ISSN: 2503-054X
Vol. 9 No: 1, April 2024

Perbandingan Algoritma Naive Bayes di dalam Scikit-Learn Python Library dengan Murni Algoritma Naive Bayes: Studi Kasus Klasifikasi Email Berbahaya

Cevi Herdian
Universitas Bunda MuliaJakarta Utara, Indonesia
cherdian@bundamulia.ac.id

Danny Revaldo
Universitas Bunda MuliaJakarta Utara, Indonesia
s36220019@student.ubm.ac.id

Monica Quinn
Universitas Bunda MuliaJakarta Utara, Indonesia
s36220018@student.ubm.ac.id

Silvia Margareta
Universitas Bunda MuliaJakarta Utara, Indonesia
s36220001@student.ubm.ac.id

***Abstract*— Saat ini, penggunaan email yang masif telah meluas dan memiliki dampak baik dan buruk. Salah satu dampak negatif adalah munculnya email spam, yang berisi promosi produk, konten pornografi, virus, dan konten tidak penting yang dikirim ke banyak orang tanpa permintaan. Hal tersebut bisa terjadi dikarenakan kebocoran data, penjualan data ilegal, dan pendaftaran kita sendiri ke berbagai grup dan milis-milis tertentu yang disalahgunakan. Untuk mengatasi masalah ini, diperlukan metodologi klasifikasi email yang dapat secara otomatis mendeteksi apakah sebuah email merupakan spam email berbahaya atau bukan. Penelitian ini bertujuan untuk mengembangkan sebuah model klasifikasi email spam dan non-spam menggunakan Algoritma Naïve Bayes dengan menggunakan dua pendekatan, yaitu penggunaan library scikit-learn dan juga matematika murni Algoritma Naïve Bayes. Penggunaan matematika murni suatu algoritma sangat jarang digunakan di dalam perbandingan antar algoritma. Biasanya memperbandingkan beberapa algoritma. Oleh karena itu, peneliti mencoba melakukan perbandingan tersebut di dalam penelitian ini. Hasil penelitian menunjukkan bahwa penggunaan Algoritma Naïve Bayes dengan library scikit-learn mampu melakukan klasifikasi dengan sangat baik, mencapai tingkat akurasi sebesar 97%, sedangkan penerapan metode matematika Naïve Bayes yang menghasilkan tingkat akurasi yang lebih besar yaitu 98%.**

Kata kunci — Algoritma Naïve Bayes, Dampak email spam, Deteksi spam, Keamanan email, Kebocoran data

I. PENDAHULUAN

Dalam kegiatan sehari-hari, banyak orang yang mengandalkan internet dan hal tersebut telah menjadi hal yang sangat umum di zaman digital yang semakin berkembang[1]. Akan tetapi, pesan spam (pesan yang tidak diinginkan) telah menjadi masalah yang sering dihadapi oleh pengguna internet[2]. Beberapa hal berbahaya yang bisa terjadi karena banyaknya email spam di dalam email adalah sebagai berikut:

- Konten Berbahaya: Email yang berisi spam seringkali mengandung konten berbahaya seperti virus, perangkat lunak berbahaya, atau upaya phishing. Elemen-elemen ini dapat menjadi ancaman serius terhadap keamanan komputer atau jaringan Anda. Perangkat lunak berbahaya dapat menginfeksi sistem Anda, mengambil data Anda, dan potensial memberikan akses tanpa izin kepada penjahat siber.
- Penipuan Keuangan: Penipuan Keuangan dalam email spam mencakup kemenangan lotere palsu, skema investasi, dan penawaran penipuan lainnya. Jika seseorang menjadi korban penipuan semacam itu, mereka dapat kehilangan uang atau menjadi korban pencurian identitas.
- Reputasi dan Kepercayaan: Jika alamat email Anda terhubung dengan banyak email spam, reputasi Anda dapat tercemar. Akibatnya alamat email Anda masuk dalam daftar hitam atau dianggap tidak dapat dipercaya oleh penyedia layanan email dan filter spam seperti yang terdapat di dalam fasilitas email google. Dampaknya adalah email sah yang Anda kirimkan dapat tersaring atau diblokir, menghambat kemampuan Anda untuk berkomunikasi secara efektif.
- Privasi: Email spam sering mengumpulkan informasi pribadi melalui taktik penipuan atau dengan mengarahkan penerima ke situs web palsu. Privasi dapat terancam ketika individu secara tidak sadar memberikan data sensitif kepada entitas jahat. Selain itu, email spam mungkin merupakan bagian dari pelanggaran data yang lebih besar, di mana informasi pribadi terungkap, menyebabkan risiko pelanggaran privasi dan pencurian identitas.

Selain dapat membahayakan keamanan pengguna internet, pesan spam juga dapat mengganggu dan memboroskan waktu pengguna internet[3]. Untuk mengatasi pesan spam, dibutuhkan metodologi yang dapat memfilter pesan spam secara otomatis yaitu salah satunya dengan menggunakan algoritma Naïve Bayes[4].Algoritma Naive Bayes adalah algoritma klasifikasi yang digunakan untuk memprediksi label atau kategori pada suatu data[5]. Metode ini menggunakan Teorema Bayes yang menghitung probabilitas suatu hipotesis atau kelas dengan memperhitungkan probabilitas dari fitur-fitur yang terkait dengan hipotesis

ISSN: 2503-054X
Vol. 9 No: 1, April 2024

tersebut[6]. Dalam konteks penyaringan email spam, untuk menghitung probabilitas bahwa email adalah spam atau bukan, algoritma Naive Bayes menggunakan informasi sebelumnya tentang email yang berupa kata-kata yang sering muncul dalam email spam[7]. Algoritma ini juga sangat efektif dalam hal klasifikasi email karena karena mampu memperhitungkan interaksi antara faktor-faktor yang terkait dengan email spam[8].

