahwa *Random Forest* mampu menghasilkan prediksi yang lebih konsisten dan minim kesalahan pada kelas *spam*, sehingga menjadi model dengan performa terbaik dalam penelitian ini.

```
mnb.fit(X_train,y_train)
y_pred2 = mnb.predict(X_test)
print(accuracy_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))
print(precision_score(y_test,y_pred2))

# Tambahan recall dan f1
print("Recall NB:", recall_score(y_test, y_pred2))
print("F1-score NB:", f1_score(y_test, y_pred2))
```

✓ 0.0s

0.9947552447552448
[[472 1]
 [2 97]]
0.9897959183673469
Recall NB: 0.9797979797979798
F1-score NB: 0.9847715736040609

Gambar 16. Tangkapan layar kode program Metrik Evaluasi (*Accuracy, Precision, Recall, F1-score*)

Performa *Naive Bayes* pada data uji menunjukkan akurasi tinggi namun *precision/recall* yang sedikit lebih rendah dibanding model *ensemble*, hal ini menggambarkan sifat *Naive Bayes* yang efisien pada teks namun rentan terhadap pola bahasa yang kompleks.

```
rfc.fit(X_train, y_train)
y_pred_rf = rfc.predict(X_test)

print("Accuracy RF:", accuracy_score(y_test, y_pred_rf))
print("Precision RF:", precision_score(y_test, y_pred_rf))
print("Confusion Matrix RF:")
print(confusion_matrix(y_test, y_pred_rf))

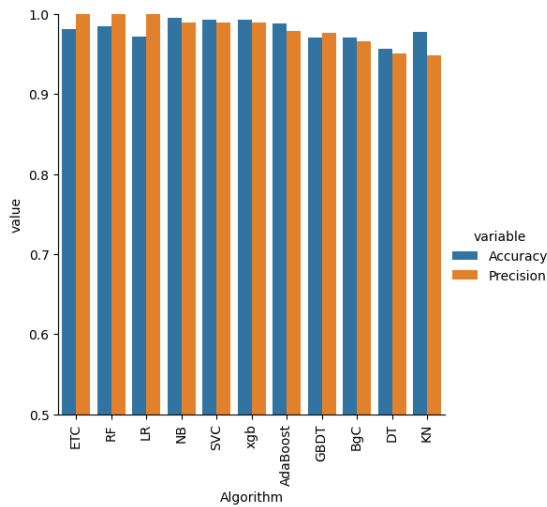
# Tambahan recall dan f1
print("Recall RF:", recall_score(y_test, y_pred_rf))
print("F1-score RF:", f1_score(y_test, y_pred_rf))
```

✓ 1.4s

Accuracy RF: 0.9842657342657343
Precision RF: 1.0
Confusion Matrix RF:
[[473 0]
 [9 90]]
Recall RF: 0.9090909090909091
F1-score RF: 0.9523809523809523

Gambar 17. Tangkapan layar kode program Metrik Evaluasi (*Accuracy, Precision, Recall, F1-score*)

Random Forest menunjukkan kombinasi metrik terbaik (*precision* dan *recall* tinggi), menandakan kemampuan model ini mengurangi kesalahan klasifikasi pada kelas minoritas (*spam*) dan memberikan performa lebih stabil dibanding *baseline*.



Gambar 18. Grafik Perbandingan Kinerja Model

Grafik perbandingan akurasi dan *precision* untuk seluruh algoritma klasifikasi. Visualisasi ini menunjukkan bahwa beberapa model, termasuk *Random Forest* dan *Naive Bayes*, memiliki performa unggul dibanding algoritma lainnya.

NB	0.9947552447552448	0.9897959183673469
RF	0.9842657342657343	1.0

Gambar 19. Highlight Grafik / Tabel *Random Forest* Terbaik

Hasil evaluasi menunjukkan bahwa *Naive Bayes* memperoleh akurasi lebih tinggi (0.9947), sedangkan *Random Forest* mencapai presisi sempurna (1.0). Artinya, *Naive Bayes* lebih unggul dalam prediksi keseluruhan, sementara *Random Forest* lebih tepat dalam memastikan setiap pantun yang terdeteksi benar-benar valid. Perbedaannya tipis, namun kedua model sama-sama menunjukkan performa yang sangat baik.

