Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 1, No. 1, Januari 2017, hlm. x-x http://j-ptiik.ub.ac.id

Klasifikasi Spam pada *Short Message Service* (SMS) menggunakan Support Vector Machine

Mutiharis Dauber Panjaitan[1], Putra Pandu Adikara[2], Budi Darma Setiawan[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]mutiharisp@gmail.com, [2]adikara.putra@ub.ac.id, [3]s.budidarma@ub.ac.id

Abstrak

Short Message Service (SMS) adalah layanan pesan singkat yang secara luas digunakan dalam berbagai aktivitas sehari-hari, termasuk pemantauan kesehatan, *mobile banking*, dan *mobile commerce*. Namun, SMS juga rentan terhadap penyalahgunaan yang dapat mengandung konten berbahaya. Pesan-pesan SMS spam dapat bercampur dengan pesan-pesan non-spam, sehingga mengganggu pengguna. Oleh karena itu, diperlukan pengelompokan pesan menjadi beberapa kategori untuk memudahkan pengguna. Dalam penelitian ini, kategori yang digunakan adalah normal, penipuan, promo, autentikasi, dan bank. Data yang digunakan berjumlah 1584 pesan, yang dibagi menjadi data latih dan data uji dengan perbandingan 75%:25%. Klasifikasi pesan dilakukan menggunakan metode SVM dengan konsep one-against-all. Penelitian ini melibatkan *preprocessing*, *term weighting*, *training*, dan *testing*. Hasil evaluasi menunjukkan tingkat akurasi sebesar 0,95, *precision* sebesar 0,96, *recall* sebesar 0,95, dan *f1-score* sebesar 0,95. Hasil ini diperoleh dengan menggunakan kombinasi parameter C = 100, *epsilon* = 10^{-5}, konstanta *gamma* = 0,01, *lambda* = 0,1, dan iterasi maksimum = 50.

Kata kunci: *klasifikasi teks, SMS, spam, support vector machine*

Abstract

Short Message Service (SMS) is a short message service that is widely used in various daily activities, including health monitoring, mobile banking, and mobile commerce. However, SMS is also vulnerable to misuse as it can contain malicious content. Spam SMS messages can be mixed with non-spam messages, thereby annoying users. Therefore, it is necessary to group messages into several categories to facilitate users. In this research, the categories used are normal, fraud, promo, authentication, and bank notification. The data used amounted to 1584 messages, which were divided into training data and test data with a ratio of 75%: 25%. Message classification was performed using the SVM method with the one-against-all concept. This research involves preprocessing, term weighting, training, and testing. The evaluation results showed an accuracy rate of 0.95, precision of 0.96, recall of 0.95, and f1-score of 0.95. These results were obtained using a combination of parameters C = 100, epsilon = 10-5, gamma constant = 0.01, lambda = 0.1, and maximum iterations = 50.

Keywords: *text classification, SMS, spam, support vector machine*

1. PENDAHULUAN

Short Message Service (SMS) adalah layanan yang memungkinkan pengiriman pesan teks singkat hingga 160 karakter tanpa memerlukan koneksi internet, menggunakan jalur komunikasi dalam jaringan seluler. (Souza *et al.*, 2011). Layanan ini sering digunakan dalam kehidupan sehari-hari seperti pemantauan kesehatan, *mobile banking*, *mobile commerce*, dan lain-lain (Saxena, Chaudhari dan Member, 2014). Secara umum, SMS dikategorikan menjadi dua jenis, yaitu spam dan non-spam. SMS non-spam merupakan pesan yang berguna dan memberikan informasi yang relevan bagi pengguna. SMS spam adalah pesan yang tidak diinginkan yang dikirim ke ponsel dan berpotensi mengandung konten berbahaya (Dharani, Hegde dan Mohana, 2023).

Berdasarkan data dari Truecaller Insight, Indonesia termasuk ke dalam daftar 20 Negara paling banyak terdampak SMS spam pada tahun 2021. Berdasarkan data tersebut, rata-rata pengguna mendapat 6 sms spam per bulan (Truecaller, 2021). Beberapa faktor yang

menyebabkan peningkatan spam SMS antara lain adalah kurangnya implementasi filter spam SMS yang efektif di tingkat operator telekomunikasi atau pada aplikasi SMS. Hal ini memungkinkan pesan spam dikirim ke ponsel pengguna tanpa penyaringan. Pesan spam ini dapat mengganggu pengguna dengan memenuhi kotak masuk dan menggeser pesan-pesan penting, sehingga menghambat pengguna dalam membaca pesan yang relevan. (Dewi *et al.*, 2017).

Beberapa penelitian yang pernah dilakukan untuk mendeteksi spam pada SMS antara lain: (Popovac *et al.*, 2018), (Jain *et al.*, 2022), dan (Bheemesh dan Deepa, 2020) telah berhasil melakukan penggolongan SMS menjadi spam dan bukan spam. Penggolongan SMS menjadi hanya dua kelas seperti ini dinilai kurang efektif karena SMS juga bisa berisi pengiriman *One-Time Password* (OTP), notifikasi transaksi perbankan dan pesan-pesan promosi (Bhatnagar dan Kumar, 2018). Jika pesan hanya diklasifikasikan ke dalam dua kategori, pesan-pesan seperti kode OTP, notifikasi perbankan, dan promosi mungkin akan dianggap sebagai spam. Oleh karena itu, dalam penelitian ini, SMS akan diklasifikasikan ke dalam lima kategori: normal, penipuan, promosi, autentikasi, dan notifikasi perbankan.