Tujuan dari penelitian ini adalah menggunakan Algoritma Naive Bayes untuk membuat metodologi filter spam yang akurat[9]. Penggunaan Algoritma Naive Bayes tersebut peneliti menggunakan dua pendekatan, yaitu pendekatan menggunakan library python Scikit-Learn dan juga pendekatan matematika murni algoritma Naive Bayes. Dataset yang dipergunakan adalah data email yang terdiri dari email yang telah dikelompokkan sebagai spam dan bukan spam yang akan digunakan untuk menguji keefektifan dari klasifikasi Algoritma Naive Bayes tersebut[10]. Dataset tersebut akan digunakan untuk melatih Algoritma Naive Bayes agar dapat mengenali pola dalam email dan membangun model klasifikasi yang memiliki tingkat akurasi yang tinggi[11]. Metodologi untuk meningkatkan kualitas filter spam menjadi lebih tepat dan efektif adalah tujuan dari penelitian ini, hal tersebut dilakukan dengan membandingkan realisasi bukan antara sesama algoritma melainkan penggunaan metode matematika murni dan penggunaan di dalam sebuah library atau package dalam hal ini adalah library scikit-learn[12]. Peneliti memperoleh wawasan mengenai pengembangan sistem filter spam menggunakan algoritma Naïve Bayes melalui beberapa jurnal sebelumnya yang membahas Algoritma Naive Bayes yang telah dipublikasikan di dalam beberapa jurnal [13], [14], [15], [16], [17], [18], [19], [20], [21].

II. METODE PENELITIAN

Berikut ini merupakan rangkaian poin penting metodologi yang dipergunakan di dalam penelitian ini, yaitu:

1. Library

   Penelitian ini menggunakan library pandas untuk analisis data, manipulasi data, dan pembersihan data[22].Menggunakan library scikit-learn untuk implementasi algoritma Naive Bayes.

2. Dataset

   Dataset adalah kumpulan dari data-data yang dikumpulkan,dataset yang digunakan untuk penelitianDataset terdiri dari kumpulan email yang telah diklasifikasikan sebagai spam dan tidak spam. Menggunakan dataset yang tersedia dari Tiago A. Almeida (talmeida@ufscar.br) Department of Computer Science, Federal University of Sao Carlos (UFSCar), Sorocaba, Sao Paulo - Brazil,JosÃ© MarÃa GÃ³mez Hidalgo (jmgomezh yahoo.es), R&D Department Optenet, Las Rozas, Madrid - Spain[23].

   Untuk dataset bisa didownload dari link berikut https://archive.ics.uci.edu/ml/datasets/sms+spam+collection

3. Code Editor Menggunakan Google Collabs sebagai code editor untuk menjalankan skrip Python.
4. Bahasa Pemrograman Menggunakan bahasa pemrograman Python untuk mengimplementasikan algoritma Naive Bayes dan langkah-langkah lainnya dalam membangun filter spam[23].
5. Algoritma (Naive Bayes)
   - Menggunakan *Algoritma Naive Bayes* sebagai metode klasifikasi untuk memfilter pesan *spam* dan tidak *spam*[24].
   - Memanfaatkan asumsi *Naive Bayes* yang menganggap setiap fitur dalam data input tidak saling terkait satu sama lain secara statistik[25].
6. Data Preprocessing
   - Melakukan tahap pembersihan data terlebih dahulu.
   - Mengubah semua huruf kapital menjadi huruf kecil agar konsistensi dalam analisis.
   - Melakukan penghapusan karakter khusus dan tanda baca yang tidak relevan.
   - Melakukan stemming atau lematisasi untuk mengubah kata-kata menjadi bentuk dasar.
7. Menghitung Konstanta Spam dan Tidak Spam
   - Menghitung jumlah total email yang diklasifikasikan sebagai spam dan tidak spam[26].
   - Menghitung probabilitas spam (P(spam)) dan probabilitas tidak spam (P(not spam)) berdasarkan jumlah email yang diklasifikasikan sebagai spam dan tidak spam[27].
8. Menghitung Parameter (XXXX) Setelah menghitung probabilitas spam dan tidak spam pada tahap sebelumnya, langkah selanjutnya adalah menghitung parameter yang akan digunakan dalam algoritma Naive Bayes[28]. Parameter ini digunakan untuk mengklasifikasikan email baru sebagai spam atau tidak spam berdasarkan probabilitas fitur-fitur yang ada dalam email tersebut[29].
9. Tahap Percobaan Jika semua sudah dibuat tahap selanjutnya adalah tahap percobaan maka akan dites model untuk melihat seberapa besar akurasi yang didapat untuk memfilter pesan spam dan tidak spam jika semakin besar akurasinya maka mode yang sudah dibuat sudah berhasil dan siap untuk dipakai
10. Memakai 2 pendekatan Metode pendekatan yang dipakai dalam jurnal ini ada 2 yaitu penggunaan library scikit-learn untuk menggunakan naive bayes dan penggunaan metode matematika naive bayes[30].
    - Penggunaan Library scikit-learn:
      - Pendekatan pertama menggunakan library scikit-learn, yang merupakan library populer dalam pemrosesan data dan pembelajaran mesin di Python[31].
      - Library ini menyediakan implementasi yang efisien dan mudah digunakan dari berbagai algoritma pembelajaran mesin, termasuk Naive Bayes[32].

ISSN: 2503-054X
Vol. 9 No: 1, April 2024

- Dengan menggunakan scikit-learn, dapat dengan mudah membangun model Naive Bayes, melatihnya dengan data training, dan menguji performanya pada data testing[33].