4. SIMPULAN

Dengan mempertimbangkan hasil evaluasi yang dilakukan, beberapa hal dapat disimpulkan sebagai berikut:

1. *Ling-Spam Dataset* memiliki karakteristik data yang tidak seimbang, yaitu jumlah *email ham* jauh lebih banyak dibandingkan *email spam*. Kondisi ini memengaruhi tingkat kesulitan model dalam mengenali *spam*, sehingga pemilihan metode pengolahan dan evaluasi menjadi faktor penting dalam proses klasifikasi.
2. Tahapan pra-pemrosesan seperti *lowercasing*, pembersihan karakter, dan *stopword removal* memberikan kontribusi signifikan dalam meningkatkan kualitas data. Pembersihan teks dan penggabungan *subject* serta *message* menghasilkan representasi yang lebih jelas sehingga model lebih mudah mengenali pola yang membedakan spam dari ham
3. Penggunaan teknik *TF-IDF* dalam proses ekstraksi fitur memberikan representasi numerik yang efektif. *TF-IDF* mampu menonjolkan kata-kata yang memiliki relevansi tinggi terhadap kategori *spam*, sehingga membantu model dalam melakukan pemisahan kelas secara lebih akurat.
4. Hasil pengujian menunjukkan bahwa algoritma *Naive Bayes* memperoleh akurasi tertinggi sebesar 99,47%, namun memiliki nilai *precision* dan *recall* yang lebih rendah pada kelas spam.
5. Algoritma *Random Forest* mencapai akurasi sebesar 98,43% dengan *precision* sebesar 100%, serta nilai *recall* dan *F1-score* yang lebih stabil. Hal ini menunjukkan bahwa *Random Forest* lebih andal dalam mendeteksi *email spam* pada *dataset* yang tidak seimbang.

6. Implementasi metode klasifikasi berbasis *machine learning* ini dapat diterapkan secara luas untuk sistem deteksi *spam* pada berbagai platform *email*. Dengan pengembangan lebih lanjut, sistem ini berpotensi membantu pengguna dalam menyaring pesan tidak diinginkan secara otomatis, meningkatkan keamanan, dan mengurangi risiko terhadap berbagai ancaman siber.

5. SARAN

Penelitian selanjutnya dapat mempertimbangkan beberapa pengembangan untuk meningkatkan hasil klasifikasi. Pertama, penggunaan teknik penyeimbangan data seperti *SMOTE* atau *oversampling* dapat dilakukan untuk menangani distribusi kelas yang tidak seimbang. Kedua, model dapat diperluas dengan mencoba algoritma lain seperti *SVM* atau *XGBoost* guna memperoleh perbandingan performa yang lebih luas. Ketiga, evaluasi dapat ditingkatkan dengan menambahkan pengujian pada dataset lain agar model lebih general dan tidak hanya bergantung pada karakteristik *Ling-Spam Dataset*. Selain itu, integrasi teknik *word embedding* seperti *Word2Vec* atau *BERT* dapat dijadikan alternatif untuk menghasilkan representasi fitur yang lebih kaya.

