

Perbandingan Metode Klasifikasi Support Vector Machine dan Naïve Bayes untuk Analisis Sentimen pada Ulasan Tekstual di Google Play Store


By Lutfi Budi Ilmawan

Perbandingan Metode Klasifikasi Support Vector Machine dan Naïve Bayes untuk Analisis Sentimen pada Ulasan Tekstual di Google Play Store

Lutfi Budi Ilmawan^{a,1}, Muh. Aliyazid Mude^{b,2}

^{a,b} Universitas Muslim Indonesia, Jl. Urip Sumoharjo KM. 5, Makassar dan 90231, Indonesia

¹ lutfibudi.ilmawan@umi.ac.id; ² aliayazid.mude@umi.ac.id

INFORMASI ARTIKEL	ABSTRAK
<p>Diterima : xx – xx – 2020 Direvisi : xx – xx – 2020 Diterbitkan : xx – xx – 2020</p> <p>Kata Kunci: Klasifikasi Analisis Sentimen Support Vector Machine Naïve Bayes Cross-validation</p>	<p>Pada penelitian ini, metode klasifikasi SVM akan dibandingkan dengan metode klasifikasi yang lain, yaitu dengan menggunakan metode klasifikasi Naïve Bayes. Metode klasifikasi Naïve Bayes merupakan metode klasifikasi yang ringan dalam pemrosesan dan memiliki akurasi yang tinggi jika diterapkan untuk klasifikasi teks menurut beberapa penelitian sebelumnya. Akurasi dari classifier diukur menggunakan metode K-fold cross validation yang hasilnya akan ditabulasikan pada tabel confusion matrix, dengan nilai K=3. Pada penelitian ini, data-data yang diolah adalah ulasan tekstual aplikasi pada Google Play Store berbahasa Indonesia yang diambil dari penelitian sebelumnya. Hasil pengujian yang didapatkan dari metode 3-folds cross-validation menghasilkan bahwa SVM Classifier memiliki nilai yang lebih tinggi jika dibandingkan dengan akurasi dari Naïve Bayes classifier untuk mengklasifikasikan ulasan tekstual berbahasa Indonesia pada Google Play Store, yakni SVM classifier mendapatkan akurasi sebesar 81,46% dan Naïve Bayes classifier sebesar 75,41%.</p> <p>ABSTRACT</p> <p><i>In this research, the performance of SVM classification method will be compared with other classification methods, by using the Naïve Bayes classification method. Naïve Bayes classification method is a light classification method and has a high accuracy if applied to the text classification according to some previous studies. The accuracy of the classifier is measured using the K-fold cross validation method whose results will be tabulated in a confusion matrix table, with a value of K = 3. In this study, the data processed are textual reviews of applications in the Indonesian language Google Play Store obtained from previous research. The test results obtained from the 3-fold cross-validation method produce that SVM Classifier has a higher value of accuracy when compared with the accuracy of the Naïve Bayes classifier, the SVM classifier gets an accuracy of 81.46% and Naïve Bayes classifier by 75.41%.</i></p> <p>This is an open access article under the CC-BY-SA license.</p> 

I. Pendahuluan

Pada penelitian sebelumnya tentang analisis sentimen pada Google Play Store [1], menunjukkan bahwa metode klasifikasi SVM (Support Vector Machine) mendapatkan nilai akurasi yang lebih baik jika dibandingkan dengan metode klasifikasi Maximum Entropy. Sehingga penulis tertarik untuk mengukur bagaimana kinerja classifier SVM jika dibandingkan dengan metode klasifikasi yang lain selain Maximum Entropy untuk analisis sentimen pada ulasan tekstual berbahasa Indonesia Google Play Store. Dari beberapa penelitian yang lain sebelumnya tentang analisis sentimen [2][3][4][5][6][7][8], terdapat tiga metode klasifikasi yang selalu mendapatkan hasil akurasi tinggi adalah Support Vector Machine, Naïve Bayes, dan Maximum Entropy.

