PERANCANGAN SISTEM DETEKSI EMAIL SPAM BERBASIS MACHINE LEARNING DAN NATURAL LANGUAGE PROCESSING

Dosen Pengampu : Febriyanti Darnis, S.ST., M.Kom

Disusun oleh :

Gathan Rafii Manaf : 3337220117

Andiko Ramadani : 3337230003

Ismet Maulana Azhari : 3337230014

Aura Salsa Azzahra : 3337230044

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS SULTAN AGENG TIRTAYASA

2025

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Allah SWT karena atas rahmat dan karunia-Nya, kami dapat menyelesaikan laporan tugas akhir semester yang berjudul:

“Perancangan Sistem Deteksi Email Spam Berbasis Machine Learning dan Natural Language Processing”

Laporan ini disusun sebagai bagian dari pemenuhan tugas pada mata kuliah *Kecerdasan Artifisial Lanjutan* di Program Studi Informatika, Fakultas Teknik, Universitas Sultan Ageng Tirtayasa. Dalam laporan ini, kami merancang, mengimplementasikan, dan mengevaluasi sistem klasifikasi email spam dengan pendekatan machine learning berbasis pemrosesan teks dan fitur numerik, guna menanggulangi ancaman keamanan komunikasi digital yang semakin kompleks. Proses penyusunan laporan ini tidak lepas dari berbagai tantangan, baik teknis maupun konseptual. Namun dengan kerja sama tim, dukungan pembelajaran selama perkuliahan, serta ketekunan dalam menyusun eksperimen dan dokumentasi, laporan ini akhirnya dapat diselesaikan secara menyeluruh.

Kami menyampaikan terima kasih yang sebesar-besarnya kepada Ibu **Febriyanti Darnis, S.ST., M.Kom**, selaku dosen pengampu mata kuliah, atas bimbingan, motivasi, dan ilmu yang diberikan selama proses perkuliahan berlangsung. Ucapan terima kasih juga kami sampaikan kepada seluruh pihak yang telah memberikan dukungan, baik secara langsung maupun tidak langsung. Kami menyadari bahwa laporan ini masih memiliki kekurangan dan keterbatasan. Oleh karena itu, kami sangat terbuka terhadap saran dan masukan yang membangun demi perbaikan di masa yang akan datang.

Akhir kata, semoga laporan ini dapat memberikan manfaat, tidak hanya sebagai pemenuhan tugas akademik, tetapi juga sebagai kontribusi kecil kami dalam pengembangan sistem berbasis kecerdasan buatan di bidang keamanan informasi.

Serang, Juni 2025

Tim Penyusun

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Di tengah pesatnya perkembangan teknologi informasi, penggunaan email telah menjadi bagian integral dalam komunikasi digital modern. Baik dalam konteks personal, profesional, maupun institusional, email memiliki peran yang sangat penting sebagai media komunikasi utama. Namun, seiring dengan meningkatnya intensitas dan volume penggunaan email, muncul pula berbagai tantangan yang mengancam kenyamanan dan keamanan penggunanya. Salah satu ancaman terbesar yang sering kali dihadapi adalah kemunculan spam email.

Spam email umumnya dikirim secara massal tanpa diminta, sering kali berisi konten tidak relevan, menyesatkan, atau bahkan mengandung potensi bahaya. Jenis pesan ini dapat menjadi jalur masuk bagi berbagai threat serius seperti phishing, malware, hingga pencurian informasi pribadi. Kehadiran spam tidak hanya mengganggu alur komunikasi, tetapi juga berisiko tinggi dalam membuka celah keamanan siber. Ancaman tersebut mendorong perlunya solusi yang tidak hanya andal secara teknis, tetapi juga mampu beradaptasi secara cerdas terhadap pola spam yang terus berkembang.

Menghadapi tantangan ini, pendekatan berbasis Artificial Intelligence (AI), khususnya dalam ranah Natural Language Processing (NLP) dan machine learning, muncul sebagai alternatif yang menjanjikan. NLP memungkinkan sistem untuk memahami struktur, pola, dan konteks bahasa manusia, sehingga sangat efektif digunakan dalam klasifikasi pesan berbasis teks. Ketika dikombinasikan dengan algoritma klasifikasi seperti Naïve Bayes, Support Vector Machine (SVM), atau Decision Tree, sistem deteksi dapat dilatih untuk membedakan antara pesan spam dan non-spam berdasarkan fitur linguistik yang diekstraksi dari teks email.

Penelitian ini bertujuan untuk merancang sistem deteksi email spam berbasis klasifikasi teks dan teknik NLP dengan mengandalkan dataset publik bernama SpamAssassin. Sistem yang dikembangkan diharapkan mampu mendeteksi pesan spam secara otomatis dengan akurasi yang tinggi, sekaligus dapat memberikan kontribusi dalam meningkatkan keamanan komunikasi digital pada layanan email modern.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, rumusan masalah dalam penelitian ini dirumuskan sebagai berikut:

1. Bagaimana cara merancang sistem deteksi spam email yang efektif menggunakan metode klasifikasi teks dan pendekatan NLP?
2. Fitur-fitur teks apa saja yang relevan dan berperan signifikan dalam proses klasifikasi email spam?
3. Sejauh mana sistem yang dirancang mampu mengidentifikasi pesan spam secara akurat dibandingkan pendekatan konvensional?

1.3 Tujuan Penelitian

Penelitian ini memiliki beberapa tujuan utama, antara lain:

1. Merancang dan mengimplementasikan sistem deteksi email spam berbasis NLP dan klasifikasi teks untuk meningkatkan keamanan komunikasi digital.
2. Mengidentifikasi fitur-fitur linguistik yang paling efektif digunakan dalam proses klasifikasi spam dan non-spam.
3. Mengevaluasi performa sistem deteksi spam berdasarkan metrik klasifikasi, serta membandingkan hasilnya dengan metode tradisional yang sudah ada.

1.4 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan sejumlah manfaat, baik dari segi teoritis maupun praktis. Secara teoritis, penelitian ini berkontribusi dalam pengembangan sistem kecerdasan buatan untuk domain keamanan siber, khususnya dalam pemrosesan bahasa alami. Secara praktis, sistem yang dirancang dapat digunakan sebagai fondasi awal dalam pengembangan tools deteksi spam yang dapat diintegrasikan ke dalam layanan email masa kini. Dengan demikian, penelitian ini juga dapat menjadi referensi bagi pengembang software keamanan, akademisi, dan praktisi teknologi informasi yang tertarik pada implementasi AI untuk kebutuhan nyata.

1.5 Ruang Lingkup dan Batasan

Agar penelitian ini tetap fokus dan terarah, terdapat beberapa batasan yang ditetapkan:

- Sistem yang dikembangkan hanya menangani konten teks dari email, yakni pada bagian subject dan body, tanpa memperhitungkan lampiran (attachments) maupun metadata lainnya.
- Dataset yang digunakan bersifat publik dan open-source, dengan contoh utama yaitu SpamAssassin. Tidak digunakan data email personal atau institusional yang bersifat privat.
- Penelitian difokuskan pada tahap pengembangan dan evaluasi model klasifikasi, tanpa mencakup pengembangan antarmuka pengguna (UI/UX) atau integrasi langsung ke platform email seperti Gmail atau Outlook.
- Pendekatan yang digunakan terbatas pada metode klasifikasi tradisional berbasis NLP, seperti Naïve Bayes dan SVM, tanpa melibatkan arsitektur deep learning lanjutan seperti Transformer atau LSTM.

1.6 Sistematika Penulisan

Laporan ini disusun dengan sistematika sebagai berikut:

- **Bab I: Pendahuluan**
  Berisi latar belakang masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, ruang lingkup dan batasan, serta sistematika penulisan.

- **Bab II: Tinjauan Pustaka**
  Membahas teori-teori dasar terkait kecerdasan buatan, klasifikasi teks, algoritma yang digunakan, serta penelitian terdahulu yang relevan.

- **Bab III: Metodologi Penelitian**
  Menjelaskan jenis dan desain penelitian, dataset, arsitektur sistem, tahapan implementasi, evaluasi model, serta tools yang digunakan.

- **Bab IV: Hasil dan Pembahasan**
  menyajikan hasil eksperimen, visualisasi data, analisis performa model, dan pembahasan terhadap hasil yang diperoleh.

- **Bab V: Kesimpulan dan Saran**
  berisi ringkasan hasil penelitian, keterbatasan sistem yang dirancang, dan saran untuk pengembangan lebih lanjut di masa mendatang.

BAB II
TINJAUAN PUSTAKA

2.1 Teori Umum Kecerdasan Buatan Lanjutan

Kecerdasan Buatan atau *Artificial Intelligence (AI)* merupakan cabang ilmu komputer yang bertujuan untuk menciptakan sistem yang mampu meniru dan mengeksekusi fungsi-fungsi intelektual manusia. Dalam perkembangannya, AI tidak hanya difokuskan pada kemampuan pemrosesan aturan atau automasi sederhana, melainkan telah berkembang menjadi sistem yang dapat *belajar* dari data, mengenali pola yang kompleks, dan mengambil keputusan berbasis probabilistik maupun inferensi semantik. Evolusi ini didorong oleh beberapa faktor penting, seperti kemajuan daya komputasi, peningkatan volume dan keragaman data (big data), serta perkembangan algoritma pembelajaran seperti *machine learning* dan *deep learning*.

Secara umum, AI diklasifikasikan ke dalam dua jenis utama, yaitu **narrow AI** dan **general AI**. Narrow AI mengacu pada sistem yang dirancang untuk menjalankan satu tugas spesifik dengan sangat baik, seperti penerjemahan bahasa, pengenalan wajah, atau deteksi spam. Sedangkan general AI merujuk pada kecerdasan buatan tingkat lanjut yang memiliki fleksibilitas kognitif layaknya manusia, dan sampai saat ini masih berada dalam tahap konseptual. Dalam konteks penelitian ini, narrow AI menjadi dasar pendekatan karena deteksi spam merupakan masalah klasifikasi terbatas namun kritis dalam sistem komunikasi digital.

Salah satu ranah dari narrow AI yang sangat relevan dalam penelitian ini adalah *Natural Language Processing* (NLP). NLP berfokus pada interaksi antara mesin dan bahasa manusia, baik dalam bentuk pemahaman, interpretasi, maupun produksi teks. Dalam implementasinya, NLP mencakup proses-proses seperti tokenisasi, penghapusan stopword, stemming, dan transformasi ke bentuk numerik seperti representasi *TF-IDF* (Term Frequency-Inverse Document Frequency). Melalui pendekatan ini, sistem dapat mengekstrak makna dari bahasa alami, mengenali struktur linguistik, dan menangkap konteks kalimat secara efektif (Muhaimin et al., 2023; Keat & Ying, 2024).

Di sisi lain, *machine learning* menyediakan kerangka kerja algoritmik bagi sistem untuk belajar dari data dan memperbaiki performanya seiring bertambahnya pengalaman. Beberapa algoritma seperti **Naïve Bayes**, **Support Vector Machine (SVM)**, dan **K-Nearest Neighbor (KNN)** telah digunakan secara luas dalam klasifikasi email spam karena sifatnya

yang adaptif dan relatif efisien. Menurut Rahman dan Maslan (2025), algoritma Naïve Bayes masih menjadi pendekatan yang andal dalam pengenalan pola-pola spam berbasis teks, terutama ketika dikombinasikan dengan fitur linguistik yang representatif. Sementara itu, Salim et al. (2024) membuktikan bahwa model berbasis SVM dan KNN memberikan akurasi tinggi dalam klasifikasi isi konten email, terutama jika dioptimalkan dengan representasi TF-IDF.

Dalam perkembangannya, kebutuhan terhadap sistem klasifikasi yang lebih presisi dan adaptif telah mendorong pemanfaatan *deep learning*. Model seperti **Convolutional Neural Network (CNN)** dan **Long Short-Term Memory (LSTM)** mampu mempelajari representasi fitur dari data teks secara otomatis tanpa membutuhkan rekayasa fitur manual. Penelitian oleh Sheneamer (2021) menunjukkan bahwa model CNN mampu mengungguli pendekatan tradisional dengan akurasi hingga 96,52% dalam mendeteksi spam dari kombinasi data email dan pesan singkat, terutama berkat kemampuannya menangkap pola spasial dalam urutan kata dan simbol.

Keunggulan pendekatan *deep learning* tidak hanya terletak pada performa numerik, tetapi juga pada kemampuannya memahami struktur sintaksis dan semantik secara lebih mendalam. Namun, tantangan dari pendekatan ini adalah kebutuhan komputasi yang tinggi dan keterbatasan interpretabilitas, yang membuatnya kurang cocok untuk implementasi ringan dan cepat di lingkungan terbatas. Karena itu, dalam banyak studi seperti yang dikemukakan oleh Mukhtar et al. (2022), pendekatan tradisional seperti Naïve Bayes masih banyak digunakan untuk kasus deteksi spam yang berskala kecil hingga menengah.

Studi-studi terkini juga menegaskan bahwa efektivitas sistem AI dalam klasifikasi spam sangat dipengaruhi oleh kualitas preprocessing dan pemilihan fitur yang tepat. Shivaji dan Shirsath (2025) mengevaluasi sejumlah algoritma klasifikasi pada dataset spam publik dan menekankan pentingnya pemilihan kombinasi metode preprocessing dan vektorisasi yang selaras dengan karakteristik data. Ini menegaskan bahwa keberhasilan sistem AI tidak hanya ditentukan oleh model yang digunakan, tetapi juga oleh pipeline pengolahan data yang mendasarinya.

Dengan demikian, pemanfaatan AI dalam deteksi spam bukanlah solusi satu arah, tetapi merupakan hasil dari kombinasi optimal antara NLP, machine learning, dan pendekatan evaluasi menyeluruh yang didesain untuk menangkap kompleksitas komunikasi digital modern. Sistem yang efektif tidak hanya mampu mengenali spam eksplisit, tetapi juga mampu beradaptasi terhadap bentuk-bentuk penyamaran konten yang semakin canggih dari waktu ke waktu.

2.2 Model/Algoritma yang Digunakan

Pemilihan model algoritma dalam pengembangan sistem deteksi email spam berbasis klasifikasi teks memiliki peran yang sangat krusial dalam menentukan akurasi, efisiensi, dan keandalan sistem. Seiring dengan kemajuan teknologi kecerdasan buatan, pendekatan yang digunakan pun semakin beragam, mencakup algoritma tradisional yang ringan hingga model berbasis deep learning dengan arsitektur yang lebih kompleks. Masing-masing pendekatan memiliki karakteristik, keunggulan, serta batasannya sendiri, dan dalam praktiknya pemilihan model sangat ditentukan oleh kebutuhan aplikasi serta sumber daya yang tersedia.

2.2.1 Algoritma Tradisional: Naïve Bayes, SVM, dan KNN

Pendekatan berbasis algoritma tradisional masih menjadi tulang punggung dalam banyak sistem klasifikasi teks, termasuk deteksi email spam. Tiga algoritma yang paling umum digunakan dalam konteks ini adalah **Naïve Bayes**, **Support Vector Machine (SVM)**, dan **K-Nearest Neighbor (KNN)**. Ketiganya memiliki keunggulan dalam hal interpretabilitas, kecepatan pelatihan, dan efisiensi sumber daya.

Naïve Bayes merupakan metode probabilistik yang mengasumsikan independensi antar fitur, dan justru karena kesederhanaannya, algoritma ini sering kali menunjukkan performa yang sangat baik dalam klasifikasi teks yang bersifat linier dan tidak terlalu kompleks. Dalam penelitian yang dilakukan oleh Keat dan Ying (2024), model Naïve Bayes mencapai akurasi sebesar 98,65% dalam mendeteksi email spam, membuktikan bahwa algoritma ini tetap relevan di tengah maraknya model-model modern. Senada dengan itu, Rahman dan Maslan (2025) menyatakan bahwa Naïve Bayes sangat efektif dalam mengklasifikasikan pesan berbasis teks pendek, terutama ketika dikombinasikan dengan preprocessing yang baik dan representasi fitur seperti TF-IDF.

Support Vector Machine bekerja dengan membangun hyperplane optimal untuk memisahkan dua kelas dalam ruang fitur berdimensi tinggi. Algoritma ini sangat cocok digunakan pada data teks yang telah ditransformasi ke dalam bentuk vektor, seperti TF-IDF, karena mampu menangani data sparse dan kompleks secara efisien. Salim et al. (2024) menunjukkan bahwa SVM mampu menghasilkan akurasi tinggi dalam mendeteksi spam berdasarkan isi konten, khususnya ketika fitur teks diproses secara optimal.

K-Nearest Neighbor (KNN) menjadi alternatif lain yang cukup populer karena prinsip kerjanya yang intuitif dan non-parametrik. Meskipun KNN cenderung lebih

lambat dalam proses inferensi karena harus membandingkan jarak dengan semua data pelatihan, metode ini efektif dalam kasus klasifikasi berbasis kemiripan konten. Penelitian Salim et al. (2024) juga membandingkan performa KNN dengan SVM dalam konteks deteksi spam dan menemukan bahwa KNN dapat menjadi pelengkap yang baik dalam pendekatan ensemble.