Penelitian yang dilakukan oleh (Jain *et al.*, 2022) terhadap klasifikasi SMS spam mendapat akurasi tertinggi dengan metode SVM dibandingkan dengan Naïve Bayes, Logistic Regression , dan Decision Tree. Akurasi yang diperoleh adalah sebesar 98,80% pada model SVM, 95.05% pada model Naïve Bayes, 97,80% pada Logistic Regression, dan 98,00% pada Decision Tree. Demikian pula penelitian yang dilakukan oleh (Bheemesh dan Deepa, 2020) mengenai klasifikasi 5573 sms ham dan spam mendapat akurasi tertinggi dengan metode SVM dibandingkan Linear Regression. Akurasi rata-rata yang diperoleh adalah 97,67% menggunakan SVM dan 92,00% menggunakan Linear Regression.

Pada penjelasan di atas, dapat dilihat bahwa penelitian mengenai klasifikasi SMS menjadi dua kelas (spam dan bukan spam) telah banyak dilaksanakan. Metode SVM juga terbukti dapat menghasilkan akurasi yang tinggi dalam pengklasifikasian SMS dibanding metode-metode lainnya. Berdasarkan temuan-temuan tersebut, maka peneliti memutuskan untuk melakukan penelitian klasifikasi SMS menjadi lima kelas (normal, penipuan, promo, autentikasi, dan notifikasi perbankan) dengan judul "Klasifikasi Spam pada *Short Message Service* (SMS) Menggunakan Metode Support Vector Machine".

2. DATA DAN METODOLOGI

2.1 Pengumpulan Data

Dalam penelitian klasifikasi SMS ini, terdapat dua sumber data yang digunakan. Sumber pertama adalah dataset SMS dengan tiga kelas, yaitu normal, penipuan, dan promo, sebanyak 1118 data (https://github.com/kmkurn/id-nlp-resource#text-classification). Sumber kedua adalah data SMS yang dikumpulkan secara mandiri oleh peneliti sebanyak 62 data dengan dua kelas, yaitu autentikasi dan bank. Data dengan kelas autentikasi dan bank ini kemudian diaugmentasi menggunakan Large Language Models (LLM) ChatGPT3.5 dari OpenAI sehingga menjadi 466 data. Total data yang digunakan dalam penelitian ini adalah 1118 ditambah 466, sehingga totalnya menjadi 1584 data.

2.2 *Short Message System* (SMS)

Short Message Service (SMS) adalah layanan yang menyediakan transfer pesan singkat tanpa koneksi internet dengan isi paling banyak 160 karakter menggunakan saluran dalam jaringan seluler (Souza *et al.*, 2011). Layanan ini sering digunakan dalam kehidupan sehari-hari seperti pemantauan kesehatan, *mobile banking, mobile commerce* dan sebagainya (Saxena, Chaudhari dan Member, 2014). Beberapa contoh penggunaan SMS adalah pengiriman *One-Time Password* (OTP), peringatan transaksi perbankan dan pesan-pesan promosi (Bhatnagar dan Kumar, 2018).

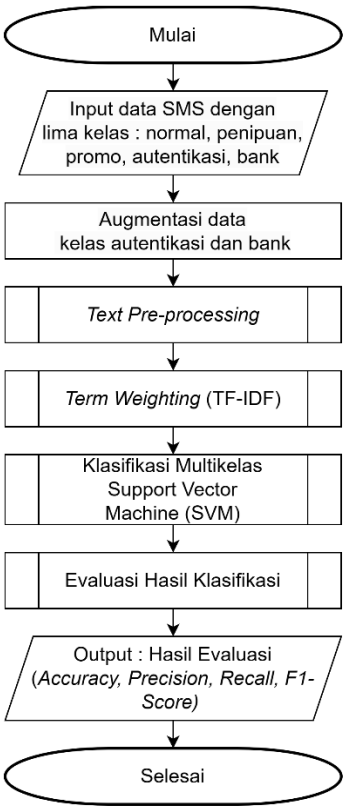
2.3 Spam

Pesan spam adalah pesan-pesan tidak diinginkan yang dikirim oleh *spammer*. *Spammer* biasanya mengirim sejumlah besar pesan kepada pengguna layanan pesan untuk keuntungan organisasi atau pribadi *spammer* (Gadde, Lakshmanarao dan Satyanarayana, 2021). *Spamming* SMS menjadi gangguan besar bagi pelanggan seluler karena sifatnya yang menyebar luas. Pesan-pesan ini sangat mengganggu pengguna karena dapat berisi

serangan terhadap informasi pribadi (Shafi'i *et al.*, 2017).

2.4 Perancangan Algoritma

Tahapan yang dilakukan pada penelitian ini adalah augmentasi data kelas autentikasi dan kelas bank, dilanjutkan *text pre-processing*, *term weighting*, klasifikasi multikelas menggunakan SVM, dan evaluasi. Diagram alir perancangan algoritma dapat dilihat pada Gambar 1.



Gambar 1. Rancangan Metodologi

2.5 Augmentasi Data Teks

Pada umumnya, augmentasi data dilakukan dengan memodifikasi data yang ada atau menghasilkan data sintetis. Augmentasi teks memiliki tantangan yang cukup besar karena perubahan kecil dapat mengubah arti kalimat. Beberapa metode augmentasi teks seperti penghapusan, penambahan, atau penggantian kata, sering menyebabkan perubahan sentimen dan menghasilkan kalimat ambigu. Oleh karena itu, augmentasi data teks yang ideal adalah dengan parafrase (Muftie dan Haris, 2023), yang dapat dilakukan menggunakan Large Language Models (LLM) seperti ChatGPT (Wake *et al.*, 2023).