Metode klasifikasi *Naïve Bayes* merupakan metode dengan algoritma yang sederhana namun memiliki kecepatan dan akurasi yang tinggi [9]. Pada penelitian yang dilakukan oleh [10], *Naïve Bayes* dapat bekerja dengan baik bahkan dengan adanya kehadiran dari fitur yang memiliki dependensi yang kuat pada dataset. Data komentar yang diambil dari Google Play untuk dijadikan *data training* memiliki jumlah yang sedikit karena sulitnya mendapatkan komentar yang sesuai [11]. Dengan permasalahan ini, metode klasifikasi *Naïve Bayes* sangat sesuai apabila digunakan sebab *Naïve Bayes Classifier* masih mampu bekerja dengan baik dengan ukuran *data training* yang kecil [12][10]. Selain dari itu, algoritma *Naïve Bayes* yang sederhana dan kecepatannya yang tinggi dalam proses pelatihan dan klasifikasi [13] membuat algoritma ini menarik untuk digunakan sebagai salah satu metode klasifikasi. Dengan karakteristik tersebut di atas, maka metode klasifikasi *Naïve Bayes* sesuai jika digunakan pada penelitian ini sebagai metode yang akan dibandingkan kinerjanya dengan metode klasifikasi SVM.

II. Metode

Tahapan dalam penelitian dimulai dari analisis sistem. Berdasarkan *requirement* yang didapatkan pada analisis sistem dilakukan proses perancangan. Hasil perancangan kemudian masuk pada tahap implementasi, dan terakhir adalah proses pengujian.

A. Analisis Sistem

Sistem yang dibangun pada penelitian ini merupakan sistem yang mampu mengukur kinerja dari *Naïve Bayes classifier* untuk analisis sentimen pada ulasan tekstual berbahasa Indonesia pada Google Play Store. Kinerja dari *Naïve Bayes classifier* akan dibandingkan dengan kinerja dari *SVM classifier*. Proses analisis sentimen dimulai dengan membangun *data training*, kemudian *data training* ini diolah dengan algoritma klasifikasi tertentu sehingga menghasilkan sebuah model klasifikasi. Adapun model ini nantinya akan digunakan oleh *classifier* sebagai dasar untuk melakukan proses klasifikasi. Dalam membangun *data training*, data awal yang berupa komentar-komentar yang telah diberi label kelas melalui tahap *pre-processing*. Tahap *preprocessing* antara lain *case folding*, proses eliminasi *stopword* dan tanda baca, dan tag negasi untuk mengganti kalimat negasi. Adapun fitur yang digunakan adalah fitur unigram.

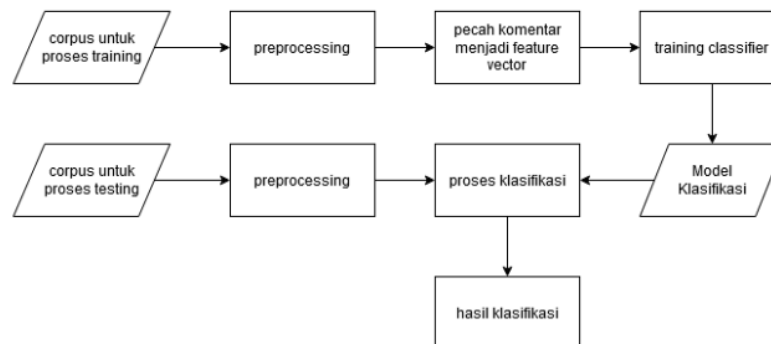
Proses evaluasi kinerja *classifier* menggunakan pendekatan *k-fold cross-validation*. Hasil klasifikasi yang diprediksi benar dan tidak benar oleh *classifier* ditabulasikan pada sebuah tabel *confusion matrix*. Informasi dalam *confusion matrix* diperlukan untuk menentukan kinerja model klasifikasi. Hasil dari *confusion matrix* digunakan untuk menentukan akurasi dari *classifier* dengan *performance metric accuracy*, untuk proses pengujian lebih detail akan dijelaskan pada sub bab berikutnya yaitu Sub Bab Pengujian Akurasi *Classifier*.

Adapun *requirement* dari hasil analisis sistem untuk sistem yang akan dibangun adalah sebagai berikut:

- Metode klasifikasi yang digunakan untuk proses analisis sentimen adalah *Naïve Bayes* dan SVM.
- Proses membangun *data training* terdiri dari *case folding*, *stopword and punctuation elimination*, dan *negation tag* dengan fitur unigram, tahapan ini disebut dengan tahap *preprocessing*.
- Kinerja *classifier* diukur dengan menghitung akurasi dari model klasifikasi yang dihasilkan menggunakan metode *k-fold cross validation*.