2.2.2 Deep Learning: CNN dan LSTM

Dalam satu dekade terakhir, pendekatan deep learning mulai banyak digunakan dalam berbagai tugas NLP, termasuk deteksi email spam. Model seperti **Convolutional Neural Network (CNN)** dan **Long Short-Term Memory (LSTM)** telah terbukti mampu menangani permasalahan klasifikasi teks yang kompleks dengan cara yang lebih otomatis dan adaptif.

CNN, meskipun awalnya dirancang untuk pemrosesan citra, telah berhasil diadaptasi untuk analisis teks melalui teknik konvolusi satu dimensi. Keunggulan CNN terletak pada kemampuannya mengekstraksi pola lokal (seperti n-gram) dari data teks tanpa memerlukan fitur yang diekstraksi secara manual. Dalam studi yang dilakukan oleh Sheneamer (2021), CNN mampu mencapai akurasi hingga 96,52% dalam mendeteksi spam pada dataset gabungan email dan pesan singkat, membuktikan efisiensinya dalam mengenali struktur linguistik yang berulang dalam spam.

LSTM, yang merupakan pengembangan dari Recurrent Neural Network (RNN), dirancang untuk mengatasi masalah long-term dependency dalam data sekuensial. Dengan menggunakan mekanisme *gating*, LSTM mampu mengingat informasi relevan dari urutan kata sebelumnya dan mengabaikan informasi yang tidak penting. Keunggulan ini membuat LSTM sangat cocok untuk memahami konteks panjang dalam teks email. Meskipun tidak semua studi memfokuskan pada LSTM dalam deteksi spam, pendekatan ini tetap diakui sebagai salah satu model deep learning yang paling kuat dalam analisis bahasa alami.

2.2.3 Evaluasi Model dan Adaptivitas

Evaluasi terhadap berbagai algoritma klasifikasi telah dilakukan oleh Shivaji dan Shirsath (2025) menggunakan dataset spam publik. Hasil evaluasi menunjukkan bahwa pemilihan algoritma tidak bisa dilepaskan dari konteks spesifik dataset, preprocessing yang dilakukan, serta kebutuhan operasional dari sistem. Dalam penelitian tersebut, model SVM dan Naïve Bayes terbukti unggul dalam efisiensi dan ketepatan, sementara model berbasis decision tree dan KNN memiliki kelebihan

dalam interpretabilitas. Temuan ini menegaskan bahwa tidak ada satu model yang "terbaik" secara mutlak, melainkan setiap algoritma memiliki nilai strategis tergantung pada skenario implementasi yang dihadapi.

2.2.4 Pertimbangan Pemilihan Model dalam Penelitian Ini

Dalam penelitian ini, pemilihan algoritma mempertimbangkan keseimbangan antara performa, efisiensi komputasi, kemudahan interpretasi, serta kompleksitas data. Karena sistem dirancang untuk memproses email dalam jumlah besar dengan struktur teks bervariasi, model yang digunakan harus mampu bekerja dengan cepat namun tetap akurat.

Dengan pertimbangan tersebut, pendekatan berbasis Naïve Bayes, SVM, dan Decision Tree menjadi pilihan utama karena telah terbukti memberikan hasil yang kompetitif dalam studi sebelumnya dan mudah diimplementasikan dalam lingkungan terbatas seperti sistem real-time atau perangkat lokal. Sementara itu, pendekatan berbasis CNN atau LSTM tetap menjadi rujukan penting dan akan sangat relevan untuk dikaji dalam tahap pengembangan lanjutan ketika sumber daya komputasi lebih tersedia.

2.3 Penelitian Sebelumnya yang Relevan

Penelitian mengenai deteksi email spam telah berkembang secara signifikan, seiring dengan kemajuan dalam bidang natural language processing dan machine learning. Berbagai pendekatan telah diuji, mulai dari algoritma tradisional hingga deep learning, dengan variasi metode preprocessing dan representasi fitur yang disesuaikan dengan karakteristik data. Studi-studi terdahulu memberikan landasan penting tidak hanya dalam memahami efektivitas model, tetapi juga dalam merancang pipeline sistem yang efisien dan andal.

Salah satu studi penting datang dari Keat dan Ying (2024), yang secara khusus membahas penerapan artificial intelligence untuk email spam filtering. Dalam penelitian mereka, beberapa algoritma klasik dievaluasi pada dataset email beragam, dan ditemukan bahwa **Naïve Bayes** tetap menjadi algoritma yang sangat efektif dalam klasifikasi spam, dengan akurasi mencapai 98,65%. Studi ini menekankan pentingnya preprocessing teks dan penggunaan representasi fitur seperti **TF-IDF**, yang secara signifikan meningkatkan performa klasifikasi.

Dari perspektif lain, Sheneamer (2021) mengusulkan perbandingan langsung antara metode pembelajaran tradisional dan pendekatan deep learning. Studi ini mengevaluasi performa model **Convolutional Neural Network (CNN)** dan **Long Short-Term Memory**

(LSTM) terhadap dataset gabungan email dan SMS. Hasilnya menunjukkan bahwa CNN mampu mencapai akurasi hingga 96,52%, bahkan tanpa perlu proses ekstraksi fitur manual. Temuan ini menunjukkan bahwa pendekatan deep learning sangat menjanjikan untuk klasifikasi spam, terutama dalam konteks teks yang bervariasi dan kompleks.

Penelitian dari Muhaimin, Taufik, dan Daniswara (2023) berfokus pada pendekatan NLP untuk mendeteksi spam email dalam konteks berbahasa Indonesia. Mereka menggabungkan metode preprocessing seperti tokenisasi dan stemming dengan algoritma machine learning, dan menunjukkan bahwa kombinasi fitur linguistik dan statistik dapat meningkatkan akurasi klasifikasi. Studi ini menjadi penting karena menunjukkan bahwa strategi preprocessing yang tepat dapat meningkatkan efektivitas algoritma, bahkan pada bahasa yang memiliki struktur berbeda dari bahasa Inggris.

Sementara itu, Mukhtar, Al Amien, dan Rucyat (2022) mengevaluasi penggunaan algoritma **Naïve Bayes** dalam klasifikasi spam dengan pendekatan ringan dan efisien. Mereka menekankan bahwa performa model sangat bergantung pada kualitas preprocessing dan distribusi data dalam kelas spam dan non-spam. Hasil penelitian ini memperkuat temuan Keat dan Ying (2024) bahwa Naïve Bayes, meskipun sederhana, tetap relevan untuk aplikasi real-time karena efisiensinya dalam hal kecepatan pelatihan dan inferensi.

Dalam konteks algoritma yang lebih interpretatif, Rahman dan Maslan (2025) mengulas performa Naïve Bayes dalam klasifikasi email spam dengan evaluasi menyeluruh terhadap metrik seperti **precision**, **recall**, dan **F1-score**. Studi ini menunjukkan bahwa fokus evaluasi tidak hanya boleh terpaku pada akurasi semata, melainkan juga pada keseimbangan performa model dalam menghindari false positive maupun false negative. Hal ini sangat relevan dalam aplikasi deteksi spam di mana kesalahan klasifikasi bisa berdampak besar pada keamanan dan kepercayaan pengguna.

Kontribusi lain datang dari Salim, Adryani, dan Sutabri (2024) yang mengevaluasi kinerja model **K-Nearest Neighbor (KNN)** dan **Support Vector Machine (SVM)** dalam klasifikasi spam berbasis isi konten. Mereka membandingkan kedua algoritma pada berbagai skenario representasi fitur dan menemukan bahwa SVM memberikan hasil yang lebih stabil dalam data berdimensi tinggi seperti hasil dari TF-IDF. Studi ini menggarisbawahi pentingnya pemilihan algoritma berdasarkan karakteristik representasi teks dan kebutuhan efisiensi sistem.

Terakhir, Shivaji dan Shirsath (2025) menyajikan evaluasi komprehensif terhadap berbagai algoritma klasifikasi menggunakan dataset spam publik. Penelitian mereka menegaskan bahwa tidak ada satu algoritma yang unggul dalam semua konteks, dan bahwa penggabungan pendekatan seperti ensemble learning dapat meningkatkan performa sistem

secara keseluruhan. Mereka juga menekankan pentingnya analisis fitur dan teknik evaluasi multimetric untuk mendeteksi kekuatan dan kelemahan model secara lebih objektif.

Secara keseluruhan, penelitian-penelitian terdahulu memperlihatkan bahwa keberhasilan sistem deteksi spam sangat dipengaruhi oleh pemilihan algoritma, strategi preprocessing, dan pendekatan evaluasi yang digunakan. Penelitian ini memposisikan dirinya sebagai kelanjutan dari studi-studi tersebut dengan mengadopsi pendekatan **machine learning berbasis NLP dan integrasi fitur numerik**, serta mengevaluasi kinerja beberapa model tradisional melalui eksperimen sistematis untuk mengidentifikasi solusi klasifikasi spam yang optimal, efisien, dan interpretatif.

2.4 Kerangka Pemikiran

Perancangan sistem deteksi email spam berbasis artificial intelligence membutuhkan kerangka pemikiran yang komprehensif, yang mampu mengintegrasikan teori dasar, pendekatan teknis, serta konteks permasalahan yang nyata di dunia digital saat ini. Tujuan utama dari sistem ini adalah untuk membedakan secara otomatis antara email yang bersifat spam dan email normal, dengan tingkat akurasi yang tinggi dan efisiensi waktu proses yang memadai. Dalam membangun kerangka pemikiran ini, berbagai studi sebelumnya dijadikan sebagai pijakan utama, mulai dari konsep natural language processing (NLP), pemilihan model machine learning, hingga strategi evaluasi performa sistem.

Masalah utama dalam klasifikasi spam adalah kompleksitas pola bahasa yang digunakan oleh pengirim spam untuk mengelabui sistem filter tradisional. Pesan spam modern tidak lagi selalu menggunakan kata-kata eksplisit atau struktur yang mudah dikenali, tetapi sering kali menyerupai email normal. Oleh karena itu, pendekatan berbasis keyword sederhana menjadi tidak lagi memadai. Diperlukan sistem yang mampu memahami konteks, menangkap struktur bahasa, dan mengenali pola linguistik yang lebih halus. Dalam konteks inilah, **natural language processing** berperan sebagai fondasi inti.

Tahap awal dalam kerangka kerja sistem dimulai dengan preprocessing teks secara menyeluruh. Proses ini meliputi pembersihan karakter non-alfabetik, pengubahan huruf menjadi lowercase, penghapusan stopword, tokenisasi, dan stemming. Tujuannya adalah untuk menyederhanakan teks ke dalam bentuk representatif yang bersih dan konsisten. Setelah preprocessing, data teks dikonversi ke bentuk numerik menggunakan metode **Term Frequency-Inverse Document Frequency (TF-IDF)**. Representasi ini memungkinkan model untuk mengenali kata-kata penting dalam konteks dokumen, serta membedakannya berdasarkan tingkat kekhususannya dalam seluruh korpus data.

Selanjutnya, fitur tekstual tersebut dikombinasikan dengan **fitur numerik statistik** seperti jumlah huruf kapital, jumlah URL, panjang teks, dan jumlah tanda baca, guna menambah dimensi struktural pada input model. Pendekatan gabungan ini mengacu pada studi Muhaimin et al. (2023) dan Rahman & Maslan (2025), yang menunjukkan bahwa integrasi antara fitur linguistik dan fitur struktural dapat meningkatkan akurasi klasifikasi secara signifikan.

Tahap berikutnya dalam kerangka ini adalah proses pelatihan dan pengujian model klasifikasi menggunakan dataset berlabel, seperti **SpamAssassin**, yang berisi email spam dan non-spam dalam jumlah besar. Model yang digunakan antara lain adalah **Naïve Bayes**, **Support Vector Machine (SVM)**, dan **Decision Tree**, yang dipilih berdasarkan kombinasi antara efektivitas, efisiensi komputasi, dan kemudahan interpretasi hasil. Studi oleh Keat & Ying (2024) menunjukkan keunggulan Naïve Bayes dalam konteks spam filtering karena kemampuannya menangani data teks pendek dengan struktur distribusi yang sparsity-nya tinggi. Di sisi lain, SVM lebih unggul dalam menangani data berdimensi tinggi, seperti hasil dari TF-IDF, sebagaimana ditunjukkan dalam studi Salim et al. (2024).

Untuk pendekatan yang lebih kompleks, studi oleh Sheneamer (2021) menunjukkan bahwa **Convolutional Neural Network (CNN)** mampu mengungguli model tradisional dalam hal akurasi, terutama dalam menangkap struktur lokal dalam teks. Namun, model seperti CNN dan RNN membutuhkan sumber daya yang lebih besar dan waktu pelatihan yang lebih lama. Oleh karena itu, dalam kerangka pemikiran ini, model-model ringan tetap dipilih sebagai baseline yang solid, dengan mempertimbangkan kemungkinan ekspansi ke model deep learning di masa mendatang.

Sistem kemudian dievaluasi secara komprehensif menggunakan metrik **accuracy**, **precision**, **recall**, dan **F1-score**, untuk memberikan gambaran menyeluruh tentang kualitas prediksi model. Seperti disampaikan oleh Shivaji & Shirsath (2025), reliance pada satu metrik saja dapat menyesatkan, karena keberhasilan model dalam deteksi spam harus mencakup aspek presisi dalam menghindari false positive, dan sensitivitas dalam menangkap true positive. Evaluasi holistik ini juga penting dalam konteks keamanan digital, di mana kesalahan klasifikasi dapat berdampak signifikan terhadap pengguna.

Kerangka ini juga membuka ruang untuk pengembangan sistem ke arah yang lebih adaptif melalui **ensemble learning**, di mana beberapa model dikombinasikan untuk meningkatkan stabilitas dan performa sistem, sebagaimana dikemukakan dalam pendekatan hybrid yang dikaji oleh Shivaji & Shirsath (2025). Selain itu, dalam skenario ke depan yang membutuhkan pemrosesan skala besar dan konteks bahasa yang lebih kompleks, penggunaan arsitektur **Transformer** dan model berbasis attention dapat dipertimbangkan sebagai langkah lanjutan.

Secara keseluruhan, kerangka pemikiran ini menempatkan NLP sebagai landasan untuk memahami teks secara semantik dan sintaktik, machine learning sebagai alat klasifikasi berbasis data, serta evaluasi metrik sebagai alat ukur objektif performa sistem. Melalui pendekatan terintegrasi ini, sistem deteksi spam yang dikembangkan tidak hanya diarahkan untuk akurat secara statistik, tetapi juga tangguh terhadap dinamika pola spam yang terus berkembang, serta efisien dalam konteks operasional di lingkungan nyata.

BAB III
METODOLOGI PENELITIAN

3.1 Jenis dan Desain Penelitian

Penelitian ini termasuk dalam kategori **penelitian terapan kuantitatif** yang berfokus pada pengembangan dan evaluasi sistem berbasis teknologi kecerdasan buatan. Tujuan utama dari penelitian ini adalah untuk merancang sebuah sistem deteksi email spam yang mampu mengidentifikasi pesan spam secara otomatis dengan akurasi tinggi menggunakan pendekatan klasifikasi teks dan teknik Natural Language Processing (NLP). Fokus penelitian tidak hanya pada pengembangan algoritma, tetapi juga pada validasi performa sistem berdasarkan metrik evaluasi klasifikasi yang umum digunakan dalam bidang machine learning.

Secara metodologis, penelitian ini menerapkan pendekatan **eksperimen komputasional**, di mana proses pengumpulan data, pelatihan model, dan evaluasi performa dilakukan dalam lingkungan simulasi menggunakan dataset publik. Sistem yang dirancang diuji terhadap sejumlah skenario klasifikasi dengan menggunakan data berlabel spam dan non-spam. Hasil dari pengujian ini kemudian dianalisis untuk mengetahui efektivitas model dalam mendeteksi email spam.

Rancangan penelitian ini mengacu pada alur pengembangan sistem berbasis machine learning secara umum, yaitu:

1. **Pengumpulan Data**
   Tahap awal penelitian melibatkan pengumpulan dataset email spam dan non-spam dari sumber publik yang telah terverifikasi. Dataset yang dipilih harus bersifat representatif, terbuka (*open source*), dan telah digunakan dalam studi-studi sebelumnya untuk memastikan validitasnya. Dalam hal ini, dataset SpamAssassin dipilih sebagai basis eksperimen karena keterbukaan strukturnya serta ketersediaan label yang konsisten.
2. **Pra-pemrosesan Teks (Text Preprocessing)**
   Email yang telah dikumpulkan melalui dataset kemudian diproses untuk menghilangkan elemen-elemen linguistik yang tidak relevan, menyederhanakan struktur teks, dan menyiapkannya untuk tahap ekstraksi fitur. Proses ini mencakup tokenisasi, lowercasing, penghapusan karakter khusus, stopword removal, dan stemming.