2.6 *Text Preprocessing*

Setelah data berhasil dikumpulkan, dibutuhkan proses *preprocessing* untuk menghindari *noise* dan hal-hal yang tidak dibutuhkan dalam dataset. Menurut (Ma'arif, 2016), dataset yang terkumpul dari proses scraping biasanya memiliki bentuk yang tidak terstruktur, maka dari itu text *preprocessing* ini sangat penting untuk dilaksanakan sebelum dataset diproses lebih lanjut.

Pada umumnya proses text *preprocessing* terdiri dari *stopword removal*, yang juga dilengkapi dengan proses *case folding*, *remove punctuation* & *number*, kemudian dilakukan *tokenizing*, dan *stemming* (Juwiantho *et al.*, 2020). Namun pada penelitian ini, tahapan *preprocessing* yang dilakukan adalah penghapusan angka dan tanda baca, *case folding*, *stemming* dan *tokenizing*.

2.7 *Term Weighting*

Term Weighting atau Pembobotan kata memegang peran penting dalam mencapai performa tinggi saat melakukan *Information Retrieval* atau klasifikasi teks. (Zhanguo *et al.*, 2011). Salah satu metode *term weighting* adalah *Term Frequency – Inverse Document Frequency* (TF-IDF). Cara kerja metode TF-IDF adalah dengan memberi bobot pada setiap *term* yang akan dimuat dalam matriks TF-IDF. Tahapan perhitungan *term weighting* TF-IDF dapat dilihat pada Persamaan (1) sampai Persamaan (4).

$$tf_{t,d} = \begin{cases} 1 + log_{10} f_{t,d}\,, & jika\ f_{t,d} > 0 \\ 0\,, & jika\ f_{t,d} = 0 \end{cases} \quad (1)$$

$$df_t = \text{banyak dokumen yang terdapat } term \text{ t} \quad (2)$$

$$idf_t = log_{10}\left(\frac{banyak\ dokumen}{df_t}\right) \quad (3)$$

$$w_{t,d} = tf_{t,d} \times idf_t \quad (4)$$

Keterangan:

$f_{t,d}$ = jumlah kemunculan *term* t pada dokumen *d*
$tf_{t,d}$ = Nilai *Term Frequency* (TF) *term t* pada dokumen *d*
df_t = Nilai *Document Frequency* (DF) *term t*
idf_t = Nilai *Inverse Document Frequency* (IDF) *term t*
$w_{t,d}$ = Nilai bobot (TF-IDF) *term t* pada dokumen *d*

2.8 Support Vector Machine

Algoritma klasifikasi Support Vector Machine (SVM) adalah algoritma yang diusulkan oleh Vapnik untuk menyelesaikan klasifikasi dua kelas (Kalcheva, Karova dan Penev, 2020). Algoritma ini bekerja dengan cara mencari sebuah *hyperplane* optimal yang dapat memisahkan sekumpulan data positif dan negatif (Moattar, Homayounpour dan Zabihzadeh, 2006).

SVM juga dapat menangani data yang tidak dapat dipisahkan secara linear dengan menggunakan teknik yang disebut trik *kernel* atau fungsi *kernel*. Fungsi *kernel* ini memungkinkan SVM untuk menemukan batas-batas non-linear dalam ruang fitur berdimensi tinggi. (Philip *et al.*, 2023). Terdapat empat fungsi *kernel* dalam algoritma SVM yaitu Linear, Sigmoid, Radial Basis Function (RBF), dan Polynomial (Ebora, Español dan Padilla, 2022).

(Vijayakumar dan Wu, 1999) mengembangkan sebuah metode yang dinamakan metode sekuensial. Pada metode sekuensial, diterapkan algoritma berbasis *sequential gradient ascent* yang memungkinkan implementasi SVM lebih cepat dan memberikan solusi yang optimal. Tahapan metode sekuensial yaitu:

1. Melakukan inisialisasi parameter awal yang digunakan yaitu λ (*lambda*) = variabel nilai skalar, $\boldsymbol{\gamma}$ (*gamma*) = nilai *learning rate*, C (*complexity*) = nilai *complexity*/kompleksitas, ε (*epsilon*) = batas perubahan nilai *alpha*, dan jumlah iterasi maksimum.
2. Melakukan perhitungan nilai *Kernel* (pada penelitian ini menggunakan *kernel* linear) dengan Persamaan (5).

$$K(x_i, x_j) = x_i \cdot x_j \quad (5)$$

Keterangan:
$K(x_i, x_j)$ = Nilai *Kernel*
x_i = data ke-i
x_j = data ke-j

3. Melakukan perhitungan nilai *Hessian matrix* dengan Persamaan (6).

$$D_{i,j} = y_i y_j (K(x_i, x_j) + \lambda^2) \quad (6)$$

Keterangan:
$D_{i,j}$ = *Hessian matrix*
y_i = kelas pada data ke-i (+1 atau -1)
y_j = kelas pada data ke-j (+1 atau -1)
$K(x_i, x_j)$ = Nilai *Kernel* dari perhitungan Nomor 2
λ (*lambda*) = variabel nilai skalar

4. Melakukan inisialisasi terhadap nilai awal parameter *alpha* ($\alpha_i = 0$) lalu menghitung nilai *error*, perubahan nilai *error* dan pembaruan nilai *alpha* menggunakan Persamaan (7) sampai Persamaan (9).