B. Perancangan Sistem

Sistem yang akan dibangun bertujuan mengukur kinerja dari *Naïve Bayes classifier* dan *SVM classifier* untuk analisis sentimen pada ulasan aplikasi Google Play Store. Analisis sentimen sendiri terdiri dari beberapa tahapan yakni *preprocessing*, *training data*, dan klasifikasi. Alur dari proses analisis sentimen dapat dilihat pada Gambar 1. Adapun pengujian nilai akurasi kemudian menggunakan pendekatan *k-fold cross validation*.



Gambar 1. Alur dari proses analisis sentimen

Data training yang akan diolah untuk menghasilkan model klasifikasi berasal dari *corpus* pada penelitian sebelumnya [1]. *Corpus* tersebut¹ sebelumnya telah diberikan label dalam 3 kelas, yaitu positif, negatif, dan crash.

Preprocessing

Sebelum disimpan sebagai *data training*, komentar-komentar tersebut terlebih dahulu melalui tahap *preprocessing*. Dalam melakukan *preprocessing* dilakukan tahapan berikut :

1. *Case folding*, yaitu penyeragaman bentuk huruf menjadi huruf kecil.
2. *Stopword and punctuation elimination*, yakni menghapus karakter-karakter yang termasuk dalam kategori *stopword* dan tanda baca. *Stopword* merupakan kata-kata yang memiliki frekuensi tinggi (seperti “atau”, “yang”, “meski”, dsb.) tapi tidak terlalu berdampak pada sentimen kalimatnya, sehingga dianggap sebagai *noise data* dan harus dihilangkan.
3. *Negation Tag*, yakni sebuah tag “NOT_” ditambahkan pada setiap kata di antara kata negasi dan kata yang mengikutinya, lalu menghapus kata negasinya. Misal terdapat potongan kalimat “tidak bagus” ini akan diubah menjadi “NOT_bagus”

Proses pemilihan fitur kemudian dilakukan setelah tahap *preprocessing*. Fitur yang digunakan adalah *unigram*, di mana setiap kata direpresentasikan sebagai *feature vector*. Setelah data melewati tahap *preprocessing* dan menjadi 24 fitur maka data tersebut akan siap disimpan sebagai *data training*. Hasil dari proses *training* adalah model klasifikasi yang akan digunakan oleh *classifier* untuk menentukan sentimen. Proses klasifikasi komentar yang juga merupakan penentuan sentimen dari aplikasi pada Google Play menggunakan metode klasifikasi *Naïve Bayes* dan akan dibandingkan dengan metode klasifikasi *Support Vector Machine* (SVM).

Perancangan Support Vector Machine Classifier

Proses penentuan sentimen atau klasifikasi dengan SVM menggunakan library LibSVM [14]. Dalam penerapannya, *library* ini melakukan proses klasifikasi dengan *binary classification*, sehingga komentar yang berupa teks dikonversi terlebih dahulu ke 20 her dengan mengasumsikan bahwa jika fitur tersebut terdapat dalam daftar fitur unik pada *data training*, maka akan bernilai 1 dan sebaliknya jika fitur tidak ada maka akan bernilai 0.

Perancangan Naïve Bayes Classifier

Proses *training classifier* menggunakan *data training* berupa komentar-komentar yang telah diberi label kelas dan telah melalui tahap *preprocessing* yang nantinya akan menghasilkan sebuah model klasifikasi. Tahapan dalam melakukan *training data* sebagai berikut :

1. Mencari jumlah keseluruhan komentar M pada *data training*. Misalkan data yang digunakan sebagai *data training* adalah data komentar yang terdapat pada Tabel 1. Maka jumlah keseluruhan komentar M = 3.

Tabel 1. Contoh *data training*

Komentar	Label
bagus NOT_rugi beli ni best	Positif
NOT_bisa_buka nyesel beli	Negatif
alur cerita NOT_jelas percuma beli	Crash

2. Mencari jumlah komentar pada masing-masing label kelas. Berdasarkan data komentar yang terdapat pada Tabel 1, maka jumlah komentar untuk kelas positif $N_{Positif} = 1$, kelas negatif $N_{Negatif} = 1$, dan kelas crash $N_{Crash} = 1$.
3. Menentukan nilai *prior probability* tiap kelasnya $P(label_kelas)$, sesuai dengan persamaan (1):

$$P(c_j) = \frac{N_{c_j}}{M} \quad (1)$$

Penjelasan dari persamaan (1) sebagai berikut :