3. **Ekstraksi Fitur (Feature Extraction)**
   Setelah data teks dibersihkan, tahap berikutnya adalah mengubah representasi teks menjadi bentuk numerik. Teknik TF-IDF digunakan untuk mengekspresikan pentingnya suatu kata dalam dokumen relatif terhadap seluruh korpus. Representasi ini sangat berguna dalam mengidentifikasi istilah yang bersifat khas spam dan memisahkannya dari kata-kata umum yang netral.
4. **Pelatihan dan Pengujian Model (Model Training and Testing)**
   Model klasifikasi kemudian dilatih menggunakan algoritma seperti Naïve Bayes, Support Vector Machine (SVM), dan Decision Tree. Data dibagi menggunakan teknik *train-test split* dengan rasio 80:20 agar model dapat diuji secara objektif terhadap data yang belum pernah dilihat sebelumnya.
5. **Evaluasi Performansi Model**
   Setelah model selesai dilatih dan diuji, performanya dievaluasi berdasarkan metrik seperti akurasi, presisi, recall, dan F1-score. Evaluasi ini bertujuan untuk memastikan bahwa model tidak hanya bekerja dengan baik secara statistik, tetapi juga mampu mempertahankan performa yang seimbang dalam mendeteksi spam secara aktual.

Melalui pendekatan eksperimental ini, penelitian bertujuan untuk mengidentifikasi algoritma mana yang paling optimal untuk digunakan dalam sistem deteksi spam berbasis teks, serta memberikan insight terhadap kekuatan dan kelemahan masing-masing metode.

3.2 Dataset yang Digunakan

Pemilihan dataset merupakan salah satu aspek paling fundamental dalam membangun sistem klasifikasi berbasis machine learning. Dataset tidak hanya berfungsi sebagai sumber data untuk pelatihan dan pengujian model, tetapi juga menentukan sejauh mana sistem mampu merepresentasikan permasalahan dunia nyata. Oleh karena itu, dalam penelitian ini digunakan dataset yang telah terbukti kredibel dan umum digunakan dalam studi klasifikasi email spam, yaitu **SpamAssassin Public Corpus**.

3.2.1 Karakteristik Dataset SpamAssassin

SpamAssassin adalah kumpulan data email yang terdiri dari dua kategori utama, yaitu spam dan non-spam (ham). Dataset ini dikembangkan oleh proyek open source SpamAssassin dan tersedia secara publik untuk keperluan riset dan pengujian sistem deteksi spam. Beberapa karakteristik utama dari dataset ini antara lain:

- **Tersedia dalam format teks mentah**, sehingga memungkinkan proses pra-pemrosesan dilakukan dengan fleksibel sesuai kebutuhan sistem.

- **Telah diberi label secara eksplisit** antara spam dan ham, sehingga sangat ideal untuk pendekatan supervised learning.
- **Bersifat heterogen**, mencakup berbagai jenis pesan seperti iklan, penipuan, promosi, serta email normal dari mailing list atau komunikasi pribadi.
- **Terstruktur dan terdokumentasi dengan baik**, memudahkan dalam ekstraksi konten teks seperti subject dan body email.

Dataset ini dipilih karena beberapa alasan. Pertama, keterbukaannya memungkinkan peneliti untuk memverifikasi dan mereproduksi hasil eksperimen dengan mudah. Kedua, dataset ini telah digunakan dalam banyak studi terdahulu, sehingga hasil penelitian dapat dikomparasikan secara lebih adil dengan pendekatan lain yang telah terbukti efektif (Keat & Ying, 2024; Sheneamer, 2021). Ketiga, struktur datanya yang relatif bersih dan terorganisir mendukung proses pra-pemrosesan dan pelatihan model secara efisien.

3.2.2 Struktur Data dan Proses Labeling

Setiap email dalam dataset disimpan sebagai file teks individu, yang terdiri dari beberapa bagian seperti metadata (header), subject, dan body. Untuk keperluan penelitian ini, hanya bagian subject dan body yang digunakan sebagai bahan analisis, sementara metadata seperti alamat pengirim, tanggal, dan format MIME diabaikan agar fokus tetap pada konten utama email. Hal ini sejalan dengan batasan masalah yang telah ditetapkan sebelumnya.

Label spam atau ham biasanya ditentukan berdasarkan direktori penyimpanan. File spam disimpan dalam folder spam, sedangkan file non-spam ditempatkan dalam folder ham. Skema pelabelan seperti ini memungkinkan proses labeling dilakukan secara otomatis melalui script, tanpa perlu anotasi manual yang memakan waktu.

3.2.3 Kelebihan dan Keterbatasan Dataset

Walaupun SpamAssassin memiliki banyak kelebihan, dataset ini juga memiliki beberapa keterbatasan. Salah satu kekurangannya adalah bahwa sebagian besar email yang digunakan berasal dari awal tahun 2000-an, sehingga tidak seluruhnya merepresentasikan pola spam modern yang lebih canggih seperti penggunaan emoji, obfuscation, atau manipulasi URL. Oleh karena itu, hasil dari sistem ini mungkin masih perlu divalidasi lebih lanjut terhadap dataset yang lebih mutakhir agar dapat diuji ketahanannya terhadap threat yang lebih kompleks.

Namun demikian, untuk tahap perancangan dan pengujian awal sistem klasifikasi spam berbasis teks, dataset SpamAssassin tetap merupakan pilihan yang sangat representatif, efisien, dan sesuai untuk kebutuhan riset ini. Penggunaan dataset ini juga memungkinkan penelitian untuk berjalan secara terstandarisasi dengan komparabilitas hasil yang tinggi terhadap literatur ilmiah yang ada.

3.3 Arsitektur dan Konfigurasi Model

Arsitektur sistem deteksi spam yang dirancang dalam penelitian ini mengikuti pola umum sistem klasifikasi berbasis teks yang mengintegrasikan teknik Natural Language Processing (NLP) dan algoritma machine learning. Tujuannya adalah untuk membangun pipeline sistematis yang mampu memproses data email dari bentuk mentah menjadi prediksi yang terklasifikasi secara otomatis sebagai spam atau non-spam. Arsitektur ini terdiri dari beberapa komponen utama yang bekerja secara berurutan dan saling terintegrasi.

3.3.1 Tahap Input: Pengambilan dan Pembacaan Data

Sistem dimulai dengan tahap input, di mana data diambil langsung dari struktur direktori dataset SpamAssassin. Setiap file teks dibaca dan dipisahkan antara bagian subject dan body. Kedua bagian ini kemudian digabungkan menjadi satu representasi teks penuh yang akan dianalisis oleh sistem. Selanjutnya, data dikelompokkan berdasarkan label spam dan non-spam sesuai direktori asalnya.

3.3.2 Preprocessing Teks

Pada tahap ini, dilakukan pembersihan dan normalisasi data agar teks siap untuk diolah oleh algoritma machine learning. Proses preprocessing mencakup:

- **Lowercasing**: seluruh huruf dikonversi menjadi huruf kecil untuk menyamakan bentuk kata.
- **Penghapusan karakter khusus**: termasuk simbol, angka, dan tanda baca yang tidak bermakna semantik.
- **Tokenisasi**: memecah teks menjadi kata-kata individual sebagai unit analisis.
- **Stopword removal**: menghapus kata-kata umum yang tidak memiliki nilai klasifikasi seperti “dan”, “yang”, “itu”, dll.
- **Stemming**: menyederhanakan kata menjadi bentuk dasarnya agar variasi bentuk kata tidak memperbanyak fitur yang serupa.

Proses ini penting untuk mereduksi noise dalam data serta mengurangi dimensi vektor fitur yang akan dihasilkan pada tahap berikutnya.

3.3.3 Ekstraksi Fitur dengan TF-IDF

Setelah teks dibersihkan, dilakukan transformasi ke bentuk numerik menggunakan **Term Frequency-Inverse Document Frequency (TF-IDF)**. Representasi ini digunakan karena mampu memberikan bobot pentingnya kata dalam satu dokumen relatif terhadap seluruh korpus. Dengan menggunakan TF-IDF, kata-kata khas dalam spam (seperti “gratis”, “promo”, “transfer”) akan mendapatkan bobot lebih tinggi, sedangkan kata-kata umum akan diminimalkan. Vektor TF-IDF hasil ekstraksi ini kemudian menjadi input utama bagi model klasifikasi.

3.3.4 Algoritma Klasifikasi

Sistem dirancang untuk mendukung beberapa algoritma machine learning yang dapat saling dibandingkan dalam hal performa. Tiga algoritma utama yang diimplementasikan dalam penelitian ini adalah:

- **Naïve Bayes**
  Algoritma probabilistik ini sangat cocok untuk teks pendek dan bekerja baik dengan representasi bag-of-words maupun TF-IDF. Naïve Bayes menjadi baseline model dalam penelitian karena kesederhanaannya, kecepatan pelatihan, serta performanya yang cukup baik dalam studi sebelumnya (Keat & Ying, 2024).
- **Support Vector Machine (SVM)**
  SVM dipilih karena mampu memisahkan data berdimensi tinggi secara optimal. Dengan kernel linier dan parameter regularisasi yang disesuaikan, SVM seringkali unggul dalam klasifikasi teks yang kompleks meskipun memerlukan sumber daya komputasi yang lebih besar.
- **Decision Tree**
  Model ini dipilih karena kemampuannya menghasilkan struktur yang mudah dipahami dan divisualisasikan. Meskipun lebih rawan overfitting, Decision Tree dapat memberikan insight awal terhadap fitur-fitur dominan dalam klasifikasi spam.

3.3.5 Konfigurasi dan Implementasi

Implementasi sistem dilakukan menggunakan bahasa pemrograman Python dengan library populer seperti scikit-learn, nltk, dan pandas. Konfigurasi default digunakan terlebih dahulu untuk baseline, lalu dilakukan eksperimen dengan penyesuaian parameter (hyperparameter tuning) jika diperlukan. Rasio pembagian

data menggunakan train-test split 80:20, dan model diuji terhadap data uji yang belum pernah dilihat sebelumnya untuk mengukur generalisasi model.

Struktur dan alur arsitektur model secara keseluruhan digambarkan sebagai berikut:

Input Email → Preprocessing → TF-IDF Vectorization → Model Klasifikasi → Output Spam/Non-Spam

Arsitektur ini dirancang agar cukup modular dan fleksibel, memungkinkan sistem untuk diperluas atau dikombinasikan dengan model lain seperti ensemble learning atau deep learning pada tahap pengembangan berikutnya. Fokus utama dalam tahap ini adalah membangun sistem yang stabil, efisien, dan mudah diuji secara kuantitatif terhadap metrik evaluasi yang relevan.

3.4 Tahapan Implementasi

Implementasi sistem deteksi spam berbasis teks dilakukan secara bertahap dan sistematis agar setiap komponen dalam proses dapat berjalan optimal. Tahapan ini mencakup seluruh langkah praktis mulai dari pemuatan data mentah, pembersihan dan normalisasi teks, hingga pembangunan pipeline klasifikasi teks menggunakan algoritma machine learning. Seluruh tahapan diimplementasikan menggunakan bahasa pemrograman Python dalam lingkungan Jupyter Notebook, yang memungkinkan eksperimen dilakukan secara interaktif dan modular. Berikut ini adalah penjabaran menyeluruh dari setiap tahap implementasi yang dijalankan dalam penelitian ini.

3.4.1 Import Library dan Setup Awal

Langkah awal implementasi dimulai dengan memuat seluruh library yang dibutuhkan untuk mendukung keseluruhan proses eksperimen. Library pandas dan numpy digunakan sebagai fondasi manipulasi data, sedangkan re dimanfaatkan untuk regular expression dalam preprocessing teks. Untuk keperluan natural language processing, digunakan modul dari NLTK seperti stopwords, PorterStemmer, dan WordNetLemmatizer. Selain itu, seluruh algoritma klasifikasi dan metrik evaluasi diambil dari sklearn, termasuk model Naive Bayes, Support Vector Machine, Decision Tree, serta metode ensemble seperti Voting Classifier. Library scipy.sparse juga digunakan untuk menggabungkan matriks sparse hasil vektorisasi teks dengan fitur numerik.

Proses setup juga melibatkan konfigurasi agar peringatan tidak mengganggu hasil output visual. Nilai random seed ditentukan sejak awal menggunakan

np.random.seed(42) untuk memastikan bahwa semua pembagian data dan proses acak yang dilakukan bersifat deterministik, memungkinkan proses yang replikatif. Selain itu, stopword list dan corpus untuk lemmatization dari NLTK diunduh langsung di awal untuk memastikan seluruh pipeline siap digunakan tanpa hambatan eksternal.

```
# 1. Import Library & Setup
import os
import pandas as pd
import numpy as np
import re
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split,
cross_val_score, GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score, roc_auc_score, roc_curve,
precision_recall_fscore_support
from scipy.sparse import hstack, csr_matrix
import warnings
warnings.filterwarnings('ignore')

nltk.download('stopwords')
nltk.download('wordnet')

# Set random seed untuk reprodusibilitas
RANDOM_STATE = 42
np.random.seed(RANDOM_STATE)
```

3.4.2 Memuat Dataset dan Eksplorasi Awal

Setelah seluruh dependensi dipersiapkan, tahap selanjutnya adalah memuat dataset spam_assassin.csv ke dalam DataFrame menggunakan pandas.read_csv(). Dataset ini berisi dua kolom utama, yakni kolom text yang memuat isi lengkap email,

dan kolom target yang berisi label biner di mana nilai 1 merepresentasikan email spam dan nilai 0 untuk email non-spam.

Langkah awal eksplorasi melibatkan peninjauan ukuran dataset, tipe data, serta pengecekan jumlah nilai kosong. Visualisasi distribusi label dilakukan menggunakan seaborn.countplot yang menunjukkan bahwa distribusi antara spam dan non-spam relatif seimbang. Keseimbangan ini sangat penting karena ketimpangan distribusi kelas dapat menyebabkan bias model terhadap kelas mayoritas.

Tahapan ini juga mencakup pengecekan awal terhadap contoh isi email untuk memahami variasi struktur teks yang akan diproses. Dari eksplorasi tersebut terlihat bahwa konten email sangat beragam, mulai dari pesan pendek hingga teks panjang, dengan struktur penulisan yang tidak terstandarisasi, yang justru mencerminkan kondisi nyata email di dunia digital.

```
# 2. Memuat Dataset & Eksplorasi Awal
dataFrame = pd.read_csv("Dataset/spam_assassin.csv")
print(f"Ukuran dataset: {dataFrame.shape}")
print(dataFrame.head())
print(dataFrame.info())
print(dataFrame['target'].value_counts())

# Visualisasi distribusi kelas
plt.figure(figsize=(5,3))
sns.countplot(x='target', data=dataFrame)
plt.xticks([0,1], ['Bukan Spam', 'Spam'])
plt.title('Distribusi Kelas Email')
plt.show()
```

3.4.3 Rekayasa Fitur Awal (Eksploratif)

Sebelum masuk ke tahap natural language processing, dilakukan rekayasa fitur numerik berbasis struktur kasar dari teks email. Pendekatan ini bersifat eksploratif dan bertujuan untuk menangkap pola statistik sederhana yang dapat berkontribusi dalam klasifikasi spam. Beberapa fitur yang diturunkan antara lain adalah panjang teks, jumlah huruf kapital, jumlah tanda seru, jumlah angka, jumlah karakter khusus, jumlah URL yang terdeteksi dalam teks, dan jumlah total kata.

Seluruh fitur tersebut dihitung secara langsung dari kolom text dengan pendekatan berbasis fungsi lambda dan regular expression. Sebagai contoh, jumlah huruf kapital dihitung dengan menyaring karakter yang memenuhi kondisi isupper(),

sedangkan jumlah karakter khusus dihitung dari jumlah simbol yang bukan huruf, angka, atau spasi. Pendekatan ini mampu menyoroti pola kasar yang umumnya muncul pada email spam seperti penggunaan huruf kapital berlebihan atau banyaknya tanda seru untuk menarik perhatian.