$$E_i = \sum_{j=1}^{n} \alpha_i D_{i,j} \quad (7)$$

$$\delta\alpha_i = \min\{\max[\gamma(1 - E_i), -\alpha_i], C - \alpha_i\} \quad (8)$$

$$\alpha_i = \alpha_i + \delta\alpha_i \quad (9)$$

Keterangan:
E_i = Nilai *error* (E) ke-i
α_j = Nilai *alpha* ke-j
$D_{i,j}$ = Nilai *Hessian matrix* baris ke-i, kolom ke-j
n = jumlah dokumen/data
$\delta\alpha_i$ = perubahan nilai α_i
C = nilai kompleksitas
α_i = Nilai *alpha* ke-i

5. Mengulangi perhitungan nomor 4 hingga iterasi maksimum atau memenuhi Persamaan (10).

$$\max(|\delta\alpha_i|) < \varepsilon \quad (10)$$

Keterangan:
$\delta\alpha_i$ = perubahan nilai α_i
ε = *epsilon*

6. Berdasarkan hasil perhitungan nomor 4, maka dipilih data dengan nilai *alpha* terbesar dari masing-masing kelas positif dan negatif. Kemudian dilakukan perhitungan *kernel* positif dan negatif dengan menggunakan Persamaan (11) dan Persamaan (12).

$$K(x_i, x^+) = x_i \cdot x^+ \quad (11)$$

$$K(x_i, x^-) = x_i \cdot x^- \quad (12)$$

Keterangan:
$K(x_i, x^+) =$ *Kernel* positif
$K(x_i, x^-) =$ *Kernel* negatif
x_i = Data ke-i
x^+ = Data dengan *alpha* tertinggi dari kelas positif
x^- = Data dengan *alpha* tertinggi dari kelas negatif

7. Melakukan perhitungan nilai *bias* dengan Persamaan (13).

$$b = -\frac{1}{2}\left(\sum_{i=0}^{n}\alpha_i y_i K(x_i, x^+) + \sum_{i=0}^{n}\alpha_i y_i K(x_i, x^-)\right) \quad (13)$$

Keterangan:
b = *bias*
α_i= Nilai *alpha* ke-i
y_i = kelas pada data ke-i (+1 atau -1)
$K(x_i, x^+) =$ *Kernel* positif
$K(x_i, x^-) =$ *Kernel* negatif
x_i = Data ke-i
x^+ = Data dengan *alpha* tertinggi dari kelas positif
x^- = Data dengan *alpha* tertinggi dari kelas negatif

8. Menghitung nilai fungsi *f(x)* dengan menggunakan Persamaan (14).

$$f(x) = \sum_{i=0}^{n}\alpha_i y_i \; K(x_i, x) + b \quad (14)$$

Keterangan:
$f(x)$ = fungsi hasil klasifikasi
α_i= Nilai *alpha* ke-i
y_i = kelas pada data ke-i (+1 atau -1)
$K(x_i, x) =$ *Kernel* data latih dan data uji
x_i = Data latih ke-i
x = Data uji
b = *bias*

2.9 *One-Against-All*

Algoritma klasifikasi Support Vector Machine (SVM) adalah algoritma yang diusulkan oleh Vapnik untuk menyelesaikan klasifikasi dua kelas (Kalcheva, Karova dan Penev, 2020). Dalam kasus klasifikasi dengan lebih dari dua kelas, maka dibutuhkan metode tambahan. Salah satu metode yang dapat digunakan untuk klasifikasi lebih dari dua kelas dengan SVM adalah *One-Against-All.*

Tahap pertama dari *one-against-all* adalah dengan melakukan pengubahan kelas menjadi +1 jika kelas sama dengan iterasi saat ini dan kelas lainnya diubah menjadi -1. Jika terdapat N jumlah kelas, maka akan dilakukan sub klasifikasi SVM sebanyak N kali sehingga akan diperoleh nilai *f(x)* sebanyak N. Untuk menentukan kelas dari sebuah dokumen setelah semua sub klasifikasi dilakukan, dapat digunakan Persamaan (15) (Putra, Indriati dan Perdana, 2023).

$$kelas\ dokumen\ x = \underset{i=1,\ldots,N}{\operatorname{argmax}}(f_i(x)) \quad (15)$$

Keterangan:
argmax = mencari kelas di mana nilai *f(x)* tertinggi
N = banyak kelas
f(x) = hasil fungsi klasifikasi yang dihasilkan menggunakan Persamaan (14)

3. HASIL DAN PEMBAHASAN

3.1 Hasil Pengujian Parameter *lambda*

Pada penelitian ini, dilakukan pengujian parameter *lambda* dengan mencoba 6 nilai yaitu 10^{-2}, 10^{-1}, 1, 10, 10^2, dan 10^3. Pengujian parameter *lambda* dilakukan dengan menggunakan nilai parameter lain yaitu *C* = 100, *epsilon* = 10^{-5}, konstanta *gamma* =10^{-2}, dan jumlah iterasi maksimum = 50. Hasil pengujian berupa *accuracy*, *precision*, *recall*, dan *f1-score* pada 6 nilai *lambda* dapat dilihat pada Tabel 1.