- M : jumlah keseluruhan dokumen
- N_{c_j} : jumlah dokumen yang terdapat pada kelas c_j

Sesuai dengan persamaan (1), maka didapatkan hasil:

$$P(Positif) = \frac{N_{Positif}}{M} = \frac{1}{3} = 0,3333333333333333$$

¹ Corpus dapat diakses pada laman <https://github.com/luthfy13/Corpus-Ulasan-Tekstual-Google-Play-Store-untuk-Analisis-Sentimen>

$$P(Negatif) = \frac{N_{Negatif}}{M} = \frac{1}{3} = 0,3333333333333333$$

$$P(Crash) = \frac{N_{Crash}}{M} = \frac{1}{3} = 0,3333333333333333$$

4. Melakukan operasi logaritma pada nilai *prior probability* untuk tiap kelasnya. Hasil yang didapatkan adalah:

$$\log_2 P(Positif) = \log_2 0,3333333333333333 = -1.5849625007211563$$

$$\log_2 P(Negatif) = \log_2 0,3333333333333333 = -1.5849625007211563$$

$$\log_2 P(Crash) = \log_2 0,3333333333333333 = -1.5849625007211563$$

5. Proses selanjutnya yakni menghitung nilai jumlah kemunculan tiap fitur untuk tiap kelasnya dan jumlah total kemunculan fitur untuk tiap kelasnya. Fitur yang dimaksud adalah tiap kata yang terdapat 18 da komentar yang telah melalui tahap *preprocessing*. Berdasarkan Tabel 1, jumlah kemunculan fitur untuk tiap kelas dapat dilihat pada Tabel 2.

Tabel 2. Jumlah kemunculan fitur untuk tiap kelas

Fitur	Jumlah Kemunculan		
	Positif	Negatif	Crash
bagus	1	0	0
NOT_rugi	1	0	0
beli	1	1	1
ni	1	0	0
best	1	0	0
NOT_bisa_buka	0	0	1
nyesal	0	0	1
alur	0	1	0
cerita	0	1	0
NOT_jelas	0	1	0
percuma	0	1	0
Total	5	5	3

Dari Tabel di atas, dihasilkan sebuah *vocabulary* V yang berisi daftar fitur unik.

6. Mencari jumlah fitur unik B yang terdapat pada *vocabulary*. Berdasarkan Tabel 2, jumlah fitur unik $B = 11$.
7. Proses selanjutnya menentukan nilai probabilitas kondisional tiap fitur untuk tiap kelas $P(\text{fitur}|\text{kelas})$ berdasarkan persamaan (2).

$$P(x_i|c_j) = \frac{N_{x_i,c_j}+1}{\sum_{x \in V} N_{x,c_j}+1} = \frac{N_{x_i,c_j}+1}{(\sum_{x \in V} N_{x,c_j})+B} \quad (2)$$

Penjelasan dari persamaan di atas sebagai berikut:

- x_i : salah satu fitur dari dokumen d .
- c_j : salah satu set kelas pada label kelas C .
- N_{x_i,c_j} : jumlah kemunculan fitur x_i pada kelas c_j .
- V : *vocabulary* yang berisi daftar fitur unik.
- $\sum_{x \in V} N_{x,c_j}$: jumlah total kemunculan fitur tiap kelasnya.
- B : jumlah dari daftar fitur pada *vocabulary* V .

Sebagai contoh nilai probabilitas kondisional untuk fitur 'bagus' pada tiap kelasnya:

$$P(\text{bagus}|\text{positif}) = \frac{1+1}{5+11} = 0.125$$

$$P(\text{bagus}|\text{Negatif}) = \frac{0+1}{5+11} = 0.0625$$

$$P(\text{bagus}|\text{Crash}) = \frac{0+1}{3+11} = 0.07142857142857142$$

8. Proses selanjutnya melakukan operasi logaritma terhadap nilai probabilitas kondisional tiap fitur untuk tiap kelas yang telah didapatkan sebelumnya.

$$\log_2 P(\text{bagus}|\text{Positif}) = \log_2 0.125 = -3.0$$

$$\log_2 P(\text{bagus}|\text{Negatif}) = \log_2 0.0625 = -4.0$$

$$\log_2 P(\text{bagus}|\text{Crash}) = \log_2 0.07142857142857142 = -3.8073549220576046$$

Apabila proses perhitungan tersebut dilakukan pada setiap fitur untuk tiap kelas, maka diperoleh hasil seperti terlihat pada Tabel 3. Nilai-nilai dari *log prior probability* untuk tiap kelas dan nilai *log probabilitas kondisional* tiap fitur untuk tiap kelas inilah yang nantinya dijadikan sebagai model klasifikasi.