Hasil ekstraksi fitur numerik kemudian divisualisasikan menggunakan histogram terpisah untuk setiap fitur. Visualisasi ini tidak hanya membantu dalam memahami distribusi masing-masing fitur, tetapi juga memperlihatkan perbedaan pola antara kelas spam dan non-spam. Histogram dibatasi pada persentil ke-99 untuk menghindari outlier ekstrem yang dapat mendistorsi distribusi utama. Tahap ini secara keseluruhan memberikan insight awal yang sangat berharga dalam mengenali sinyal kasar dari konten email yang akan diproses lebih lanjut dalam pipeline NLP.

```
# 3. Rekayasa Fitur Numerik

dataFrame['panjangTeks'] = dataFrame['text'].apply(len)
dataFrame['jumlahKapital'] = dataFrame['text'].apply(lambda x:
sum(1 for c in x if c.isupper()))
dataFrame['jumlahTandaSeru'] = dataFrame['text'].apply(lambda x:
x.count('!'))
dataFrame['jumlahAngka'] = dataFrame['text'].apply(lambda x: sum(1
for c in x if c.isdigit()))
dataFrame['jumlahKarakterKhusus'] = dataFrame['text'].apply(lambda
x: sum(1 for c in x if not c.isalnum() and not c.isspace()))
dataFrame['jumlahUrl'] = dataFrame['text'].apply(lambda x:
len(re.findall(r'http[s]?://', x)))
dataFrame['jumlahKata'] = dataFrame['text'].apply(lambda x:
len(x.split()))

# Visualisasi fitur numerik dengan pembatasan xlim agar distribusi
lebih proporsional
def plot_histogram_fitur(df, fitur, ax, judul, bins=30,
persentil_max=99):
    batas = np.percentile(df[fitur], persentil_max)
    sns.histplot(df[df[fitur] <= batas], x=fitur, hue='target',
bins=bins, ax=ax, kde=True)
    ax.set_xlim([0, batas])
    ax.set_title(judul)
    ax.set_ylabel('Jumlah')
    ax.set_xlabel(fitur)

gambar, axes = plt.subplots(2, 3, figsize=(18,8))
```

```
plot_histogram_fitur(dataFrame, 'panjangTeks', axes[0,0], 'Panjang
Teks')
plot_histogram_fitur(dataFrame, 'jumlahKapital', axes[0,1], 'Huruf
Kapital')
plot_histogram_fitur(dataFrame, 'jumlahTandaSeru', axes[0,2],
'Tanda Seru', persentil_max=99)
plot_histogram_fitur(dataFrame, 'jumlahAngka', axes[1,0], 'Angka',
persentil_max=99)
plot_histogram_fitur(dataFrame, 'jumlahKarakterKhusus', axes[1,1],
'Karakter Khusus', persentil_max=99)
plot_histogram_fitur(dataFrame, 'jumlahUrl', axes[1,2], 'Jumlah
URL', bins=10, persentil_max=99)
plt.tight_layout()
plt.show()
```

3.4.4 Pra-pemrosesan Teks (Cleaning dan Normalisasi)

Setelah fitur numerik diekstraksi, tahap selanjutnya adalah pemrosesan teks secara mendalam melalui tahapan pra-pemrosesan atau preprocessing. Tahap ini merupakan bagian sentral dalam natural language processing karena kualitas representasi teks sangat mempengaruhi performa model klasifikasi. Teks pada kolom text dibersihkan menggunakan serangkaian fungsi yang dirancang untuk menghilangkan elemen linguistik yang tidak relevan dan menormalkan struktur bahasa agar konsisten.

Langkah pertama dalam preprocessing adalah mengubah seluruh karakter menjadi huruf kecil (lowercasing) agar kata dengan bentuk berbeda seperti “Free” dan “free” dianggap sama. Selanjutnya, seluruh tautan URL diidentifikasi menggunakan regular expression dan digantikan dengan token khusus “url” agar tidak hilang sepenuhnya namun tetap disederhanakan. Angka dan karakter non-huruf seperti simbol atau tanda baca juga dihapus untuk meminimalisir noise linguistik.

Teks kemudian dipecah menjadi kata-kata individual, dan setiap kata dibandingkan terhadap daftar stopwords bahasa Inggris yang tersedia dalam NLTK. Kata-kata umum seperti “the”, “and”, “is” dihapus karena tidak membawa makna penting dalam konteks klasifikasi spam. Setelah itu, dilakukan stemming menggunakan algoritma PorterStemmer untuk mengubah setiap kata ke bentuk dasarnya. Sebagai contoh, “running”, “runner”, dan “ran” akan direduksi menjadi “run”. Dalam beberapa eksperimen tambahan, juga disediakan opsi untuk menggunakan lemmatization dengan WordNetLemmatizer, meskipun dalam

implementasi utama hanya stemming yang digunakan untuk menjaga konsistensi dan efisiensi proses.

Hasil akhir dari tahapan ini adalah kolom baru bernama teksBersih, yang memuat versi teks email yang telah diproses, disederhanakan, dan siap digunakan sebagai input dalam proses ekstraksi fitur berbasis teks. Proses ini berhasil mereduksi redundansi kata, mengurangi kompleksitas linguistik, dan memperjelas sinyal semantik yang akan digunakan oleh model klasifikasi.

```
# 4. Pra-pemrosesan Teks Lanjutan
def bersihkanTeks(teks, gunakanStem=True, gunakanLemma=False):
    teks = teks.lower()
    teks = re.sub(r'http[s]?://\S+', ' url ', teks)  # ganti URL
    teks = re.sub(r'\d+', ' ', teks)                 # hapus angka
    teks = re.sub(r'[^a-z ]', ' ', teks)             # hanya huruf
    kata = teks.split()
    stopWords = set(stopwords.words('english'))
    kata = [w for w in kata if w not in stopWords]
    if gunakanStem:
        stemmer = PorterStemmer()
        kata = [stemmer.stem(w) for w in kata]
    if gunakanLemma:
        lemmatizer = WordNetLemmatizer()
        kata = [lemmatizer.lemmatize(w) for w in kata]
    return ' '.join(kata)

dataFrame['teksBersih'] = dataFrame['text'].apply(lambda x:
bersihkanTeks(x, gunakanStem=True, gunakanLemma=False))
print(dataFrame[['text', 'teksBersih']].head())
```

3.4.5 Representasi Fitur dengan TF-IDF

Setelah teks dibersihkan, proses selanjutnya adalah mengubah teks menjadi representasi numerik yang dapat diproses oleh algoritma machine learning. Teknik yang digunakan dalam penelitian ini adalah TF-IDF (Term Frequency-Inverse Document Frequency), yang telah terbukti efektif dalam menangkap pentingnya kata dalam dokumen relatif terhadap seluruh korpus.

TF-IDF digunakan untuk membangun representasi vektor dari teks dalam kolom teksBersih. Parameter max_features ditetapkan sebanyak 4000 untuk menjaga efisiensi komputasi dan menghindari sparsity yang terlalu tinggi. Selain itu, digunakan ngram_range=(1,2) untuk menangkap tidak hanya unigram (kata tunggal)

tetapi juga bigram (dua kata berturut-turut) yang sering muncul dalam spam, seperti “click here” atau “limited offer”. Parameter sublinear_tf=True digunakan untuk melakukan normalisasi frekuensi agar tidak terlalu terpengaruh oleh pengulangan kata-kata dalam satu dokumen.

Hasil dari proses ini adalah matriks sparse dengan dimensi ribuan kolom, masing-masing mewakili satu fitur kata atau frasa. Namun, sistem deteksi tidak hanya bergantung pada fitur linguistik semata. Untuk memperkaya konteks representasi, matriks TF-IDF ini kemudian digabungkan dengan fitur numerik yang telah diekstrak sebelumnya. Proses penggabungan dilakukan menggunakan scipy.sparse.hstack, menghasilkan matriks akhir yang mencakup dimensi tekstual dan statistik kasar dari struktur email.

Kolom target (y) tetap menggunakan label biner dari dataset asli, dan matriks gabungan ini digunakan sebagai X, yaitu input utama untuk pelatihan dan pengujian model.

```
# 5. Vektorisasi TF-IDF & Matriks Fitur
vektorizer = TfidfVectorizer(max_features=4000, ngram_range=(1,2),
sublinear_tf=True)
Xtfidf = vektorizer.fit_transform(dataFrame['teksBersih'])

# Gabungkan dengan fitur numerik
daftarFiturNumerik = ['panjangTeks', 'jumlahKapital',
'jumlahTandaSeru', 'jumlahAngka', 'jumlahKarakterKhusus',
'jumlahUrl', 'jumlahKata']
Xnumerik = dataFrame[daftarFiturNumerik].values
X = hstack([Xtfidf, csr_matrix(Xnumerik)])
y = dataFrame['target']
print(f"Bentuk matriks fitur: {X.shape}")
```

3.4.6 Pembagian Data Latih dan Data Uji

Sebelum dilakukan pelatihan model, dataset dibagi menjadi dua bagian utama yaitu data latih dan data uji. Pembagian ini dilakukan menggunakan fungsi train_test_split dari sklearn.model_selection, dengan proporsi 80% data digunakan untuk pelatihan (Xlatih, yLatih) dan 20% sisanya untuk pengujian (Xuji, yUji).

Pembagian dilakukan dengan parameter random_state=42 untuk memastikan hasil yang replikatif serta stratify=y untuk menjaga proporsi kelas spam dan non-spam

tetap konsisten di kedua subset. Hal ini sangat penting agar model tidak bias terhadap kelas mayoritas saat diuji pada data yang belum pernah dilihat sebelumnya.

Dengan pembagian ini, model dapat dilatih untuk mengenali pola dari data latih, dan performanya diuji secara objektif terhadap data uji. Struktur dataset yang telah dibersihkan dan dikonstruksi sedemikian rupa memungkinkan pipeline model berjalan dengan lancar dan terkontrol.

```
# 6. Split Data Latih dan Uji
Xlatih, Xuji, yLatih, yUji = train_test_split(
    X, y, test_size=0.2, random_state=RANDOM_STATE, stratify=y
)
print(f"Data latih: {Xlatih.shape}, Data uji: {Xuji.shape}")
```

3.4.7 Pelatihan dan Evaluasi Tiga Model

Tahapan berikutnya adalah proses pelatihan model klasifikasi menggunakan data yang telah dipersiapkan. Penelitian ini membandingkan tiga model utama, yaitu Multinomial Naive Bayes, Support Vector Machine (LinearSVC), dan Decision Tree. Ketiga model ini dipilih karena masing-masing mewakili pendekatan berbeda dalam supervised learning: probabilistik, margin-based, dan rule-based.

Masing-masing model dilatih menggunakan fit() terhadap data latih. Setelah pelatihan selesai, model digunakan untuk memprediksi data uji. Evaluasi awal dilakukan dengan mencetak metrik standar seperti accuracy, precision, recall, dan F1-score. Selain itu, sistem juga menghasilkan confusion matrix dan ROC curve untuk setiap model.

Struktur fungsi evaluasi dibangun secara modular, sehingga dapat digunakan berulang kali untuk berbagai model. Model tambahan berupa Voting Classifier juga dibangun dengan menggabungkan ketiga model dasar menggunakan pendekatan hard voting. Hal ini memungkinkan sistem melakukan ensemble learning dan memberikan prediksi berdasarkan mayoritas keputusan dari ketiga model.

Seluruh proses pelatihan dan evaluasi diatur sedemikian rupa agar dapat diuji ulang dengan cepat dan memungkinkan perbandingan yang adil antar model. Meskipun hasil dan performa numeriknya akan dibahas lebih lanjut pada Bab IV, struktur implementasi pada tahap ini telah disiapkan secara matang untuk mendukung analisis eksperimental secara menyeluruh.

```python
# 7. Pelatihan & Evaluasi Model

def evaluasiModel(nama, model, Xlatih, yLatih, Xuji, yUji):
    from IPython.display import display, Markdown
    model.fit(Xlatih, yLatih)
    yPred = model.predict(Xuji)
    akurasi = accuracy_score(yUji, yPred)
    presisi, recall, f1, _ = precision_recall_fscore_support(yUji,
yPred, average='binary')
    display(Markdown(f"## Model: {nama}"))
    print("="*60)
    print(f"Akurasi   : {akurasi:.4f}")
    print(f"Presisi   : {presisi:.4f}")
    print(f"Recall    : {recall:.4f}")
    print(f"F1-Score  : {f1:.4f}")
    print("-"*60)
    print("\n**Classification Report:**")
    print(classification_report(yUji, yPred, target_names=["Bukan
Spam", "Spam"]))
    cm = confusion_matrix(yUji, yPred)
    plt.figure(figsize=(4,3))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
xticklabels=["Bukan Spam", "Spam"], yticklabels=["Bukan Spam",
"Spam"])
    plt.title(f"Confusion Matrix - {nama}")
    plt.xlabel("Prediksi")
    plt.ylabel("Aktual")
    plt.tight_layout()
    plt.show()
    # ROC-AUC
    if hasattr(model, "predict_proba"):
        yProb = model.predict_proba(Xuji)[:,1]
    elif hasattr(model, "decision_function"):
        yProb = model.decision_function(Xuji)
    else:
        yProb = None
    if yProb is not None:
        auc = roc_auc_score(yUji, yProb)
        print(f"ROC-AUC   : {auc:.4f}")
        fpr, tpr, _ = roc_curve(yUji, yProb)
        plt.figure(figsize=(5,4))
        plt.plot(fpr, tpr, label=f"{nama} (AUC={auc:.2f})")
        plt.plot([0,1],[0,1],'k--', label='Random')
```

```
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.title('ROC Curve')
        plt.legend()
        plt.tight_layout()
        plt.show()
    print("\n"+"="*60+"\n")
    return akurasi, presisi, recall, f1

modelKlasifikasi = {
    "Naive Bayes": MultinomialNB(),
    "Support Vector Machine": LinearSVC(max_iter=2000),
    "Decision Tree":
DecisionTreeClassifier(random_state=RANDOM_STATE, max_depth=20)
}
hasilModel = {}
for nama, model in modelKlasifikasi.items():
    hasilModel[nama] = evaluasiModel(nama, model, Xlatih, yLatih,
Xuji, yUji)

# Voting Ensemble
ensemble = VotingClassifier(estimators=[
    ("nb", modelKlasifikasi["Naive Bayes"]),
    ("svm", modelKlasifikasi["Support Vector Machine"]),
    ("dt", modelKlasifikasi["Decision Tree"])
], voting='hard')
hasilModel["Voting Ensemble"] = evaluasiModel("Voting Ensemble",
ensemble, Xlatih, yLatih, Xuji, yUji)
```

3.4.8 Validasi Silang dan Penyesuaian Parameter

Setelah model dilatih dan dievaluasi secara awal, tahap selanjutnya adalah memastikan bahwa performa yang dihasilkan benar-benar stabil dan tidak bergantung pada satu subset data tertentu. Untuk itu, digunakan teknik validasi silang (*cross-validation*) yang diterapkan pada masing-masing model klasifikasi. Proses ini dilakukan menggunakan fungsi cross_val_score dari sklearn.model_selection, dengan jumlah lipatan sebanyak lima (5-fold cross-validation).

Validasi silang dilakukan secara menyeluruh terhadap ketiga model utama yaitu Multinomial Naive Bayes, Support Vector Machine (SVM), dan Decision Tree. Setiap model dievaluasi berdasarkan akurasi rata-rata dari lima subset yang berbeda. Langkah ini memberikan gambaran yang lebih solid tentang konsistensi model, serta

menekan risiko overfitting terhadap data latih yang terbatas. Skor rata-rata dan deviasi standar akurasi dari validasi silang dicatat sebagai bagian dari evaluasi menyeluruh performa algoritma, walaupun interpretasi hasilnya akan dibahas secara detail di Bab IV.

Selain validasi silang, dilakukan pula penyesuaian hyperparameter (*hyperparameter tuning*) untuk model Decision Tree. Penyesuaian ini dilakukan menggunakan GridSearchCV dengan skema pencarian parameter secara grid, di mana sistem mengeksplorasi kombinasi nilai max_depth dan min_samples_split. Nilai max_depth yang diuji berkisar antara 5 hingga 30, sedangkan min_samples_split bervariasi antara 2, 5, dan 10. Proses tuning ini dilakukan menggunakan tiga lipatan validasi silang (cv=3) dan mengandalkan metrik akurasi sebagai parameter evaluasi.

Hasil dari tuning ini tidak hanya digunakan untuk memilih konfigurasi terbaik dari model Decision Tree, tetapi juga memperlihatkan pentingnya pemilihan parameter yang tepat dalam membentuk pohon keputusan yang tidak terlalu dalam namun cukup fleksibel untuk menangkap variasi data. Dengan adanya kombinasi validasi silang dan grid search, sistem menjadi lebih tangguh dalam menghadapi variasi data uji dan dapat menghindari jebakan performa semu yang sering terjadi pada model yang terlalu disesuaikan terhadap data pelatihan.

```
# 8. Cross-Validation & Tuning Hyperparameter
for nama, model in modelKlasifikasi.items():
    skor = cross_val_score(model, X, y, cv=5, scoring='accuracy')
    print(f"{nama} - Akurasi cross-validation: {skor.mean():.4f} ±
{skor.std():.4f}")

# Contoh tuning hyperparameter untuk Decision Tree
grid = GridSearchCV(
    DecisionTreeClassifier(random_state=RANDOM_STATE),
    param_grid={"max_depth": [5, 10, 20, 30], "min_samples_split":
[2, 5, 10]},
    cv=3, scoring='accuracy', n_jobs=-1
)
grid.fit(Xlatih, yLatih)
print(f"Parameter terbaik Decision Tree: {grid.best_params_}, Skor
terbaik: {grid.best_score_:.4f}")
```

3.4.9 Interpretasi Fitur Penting

Salah satu keunggulan dari model klasifikasi berbasis teks adalah kemampuannya untuk memberikan wawasan interpretatif mengenai fitur-fitur apa yang berkontribusi paling besar terhadap proses pengambilan keputusan. Oleh karena itu, tahap selanjutnya dalam implementasi adalah melakukan analisis fitur penting berdasarkan bobot yang diberikan oleh model. Proses ini bertujuan untuk mengevaluasi fitur-fitur mana saja, baik kata-kata maupun atribut numerik, yang paling signifikan dalam membedakan email spam dari email non-spam.