- Penggunaan Metode Matematika Naive Bayes:
  - Pendekatan kedua adalah menggunakan metode matematika Naive Bayes untuk mengimplementasikan algoritma secara langsung[34].
  - Metode ini melibatkan perhitungan probabilitas dan teorema Bayes untuk mengklasifikasikan email sebagai spam atau tidak spam[35].
  - Dalam pendekatan ini, kita akan menghitung probabilitas spam dan tidak spam berdasarkan data training, serta menghitung parameter seperti probabilitas kemunculan kata dalam email spam dan tidak spam[36].
  - Dengan menggunakan rumus-rumus matematika Naive Bayes, kita dapat mengklasifikasikan email baru sebagai spam atau tidak spam berdasarkan perhitungan probabilitas tersebut[37].

Tahap Pembuatan

Melakukan *Install* dan *import library* yang dibutuhkan

- Pertama *install* beberapa *library* yang diperlukan, yaitu *Pandas* dan *Scikit-learn*
- Lalu *import library* yang sudah ditentukan yaitu *Pandas* dan *Scikit-learn.*

Setelah *Import Library* yang diperlukan selanjutnya *Import Dataset*

- Lalu import dataset yang bernama “SMSSpamCollection“ lalu tampilkan data yang ada dengan kode

```
url = 'https://raw.githubusercontent.com/itsmecevi/dataset/main/SMSSpamCollection'
pesan_spam = pd.read_csv(url, delimiter="\t", header=None, names=['Label', 'Text'])

print(pesan_spam.shape)
pesan_spam.head()
```

Gambar 1 Membaca Dataset “SMSSpamCollection”

- Lalu hitung berapa jumlah prosentase data yang berlabel ham atau spam pada kolom label

```
pesan_spam['Label'].value_counts(normalize=True)
```

Gambar 2 Perhitungan kepada data yang berlabel Ham atau Spam

Tahap pertama setelah *Import* adalah *Split Data Training* dan *Data Testing*

Lalu kita acak dataset dengan parameter frac=1 berarti semua dataset diacak dan memastikan bahwa hasil acak yang sama akan dihasilkan setiap kali kode dijalankan. Setelah itu kita hitung jumlah indeks yang berada di dataset untuk dibagi menjadi data pelatihan (data training) dan data untuk pengetesan dimana 80% untuk data training dan 20% untuk data testing. Lalu kita latih data yang sudah kita pisahkan dan diacak tadi lalu diulang kembali dimana nilai indeks yang sudah dipakai tidak dimasukan kembali Lalu kita tes data yang sudah dilatih dan diulang terus menerus dimana nilai indeksnya yang sudah diuji tidak dipakai lagi. Lalu kita tampilkan bentuk (shape) dari subset pelatihan dan pengujian, yaitu jumlah baris dan kolomnya. Hal ini memberikan informasi tentang berapa banyak baris yang ada dalam subset pelatihan dan pengujian.

Tahap Selanjutnya melakukan *Data Preparation*

Tampilkan data sebelum ditransformasi terlebih dahulu. Data transformasi dalam hal ini adalah membuat text menjadi lowercase (huruf kecil semua).

	Label	Text
0	ham	Yep, by the pretty sculpture
1	ham	Yes, princess. Are you going to make me moan?
2	ham	Welp apparently he retired
3	ham	Havent.
4	ham	I forgot 2 ask ü all smth.. There's a card on ...

Gambar 3 Dataframe dari Dataset “SMSSpamCollection”

ISSN: 2503-054X
Vol. 9 No: 1, April 2024

Tampilkan data sesudah diTransformasi

	Label	Text
0	ham	yep by the pretty sculpture
1	ham	yes princess are you going to make me moan
2	ham	welp apparently he retired
3	ham	havent
4	ham	i forgot 2 ask ü all smth there s a card on ...

Gambar 4 Dataframe yang sudah di Transformasi

Peneliti melakukan *Vectorizer*: Konversi teks menjadi matriks numerik

Lalu konversikan teks menjadi representasi vektor berdasarkan frekuensi kata dengan metode "*Vectorizer*". *CountVectorizer* adalah metode ekstraksi fitur teks yang disediakan oleh perpustakaan scikit-learn dalam bahasa pemrograman *Python*. Metode ini digunakan untuk mengkonversi dokumen teks menjadi representasi matriks numerik. Terlebih dahulu kita konversikan data training ke dalam bentuk matriks numerik dengan bantuan modul *CountVectorizer* di dalam *library Scikit-learn Python*.

Cara kerja *CountVectorizer* adalah membuat sebuah teks menjadi *Vector Matrix Numeric*. Berikut dijelaskan secara singkat bagaimana cara kerjanya dengan bantuan *Scikit-learn* dari *Python library*

```
Document-Term Matrix:
[[1 0 0 0 0 0 1 0 1 0]
 [0 0 1 1 0 1 0 1 1 0]
 [1 1 0 0 1 0 0 0 1 1]]

Feature Names:
dict_keys(['love', 'eating', 'pizza', 'is', 'my', 'favorite', 'food', 'enjoy', 'with', 'friends'])
```

Gambar 5 Hasil dari CountVectorizer terhadap DatasetMelakukan *Modeling*

Lalu latih model Naive Bayes menggunakan data pelatihan (training dataset) yang telah diubah menjadi vektor matriks numerik.

```
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(X_train, y_train)
```

Gambar 6 Melatih Data Training dengan Library Scikit-learn

Selanjutnya dilakukan konversi dari text menjadi vektor matriks numerik untuk dataset testing.

```
0        ham
1        ham
2       spam
3        ham
4        ham
       ...
1109     ham
1110     ham
1111     ham
1112    spam
1113     ham
Name: Label, Length: 1114, dtype: object
```

Gambar 7 Hasil Konversi pada Label Data Testing

Lalu lakukan prediksi data pada data tes (dataset testing)

ISSN: 2503-054X
Vol. 9 No: 1, April 2024

Index	Predicted	Original
0	ham	ham
1	ham	ham
2	spam	spam
3	ham	ham
4	ham	ham
5	ham	ham
6	ham	ham
7	ham	ham
8	ham	ham
9	spam	ham

Gambar 8 Hasil Prediksi terhadap Data Testing

Lalu coba hitung akurasi yang menggunakan library scikit-learn. Perhitungan akurasi merujuk pada ukuran seberapa sering model klasifikasi memprediksi dengan benar kelas atau label yang sebenarnya dari suatu contoh tertentu. Ini adalah metrik evaluasi umum yang digunakan untuk menilai kinerja model klasifikasi. Secara matematis, akurasi dihitung dengan membagi jumlah contoh yang terklasifikasi dengan benar dengan jumlah total contoh dalam dataset. Biasanya dinyatakan dalam bentuk persentase.