Tabel 1. Hasil Pengujian Parameter *lambda*

Nilai *lambda*	*Accuracy*	*Precision*	*Recall*	*F1-Score*
10^{-2}	0,95	0,96	0,94	0,95
10^{-1}	0,95	0,96	0,95	0,95
1	0,91	0,93	0,91	0,91
10	0,87	0,92	0,86	0,86
10^2	0,91	0,92	0,91	0,91
10^3	0,91	0,92	0,91	0,91

Parameter *lambda* adalah salah satu unsur dalam perhitungan *hessian matrix.* Apabila nilai *lambda* semakin besar, maka pengaruh nilai *kernel* akan lebih kecil dalam perhitungan nilai *hessian matrix,* dan sebaliknya apabila nilai *lambda* semakin kecil, maka pengaruh nilai *kernel* akan lebih besar pada *hessian matrix*. Berdasarkan hasil pengujian pada Tabel 1, dapat dilihat bahwa hasil tertinggi diperoleh saat parameter *lambda* bernilai 10^{-1}, sedangkan hasil terendah diperoleh saat parameter *lambda* bernilai 10. Pada nilai *lambda* = 10^{-1}, diperoleh hasil *accuracy* = 0,95, *precision* = 0,96, *recall* = 0,95, dan *f1-score* = 0,95. Pada nilai *lambda* = 10, diperoleh hasil *accuracy* = 0,87, *precision* = 0,92, *recall* = 0,86, dan *f1-score* = 0,86. Nilai *lambda* = 10^{-1} akan digunakan pada pengujian parameter berikutnya karena memperoleh hasil yang tertinggi.

3.2 Hasil Pengujian Parameter konstanta *gamma*

Pada penelitian ini, dilakukan pengujian parameter *gamma* dengan mencoba 6 nilai yaitu 10^{-4}, 10^{-3}, 10^{-2}, 1, 10, dan 10^{2}. Pengujian parameter konstanta *gamma* dilakukan dengan menggunakan nilai parameter lain yaitu *C* = 100, *epsilon* = 10^{-5}, *lambda* = 10^{-1}, dan jumlah iterasi maksimum = 50. Hasil pengujian berupa *accuracy*, *precision*, *recall*, dan *f1-score* pada 6 nilai *gamma* dapat dilihat pada Tabel 2.

Tabel 2. Hasil Pengujian Parameter konstanta *gamma*

Nilai *gamma*	*Accuracy*	*Precision*	*Recall*	*F1-Score*
10^{-4}	0,91	0,92	0,91	0,91
10^{-3}	0,91	0,92	0,91	0,91
10^{-2}	0,95	0,96	0,95	0,95
1	0,47	0,65	0,57	0,46
10	0,47	0,74	0,51	0,42
10^{2}	0,5	0,72	0,51	0,44

Parameter konstanta *gamma (learning rate)* adalah salah satu unsur dalam perhitungan perubahan nilai *alpha (delta alpha)*. Apabila nilai *gamma* semakin besar, perubahan nilai *alpha* akan semakin besar dan sebaliknya apabila nilai konstanta *gamma* semakin kecil, maka perubahan nilai *alpha* juga akan semakin kecil. Berdasarkan hasil pengujian pada Tabel 2, dapat dilihat bahwa hasil tertinggi diperoleh saat parameter *gamma* bernilai 10^{-2}, sedangkan hasil terendah diperoleh saat parameter *gamma* bernilai 1.

Hasil klasifikasi menurun seiring bertambah besarnya nilai konstanta *gamma*, hal ini disebabkan oleh nilai konstanta *gamma* yang besar menyebabkan perubahan nilai *alpha* yang terlalu besar sehingga tidak optimal. Pada nilai konstanta *gamma* = 10^{-2}, diperoleh hasil *accuracy* = 0,95, *precision* = 0,96, *recall* = 0,95, dan *f1-score* = 0,95. Pada nilai konstanta *gamma* = 1, diperoleh hasil *accuracy* = 0,47, *precision* = 0,65, *recall* = 0,57, dan *f1-score* = 0,46. Nilai konstanta *gamma* = 10^{-2} akan digunakan pada pengujian parameter berikutnya karena memperoleh hasil yang tertinggi.

3.3 Hasil Pengujian Parameter *C*

Pada penelitian ini, dilakukan pengujian parameter *C* dengan mencoba 6 nilai yaitu 10^{-10}, 10^{-5}, 10^{-1}, 1, 10, dan 10^{2}. Pengujian parameter *C* dilakukan dengan menggunakan nilai parameter lain yaitu *gamma* = 10^{-2}, *epsilon* = 10^{-5}, *lambda* = 10^{-1}, dan jumlah iterasi maksimum = 50. Hasil pengujian berupa *accuracy*, *precision*, *recall*, dan *f1-score* pada 6 nilai *C* dapat dilihat pada Tabel 3.

Tabel 3. Hasil Pengujian Parameter *C*

Nilai *C*	*Accuracy*	*Precision*	*Recall*	*F1-Score*
10^{-10}	0,91	0,92	0,91	0,91
10^{-5}	0,91	0,92	0,91	0,91
10^{-1}	0,95	0,96	0,95	0,95
1	0,95	0,96	0,95	0,95
10	0,95	0,96	0,95	0,95
10^{2}	0,95	0,96	0,95	0,95

Parameter *C (complexity)* adalah salah satu unsur dalam perhitungan perubahan nilai *alpha (delta alpha)*. Apabila nilai *C* semakin besar, perubahan nilai *alpha* akan semakin besar dan sebaliknya apabila nilai *C* semakin kecil, maka perubahan nilai *alpha* juga akan semakin kecil. Berdasarkan hasil pengujian pada Tabel 3, dapat dilihat bahwa hasil tertinggi diperoleh saat parameter *C* bernilai 10^{-1}, 1, 10, dan 10^{2}, sedangkan hasil terendah diperoleh saat parameter *C* bernilai 10^{-10} dan 10^{-5}.