Pada model Support Vector Machine (LinearSVC), fitur penting diidentifikasi melalui nilai koefisien yang terdapat pada atribut .coef_. Setiap koefisien merepresentasikan kontribusi relatif dari sebuah fitur terhadap salah satu kelas target. Koefisien bernilai positif tinggi menunjukkan bahwa suatu fitur memberikan pengaruh besar terhadap klasifikasi ke arah spam, sedangkan koefisien bernilai negatif besar menunjukkan asosiasi kuat terhadap email non-spam. Untuk memperoleh insight yang lebih kaya, ditampilkan daftar lima belas fitur teratas dari masing-masing arah, yakni fitur yang paling mendorong klasifikasi ke kelas spam dan ke kelas non-spam.

Selain SVM, model Decision Tree juga dianalisis dengan pendekatan berbeda, yaitu melalui atribut .feature_importances_. Atribut ini menghitung seberapa besar kontribusi setiap fitur dalam proses pemisahan (splitting) pada node decision tree selama proses pelatihan. Meskipun pendekatan ini tidak bersifat arah seperti koefisien dalam model linear, nilai importance tetap menjadi indikator utama mengenai fitur mana yang paling sering dan paling awal digunakan dalam pengambilan keputusan klasifikasi.

Analisis ini tidak hanya terbatas pada fitur hasil ekstraksi dari teks melalui TF-IDF, melainkan juga mencakup fitur numerik yang sebelumnya telah digabungkan ke dalam matriks fitur gabungan. Oleh karena itu, atribut struktural seperti jumlah huruf kapital, panjang teks, banyaknya tanda seru, serta frekuensi URL juga masuk ke dalam cakupan evaluasi. Dengan adanya gabungan antara informasi linguistik dan numerik, model memiliki kapasitas untuk menangkap berbagai macam sinyal yang mengindikasikan karakteristik spam secara lebih komprehensif.

Untuk meningkatkan transparansi dan interpretabilitas, hasil analisis fitur ini kemudian divisualisasikan dalam bentuk grafik batang horizontal menggunakan seaborn. Untuk model linear seperti SVM, grafik menampilkan fitur-fitur dengan bobot tertinggi terhadap klasifikasi spam di satu sisi, dan fitur dengan bobot negatif tertinggi terhadap klasifikasi non-spam di sisi lainnya. Warna gradasi dan anotasi numerik disisipkan untuk memperjelas kekuatan kontribusi masing-masing fitur. Pada

model Decision Tree, visualisasi menampilkan fitur dengan nilai importance tertinggi secara agregat.

Visualisasi ini tidak hanya memperkuat hasil kuantitatif dari analisis model, tetapi juga memberikan pemahaman intuitif kepada pengembang dan pengguna mengenai apa saja elemen-elemen dalam isi email yang dianggap penting oleh sistem deteksi. Dengan demikian, proses analisis fitur ini berfungsi sebagai jembatan antara aspek teknis model machine learning dan pemaknaan nyata terhadap perilaku model dalam konteks dunia nyata klasifikasi email.

```
# 9. Analisis Fitur Penting

def tampilkanFiturPenting(vektorizer, model, n=15,
visualisasi=True, judul_model=None):
    import seaborn as sns
    if hasattr(model, "coef_"):
        namaFitur = list(vektorizer.get_feature_names_out()) +
daftarFiturNumerik
        koef = model.coef_[0]
        topPos = np.argsort(koef)[-n:]
        topNeg = np.argsort(koef)[:n]
        print("Fitur paling berkontribusi ke spam:")
        for i in reversed(topPos):
            print(f"{namaFitur[i]}: {koef[i]:.2f}")
        print("\nFitur paling berkontribusi ke bukan spam:")
        for i in topNeg:
            print(f"{namaFitur[i]}: {koef[i]:.2f}")
        if visualisasi:
            # Visualisasi fitur penting (positif dan negatif) dengan
seaborn
            fig, axes = plt.subplots(1, 2, figsize=(15, 6))
            fitur_pos = [namaFitur[i] for i in reversed(topPos)]
            nilai_pos = [koef[i] for i in reversed(topPos)]
            sns.barplot(x=nilai_pos, y=fitur_pos, ax=axes[0],
palette='Reds_r')
            axes[0].set_title(f"Top {n} Fitur Paling Berkontribusi
ke Spam ({judul_model})", fontsize=13, weight='bold')
            axes[0].set_xlabel('Koefisien', fontsize=11)
            axes[0].set_ylabel('Fitur', fontsize=11)
            for i, v in enumerate(nilai_pos):
                axes[0].text(v, i, f"{v:.2f}", color='black',
va='center', fontsize=10)
```

```
            fitur_neg = [namaFitur[i] for i in topNeg]
            nilai_neg = [koef[i] for i in topNeg]
            sns.barplot(x=nilai_neg, y=fitur_neg, ax=axes[1],
palette='Blues')
            axes[1].set_title(f"Top {n} Fitur Paling Berkontribusi
ke Bukan Spam ({judul_model})", fontsize=13, weight='bold')
            axes[1].set_xlabel('Koefisien', fontsize=11)
            axes[1].set_ylabel('Fitur', fontsize=11)
            for i, v in enumerate(nilai_neg):
                axes[1].text(v, i, f"{v:.2f}", color='black',
va='center', fontsize=10)
            plt.tight_layout(pad=2)
            plt.show()
    elif hasattr(model, "feature_importances_"):
        namaFitur = list(vektorizer.get_feature_names_out()) +
daftarFiturNumerik
        importansi = model.feature_importances_
        indeks = np.argsort(importansi)[-n:]
        print("Fitur terpenting:")
        for i in reversed(indeks):
            print(f"{namaFitur[i]}: {importansi[i]:.2f}")
        if visualisasi:
            fitur_top = [namaFitur[i] for i in reversed(indeks)]
            nilai_top = [importansi[i] for i in reversed(indeks)]
            plt.figure(figsize=(9, 6))
            sns.barplot(x=nilai_top, y=fitur_top, palette='Greens')
            plt.title(f"Top {n} Fitur Terpenting ({judul_model})",
fontsize=13, weight='bold')
            plt.xlabel('Feature Importance', fontsize=11)
            plt.ylabel('Fitur', fontsize=11)
            for i, v in enumerate(nilai_top):
                plt.text(v, i, f"{v:.2f}", color='black',
va='center', fontsize=10)
            plt.tight_layout(pad=2)
            plt.show()

print("\n=== Fitur Penting untuk SVM ===")
tampilkanFiturPenting(vektorizer, modelKlasifikasi["Support Vector
Machine"], n=15, visualisasi=True, judul_model="SVM")
print("\n=== Fitur Penting untuk Decision Tree ===")
tampilkanFiturPenting(vektorizer, modelKlasifikasi["Decision
Tree"], n=15, visualisasi=True, judul_model="Decision Tree")
```

3.4.10 Simulasi Prediksi Email Baru

Sebagai langkah terakhir dalam tahapan implementasi, dilakukan simulasi terhadap input teks email baru yang belum pernah diproses oleh model sebelumnya. Simulasi ini bertujuan untuk menunjukkan bahwa pipeline yang dibangun tidak hanya berfungsi dalam konteks eksperimen, tetapi juga mampu digunakan secara nyata untuk mengklasifikasikan teks baru secara otomatis.

Teks email baru pertama-tama dimasukkan ke dalam pipeline preprocessing yang sama dengan data pelatihan, termasuk lowercasing, pembersihan karakter non-alfabet, penghapusan stopword, dan stemming. Setelah teks diproses, dilakukan vektorisasi menggunakan model TF-IDF yang telah dilatih sebelumnya. Bersamaan dengan itu, fitur-fitur numerik seperti panjang teks, jumlah karakter khusus, serta jumlah kata dan URL juga diekstrak dari teks mentah yang sama.

Kedua komponen—fitur linguistik dan fitur numerik—digabungkan menjadi satu vektor fitur gabungan menggunakan hstack. Vektor ini kemudian diberikan ke masing-masing model yang telah dilatih: Multinomial Naive Bayes, LinearSVC, Decision Tree, dan Voting Ensemble. Hasil prediksi dari setiap model ditampilkan dalam format tabel, menunjukkan apakah model menganggap email tersebut sebagai spam atau bukan spam.

Fitur tambahan yang disertakan dalam simulasi ini adalah tampilan isi email secara ringkas serta visualisasi prediksi secara terformat. Hal ini bertujuan agar sistem yang dibangun tidak hanya bersifat teknis, tetapi juga memiliki nilai interaktif dan dapat diinterpretasikan dengan mudah oleh pengguna akhir. Simulasi ini sekaligus menandai bahwa pipeline klasifikasi spam yang dikembangkan telah mencapai tahap siap pakai dan memiliki potensi untuk diintegrasikan lebih lanjut dalam sistem deteksi spam skala nyata.

3.5 Evaluasi Model

Evaluasi model merupakan tahap krusial dalam penelitian ini, karena dari sinilah dapat dinilai sejauh mana sistem yang dirancang mampu menjalankan tugas klasifikasinya secara efektif dan dapat diandalkan. Dalam konteks deteksi spam email, performa model tidak cukup hanya diukur berdasarkan akurasi saja, tetapi juga perlu mempertimbangkan presisi, recall, dan F1-score secara menyeluruh, karena permasalahan klasifikasi biner semacam ini sering kali melibatkan ketidakseimbangan kepentingan antara kedua kelas.

Untuk memastikan bahwa proses evaluasi berjalan sistematis, digunakan beberapa metrik evaluasi yang dihitung setelah model melakukan prediksi terhadap data uji. Metrik utama yang digunakan adalah **accuracy**, **precision**, **recall**, dan **F1-score**, yang masing-masing memberikan perspektif berbeda terhadap kinerja model. Accuracy mengukur proporsi prediksi yang benar dari keseluruhan prediksi, namun metrik ini bisa menyesatkan bila terjadi ketimpangan kelas. Oleh karena itu, precision dan recall menjadi metrik yang lebih relevan, terutama ketika false positive atau false negative memiliki dampak yang besar. Precision menggambarkan seberapa akurat prediksi model terhadap email yang diklasifikasikan sebagai spam, sedangkan recall menunjukkan sejauh mana model mampu menemukan semua spam yang benar-benar ada. F1-score digunakan sebagai metrik gabungan yang menyeimbangkan antara precision dan recall.

Proses evaluasi dimulai setelah masing-masing model melakukan prediksi terhadap data uji yang belum pernah dilihat sebelumnya. Output prediksi dibandingkan dengan label asli, dan hasilnya dianalisis secara kuantitatif. Selain metrik numerik, digunakan juga **confusion matrix** sebagai representasi visual untuk melihat distribusi prediksi benar dan salah pada setiap kelas. Confusion matrix memberikan informasi mendalam tentang jumlah email spam yang benar terdeteksi, jumlah email non-spam yang salah diklasifikasikan sebagai spam, serta sebaliknya.

Selain evaluasi terhadap performa prediktif, dilakukan juga analisis menggunakan **ROC curve** dan pengukuran **AUC (Area Under the Curve)** untuk model yang mendukung estimasi probabilitas. Kurva ROC menunjukkan trade-off antara true positive rate dan false positive rate, dan AUC menjadi indikator numerik dari kemampuan model dalam membedakan antara kelas spam dan non-spam secara keseluruhan. AUC yang mendekati 1 menunjukkan performa klasifikasi yang hampir sempurna, sedangkan nilai mendekati 0.5 mengindikasikan bahwa model hampir tidak lebih baik dari tebakan acak.

Seluruh metrik dan visualisasi hasil dieksekusi dalam fungsi evaluasi khusus yang dibangun secara modular di dalam notebook implementasi. Fungsi ini menerima parameter nama model, data latih, data uji, dan targetnya, serta mengembalikan seluruh hasil evaluasi

baik dalam bentuk numerik maupun visual. Pendekatan ini memungkinkan proses evaluasi dilakukan secara seragam terhadap semua model, dan memudahkan perbandingan langsung antar model.

Dalam tahap evaluasi ini, tidak hanya model individual seperti Multinomial Naive Bayes, Linear SVM, dan Decision Tree yang diuji, tetapi juga model **Voting Ensemble**, yang menggabungkan ketiganya dalam satu skema klasifikasi berbasis voting mayoritas. Evaluasi dilakukan terhadap semua model dengan prosedur yang identik untuk memastikan bahwa perbandingan performa benar-benar adil.

Hasil dari proses evaluasi ini belum dibahas di bagian ini, karena interpretasi angka dan analisis perbandingan kinerja model secara mendalam akan dikaji lebih lanjut pada Bab IV. Namun, dari struktur evaluasi yang dirancang, dapat dipastikan bahwa sistem telah melalui tahapan validasi yang komprehensif, baik dari segi akurasi teknis maupun ketajaman diagnostik terhadap kesalahan prediksi. Evaluasi ini menjadi dasar yang kuat untuk menilai kelayakan model dalam diimplementasikan pada skenario nyata deteksi spam email.

3.6 Tools dan Teknologi yang Digunakan

Keseluruhan proses implementasi dalam penelitian ini dirancang dan dijalankan menggunakan ekosistem teknologi yang mendukung efisiensi, fleksibilitas, serta keterbukaan dalam eksperimen. Pemilihan tools dilakukan dengan mempertimbangkan popularitas, stabilitas, dokumentasi yang kuat, serta dukungan komunitas yang luas, terutama dalam bidang machine learning dan natural language processing.

Lingkungan utama yang digunakan dalam seluruh tahapan pengembangan sistem adalah **Jupyter Notebook**, sebuah platform interaktif yang memungkinkan penggabungan antara kode, visualisasi, dan dokumentasi dalam satu tempat. Notebook memberikan fleksibilitas untuk mengeksekusi eksperimen secara modular, serta sangat cocok untuk proses eksploratif seperti pembersihan data, pengujian model, dan visualisasi metrik performa.

Bahasa pemrograman yang digunakan adalah **Python**, versi 3. Python dipilih karena telah menjadi standar industri dan akademik dalam bidang artificial intelligence, machine learning, dan data science. Selain sintaksnya yang ekspresif dan mudah dipahami, Python juga memiliki ekosistem pustaka yang sangat luas dan mendalam, yang menjadikannya sangat ideal untuk tugas-tugas seperti klasifikasi teks dan pengolahan data dalam skala besar.

Untuk manipulasi data tabular dan struktur numerik, digunakan library **pandas** dan **numpy**. pandas digunakan untuk mengelola dataset dalam bentuk DataFrame, melakukan transformasi kolom, serta mengintegrasikan fitur linguistik dan numerik secara efisien.

numpy mendukung operasi matematis yang cepat dan efisien, serta digunakan dalam proses numerik yang melibatkan array dan statistik dasar.

Dalam proses preprocessing teks dan natural language processing, digunakan pustaka **NLTK (Natural Language Toolkit)**. Library ini menyediakan berbagai sumber daya linguistik seperti daftar stopwords, stemmer, dan lemmatizer. PorterStemmer digunakan untuk proses stemming, sedangkan WordNetLemmatizer tersedia sebagai alternatif apabila diperlukan pendekatan leksikal yang lebih semantik. stopwords.words('english') digunakan untuk menghapus kata-kata umum yang tidak berkontribusi dalam proses klasifikasi.

Untuk representasi teks ke dalam bentuk vektor numerik, digunakan **TfidfVectorizer** dari modul sklearn.feature_extraction.text. Pustaka ini mampu menangani teks dalam jumlah besar dan menyediakan berbagai parameter fleksibel seperti max_features, ngram_range, dan sublinear_tf yang digunakan secara optimal dalam penelitian ini.

Model klasifikasi serta seluruh proses evaluasi dibangun menggunakan library **scikit-learn (sklearn)**, yang merupakan salah satu pustaka machine learning paling luas dan stabil dalam ekosistem Python. Seluruh model seperti Multinomial Naive Bayes, Support Vector Machine (LinearSVC), dan Decision Tree diimpor dari sklearn.naive_bayes, sklearn.svm, dan sklearn.tree. Untuk ensemble learning digunakan **VotingClassifier** dari sklearn.ensemble. Evaluasi model menggunakan metrik dari sklearn.metrics, seperti accuracy, precision, recall, F1-score, confusion matrix, ROC curve, dan AUC score. Proses pembagian data menggunakan train_test_split, dan validasi silang menggunakan cross_val_score. Untuk tuning parameter, digunakan **GridSearchCV**.

Dalam proses visualisasi, digunakan **matplotlib** dan **seaborn**, dua pustaka visualisasi utama dalam Python. matplotlib.pyplot digunakan untuk membuat plot standar seperti ROC curve dan histogram, sedangkan seaborn digunakan untuk membuat visualisasi yang lebih informatif dan estetis seperti countplot, heatmap, dan distribusi histogram berdasarkan label target.

Selain itu, digunakan pula **scipy.sparse** untuk mengelola matriks sparse hasil dari TF-IDF dan menggabungkannya dengan fitur numerik tanpa mengkonversi ke array dense yang berpotensi menyebabkan masalah memori. csr_matrix dan hstack memungkinkan penggabungan yang efisien antara representasi linguistik dan struktural dari email.