Akurasi = (Jumlah contoh yang terklasifikasi dengan benar) / (Total jumlah contoh)

Sebagai contoh, jika model klasifikasi memprediksi dengan benar kelas dari 80 dari 100 contoh dalam dataset, akurasi akan menjadi 80%.

```
# Calculate accuracy of predictions
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2%}")
```

```
Accuracy: 97.94%
```

Gambar 9 Hasil Akurasi yang didapat menggunakan Scikit-learn

Model tersebut menjelaskan bahwa dari 100 data klasifikasi spam, yang benar adalah 97 dan yang salah adalah 3 kali (error)

Melakukan pendekatan kedua yaitu menggunakan Matematika Algoritma *Naive Bayes*

Lalu sekarang peneliti akan mencoba menggunakan metode matematika Naive Bayes. Pendekatan matematika Naive Bayes sama dengan pendekatan implementasi di dalam library scikit-learn. Dibawah ini adalah penjelasan dari kedua pendekatan tersebut. Pendekatan Algoritma Naive Bayes di scikit-learn python library:

- Implementasi Naive Bayes di scikit-learn menyediakan algoritma yang sudah dikodekan dan siap digunakan.
- Menggunakan representasi data dalam bentuk matriks atau vektor numerik.
- Menggunakan matriks atau vektor numerik untuk menghitung kemungkinan likelihood dan prior secara efisien.
- Mendukung beberapa variasi Naive Bayes seperti Naive Bayes Gaussian, Naive Bayes Multinomial, dan Naive Bayes Bernoulli.
- Menyediakan fungsi-fitur tambahan seperti smoothing dan penanganan fitur bernilai nol.

Pendekatan Matematika Naive Bayes:

- Pada pendekatan matematika Naive Bayes, algoritma didasarkan pada teori probabilitas dan teorema Bayes.
- Menggunakan asumsi independensi kondisional, yang dikenal sebagai Naive Bayes assumption, yang menyatakan bahwa setiap fitur adalah independen secara kondisional terhadap kelas yang ditentukan.
- Memerlukan perhitungan probabilitas untuk mengestimasi likelihood dan prior yang digunakan untuk membuat prediksi.
- Membutuhkan manipulasi dan perhitungan matematika yang rumit untuk menghitung probabilitas.

Dengan menggunakan library scikit-learn, implementasi Naive Bayes menjadi lebih mudah dan praktis karena telah disediakan berbagai fitur dan fungsionalitas yang siap pakai. Pendekatan matematika Naive Bayes, di sisi lain, lebih berfokus pada teori dan perhitungan probabilitas yang mendasari algoritma.

Setelah membahas Naive Bayes menggunakan library scikit-learn python. Selanjutnya pembahasan detail bagaimana Naive Bayes secara awalnya dibuat dengan menggunakan pendekatan matematika manual yaitu menggunakan teori peluang.Pertama pisahkan setiap kata-kata dalam setiap email menjadi satu persatu lalu dibersihkan dari duplikasi kata-kata yang sama lalu dikonversikan menjadi list. Jadi dengan menjalankan kode tersebut, setiap pesan dalam kolom ‘Text’ subset pelatihan (data training) akan dipecah menjadi kata-kata individu, dan seluruh kata tersebut akan disimpan dalam list kosakata setelah menghilangkan duplikasi. Sebagai hasilnya, kosakata akan berisi semua kata unik yang muncul dalam pesan-pesan pelatihan.

ISSN: 2503-054X
Vol. 9 No: 1, April 2024

```
train_dataset['Text'] = train_dataset['Text'].str.split()

kosakata = []
for pesan in train_dataset['Text']:
    for kata in pesan:
        kosakata.append(kata)

kosakata = list(set(kosakata))
```

Gambar 10 Pembuatan Kamus yang terdiri dari kosakata untuk kata unik

Lalu buat sebuah kamus dari beberapa kata-kata mendapatkan kamus "jumlah_kata_perpesan" yang menyimpan jumlah kemunculan setiap kata dalam setiap pesan dalam subset pelatihan (data training). Kamus ini akan digunakan sebagai fitur untuk melatih model atau melakukan analisis lebih lanjut pada data pelatihan. Setelah itu buat DataFrame baru dengan nama jumlah_kata. DataFrame dibuat berdasarkan kamus jumlah_kata_perpesan. Setiap kunci (key) dalam kamus akan menjadi nama kolom dalam DataFrame, dan nilai-nilai dari setiap kunci akan menjadi nilai-nilai dalam kolom yang sesuai. Dengan kata lain, setiap kolom pada DataFrame akan mewakili kata unik, dan setiap baris akan mewakili pesan dalam subset pelatihan.

	per	sathy	airport	373	meaningful	mutai	08712402902	who	dust	lecture	...	money	beer	lrg	lots	computational	orchard	entitled	talked	drink	mahfuuz
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

Gambar 11 Tabel yang berisikan jumlah kata pada setiap pesan

Tampilkan DataFrame train_dataset_clean yang merupakan gabungan antara DataFrame "train_dataset" dan DataFrame "jumlah_kata". DataFrame ini akan berisi semua kolom dari DataFrame "train_dataset" serta kolom-kolom baru yang berasal dari DataFrame "jumlah_kata".