Tabel 3. Nilai log probabilitas kondisional fitur untuk tiap kelas

Fitur	log P(fitur kelas)		
	Positif	Negatif	Crash
bagus	-3.0	-4.0	-3.8073549220576046
NOT_rugi	-3.0	-4.0	-3.8073549220576046
beli	-3.0	-3.0	-2.8073549220576046
ni	-3.0	-4.0	-3.8073549220576046
best	-3.0	-4.0	-3.8073549220576046
NOT_bisa_buka	-4.0	-4.0	-2.8073549220576046
nyesel	-4.0	-4.0	-2.8073549220576046
Alur	-4.0	-3.0	-3.8073549220576046
Cerita	-4.0	-3.0	-3.8073549220576046
NOT_jelas	-4.0	-3.0	-3.8073549220576046
Percuma	-4.0	-3.0	-3.8073549220576046
Log P(kelas)	-1.584962500721	-1.584962500721	-1.584962500721

Setelah pembentukan model klasifikasi selesai dilakukan, maka proses klasifikasi siap untuk dilakukan. Misalkan sebuah komentar yang berisi kalimat '*percuma, aplikasi ini gak jelas, nyesel*' yang akan ditentukan sentimennya. Maka tahap untuk melakukan proses klasifikasi pada kalimat tersebut adalah:

1. Komentar melalui tahap *preprocessing*, hasil *preprocessing* berdasarkan kalimat '*percuma aplikasi ini saya beli, nyesel*' adalah '*percuma NOT_jelas nyesel*'.
2. Menentukan nilai log probabilitas kondisional tiap fitur yang terdapat pada komentar hasil *preprocessing* '*percuma NOT_jelas nyesel*' dengan mengambil nilai log probabilitas kondisional tiap fitur untuk tiap kelas yang terdapat pada model klasifikasi. Hasilnya dapat dilihat pada Tabel 4.

Tabel 4. Nilai log probabilitas kondisional tiap fitur

Fitur	log P(fitur kelas)		
	Positif	Negatif	Crash
percuma	-4.0	-3.0	-3.8073549220576046
NOT_jelas	-4.0	-3.0	-3.8073549220576046
nyesel	-4.0	-4.0	-2.8073549220576046

3. Mengambil nilai log prior probability tiap kelas log P(kelas) pada model klasifikasi.
4. Penentuan kelas sentimen berdasarkan persamaan (3):

$$\begin{aligned}
 c_{\text{map}} &= \arg \max_{c_j \in C} \log(P(c_j) \prod_{i=1}^m P(x_i | c_j)) \\
 &= \arg \max_{c_j \in C} \log P(c_j) + \sum_{i=1}^m P(x_i | c_j)
 \end{aligned} \quad (3)$$

Dengan mencari nilai maksimum dari nilai probabilitas akhir tiap kelas $P(\text{kelas} | \text{komentar})$.

$$P(\text{Positif} | \text{'percuma NOT_jelas nyesel'}) = -1.5849625007211563 + -4.0 + -4.0 + -4.0$$

$$= -13.584962500721156$$

$$P(\text{Negatif} | \text{'percuma NOT_jelas nyesel'}) = -1.5849625007211563 + -3.0 + -3.0 + -4.0$$

$$= -11.584962500721156$$

$$P(\text{Crash} | \text{'percuma NOT_jelas nyesel'}) = -1.32192809488736 + -3.8073549220576046$$

$$-3.8073549220576046 + -2.8073549220576046$$

$$= -12.00702726689397$$

$$c_{\text{map}} = \max(P(\text{Positif} | \text{'percuma NOT_jelas nyesel'}), P(\text{Negatif} | \text{'percuma NOT_jelas nyesel'}), P(\text{Crash} | \text{'percuma NOT_jelas nyesel'}))$$