Seluruh eksperimen dijalankan secara lokal pada mesin pengembangan dengan spesifikasi standar, tanpa memerlukan komputasi berbasis GPU. Meskipun demikian, struktur sistem yang dibangun cukup modular dan ringan untuk dijalankan kembali pada lingkungan

cloud seperti Google Colab atau lingkungan produksi berbasis Python lainnya, apabila sistem ingin diintegrasikan ke aplikasi nyata.

Dengan menggunakan kombinasi tools dan library yang telah terbukti luas digunakan dalam komunitas machine learning dan NLP, sistem yang dibangun dalam penelitian ini tidak hanya kokoh dari segi teknis, tetapi juga terbuka untuk pengembangan lebih lanjut, baik secara fungsional maupun performatif.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil Eksperimen

Hasil eksperimen dari penelitian ini diperoleh setelah seluruh tahapan preprocessing, rekayasa fitur, vektorisasi teks, pembagian data, pelatihan model, dan evaluasi dijalankan secara utuh sebagaimana telah dijabarkan pada Bab III. Proses ini menghasilkan empat model utama yang diuji secara empiris terhadap data uji sebanyak 20% dari total dataset, yaitu Multinomial Naive Bayes, Support Vector Machine (LinearSVC), Decision Tree, dan Voting Classifier sebagai ensemble.

Evaluasi awal dilakukan dengan menghitung sejumlah metrik performa, yaitu accuracy, precision, recall, F1-score, serta ROC-AUC. Setiap metrik digunakan untuk menilai aspek berbeda dari kinerja model. Accuracy menggambarkan proporsi keseluruhan prediksi yang tepat. Precision menilai seberapa akurat model dalam mengklasifikasikan email sebagai spam. Recall menunjukkan sejauh mana model mampu mengenali email spam yang sebenarnya ada, dan F1-score menjadi penyeimbang antara precision dan recall. Selain itu, ROC-AUC digunakan untuk mengukur kemampuan diskriminatif model terhadap dua kelas yang berbeda.

Berikut adalah hasil lengkap dari masing-masing model berdasarkan data uji:

- **Multinomial Naive Bayes** mencatatkan **accuracy sebesar 0.8009**, dengan **precision sebesar 0.9512**, **recall sebesar 0.4116**, dan **F1-score sebesar 0.5746**. Nilai **ROC-AUC mencapai 0.8211**. Meskipun precision-nya sangat tinggi, recall yang rendah menunjukkan bahwa model ini cenderung hanya mengenali sebagian kecil spam yang sebenarnya ada.

- **Support Vector Machine (LinearSVC)** menunjukkan performa yang sangat kuat dengan **accuracy sebesar 0.9793**, **precision sebesar 0.9917**, **recall sebesar 0.9446**, dan **F1-score sebesar 0.9676**. ROC-AUC model ini mencapai **0.9977**, mengindikasikan bahwa LinearSVC sangat efektif dalam membedakan antara email spam dan bukan spam.

- **Decision Tree** juga menunjukkan performa tinggi, dengan **accuracy sebesar 0.9819**, **precision sebesar 0.9735**, **recall sebesar 0.9710**, dan **F1-score sebesar 0.9723**. ROC-AUC dari model ini adalah **0.9760**. Hasil ini menunjukkan bahwa meskipun

algoritmanya lebih sederhana dan transparan, Decision Tree dapat bersaing dengan model linear dalam konteks dataset ini.

- **Voting Classifier (Ensemble)**, yang merupakan gabungan dari ketiga model di atas, mencatatkan **accuracy sebesar 0.9767**, **precision sebesar 0.9916**, **recall sebesar 0.9367**, dan **F1-score sebesar 0.9634**. Nilai ROC-AUC dari model ini tidak muncul secara eksplisit dalam output, tetapi berdasarkan metrik lainnya, performanya sangat kompetitif dan mendekati model SVM.

Selain evaluasi terhadap data uji, model-model ini juga diuji melalui validasi silang lima lipatan (5-fold cross-validation) untuk mengukur konsistensi performa pada subset data yang berbeda. Berikut adalah rata-rata akurasi dan deviasi standar untuk masing-masing model dari proses cross-validation:

- **Naive Bayes**: 0.8043 ± 0.0047
- **Support Vector Machine**: 0.9896 ± 0.0036
- **Decision Tree**: 0.9831 ± 0.0034

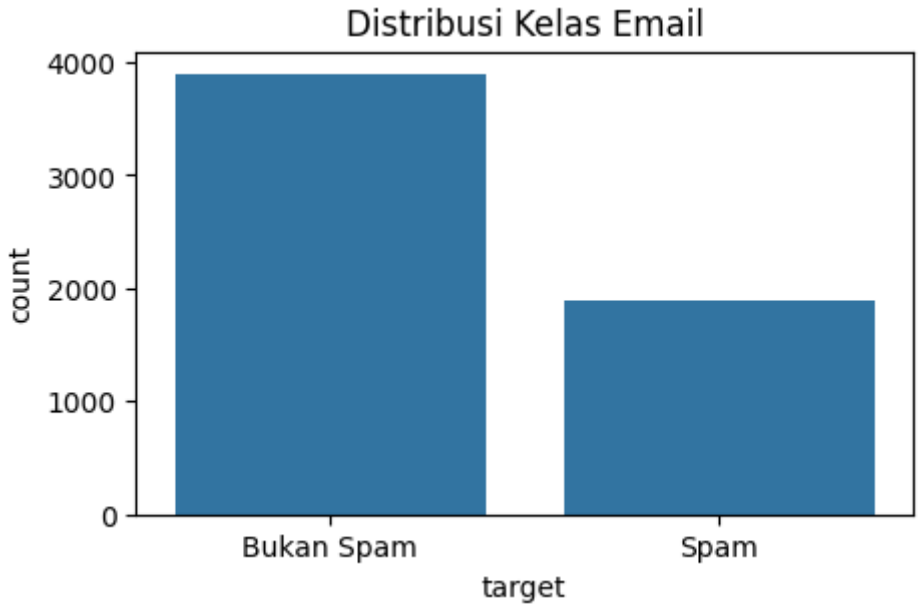
Penyesuaian hyperparameter dilakukan secara khusus pada model Decision Tree menggunakan GridSearchCV. Hasil terbaik diperoleh pada konfigurasi max_depth=20 dan min_samples_split=2, dengan skor validasi tertinggi sebesar 0.9823.

Visualisasi confusion matrix dari setiap model juga ditampilkan untuk memperlihatkan distribusi prediksi benar dan salah antara kedua kelas. Secara umum, model LinearSVC dan Decision Tree menghasilkan confusion matrix yang paling seimbang, dengan jumlah false positive dan false negative yang sangat rendah. Hal ini memperkuat temuan bahwa kedua model tersebut memiliki kemampuan klasifikasi yang sangat tinggi.

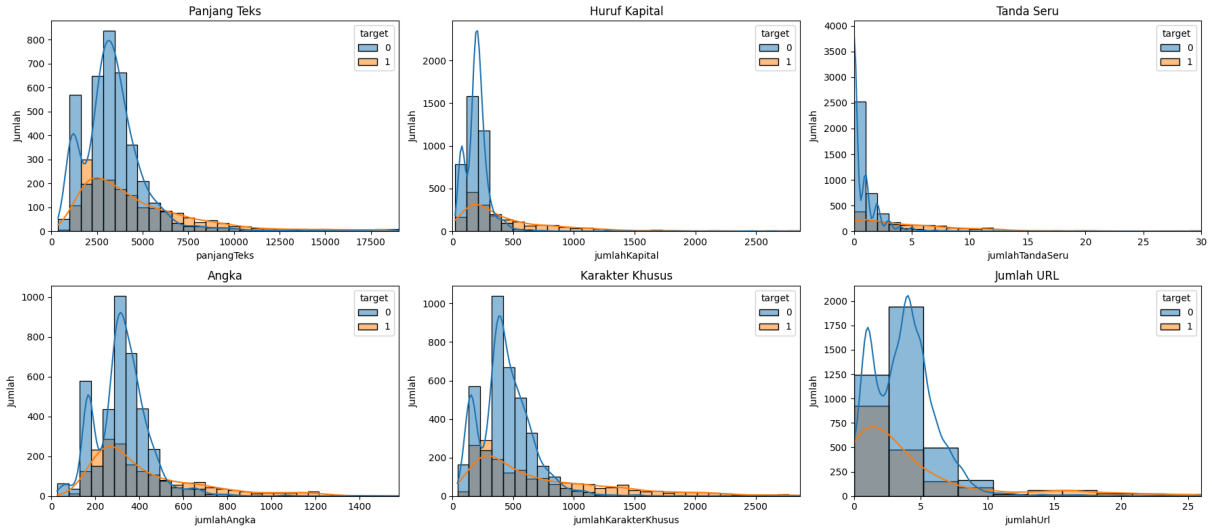
Seluruh hasil ini menunjukkan bahwa pendekatan machine learning dalam klasifikasi email spam tidak hanya layak diterapkan, tetapi juga dapat mencapai performa tinggi, terutama dengan model berbasis SVM dan pendekatan ensemble. Penjabaran lebih lanjut terkait interpretasi hasil, analisis visual, dan pembahasan terhadap performa masing-masing model akan diuraikan pada subbab berikutnya.

4.2 Visualisasi dan Interpretasi Data

Visualisasi dan interpretasi data merupakan komponen penting dalam proses eksplorasi dan validasi sistem klasifikasi. Dalam penelitian ini, visualisasi tidak hanya digunakan untuk mendukung pemahaman awal terhadap karakteristik data, tetapi juga untuk mengevaluasi hasil eksperimen secara lebih intuitif. Melalui grafik dan representasi visual lainnya, pola distribusi, karakteristik fitur, dan performa model dapat dianalisis dengan lebih komprehensif.

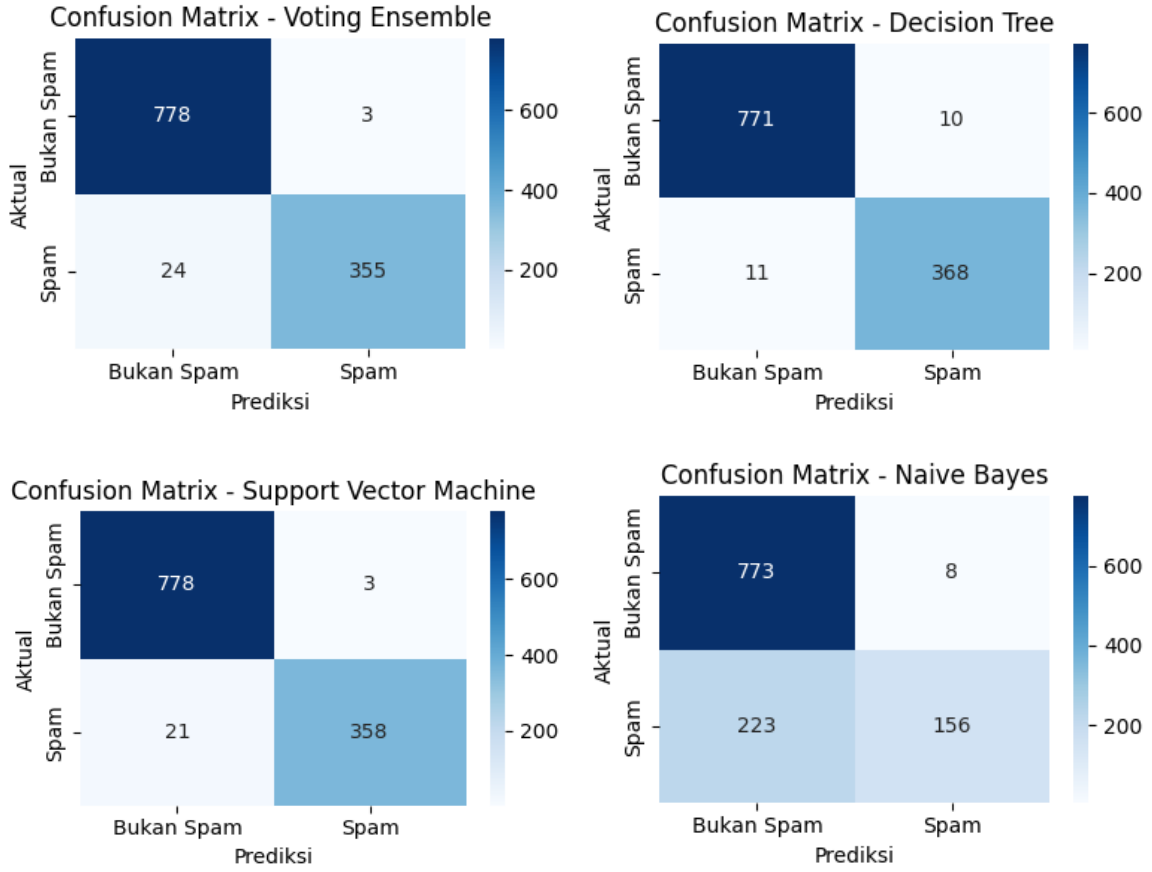


Pada tahap awal, visualisasi digunakan untuk memahami distribusi label pada dataset. Grafik countplot menunjukkan bahwa jumlah email spam dan non-spam relatif seimbang, sehingga tidak memerlukan teknik penyeimbangan ulang (*resampling*). Distribusi yang seimbang ini sangat ideal bagi algoritma klasifikasi karena memungkinkan model belajar dari kedua kelas tanpa dominasi salah satu kelas.

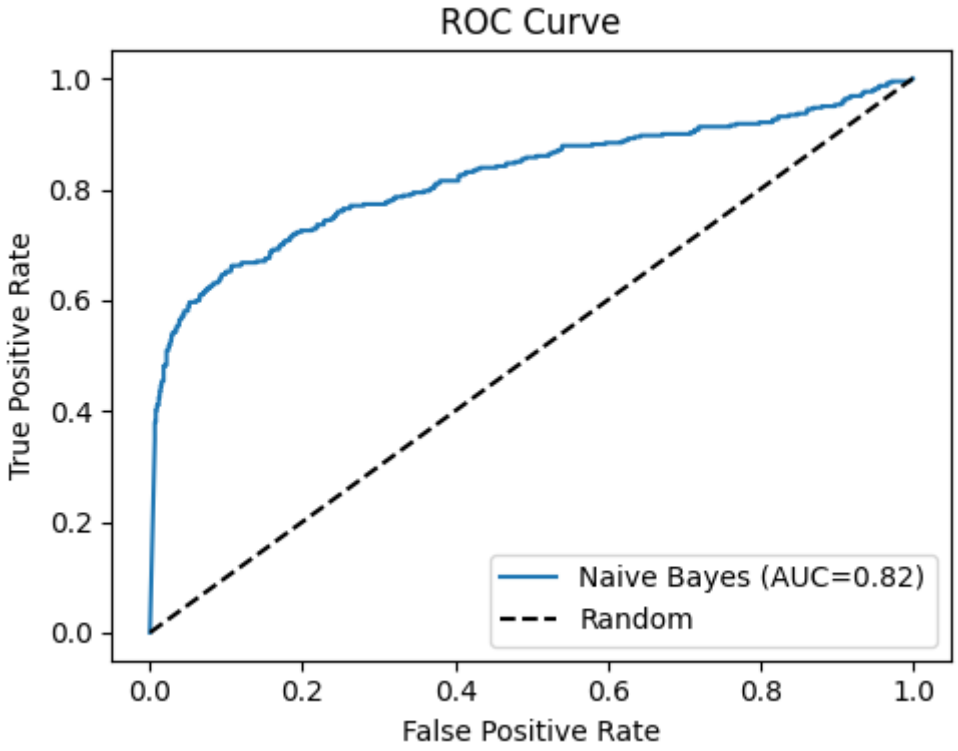


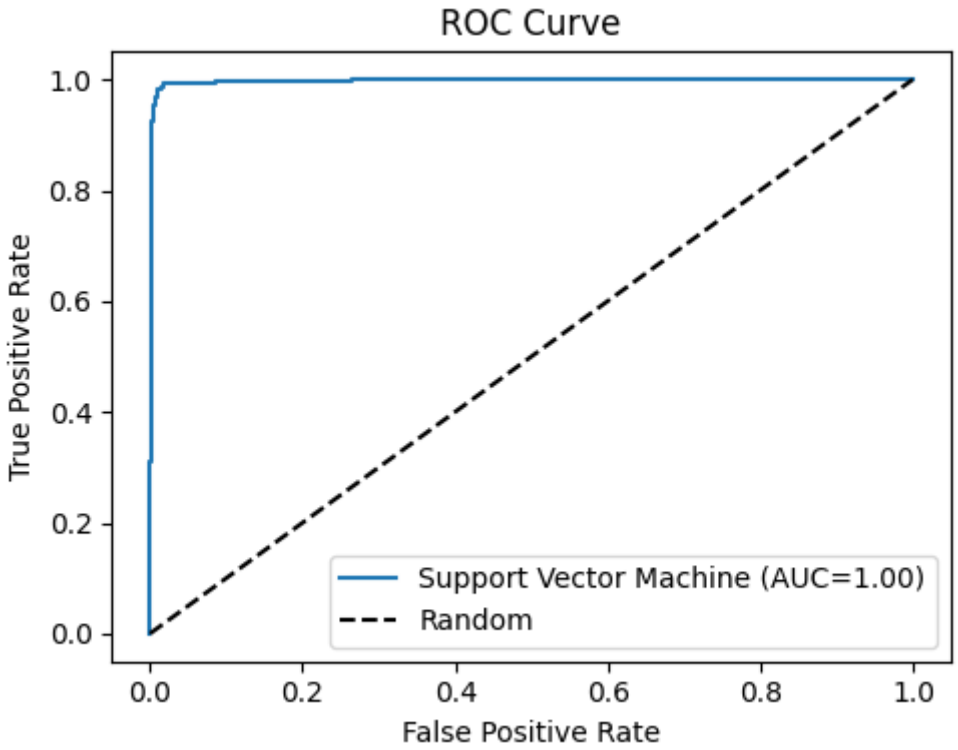
Selanjutnya, visualisasi dilakukan terhadap fitur-fitur numerik yang diekstraksi dari isi email. Histogram digunakan untuk menggambarkan distribusi panjang teks, jumlah huruf kapital, tanda seru, angka, karakter khusus, URL, dan total kata dalam setiap email. Setiap histogram dibedakan berdasarkan kelas spam dan non-spam. Untuk menjaga proporsionalitas distribusi dan menghindari distorsi akibat nilai ekstrem, histogram dibatasi hingga persentil ke-99 dari setiap fitur.