DAFTAR PUSTAKA

- [1] M. Basil Musyaffa Amin *et al.*, “Deteksi Spam Berbahasa Indonesia Berbasis Teks Menggunakan Model Bert,” *J. Teknol. Inf. dan Ilmu Komput.*, vol. 11, no. 6, pp. 1291–1302, Dec. 2024, doi: 10.25126/JTIK.2024118121.
- [2] M. Spam *et al.*, “Memahami Spam Terhadap Digitalisasi Masyarakat Desa Perkebunan Sei Balai Kecamatan Sei Balai Kabupaten Batubara,” *J. Pengabd. Masy.*, vol. 2, no. 2, pp. 151–159, Dec. 2023, doi: 10.70340/JAPAMAS.V2I2.90.
- [3] M. Sari and G. Mahalisa, “NAIVE BAYES CLASSIFIER UNTUK DETEKSI EMAIL SPAM - Google Scholar,” *J. Inform.*, vol. 15, no. 4, pp. 675–680, 2024, [Online]. Available: https://scholar.google.com/scholar?hl=id&as_sdt=0%2C5&q=NAIVE+BAYES+CLASSIFIER+UNTU K+DETEKSI+EMAIL+SPAM&btnG=
- [4] M. Rustam, A. Brotokuncoro, and R. Roestam, “Deteksi Email Spam dengan Continuous Bag-Of-Words dan Random Forest,” *Ranah Res. J. Multidiscip. Res. Dev.*, vol. 6, no. 4, pp. 758–765, Jun. 2024, doi: 10.38035/RRJ.V6I4.873.
- [5] S. A. Aklani, H. Haeruddin, and N. Putri, “IMPLEMENTASI MAIL GATEWAY SECURITY DALAM MENINGKATKAN KEAMANAN EMAIL,” *J. Inf. Syst. Manag.*, vol. 5, no. 2, pp. 150–155, Jan. 2024, doi: 10.24076/JOISM.2024V5I2.1378.
- [6] M. A. Alfayyed, D. Sandra, and Jasmir, “Perbandingan Algoritma Naive Bayes Dan K-Nearest Neighbor Pada Klasifikasi Email,” *J. Inform. Dan Rekayasa Komputer(JAKAKOM)*, vol. 5, no. 2, pp. 1696–1705, Sep. 2025, doi: 10.33998/JAKAKOM.2025.5.2.2378.
- [7] H. Mukhtar, J. Al Amien, and M. A. Rucyat, “Filtering Spam Email menggunakan Algoritma Naïve Bayes,” *J. CoSciTech (Computer Sci. Inf. Technol.)*, vol. 3, no. 1, pp. 9–19, 2022, doi: 10.37859/coscitech.v3i1.3652.
- [8] A. Nur *et al.*, “Implementasi Algoritma Regresi Logistik untuk Binary Classification dalam Spam SMS dan WhatsApp,” *Pros. SEMNAS INOTEK (Seminar Nas. Inov. Teknol.)*, vol. 7, no. 1, pp. 80–93, Jul. 2023, doi: 10.29407/INOTEK.V7I1.3413.
- [9] G. Putra, “KLASIFIKASI EMAIL SPAM MENGGUNAKAN ALGORITMA ARTIFICIAL NEURAL NETWORK DAN SUPPORT VECTOR MACHINE,” *J. Komput. dan Inform.*, vol. 20, no. 1, pp. 09–15, Apr. 2025, Accessed: Nov. 05, 2025. [Online]. Available: <https://journal.untar.ac.id/index.php/JKI/article/view/34646>
- [10] E. S. Ainun, U. Inayah, and M. Ilmih, “Klasifikasi Email Spam Dan Ham Menggunakan Algoritma Support Vector Machine, Naive Bayes Dan Logistic Regression,” *Sci. J. Comput. Sci. Informatics*, vol. 2, no. 2, pp. 77–84, Jul. 2025, doi: 10.34304/SCIENTIFIC.V2I2.399.
- [11] A. Purnama and D. Hamidin, “Metode Algoritma Logistic Regression dalam Klasifikasi Email Spam,” *J. Software, Hardw. Inf. Technol.*, vol. 5, no. 1, pp. 39–47, Jan. 2025, doi: 10.24252/SHIFT.V5I1.159.
- [12] N. M. Damayanti, I. D. Ariningtyas, M. Izuddin, A. Icham, and A. P. Sari, “ANALISIS SENTIMEN PUBLIK PADA TAGAR #BTSCOMEBACK DI PLATFORM X MENGGUNAKAN INDOBERTWEET,” *J. Inform. dan Tek. Elektro Terap.*, vol. 13, no. 3, pp. 2830–7062, Jul. 2025, doi:

- 10.23960/JITET.V13I3.7176.
- [13] S. H. Rukiman and A. Rahmatulloh, “Evaluasi Efektivitas Penggunaan FastText Embedding dan LSTM Networks dalam Deteksi Phishing Email,” *Fakt. Exacta*, vol. 18, no. 2, pp. 194–200, Oct. 2025, doi: 10.30998/FAKTOREXACTA.V18I2.26769.
 - [14] A. H. Nurridha, W. Hidayat, and A. Erfina, “Sentiment Analysis of the Issue of Eliminating the Independent Curriculum using the Naïve Bayes Classifier Algorithm,” *Sist. J. Sist. Inf.*, vol. 14, no. 2, pp. 713–725, Mar. 2025, doi: 10.32520/STMSI.V14I2.5039.
 - [15] M. P. Syah, A. P. Wardani, M. Idhom, and Trimono, “Perbandingan Representasi Teks Tf-Idf Dan Bert Terhadap Akurasi Cosine Similarity Dalam Penilaian Otomatis Jawaban Berbasis Teks,” *Data Sci. Indones.*, vol. 5, no. 1, pp. 47–59, Jul. 2025, doi: 10.47709/DSI.V5I1.6021.
 - [16] F. Agil Firmansyah, U. Enri, I. Maulana, J. H. Ronggowaluyo, T. Timur, and J. Barat, “PENERAPAN ALGORITMA NAIVE BAYES DENGAN CHI-SQUARE UNTUK KLASIFIKASI SPAM EMAIL BERBASIS KATA DAN FREKUENSI,” *J. Inform. dan Tek. Elektro Terap.*, vol. 13, no. 1, pp. 2830–7062, Jan. 2025, doi: 10.23960/JITET.V13I1.5506.
 - [17] A. K. Kencana, F. D. Ananda, A. D. Hartanto, and H. Hartatik, “Implementasi Metode Random Forest Klasifikasi untuk Phishing Link Detection,” *Intechno J. Inf. Technol. J.*, vol. 4, no. 2, pp. 55–59, Dec. 2022, doi: 10.24076/INTECHNOJOURNAL.2022V4I2.1562.