Hasil klasifikasi meningkat seiring dengan bertambah besarnya nilai *C*, hal ini menunjukkan pada nilai *alpha* yang optimal diperoleh saat *C* bernilai tinggi. Pada nilai *C* = 10^{-10} dan 10^{-5}, diperoleh hasil *accuracy* = 0,91, *precision* = 0,92, *recall* = 0,91, dan *f1-score* = 0,91. Pada nilai *C* = 10^{-1}, 1, 10, dan 10^{2}, diperoleh hasil *accuracy* = 0,95, *precision* = 0,96, *recall* = 0,95, dan *f1-score* = 0,96. Nilai *C* = 10^{2} akan digunakan pada pengujian parameter berikutnya karena memperoleh hasil yang tertinggi.

3.4 Hasil Pengujian Parameter *epsilon*

Pada penelitian ini, dilakukan pengujian parameter *epsilon* dengan mencoba 6 nilai yaitu 10^{-10}, 10^{-5}, 10^{-1}, 1, 10, dan 10^{2}. Pengujian parameter *epsilon* dilakukan dengan menggunakan nilai parameter lain yaitu *C*=10^{2}, *gamma* = 10^{-2}, *lambda* = 10^{-1}, dan jumlah iterasi maksimum = 50. Hasil pengujian berupa *accuracy*, *precision*, *recall*, dan *f1-score* pada 6 nilai *epsilon* dapat dilihat pada Tabel 4.

Tabel 4. Hasil Pengujian Parameter *C*

Nilai *epsilon*	*Accuracy*	*Precision*	*Recall*	*F1-Score*
10^{-10}	0,95	0,96	0,95	0,95
10^{-5}	0,95	0,96	0,95	0,95
10^{-1}	0,91	0,92	0,91	0,91
1	0,91	0,92	0,91	0,91

Nilai *epsilon*	*Accuracy*	*Precision*	*Recall*	*F1-Score*
10	0,91	0,92	0,91	0,91
10^2	0,91	0,92	0,91	0,91

Parameter *epsilon* adalah salah satu penentu berhentinya iterasi untuk mencari nilai *alpha*. Jika perubahan nilai *alpha* sudah kurang dari nilai *epsilon*, maka iterasi akan berhenti. Apabila nilai *epsilon* terlalu besar, maka iterasi akan cepat berhenti dan sebaliknya apabila nilai *epsilon* terlalu kecil, maka iterasi akan berlangsung lebih lama. Berdasarkan hasil pengujian pada Tabel 4, dapat dilihat bahwa hasil tertinggi diperoleh saat parameter *epsilon* bernilai 10^{-10}, dan 10^{-5}, sedangkan hasil terendah diperoleh saat parameter *epsilon* bernilai 10^{-1}, 1, 10, dan 10^2.

Hasil klasifikasi menurun seiring dengan bertambah besarnya nilai *epsilon*, hal ini menunjukkan pada nilai *alpha* yang optimal diperoleh saat *epsilon* bernilai rendah. Pada nilai *epsilon* = 10^{-10} dan 10^{-5}, diperoleh hasil *accuracy* = 0,95, *precision* = 0,96, *recall* = 0,95, dan *f1-score* = 0,95. Pada nilai *epsilon* = 10^{-1}, 1, 10, dan 10^2, diperoleh hasil *accuracy* = 0,91, *precision* = 0,92, *recall* = 0,91, dan *f1-score* = 0,91. Nilai *epsilon* = 10^{-5} akan digunakan pada pengujian parameter berikutnya karena memperoleh hasil yang tertinggi.

3.5 Hasil Pengujian Parameter Jumlah Iterasi Maksimum

Pada penelitian ini, dilakukan pengujian parameter jumlah iterasi maksimum dengan mencoba 6 nilai yaitu 5, 10, 20, 50, 100, dan 200. Pengujian parameter jumlah iterasi maksimum dilakukan dengan menggunakan nilai parameter lain yaitu C=10^2, *gamma* = 10^{-2}, *lambda* = 10^{-1}, dan *epsilon* = 10^{-5}. Hasil pengujian berupa *accuracy*, *precision*, *recall*, dan *f1-score* pada 6 nilai jumlah iterasi maksimum dapat dilihat pada Tabel 5.

Tabel 5. Hasil Pengujian Parameter Jumlah Iterasi Maksimum

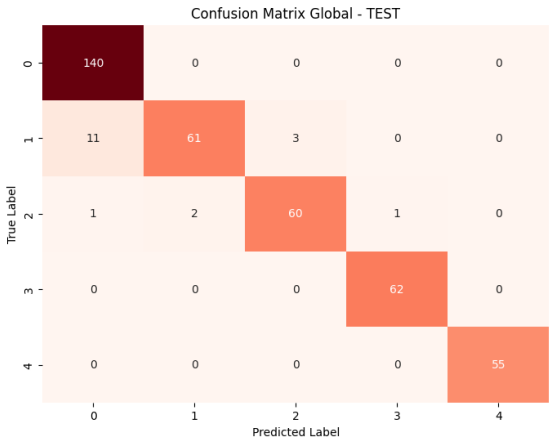
Nilai iterasi	*Accuracy*	*Precision*	*Recall*	*F1-Score*
5	0,95	0,96	0,94	0,95
10	0,94	0,96	0,94	0,94
20	0,94	0,96	0,94	0,95
50	0,95	0,96	0,95	0,95
100	0,79	0,86	0,87	0,83
200	0,44	0,54	0,55	0,41