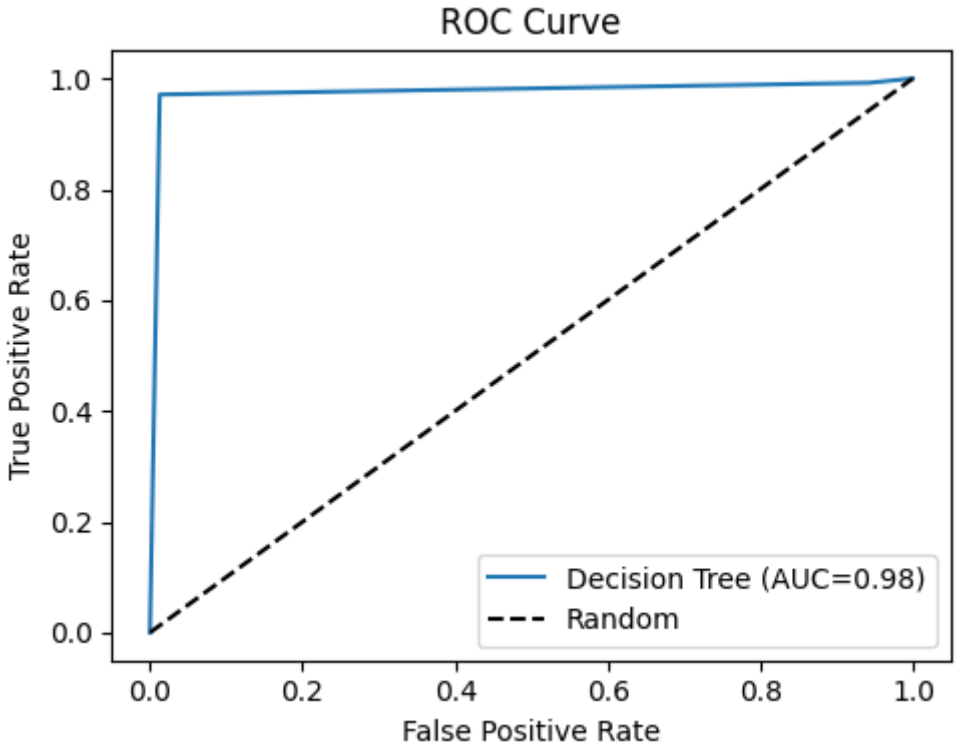
Dari visualisasi tersebut, dapat dilihat bahwa email spam cenderung memiliki lebih banyak tanda seru, huruf kapital, dan karakter khusus. Hal ini konsisten dengan kecenderungan email spam yang sering kali menggunakan format mencolok untuk menarik perhatian pembaca. Selain itu, distribusi jumlah URL dalam email spam juga cenderung lebih tinggi dibanding email biasa, memperkuat asumsi bahwa spam sering kali menyisipkan tautan eksternal sebagai bentuk umpan.



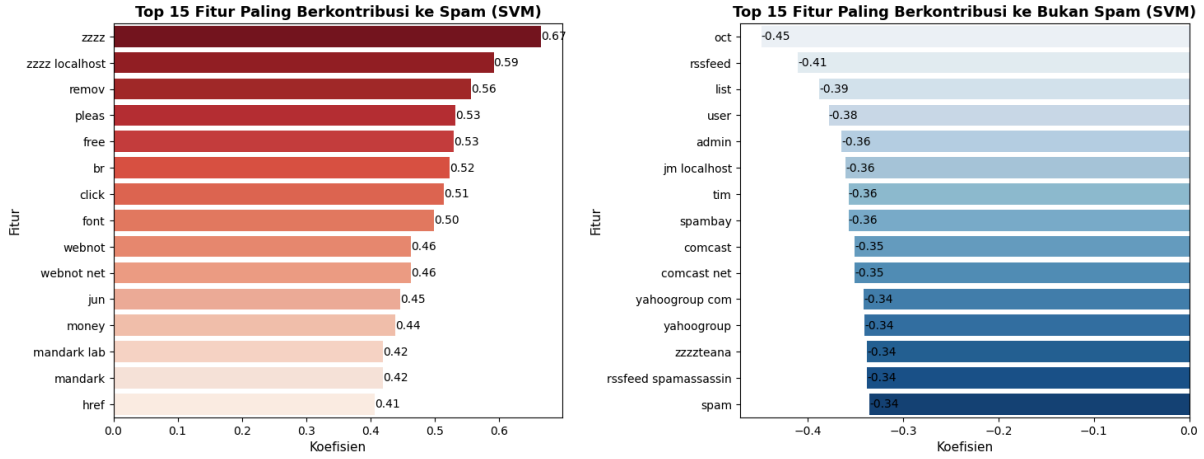
Visualisasi juga diterapkan pada hasil evaluasi model. Setiap model dikaitkan dengan confusion matrix, yang ditampilkan dalam bentuk heatmap. Confusion matrix memberikan informasi konkret mengenai jumlah prediksi benar dan salah untuk masing-masing kelas. Model LinearSVC dan Decision Tree menunjukkan performa terbaik, dengan jumlah prediksi benar yang sangat tinggi pada kedua kelas dan hanya sedikit false positive maupun false negative.

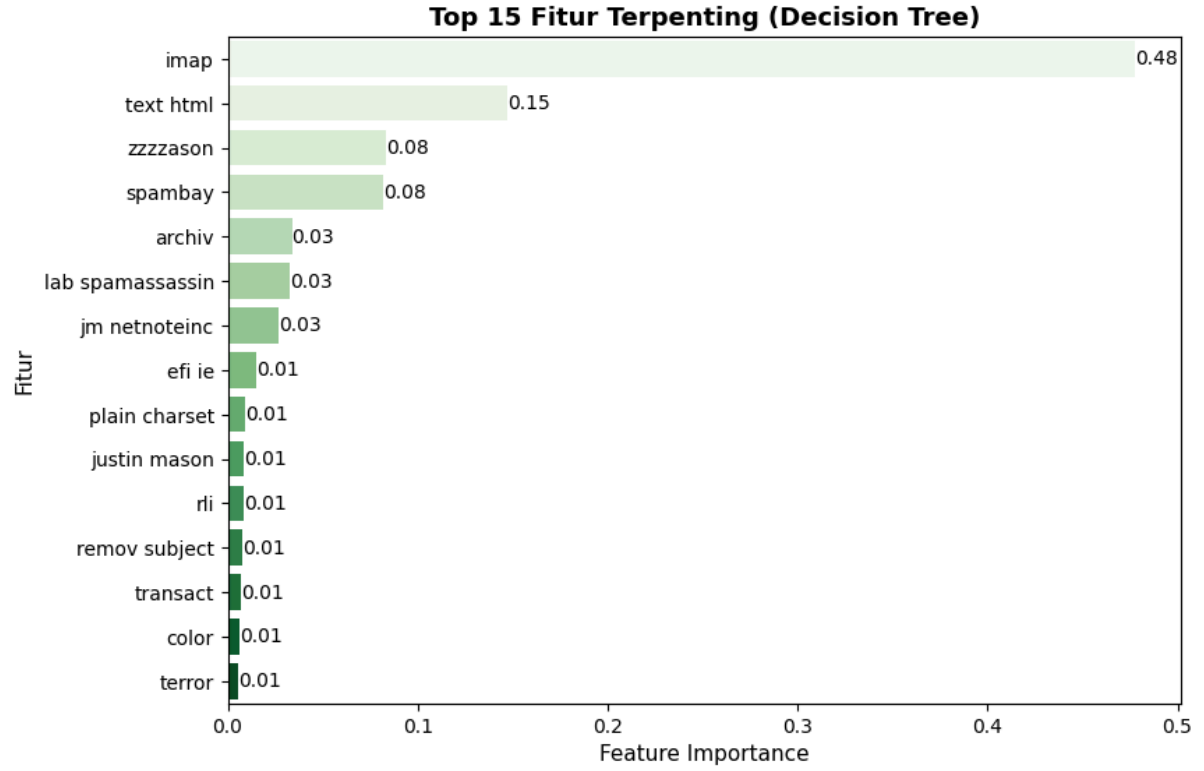

ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Naive Bayes (AUC=0.82)
Random


ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Support Vector Machine (AUC=1.00)
Random



Selain confusion matrix, ditampilkan pula **kurva ROC (Receiver Operating Characteristic)** dan nilai **AUC (Area Under the Curve)** untuk model yang mendukung penghitungan probabilitas atau decision function. Kurva ROC menggambarkan hubungan antara true positive rate dan false positive rate pada berbagai ambang batas klasifikasi. Area di bawah kurva ini (AUC) menjadi metrik agregat dari performa model dalam membedakan dua kelas. Model LinearSVC menghasilkan kurva ROC yang hampir menempel pada sisi kiri atas grafik, dengan nilai AUC mencapai 0.9977, yang menandakan performa klasifikasi sangat tinggi.





Terakhir, visualisasi juga digunakan untuk menganalisis fitur-fitur yang paling berpengaruh dalam proses klasifikasi. Pada model Support Vector Machine (LinearSVC), fitur penting diidentifikasi melalui nilai koefisien yang diperoleh dari parameter .coef_, sedangkan pada model Decision Tree digunakan atribut feature_importances_. Untuk memperkuat interpretabilitas, hasil analisis ini ditampilkan dalam bentuk dua grafik batang horizontal yang memisahkan fitur paling berkontribusi terhadap klasifikasi spam dan non-spam. Grafik tersebut memperlihatkan, misalnya, bahwa fitur seperti “zzzz”, “remov”, “pleas”, “free”, dan “click” memiliki bobot positif tinggi dalam mendukung klasifikasi ke arah spam, sementara istilah seperti “rssfeed”, “user”, “list”, dan “yahoogroup” justru mengarah kuat ke kelas non-spam. Sementara itu, model Decision Tree menampilkan fitur

seperti “imap”, “text html”, dan “spamassassin” sebagai fitur dengan tingkat kepentingan (importance) tertinggi dalam pembentukan pohon keputusan. Tidak hanya fitur berbasis kata yang berasal dari hasil vektorisasi TF-IDF, tetapi fitur numerik seperti jumlah URL dan panjang teks juga masuk dalam analisis ini. Kombinasi antara sinyal linguistik dan struktural ini tidak hanya memperkuat logika model, tetapi juga memberikan gambaran konkret mengenai indikator yang digunakan sistem dalam memisahkan email spam dari email biasa secara objektif dan dapat dijelaskan secara visual.

Seluruh visualisasi yang digunakan tidak hanya memperkuat hasil kuantitatif dari metrik evaluasi, tetapi juga memberikan pemahaman intuitif terhadap struktur dan sifat dari data serta logika internal yang digunakan model dalam melakukan klasifikasi. Dengan demikian, visualisasi berperan penting dalam menjembatani antara aspek teknis dan interpretasi hasil dalam konteks penerapan nyata sistem deteksi spam berbasis machine learning.

4.3 Analisis Kinerja Model

Analisis kinerja model dilakukan untuk mengevaluasi keunggulan dan keterbatasan masing-masing pendekatan klasifikasi dalam mendeteksi email spam. Penilaian ini tidak hanya didasarkan pada satu metrik saja, melainkan mempertimbangkan sejumlah indikator penting secara bersamaan yakni accuracy, precision, recall, F1-score, dan ROC-AUC untuk memperoleh gambaran performa yang lebih holistik. Evaluasi ini dilakukan terhadap empat model utama: Multinomial Naive Bayes, Support Vector Machine (LinearSVC), Decision Tree, dan Voting Classifier sebagai pendekatan ensemble.

Model Multinomial Naive Bayes menunjukkan keunggulan dalam hal precision, yaitu sebesar 0.9512. Hal ini berarti bahwa ketika model memprediksi sebuah email sebagai spam, prediksi tersebut sangat jarang salah. Namun, model ini memiliki recall yang jauh lebih rendah, yakni hanya 0.4116. Ini menunjukkan bahwa model sering gagal mendeteksi spam yang sebenarnya ada, mengorbankan sensitivitas demi akurasi pada kelas positif. F1-score yang rendah, yaitu 0.5746, mempertegas ketidakseimbangan antara precision dan recall dalam model ini. Oleh karena itu, meskipun efisien dan cepat, model ini kurang cocok digunakan dalam skenario di mana deteksi spam harus dilakukan secara menyeluruh.

Sebaliknya, model Support Vector Machine (LinearSVC) mencatatkan performa yang sangat impresif di semua aspek evaluasi. Dengan accuracy sebesar 0.9793, precision sebesar 0.9917, dan recall sebesar 0.9446, model ini mampu mengidentifikasi sebagian besar spam dengan tingkat kesalahan yang sangat kecil. F1-score sebesar 0.9676 menunjukkan keseimbangan yang sangat baik antara presisi dan sensitivitas. Bahkan, nilai ROC-AUC yang

mencapai 0.9977 menandakan bahwa kemampuan diskriminatif model ini terhadap kedua kelas sangat tinggi, hampir mendekati ideal. Hal ini menempatkan SVM sebagai model yang paling unggul secara keseluruhan dalam konteks penelitian ini.

Model Decision Tree juga menunjukkan kinerja yang kuat dengan accuracy sebesar 0.9819, precision sebesar 0.9735, recall sebesar 0.9710, dan F1-score sebesar 0.9723. Dengan performa yang sangat kompetitif terhadap SVM, Decision Tree memiliki keunggulan tambahan berupa interpretabilitas tinggi, yang sangat berguna untuk menjelaskan alur keputusan secara visual. Namun, karena model ini cenderung lebih sensitif terhadap overfitting, performa pada data baru dapat lebih fluktuatif jika tidak ditangani dengan parameter tuning yang tepat. Hal ini sudah diantisipasi dalam penelitian melalui GridSearchCV yang mengoptimalkan parameter seperti kedalaman pohon dan minimal jumlah data dalam setiap pemisahan.

Voting Classifier sebagai pendekatan ensemble menggabungkan tiga model sebelumnya dengan skema voting mayoritas. Hasilnya mencerminkan kombinasi kekuatan dari ketiganya, menghasilkan accuracy sebesar 0.9767, precision sebesar 0.9916, recall sebesar 0.9367, dan F1-score sebesar 0.9634. Meskipun tidak melampaui performa terbaik dari model individu, Voting Classifier tetap menjadi solusi yang stabil dan kompetitif, terutama dalam skenario di mana kestabilan dan generalisasi lebih diutamakan dibanding performa ekstrem pada satu metrik tertentu.

Selain evaluasi terhadap data uji, performa setiap model diuji ulang melalui validasi silang 5-fold untuk mengukur kestabilan terhadap variasi subset data. Support Vector Machine kembali menunjukkan konsistensi tinggi dengan rata-rata akurasi sebesar 0.9896 dan deviasi standar yang kecil. Decision Tree mengikuti di belakang dengan akurasi 0.9831, sementara Naive Bayes mencatatkan akurasi yang jauh lebih rendah yaitu 0.8043. Hasil ini mengonfirmasi bahwa model SVM tidak hanya unggul dalam satu pengujian, tetapi juga konsisten dalam berbagai skenario data.

Dari seluruh hasil yang diperoleh, dapat disimpulkan bahwa Support Vector Machine merupakan model terbaik dalam konteks penelitian ini, dengan keseimbangan yang sangat kuat antara akurasi, presisi, dan recall, serta performa yang stabil dalam validasi silang. Decision Tree menawarkan performa mendekati dengan interpretabilitas yang tinggi, sementara Naive Bayes tetap relevan sebagai baseline yang ringan dan cepat, namun kurang andal dalam deteksi spam yang menyeluruh. Voting Classifier berhasil memadukan kekuatan ketiganya dalam solusi yang cukup seimbang dan praktis untuk implementasi.