	Label	Text	per	sathy	airport	373	meaningful	mutai	08712402902	who	...	money	beer	lrg	lots	computational	orchard	entitled	talked	drink	mahfuuz
0	ham	[yep, by, the, pretty, sculpture]	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	ham	[yes, princess, are, you, going, to, make, me,...	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	ham	[welp, apparently, he, retired]	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	ham	[havent]	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	ham	[i, forgot, 2, ask, ü, all, smth, there, s, a,...	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

Gambar 12 Tabel yang berisikan jumlah kata dengan teksnya juga untuk melakukan perbandingan

Buat perhitungan untuk mengidentifikasi yang mana spam dan tidak spam untuk mendapatkan beberapa statistik dan parameter yang digunakan dalam model Naive Bayes, termasuk probabilitas kemunculan spam dan ham (non spam), jumlah kata dalam pesan spam dan ham, jumlah kata unik dalam dataset, serta nilai parameter dalam data pelatihan yang sudah dibersihkan.

```
# P(Spam) and P(Ham)
p_spam = len(pesan_spam) / len(train_dataset_clean)
p_ham = len(pesan_ham) / len(train_dataset_clean)

# N_Spam
n_kata_perpesan_spam = pesan_spam['Text'].apply(len)
n_spam = n_kata_perpesan_spam.sum()

# N_Ham
n_kata_perpesan_ham = pesan_ham['Text'].apply(len)
n_ham = n_kata_perpesan_ham.sum()
```

Gambar 13 Rumus untuk mengidentifikasi pesan "Spam" dan "Ham"

ISSN: 2503-054X
Vol. 9 No: 1, April 2024

Lakukan inisialisasi dan perhitungan yang dimana model akan menginisialisasikan terlebih dahulu parameternya lalu menghitung berapa banyak parameternya yang dihasilkannya.

```
# insialisasi parameters
parameter_spam = {kata_unik:0 for kata_unik in koskata}
parameter_ham = {kata_unik:0 for kata_unik in koskata}
```

Gambar 14 Penginisialisasi parameter terhadap model

Lalu melakukan percobaan kepada model untuk mengklasifikasikan pesan (teks) sebagai spam atau ham menggunakan model Naive Bayes yang telah dilatih sebelumnya. Yang dimana akan melakukan perulangan jika pesan mayoritas dikatakan Ham akan mengeluarkan output pesan berisikan ham jika sebaliknya maka akan mengeluarkan output Spam

```
msg = re.sub('\W', ' ', msg)
    msg = msg.lower().split()

    p_isi_pesan_spam = p_spam
    p_isi_pesan_ham = p_ham

    for kata in msg:
        if kata in parameter_spam:
            p_isi_pesan_spam *= parameter_spam[kata]

        if kata in parameter_ham:
            p_isi_pesan_ham *= parameter_ham[kata]
```

Gambar 15 Melakukan pengklasifikasian terhadap pesan(teks) yang di anggap “Spam” atau “Ham”

Lalu tampilkan fungsi classify untuk mengklasifikasikan pesan "tes 1" dan “tes 2” sebagai spam atau ham menggunakan model Naive Bayes yang telah dilatih sebelumnya.

```
[35] classify('tes 1')

     P(Spam|pesan): 0.0005331321413709429
     P(Ham|pesan): 0.000692115439788068
     Label: Ham

[36] classify("tes 2")

     P(Spam|pesan): 0.0009725267633799617
     P(Ham|pesan): 0.003500506935851191
     Label: Ham
```

Gambar 16 Menampilkan hasil dari Classify sebanyak 2 kali percobaan

Lalu buat sebuah fungsi, yang dimana digunakan untuk mengklasifikasikan pesan “msg” sebagai spam atau ham berdasarkan model Naive Bayes yang telah dilatih sebelumnya.

- msg = re.sub('\W', ' ', msg): Baris ini menggunakan modul re untuk mengganti semua karakter non-alphanumerik dalam msg menjadi spasi. Ini dilakukan dengan menggunakan fungsi re.sub() dengan pola \W yang mencocokkan karakter non-alphanumerik dan menggantinya dengan spasi.
- msg = msg.lower().split(): Baris ini mengubah semua karakter dalam msg menjadi huruf kecil menggunakan metode lower() dan kemudian memisahkan kata-kata dalam pesan menjadi sebuah daftar menggunakan metode split(). Ini akan menghasilkan variabel msg yang berisi daftar kata-kata dalam pesan yang telah diubah menjadi huruf kecil.
- p_isi_pesan_spam = p_spam dan p_isi_pesan_ham = p_ham: Baris ini menginisialisasi variabel p_isi_pesan_spam dan p_isi_pesan_ham dengan probabilitas a priori P(Spam) dan P(Ham) yang telah dihitung sebelumnya. Perulangan for kata in msg: Baris ini melakukan perulangan untuk setiap kata dalam msg. Peluang tersebut dijalankan dengan fungsi If kondisi a dan kondisi b.
    a. if kata in parameter_spam: p_isi_pesan_spam *= parameter_spam[kata]: Baris ini memeriksa apakah kata tersebut ada dalam kamus parameter_spam. Jika kata tersebut ada, maka probabilitas kondisional P(kata|Spam) dikalikan dengan p_isi_pesan_spam.
    b. if kata in parameter_ham: p_isi_pesan_ham *= parameter_ham[kata]: Baris ini memeriksa apakah kata tersebut ada dalam kamus parameter_ham. Jika kata tersebut ada, maka probabilitas kondisional

ISSN: 2503-054X
Vol. 9 No: 1, April 2024

P(kata|Ham) dikalikan dengan p_isi_pesan_ham.