Parameter jumlah iterasi maksimum adalah salah satu penentu (bersama dengan *epsilon*) berhentinya iterasi untuk mencari nilai *alpha*. Iterasi perhitungan nilai *alpha* akan berhenti dilakukan jika sudah mencapai jumlah iterasi maksimum. Semakin tinggi nilai jumlah iterasi maksimum, maka iterasi akan berlangsung lebih lama. Berdasarkan hasil pengujian pada Tabel 5, dapat dilihat bahwa hasil tertinggi diperoleh saat parameter jumlah iterasi maksimum bernilai 50 sedangkan hasil terendah diperoleh saat parameter jumlah iterasi maksimum bernilai 200.

Hasil ini menunjukkan bahwa jumlah iterasi yang sedikit mengakibatkan perhitungan *alpha* terlalu cepat berhenti sehingga nilai *alpha* tidak optimal dan jumlah iterasi yang terlalu banyak mengakibatkan perhitungan *alpha* terlalu lama berhenti sehingga nilai *alpha* juga tidak optimal. Pada nilai iterasi = 50, diperoleh hasil *accuracy* = 0,95, *precision* = 0,96, *recall* = 0,95, dan *f1-score* = 0,95. Pada nilai iterasi = 200, diperoleh hasil *accuracy* = 0,44, *precision* = 0,54, *recall* = 0,55, dan *f1-score* = 0,41.

3.6 Hasil Evaluasi Klasifikasi

Hasil evaluasi klasifikasi spam pada SMS menggunakan metode SVM dengan lima kelas (normal, penipuan, promo, autentikasi, dan bank) dalam bentuk *confusion matrix* dapat dilihat pada Gambar 2.



Gambar 2. *Confusion matrix*

Keterangan: Label 0 adalah normal, 1 adalah penipuan, 2 adalah promo, 3 adalah autentikasi, 4 adalah bank.

Pada Gambar 2, dapat dilihat bahwa kesalahan prediksi terbanyak, yaitu sebanyak 11 data, terjadi pada data dengan kelas asli 1

(FRAUD) yang diprediksi sebagai kelas 0 (NORMAL). Mayoritas kesalahan prediksi ini disebabkan oleh pesan-pesan yang sulit diklasifikasikan berdasarkan sintaksisnya. Dengan kata lain, pesan-pesan ini memiliki struktur yang membuatnya sulit untuk diidentifikasi dengan benar. Kelas 4 (BANK) menunjukkan hasil evaluasi tertinggi jika dibandingkan dengan kelas-kelas lainnya. Hal ini karena pesan-pesan dalam kelas 4 (BANK) umumnya memiliki struktur yang seragam dan cenderung minim variasi. Pesan-pesan ini sudah memiliki sintaksis yang tetap dan kata-kata yang digunakan juga tidak terlalu beragam, sehingga lebih mudah untuk diklasifikasikan dengan benar.

4. PENUTUP

Klasifikasi Spam pada SMS menggunakan metode SVM dimulai dengan augmentasi pada kelas data yang sedikit, kemudian digabungkan dengan data awal. Proses *pre-processing* meliputi *cleaning*, *case folding*, *stemming*, dan *tokenizing*, diikuti penghitungan *term weighting* dengan metode TF-IDF. Klasifikasi dilakukan menggunakan SVM dengan konsep one-against-all karena SVM hanya dapat mengklasifikasi dua kelas. Proses ini terdiri dari tahap *training* dan *testing*. Pada tahap *training*, dihitung *kernel*, *hessian matrix, alpha*, dan *bias*, sementara pada tahap *testing* dihitung skor prediksi *f(x)*, dan label data ditentukan berdasarkan nilai *f(x)* tertinggi. Hasil evaluasi menunjukkan *accuracy* 0,95, *precision* 0,96, *recall* 0,95, dan *f1-score* 0,95. Kelas bank paling akurat diprediksi dengan nilai rata-rata 1, sementara kelas penipuan paling tidak akurat dengan rata-rata 0,88. Kelas normal memiliki rata-rata 0,95, kelas promo 0,93, dan kelas autentikasi 0,99. Parameter yang digunakan adalah C = 100, *epsilon* = 10^{-5}, konstanta *gamma* = 0,01, *lambda* = 0,1, dan iterasi maksimum = 50.

5. DAFTAR PUSTAKA

Bhatnagar, S. dan Kumar, A. (2018) "A Rule-Based Classification of Short Message Service Type," *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, (Icisc), hal. 1139–1142. Tersedia pada: https://doi.org/10.1109/ICISC.2018.8398982.

Bheemesh, K.R. dan Deepa, N. (2020) "Accurate SMS Spam Detection Using Support Vector Machine In Comparison With Linear Regression," *2023 Fifth International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, hal. 1–4. Tersedia pada: https://doi.org/10.1109/ICECCT56650.2023.10179827.

Dewi, F.K. *et al.* (2017) "Multiclass SMS Message Categorization : Beyond Spam Binary Classification," *2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, hal. 210–215. Tersedia pada: https://doi.org/10.1109/ICACSIS.2017.8355035.

Dharani, V., Hegde, D. dan Mohana (2023) "Spam SMS (or) Email Detection and Classification using Machine Learning," *2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, (Icssit), hal. 1104–1108. Tersedia pada: https://doi.org/10.1109/ICSSIT55814.2023.10060908.