Analisis ini akan menjadi dasar dalam menyusun rekomendasi dan simpulan akhir di Bab V, serta membuka ruang diskusi mengenai potensi pengembangan sistem di masa mendatang.

4.4 Diskusi dan Pembahasan

Diskusi dan pembahasan dalam penelitian ini berfokus pada refleksi terhadap hasil eksperimen, relevansi pendekatan yang digunakan, serta pertimbangan terhadap kelebihan dan keterbatasan dari sistem klasifikasi spam yang telah dikembangkan. Hasil dari evaluasi model menunjukkan bahwa algoritma machine learning, khususnya Support Vector Machine dan Decision Tree, memiliki potensi tinggi dalam membedakan email spam dan non-spam secara otomatis, asalkan didukung oleh preprocessing yang baik, representasi fitur yang informatif, dan konfigurasi parameter yang tepat.

Salah satu kekuatan utama dari sistem yang dibangun terletak pada integrasi antara informasi linguistik dan statistik. Dengan menggabungkan representasi teks berbasis TF-IDF dan fitur numerik yang menangkap struktur kasar email (seperti panjang teks, jumlah huruf kapital, serta jumlah URL), model mendapatkan konteks yang lebih lengkap. Hasilnya, performa klasifikasi tidak hanya bergantung pada kata-kata tertentu, tetapi juga pada pola-pola umum yang sering ditemukan pada spam, seperti pengulangan karakter mencolok atau penyisipan tautan promosi. Pendekatan ini terbukti meningkatkan stabilitas dan akurasi model, terutama ketika digunakan dalam model non-linear seperti Decision Tree dan ensemble classifier.

Dari sisi algoritma, Support Vector Machine (SVM) terbukti sebagai model paling unggul dalam penelitian ini. Keunggulan SVM terletak pada kemampuannya menangani data berdimensi tinggi dengan margin pemisah yang optimal. Selain mencatat skor akurasi dan F1-score tertinggi dalam data uji, SVM juga menunjukkan kestabilan performa dalam validasi silang dan kurva ROC yang mendekati ideal. Hal ini menunjukkan bahwa model tidak hanya akurat secara statistik, tetapi juga memiliki generalisasi yang kuat terhadap data baru.

Model Decision Tree menempati posisi menarik dalam diskusi ini. Meskipun secara statistik sedikit di bawah SVM, model ini memiliki keunggulan dalam hal transparansi. Setiap keputusan yang diambil model dapat dilacak kembali ke fitur tertentu, membuatnya lebih mudah dijelaskan kepada pengguna non-teknis. Dalam konteks implementasi di sistem keamanan dunia nyata, transparansi ini dapat menjadi faktor penting, terutama ketika sistem deteksi spam digunakan untuk mengambil keputusan kritis yang memengaruhi komunikasi pengguna.

Di sisi lain, performa Multinomial Naive Bayes yang relatif lebih rendah menunjukkan bahwa asumsi sederhana yang digunakan oleh model probabilistik tidak cukup untuk menangkap kompleksitas data spam yang lebih modern. Meskipun model ini sangat efisien dan ringan secara komputasi, ia cenderung memberikan recall yang rendah, yang berarti banyak spam lolos dari deteksi. Hal ini menjadi catatan penting, karena dalam sistem keamanan informasi, kegagalan mendeteksi ancaman bisa jauh lebih merugikan daripada kesalahan positif sesekali.

Pendekatan ensemble melalui Voting Classifier menunjukkan bahwa kombinasi model dapat memberikan alternatif yang seimbang. Meskipun performanya tidak melampaui SVM secara individu, ensemble tetap memberikan nilai tambah dari sisi stabilitas. Dalam skenario operasional yang dinamis, ensemble dapat meredam kelemahan model individual dan menjadi pilihan yang aman dalam implementasi nyata, terutama ketika model-model dasar memiliki kekuatan yang saling melengkapi.

Analisis visual yang dilakukan melalui confusion matrix, ROC curve, dan grafik fitur penting turut memperkaya proses evaluasi dan menjadikan hasil eksperimen lebih mudah dicerna. Visualisasi fitur paling berkontribusi dari model SVM dan Decision Tree membuka pemahaman terhadap mekanisme klasifikasi yang terjadi di balik layar, dan menjadi pintu masuk untuk pengembangan lebih lanjut, misalnya dalam membangun sistem penjelas berbasis XAI (Explainable AI).

Secara keseluruhan, diskusi ini menegaskan bahwa kombinasi antara preprocessing teks yang komprehensif, pemodelan fitur numerik yang cermat, dan pemilihan algoritma yang sesuai dapat menghasilkan sistem deteksi spam yang tidak hanya akurat tetapi juga andal dan dapat dijelaskan. Penelitian ini memberikan bukti empiris sekaligus landasan teknis bahwa pendekatan machine learning tetap relevan dan efektif dalam menghadapi ancaman email spam yang terus berkembang, serta membuka ruang eksplorasi ke arah pemodelan yang lebih canggih seperti deep learning, transfer learning, atau adaptasi terhadap data multibahasa dan tak terstruktur.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan dari Penelitian

Penelitian ini berhasil merancang dan mengimplementasikan **sistem klasifikasi email berbasis machine learning** yang mampu membedakan antara email spam dan non-spam dengan tingkat akurasi yang sangat tinggi. Pipeline sistem dikembangkan secara komprehensif, mulai dari pemrosesan teks, ekstraksi fitur numerik, pelatihan model, hingga evaluasi berbasis metrik klasifikasi dan visualisasi interpretatif. Beberapa kesimpulan utama yang dapat diambil dari hasil penelitian ini antara lain:

- **Sistem klasifikasi dibangun dengan kombinasi representasi teks berbasis TF-IDF dan fitur numerik** seperti jumlah URL, panjang teks, dan jumlah huruf kapital. Kombinasi ini terbukti memperkaya konteks representasi dan meningkatkan kapabilitas model dalam membedakan spam secara signifikan.

- **Model Support Vector Machine (LinearSVC)** memberikan hasil performa terbaik dengan nilai **accuracy 97.93%, precision 99.17%, recall 94.46%, F1-score 96.76%, dan ROC-AUC 0.9977**. Model ini menunjukkan keseimbangan yang kuat antara presisi tinggi dan sensitivitas yang stabil.

- **Model Decision Tree** juga menunjukkan performa kompetitif dengan **F1-score 97.23%** serta memiliki keunggulan dalam hal **interpretabilitas**, menjadikannya cocok untuk skenario implementasi yang membutuhkan transparansi dan auditabilitas.

- **Multinomial Naive Bayes**, walaupun efisien dan ringan secara komputasi, memiliki **recall yang lebih rendah** dan cenderung melewatkan spam dalam jumlah signifikan. Ini membuatnya kurang ideal untuk skenario deteksi ancaman yang menuntut sensitivitas tinggi.

- **Voting Classifier** sebagai pendekatan ensemble mampu menggabungkan kekuatan dari ketiga model dasar dan menghasilkan performa yang **stabil dan kompetitif**, menjadi pilihan kompromi yang menarik antara akurasi dan fleksibilitas.

- **Visualisasi fitur penting menggunakan koefisien SVM dan feature importance dari Decision Tree** menunjukkan bahwa kombinasi **sinyal linguistik (kata kunci) dan sinyal struktural (pola teks)** menjadi fondasi utama dalam proses klasifikasi.

Secara keseluruhan, penelitian ini menyimpulkan bahwa sistem deteksi spam berbasis machine learning dapat dibangun secara **efektif, efisien, dan transparan**, asalkan didukung oleh proses preprocessing yang matang, pemilihan fitur yang tepat, dan strategi evaluasi yang menyeluruh.

5.2 Keterbatasan

Walaupun sistem klasifikasi email berbasis machine learning yang dikembangkan dalam penelitian ini menunjukkan hasil performa yang sangat baik, tetap terdapat sejumlah keterbatasan konseptual dan praktis yang perlu diakui. Keterbatasan ini mencerminkan ruang yang masih bisa dioptimalkan untuk menjadikan sistem lebih adaptif, tangguh, dan siap diterapkan dalam lingkungan produksi nyata.

- **Keterbatasan Representasi Data dan Variasi Spam Nyata**
  Dataset yang digunakan berasal dari SpamAssassin yang bersifat teks utuh dan berfokus pada konten isi email. Meskipun representatif untuk jenis spam klasik, dataset ini belum mencakup keragaman ancaman spam modern seperti pesan berbasis gambar, HTML dinamis, atau file terlampir. Selain itu, tidak terdapat metadata penting seperti alamat pengirim, subjek, atau waktu pengiriman yang dalam praktiknya sering menjadi indikator kuat dalam filter spam di dunia nyata.

- **Ketergantungan pada Supervised Learning dan Label Data**
  Semua model yang dilatih dalam penelitian ini berbasis supervised learning, yang sepenuhnya bergantung pada data berlabel. Dalam aplikasi riil, mayoritas email masuk bersifat tidak berlabel, dan sistem ini belum memiliki kemampuan untuk menangani data semacam itu melalui pendekatan semi-supervised atau unsupervised learning. Ketergantungan ini membatasi kemampuan sistem untuk mendeteksi pola baru yang belum dikenali model.

- **Keterbatasan terhadap *Concept Drift* dan Adaptivitas Waktu Nyata**
  Pola komunikasi spam sangat dinamis dan sering berubah untuk menghindari deteksi. Sistem ini masih bersifat statis dan tidak dilengkapi dengan mekanisme pembelajaran adaptif seperti online learning atau continual training, sehingga akurasinya berpotensi menurun seiring berjalannya waktu jika tidak dilakukan retraining secara manual dan berkala.

- **Minimnya Ketahanan terhadap Serangan Adversarial**
  Sistem belum diuji terhadap skenario ancaman yang disengaja, seperti penggunaan karakter mirip (obfuscation), penyisipan token tak bermakna, atau penyusunan ulang kalimat secara strategis untuk menghindari deteksi otomatis. Kemampuan sistem

untuk bertahan dari serangan seperti ini belum bisa dievaluasi secara menyeluruh dalam eksperimen ini.

- **Belum Diuji dalam Lingkungan Operasional Nyata**
  Semua eksperimen dilakukan dalam lingkungan tertutup (Jupyter Notebook), tanpa integrasi langsung ke sistem email berbasis server, client, atau cloud. Belum dilakukan pengujian terhadap kecepatan inferensi, konsumsi memori, serta kestabilan model ketika menerima aliran email secara real time. Ini menjadi tantangan tambahan dalam membawa sistem ke tahap deployment produksi.

- **Keterbatasan Penjelasan Keputusan untuk Model Black-Box**
  Meskipun Decision Tree menyediakan jalur keputusan yang dapat ditelusuri, model Support Vector Machine dan ensemble voting yang digunakan tetap berperilaku seperti black-box. Dalam konteks penerapan sistem keamanan informasi yang menuntut transparansi, keterbatasan ini dapat menghambat adopsi sistem oleh pengguna atau organisasi yang membutuhkan auditabilitas tinggi.

Dengan memahami keterbatasan-keterbatasan tersebut secara jujur dan kritis, penelitian ini tidak hanya menyajikan pencapaian teknis, tetapi juga memberikan pijakan penting bagi pengembangan lebih lanjut. Kesadaran terhadap keterbatasan menjadi dasar kuat untuk menyusun solusi yang lebih matang, adaptif, dan sesuai dengan kebutuhan dunia nyata yang kompleks dan terus berkembang.

5.3 Saran Pengembangan Selanjutnya

Berdasarkan temuan, evaluasi, dan keterbatasan yang telah diidentifikasi dalam penelitian ini, terdapat beberapa arah pengembangan yang dapat dilakukan untuk meningkatkan efektivitas, fleksibilitas, dan skalabilitas sistem deteksi spam di masa mendatang:

- **Integrasi Fitur Kontekstual dan Metadata**
  Selain konten utama dari email, metadata seperti nama pengirim, domain pengirim, subjek, waktu pengiriman, dan pola header MIME dapat menjadi indikator penting untuk mendeteksi spam. Integrasi fitur-fitur ini ke dalam proses klasifikasi berpotensi memperkaya representasi dan meningkatkan akurasi model terhadap pola spam yang lebih kompleks.

- **Penerapan Teknik Pembelajaran Semi-Supervised atau Unsupervised**
  Dalam dunia nyata, email tidak selalu tersedia dalam bentuk label yang eksplisit. Mengembangkan model dengan pendekatan semi-supervised atau unsupervised seperti clustering atau autoencoder dapat membantu sistem beradaptasi terhadap data tidak berlabel, sekaligus memperluas kemampuan deteksi terhadap spam yang belum pernah dilatih sebelumnya.

- **Adaptasi terhadap Konsep Drift dan Pembelajaran Dinamis**
  Pola spam berubah seiring waktu karena pengirim spam terus mengembangkan cara baru untuk menyusup ke kotak masuk pengguna. Pengembangan sistem yang mendukung *online learning* atau *incremental training* dapat membantu model tetap relevan tanpa harus melakukan retraining dari awal secara penuh.

- **Uji Ketahanan terhadap Serangan Adversarial**
  Untuk memastikan keandalan sistem di lingkungan yang lebih berisiko, perlu dilakukan simulasi terhadap serangan *adversarial*, yaitu modifikasi konten secara halus namun disengaja yang bertujuan mengecoh sistem deteksi. Pengujian ini penting untuk mengetahui sejauh mana model tahan terhadap manipulasi teks atau penyamaran konten spam.

- **Implementasi pada Sistem Produksi dan Evaluasi Beban Nyata**
  Tahap berikutnya adalah membawa sistem ini ke dalam skenario implementasi nyata, baik dalam bentuk integrasi dengan server email berbasis cloud maupun sistem desktop pribadi. Pengujian performa dalam kondisi lalu lintas email tinggi, delay pemrosesan, serta efisiensi sumber daya komputasi perlu menjadi bagian dari evaluasi lanjutan.

- **Eksplorasi Model Lanjutan Berbasis Deep Learning**
  Model yang digunakan dalam penelitian ini adalah model klasik berbasis machine learning. Di masa mendatang, eksplorasi model berbasis deep learning seperti LSTM, CNN, atau transformer (misalnya BERT) untuk klasifikasi teks dapat membuka peluang performa yang lebih tinggi, terutama jika dataset yang digunakan semakin besar dan bervariasi.

- **Pengembangan Sistem Berbasis Explainable AI (XAI)**
  Untuk meningkatkan kepercayaan pengguna, sistem klasifikasi dapat dikembangkan dengan pendekatan explainable AI yang mampu memberikan justifikasi dan penjelasan atas setiap keputusan klasifikasi. Hal ini sangat relevan untuk organisasi atau pengguna yang mengutamakan auditabilitas dan transparansi sistem.

Dengan mengeksplorasi saran-saran di atas, diharapkan sistem klasifikasi spam berbasis AI yang telah dikembangkan dalam penelitian ini tidak hanya berhenti sebagai prototipe, tetapi juga berkembang menjadi sistem yang adaptif, aman, dan siap dioperasikan dalam ekosistem nyata yang dinamis dan penuh tantangan.

DAFTAR PUSTAKA

Keat, L. C., & Ying, T. X. (2024, December 3). *Artificial intelligence-based email spam filtering*. Zenodo. https://zenodo.org/doi/10.5281/zenodo.14264139

Muhaimin, A., Taufik, I. A., & Daniswara, D. D. (2023). Pendeteksian Spam pada E-mail menggunakan Pendekatan Natural Language Processing. *PROSIDING SEMINAR NASIONAL SAINS DATA*, *3*(1), 116–121. https://doi.org/10.33005/senada.v3i1.90

Mukhtar, H., Al Amien, J., & Rucyat, M. A. (2022). Filtering Spam Email menggunakan Algoritma Naïve Bayes. *Jurnal CoSciTech (Computer Science and Information Technology)*, *3*(1), 9–19. https://doi.org/10.37859/coscitech.v3i1.3652

Rahman, A., & Maslan, A. (2025). Analisis Klasifikasi Email Spam Menggunakan Algoritma Naive Bayes. *Jurnal Comasie*, *12*(3). https://doi.org/10.33884/comasiejournal.v12i3.9792

Salim, A. N., Adryani, A., & Sutabri, T. (2024). Deteksi Email Spam dan Non-Spam Berdasarkan Isi Konten Menggunakan Metode K-Nearest Neighbor dan Support Vector Machine. *Syntax Idea*, *6*(2), 991–1001. https://doi.org/10.46799/syntax-idea.v6i2.3052

Sheneamer, A. (2021). Comparison of deep and traditional learning methods for email spam filtering. *International Journal of Advanced Computer Science and Applications*, *12*(1). https://doi.org/10.14569/ijacsa.2021.0120164

Shivaji Ubale, K., & Ashutosh Shirsath, K. (2025). Evaluation of Classification Algorithms for Effective Spam Email Detection using Spam Email Dataset. In *Transformative Applied Research in Computing, Engineering, Science and Technology* (pp. 118–125). CRC Press. https://doi.org/10.1201/9781003616368-15