Dalam langkah ini, setiap kata dalam pesan dikalikan dengan probabilitas kondisional yang telah dihitung sebelumnya. Ini dilakukan untuk menghasilkan probabilitas posterior P(Spam|pesan) dan P(Ham|pesan). Pada bagian ini, dilakukan perbandingan antara probabilitas posterior P(Ham|pesan) dan P(Spam|pesan) untuk menentukan label (spam atau ham) dari pesan. Jika p_isi_pesan_ham lebih besar dari p_isi_pesan_spam, maka pesan diklasifikasikan sebagai ham. Jika p_isi_pesan_spam lebih besar dari p_isi_pesan_ham, maka pesan diklasifikasikan sebagai spam. Jika keduanya sama, maka dikembalikan label "Butuh perhitungan ulang" karena tidak bisa menentukan label secara pasti.

```
def classify_tes_dataset(msg):
    '''
    message: a string
    '''

    msg = re.sub('\W', ' ', msg)
    msg = msg.lower().split()

    p_isi_pesan_spam = p_spam
    p_isi_pesan_ham = p_ham

    for kata in msg:
        if kata in parameter_spam:
            p_isi_pesan_spam *= parameter_spam[kata]

        if kata in parameter_ham:
            p_isi_pesan_ham *= parameter_ham[kata]

    if p_isi_pesan_ham > p_isi_pesan_spam:
        return 'ham'
    elif p_isi_pesan_spam > p_isi_pesan_ham:
        return 'spam'
    else:
        return 'Butuh perhitungan ulang'
```

Gambar 17 Perbandingan antara probabilitas posterior P(Ham|pesan) dan P(Spam|pesan)

Fungsi “tes_dataset['predicted'] = tes_dataset['Text'].apply(classify_tes_dataset)” digunakan untuk mengklasifikasikan set data uji (“tes_dataset”) menggunakan fungsi classify_tes_dataset. Hasil klasifikasi akan disimpan dalam kolom 'predicted' dalam “tes_dataset”.

	Label	Text	predicted
0	ham	Later i guess. I needa do mcat study too.	ham
1	ham	But i haf enuff space got like 4 mb...	ham
2	spam	Had your mobile 10 mths? Update to latest Oran...	spam
3	ham	All sounds good. Fingers . Makes it difficult ...	ham
4	ham	All done, all handed in. Don't know if mega sh...	ham

Gambar 18 Hasil dari prediksi model terhadap pesan

Lalu hitung akurasi dari model klasifikasi yang telah dilakukan pada “tes_dataset”. Maka didapat akurasi sebesar 98.7% untuk penggunaan metode matematika naive bayes.

```
Benar: 1100
Salah: 14
Akurasi: 0.9874326750448833
```

Gambar 19 Hasil akurasi jika menggunakan Matematika Murni Naive Bayes

III. Hasil dan Pembahasan

Berikut hasil dari 2 pendekatan yang dilakukan yaitu menggunakan *library Scikit-learn* dan matematika murni *Naive Bayes*

Pendekatan menggunakan *Scikit-Learn*:

- Dalam pendekatan ini, algoritma *Naive Bayes* diimplementasikan menggunakan *library Scikit-learn* dengan penggunaan parameter seperti jenis *Naive Bayes*.
- Percobaan menunjukkan bahwa pendekatan ini berhasil mencapai tingkat akurasi sebesar 97%. Analisis lebih lanjut menunjukkan bahwa penggunaan jenis *Naive Bayes* tertentu memberikan hasil yang lebih baik tergantung pada distribusi data dan kompleksitas kasus.

```
Accuracy: 97.94%
```

ISSN: 2503-054X
Vol. 9 No: 1, April 2024

Gambar 20 Hasil akurasi jika menggunakan Library Scikit-learn Naive Bayes

Pendekatan menggunakan metode matematika murni *Naive Bayes*:

- Parameter: Metode matematika murni *Naive Bayes* diimplementasikan dengan mempertimbangkan langkah-langkah seperti perhitungan probabilitas posterior, dan pemilihan atribut yang relevan. Metode ini memperhitungkan distribusi probabilitas yang mendasari data dengan cermat.
- Akurasi: Hasil penelitian menunjukkan peningkatan yang signifikan dalam akurasi dengan mencapai 98.7%. Pendekatan ini menekankan pentingnya pemilihan atribut yang akurat dan perhitungan probabilitas yang cermat, yang menghasilkan model yang sangat tepat pada data uji.

```
Benar: 1100
Salah: 14
Akurasi: 0.9874326750448833
```

Gambar 21 Hasil akurasi jika menggunakan Matematika Murni Naive Bayes Analisis Faktor-Faktor Pengaruh:

1. Perbandingan Performa: Perbandingan antara dua pendekatan ini mengungkapkan perbedaan performa yang signifikan. Pendekatan menggunakan metode matematika murni Naive Bayes menunjukkan keunggulan dalam memahami hubungan kompleks antar atribut, terutama pada dataset yang memiliki pola yang rumit. Di sisi lain, pendekatan Scikit-learn memberikan hasil yang baik pada data dengan distribusi yang lebih sederhana.
2. Kelebihan dan Kekurangan: Pendekatan Scikit-learn menawarkan kemudahan implementasi dan penggunaan berbagai jenis Naive Bayes yang telah dioptimalkan untuk berbagai kasus. Namun, metode matematika murni membutuhkan pemahaman statistik yang lebih mendalam dan penanganan data yang lebih teliti. Keakuratannya lebih tinggi pada data yang kompleks, tetapi membutuhkan penyesuaian parameter yang cermat.