Ebora, J.G.O., Español, J.C.N. dan Padilla, D.A. (2022) "Text Classification of Facebook Messages Using Multiclass Support Vector Machine," *2022 13th International Conference on Computing Communication and Networking Technologies, ICCCNT 2022*, hal. 1–6. Tersedia pada: https://doi.org/10.1109/ICCCNT54827.2022.9984554.

Gadde, S., Lakshmanarao, A. dan Satyanarayana, S. (2021) "SMS Spam Detection using Machine Learning and Deep Learning Techniques," *2021 7th International Conference on Advanced Computing and Communication Systems, ICACCS 2021*, hal. 358–362. Tersedia pada: https://doi.org/10.1109/ICACCS51430.2021.9441783.

Jain, T. *et al.* (2022) "SMS Spam Classification Using Machine Learning Techniques," *2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, hal. 273–279. Tersedia pada: https://doi.org/10.1109/Confluence5298

9.2022.9734128.

Juwiantho, H. *et al.* (2020) "Sentiment Analysis Twitter Bahasa Indonesia Berbasis WORD2VEC Menggunakan Deep Convolutional Neural Network," *Jurnal Teknologi Informasi dan Ilmu Komputer*, 7(1), hal. 181–188. Tersedia pada: https://doi.org/10.25126/jtiik.202071758.

Kalcheva, N., Karova, M. dan Penev, I. (2020) "Comparison of the accuracy of SVM kemel functions in text classification," *Proceedings of the International Conference on Biomedical Innovations and Applications, BIA 2020*, hal. 141–145. Tersedia pada: https://doi.org/10.1109/BIA50171.2020.9244278.

Ma'arif, M.R. (2016) "Integrasi Laman Web Tentang Pariwisata Daerah Istimewa Yogyakarta Memanfaatkan Teknologi Web Scraping Dan Text Mining," *Teknomatika*, 9(1), hal. 71–80. Tersedia pada: http://eprints.binadarma.ac.id/3554/1/Ma%27arif2016 TEKNOMATIKA %5BIntegrasi Laman Web Tentang Pariwisata Daerah Istimewa Yogyakarta Memanfaatkan Teknologi Web Scraping Dan Text Mining%5D.pdf.

Moattar, M.H., Homayounpour, M.M. dan Zabihzadeh, D. (2006) "Persian Text Normalization using Classification Tree and Support Vector Machine," hal. 1308–1311. Tersedia pada: https://doi.org/10.1109/ictta.2006.1684569.

Muftie, F. dan Haris, M. (2023) "IndoBERT Based Data Augmentation for Indonesian Text Classification," *2023 International Conference on Information Technology Research and Innovation (ICITRI)*, hal. 128–132. Tersedia pada: https://doi.org/10.1109/ICITRI59340.2023.10250061.

Philip, J. *et al.* (2023) "A Comparative Study of Text Classification using Selective Machine Learning Algorithms," hal. 482–484. Tersedia pada: https://doi.org/10.1109/iciccs56967.2023.10142474.

Popovac, M. *et al.* (2018) "Convolutional Neural Network based SMS Spam Detection," *2018 26th Telecommunications Forum (TELFOR)*, hal. 1–4.

Putra, P.R.B., Indriati dan Perdana, R.S. (2023) "Klasifikasi Judul Berita Onlinemenggunakan Metode Support Vector Machine(SVM) dengan Seleksi Fitur Chi-square.pdf," *Jurnal Pengemb*, 7(5), hal. 2132–2141.

Saxena, N., Chaudhari, N.S. dan Member, S. (2014) "EasySMS : A Protocol for End-to-End Secure Transmission of SMS," *IEEE Transactions on Information Forensics and Security*, 9(7), hal. 1157–1168. Tersedia pada: https://doi.org/10.1109/TIFS.2014.2320579.

Shafi'i, M. *et al.* (2017) "A Review on Mobile SMS Spam Filtering Techniques," *IEEE Access*, 5, hal. 15650–15666. Tersedia pada: https://doi.org/10.1109/ACCESS.2017.2666785.

Souza, R.D. *et al.* (2011) "Protocol Implementation for Short Message Service over IP," *2011 6th International Conference on Industrial and Information Systems*, hal. 443–447. Tersedia pada: https://doi.org/10.1109/ICIINFS.2011.6038110.

Truecaller (2021) *Truecaller Insights: Top 20 Countries Affected By Spam Calls In 2021*. Tersedia pada: https://www.truecaller.com/blog/insights/top-20-countries-affected-by-spam-calls-in-2021 (Diakses: 10 September 2023).

Vijayakumar, S. dan Wu, S. (1999) "Sequential Support Vector Classi ers and Regression 1 Abstract 2 Introduction," *Proc. International Conference on Soft Computing*, ((SOCO'99),Genoa, Italy), hal. 610–619.

Wake, N. *et al.* (2023) "ChatGPT Empowered Long-Step Robot Control in Various Environments : A Case Application," *IEEE Access*, 11(August), hal. 95060–95078. Tersedia pada: https://doi.org/10.1109/ACCESS.2023.3310935.

Zhanguo, M. *et al.* (2011) "Improved terms weighting algorithm of text," *Proceedings - 2011 International*

Conference on Network Computing and Information Security, NCIS 2011, 2, hal. 367–370. Tersedia pada: https://doi.org/10.1109/NCIS.2011.171.