IV. KESIMPULAN DAN SARAN

Penggunaan matematika peluang Naive Bayes lebih menghasilkan hasil yang lebih akurat jika dibandingkan dengan penggunaan library scikit-learn. Pernyataan tersebut tidak sepenuhnya benar. Akurasi yang dihasilkan oleh suatu model Supervised Learning dalam hal ini adalah klasifikasi termasuk Naive Bayes dipengaruhi oleh beberapa hal. Kemungkinan beberapa faktor yang menjelaskan bahwa akurasi metode manual Naive Bayes dan Penggunaan Naive Bayes dengan scikit-learn bisa berbeda tergantung studi kasusnya dikarenakan beberapa hal sebagai berikut:

1. Implementasi yang berbeda: Implementasi manual Naive Bayes dengan matematika peluang murni cenderung lebih fleksibel dan dapat disesuaikan secara khusus untuk dataset dan masalah yang sedang dihadapi. Di sisi lain, implementasi Naive Bayes di scikit-learn memiliki pendekatan yang lebih umum dan mencakup beberapa variasi Naive Bayes (seperti Gaussian, Multinomial, dan Bernoulli). Pendekatan yang lebih umum ini dapat mempengaruhi akurasi jika asumsi atau karakteristik data tidak sepenuhnya cocok dengan salah satu variasi tersebut.
2. Preprocessing data: Penggunaan scikit-learn membutuhkan preprocessing data yang sesuai dengan format yang diterima oleh library, seperti representasi vektor atau matriks. Jika preprocessing tidak dilakukan dengan benar atau tidak sesuai dengan karakteristik data, ini dapat mempengaruhi performa dan akurasi model. Dalam implementasi manual, Anda memiliki kendali penuh atas preprocessing data dan dapat menyesuaikannya sesuai dengan kebutuhan.
3. Parameter dan tuning: Scikit-learn menyediakan metode untuk tuning parameter model Naive Bayes secara otomatis, seperti dengan menggunakan teknik validasi silang (cross-validation) dan optimasi parameter. Dengan penyetelan parameter yang tepat, performa dan akurasi model Naive Bayes di scikit-learn dapat ditingkatkan. Dalam implementasi manual, Anda perlu secara manual menentukan parameter dan melakukan penyetelan sendiri, yang memerlukan pengetahuan yang mendalam tentang algoritma dan data yang digunakan.

Bisa diambil kesimpulan bahwa Naive Bayes, baik dalam implementasi Scikit-Learn maupun dalam metode matematika, adalah pilihan yang sangat baik untuk filtering spam. Tingkat akurasi yang tinggi dari kedua pendekatan menunjukkan bahwa mereka dapat digunakan secara luas dalam industri email untuk mengurangi jumlah email spam yang diterima oleh pengguna. Penelitian ini juga memberikan dasar untuk pengembangan teknik filtering spam yang lebih canggih dan akurat di masa depan. Oleh karena itu, baik Scikit-Learn dengan algoritma Naive Bayes maupun metode matematika Naive Bayes merupakan alternatif yang kuat untuk meningkatkan keefektifan filtering spam. Dalam banyak kasus, penggunaan Naive Bayes dengan scikit-learn yang menggunakan implementasi yang disediakan oleh library dapat memberikan hasil yang baik dan mudah digunakan, terutama untuk kasus umum. Namun dalam penelitian kali ini, kasus metode manual lebih baik.

DAFTAR PUSTAKA

[1] P. DiMaggio, E. Hargittai, W. R. Neuman, dan J. P. Robinson, "Social implications of the Internet," Annual review of sociology, vol. 27, no. 1, hal. 307-336, 2001.
[2] I. N. Dewi dan C. Supriyanto, "Klasifikasi Teks Pesan Spam Menggunakan Algoritma Naïve Bayes," Semantik, vol. 3, no. 1, 2013.
[3] A. Setiyono dan H. F. Pardede, "Klasifikasi SMS Spam Menggunakan Support Vector Machine," Jurnal Pilar Nusa Mandiri, vol. 15, no. 2, hal. 275-280,

ISSN: 2503-054X
Vol. 9 No: 1, April 2024

2019.
[4] A. Wibisono, "Filtering Spam Email Menggunakan Metode Naive Bayes," Jurnal Teknologi Pintar, vol. 3, no. 4, 2023.
[5] D. Sartika dan D. I. Sensuse, "Perbandingan algoritma klasifikasi Naive Bayes, Nearest Neighbour, dan Decision Tree pada studi kasus pengambilan keputusan pemilihan pola pakaian," JATISI (Jurnal Teknik Informatika dan Sistem Informasi), vol. 3, no. 2, hal. 151-161, 2017.
[6] A. Deolika, K. Kusrini, dan E. T. Luthfi, "Analisis pembobotan kata pada klasifikasi text mining," Jurnal Teknologi Informasi (JurTI), vol. 3, no. 2, hal. 179-184, 2019.
[7] 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.
[8] A. Wibisono, "Filtering Spam Email Menggunakan Metode Naive Bayes," Jurnal Teknologi Pintar, vol. 3, no. 4, 2023.
[9] A. Hidayat, "Klasifikasi Spam Email Menggunakan Metode Naive Bayes," Jurnal Teknologi Pintar, vol. 3, no. 2, 2023.
[10] A. Wibisono, "Filtering Spam Email Menggunakan Metode Naive Bayes," Jurnal Teknologi Pintar, vol. 3, no. 4, 2023.
[11] P. Nagaraj, V. Muneeswaran, G. S. S. Reddy, V. B. Kumar, B. M. Mohan, dan S. Kumar, "Automatic Email Spam Classification Using Naïve Bayes," dalam 2023 International Conference on Computer Communication and Informatics (ICCCI), Januari 2023, hal. 1-5, IEEE.
[12] B. Hemapriya, K. Devi, dan K. Harini, "Automatic Scikit-learn based detection and classification of Breast Cancer using Machine Learning techniques," dalam 2023 Third International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), Januari 2023, hal. 1-8, IEEE.
[13] H. Zhang, N. Cheng, Y. Zhang, dan Z. Li, "Label flipping attacks against Naive Bayes on spam filtering systems," Applied Intelligence, vol. 51, hal. 4503-4514, 2021.
[14] T. Lv, P. Yan, H. Yuan, dan W. He, "Spam filter based on naive Bayesian classifier," dalam Journal of Physics: Conference Series, vol. 1575, no. 1, hal. 012054, IOP Publishing, Juni 2020.
[15] S. J. S. Daisy dan A. R. Begum, "Smart material to build mail spam filtering technique using Naive Bayes and MRF methodologies," Materials Today: Proceedings, vol. 47, hal. 446-452, 2021.
[16] B. Kuchipudi, R. T. Nannapaneni, dan Q. Liao, "Adversarial machine learning for spam filters," dalam Proceedings of the 15th International Conference on Availability, Reliability and Security, Agustus 2020, hal. 1-6.
[17] S. Rapacz, P. Chołda, dan M. Natkaniec, "A method for fast selection of machine-learning classifiers for spam filtering," Electronics, vol. 10, no. 17, hal. 2083, 2021.
[18] I. Wickramasinghe dan H. Kalutarage, "Naive Bayes: applications, variations and vulnerabilities: a review of literature with code snippets for implementation," Soft Computing, vol. 25, no. 3, hal. 2277-2293, 2021.
[19] X. Yang, H. Yu, dan Z. Jia, "Research on spam filtering algorithm based on mutual information and weighted naive Bayesian classification," International Journal of Ad Hoc and Ubiquitous Computing, vol. 37, no. 4, hal. 240-248, 2021.
[20] M. Novo-Loures, D. Ruano-Ordas, R. Pavon, R. Laza, S. Gomez-Meire, dan J. R. Mendez, "Enhancing representation in the context of multiple-channel spam filtering," Information Processing & Management, vol. 59, no. 2, hal. 102812, 2022.
[21] T. Mehrotra, G. K. Rajput, M. Verma, B. Lakhani, dan N. Singh, "Email spam filtering technique from various perspectives using machine learning algorithms," dalam Data Driven Approach Towards Disruptive Technologies: Proceedings of MIDAS 2020, hal. 423-432, Springer Singapore, 2021.
[22] S. Raschka, J. Patterson, dan C. Nolet, "Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence," Information, vol. 11, no. 4, hal. 193, 2020.
[23] A. Z. Farmadiansyah, A. F. Hidayatullah, dan F. Rahma, "Deteksi Email Spam dan Non Spam Bahasa Indonesia Menggunakan Metode Naïve Bayes," AUTOMATA, vol. 2, no. 2, 2021.
[24] A. Wibisono, "Filtering Spam Email Menggunakan Metode Naive Bayes," Jurnal Teknologi Pintar, vol. 3, no. 4, 2023.
[25] S. Chen, G. I. Webb, L. Liu, dan X. Ma, "A novel selective naïve Bayes algorithm," Knowledge-Based Systems, vol. 192, hal. 105361, 2020.
[26] A. Wibisono, "Filtering Spam Email Menggunakan Metode Naive Bayes," Jurnal Teknologi Pintar, vol. 3, no. 4, 2023.
[27] A. Hidayat, "Klasifikasi Spam Email Menggunakan Metode Naive Bayes," Jurnal Teknologi Pintar, vol. 3, no. 2, 2023.
[28] A. Putri, "Kinerja Naïve Bayes Classifier pada Penyaringan Short Message Service (SMS) Spam," 2023.
[29] H. Mukhtar, J. Al Amien, dan M. A. Rucyat, "Filtering Spam Email menggunakan Algoritma Naïve Bayes," Jurnal CoSciTech (Computer Science and Information Technology), vol. 3, no. 1, hal. 9-19, 2022.
[30] M. Anita, B. Susanto, dan L. Larwuy, "Perbandingan Metode Random Forest dan Naïve Bayes dalam Email Spam Filtering," KUBIK: Jurnal Publikasi Ilmiah Matematika, vol. 7, no. 2, hal. 88-96, 2022.
[31] R. P. Cota dan D. Zinca, "Comparative results of spam email detection using machine learning algorithms," dalam 2022 14th International Conference on Communications (COMM), Juni 2022, hal. 1-5, IEEE.
[32] G. Mohanan, D. M. Padmanabhan, dan G. S. Anisha, "Analyzing Random Forest, Naive Bayes, and SVM to Filter Spam Emails Across Multiple Datasets," dalam ICCCE 2021: Proceedings of the 4th International Conference on Communications and Cyber Physical Engineering, hal. 325-332, Springer Nature Singapore, Mei 2022.
[33] G. Mohanan, D. M. Padmanabhan, dan G. S. Anisha, "Classifying Emails into Spam or Ham Using ML Algorithms," dalam Data Science and Security: Proceedings of IDSCS 2021, hal. 214-221, Springer Singapore, 2021.
[34] K. F. Rafat, Q. Xin, A. R. Javed, Z. Jalil, dan R. Z. Ahmad, "Evading obscure communication from spam emails," Math. Biosci. Eng., vol. 19, no. 2, hal. 1926-1943, 2022.
[35] S. Gibson, B. Issac, L. Zhang, dan S. M. Jacob, "Detecting spam email with machine learning optimized with bio-inspired metaheuristic algorithms," IEEE Access, vol. 8, hal. 187914-187932, 2020.
[36] R. S. Lutfiyani dan N. Retnowati, "IMPLEMENTASI PENDETEKSIAN SPAM EMAIL MENGGUNAKAN METODE TEXT MINING DENGAN ALGORITMA NAIVE BAYES DAN DECISION TREE J48," J-Icon: Jurnal Komputer dan Informatika, vol. 9, no. 2, hal. 244-252, 2021.
[37] A. Hosseinalipour dan R. Ghanbarzadeh, "A novel approach for spam detection using horse herd optimization algorithm," Neural Computing and Applications, vol. 34, no. 15, hal. 13091-13105, 2022.