$$= \max(-13.584962500721156, -11.584962500721156, -12.00702726689397)$$

$$= -11.584962500721156$$

1. Komentar '*percuma, aplikasi ini gak jelas, nyesel*' diklasifikasikan dalam sentimen kelas negatif.

Perancangan Proses Pengujian Akurasi Classifier

Pengujian untuk evaluasi *classifier* menggunakan metode *k-fold cross-validation* dengan nilai $k=3$. Data untuk pengujian diambil dari data awal yang digunakan sebagai *data training*. Jumlah data yang diambil, sama untuk setiap kelas sentimennya. Data kemudian dibagi menjadi tiga partisi yang terdiri dari komentar-komentar yang jumlahnya sama dari setiap label kelas. Salah satu partisi digunakan untuk proses *testing* sementara dua partisi lainnya digunakan untuk proses *training*. Prosedur ini diulangi sebanyak 3 kali

sedemikian sehingga setiap partisi digunakan sebagai *data testing* tepat satu kali. Total error ditentukan dengan menjumlahkan error yang didapatkan selama proses evaluasi. Hasil evaluasi kemudian ditabulasikan pada sebuah *confusion matrix*, selanjutnya menghitung akurasi *classifier* dengan menggunakan persamaan (4).

$$\text{Akurasi} = \frac{\text{Jumlah klasifikasi benar}}{\text{Jumlah Data Uji}} \times 100\% \quad (4)$$

C. Implementasi

Pada penelitian ini, pembuatan aplikasi diimplementasikan dengan bahasa pemrograman python. Seluruh proses mulai dari *preprocessing* sampai pada pengujian akurasi menggunakan bahasa pemrograman python. Proses klasifikasi dengan metode *Naïve Bayes* diimplementasikan pada python tanpa bantuan *library* tambahan. Sedangkan untuk pengimplementasian metode klasifikasi *SVM* menggunakan *library LibSVM* [14] yang digunakan pada penelitian sebelumnya [1].

III. Hasil dan Pembahasan

Tahap ini merupakan tahap pengujian sistem untuk mengetahui performansi dari *classifier* telah dibangun, parameter yang digunakan untuk mengukur performansi dari *classifier* pada penelitian ini adalah akurasi. Percobaan untuk proses evaluasi *classifier* menggunakan metode *k-fold cross-validation* dengan nilai $k = 3$. Sesuai dengan penelitian sebelumnya [15][7] yang menggunakan *3-folds cross-validation* karena dataset yang digunakan cukup besar yakni 1818 data sehingga proses *training classifier* membutuhkan waktu yang lama. Hasil klasifikasi yang diprediksi benar dan tidak benar oleh model klasifikasi kemudian ditabulasikan pada tabel *confusion matrix*.

A. Data

Data yang digunakan untuk proses pengujian akurasi *classifier* diambil dari *corpus* penelitian sebelumnya dengan objek yang sama, yaitu ulasan tekstual berbahasa Indonesia pada Google Play Store [1] dengan total 1818 komentar yang terdiri dari 606 komentar dengan sentimen positif, 606 komentar untuk sentimen negatif, dan 606 komentar untuk sentimen *crash*.

B. Evaluasi dan Pengukuran Akurasi Classifier

Evaluasi *classifier* menggunakan metode *3-folds cross-validation*. Selama proses pengujian digunakan 3 partisi data yang berukuran sama, salah satu dari partisi dipilih untuk testing, sedangkan dua partisi lainnya digunakan untuk training. Prosedur ini diulangi sebanyak 3 kali sedemikian sehingga setiap partisi digunakan sebagai *data testing* tepat satu kali. Total error ditentukan dengan menjumlahkan error yang didapatkan selama proses pengujian.

Data untuk pengujian diambil dari *corpus* dengan membagi *corpus* tersebut ke dalam 3 partisi. Dalam 1 partisi, terdapat 202 komentar dengan label kelas positif, 202 komentar dengan label kelas negatif, dan 202 komentar dengan label kelas *crash*. Jadi, total data dari tiap partisi adalah 606 data. Hasil akurasi *classifier* dihitung berdasarkan persamaan (4). Data set inilah yang digunakan untuk pengujian akurasi dari *Naïve Bayes classifier* dan *SVM classifier*.

Gambar 2.a menunjukkan output dari proses evaluasi *Naïve Bayes classifier* menggunakan *3-folds cross-validation*. Pada iterasi pertama, *data training* berasal dari gabungan partisi pertama dan partisi kedua, dan partisi ketiga digunakan sebagai *data testing*. Di sini terlihat bahwa dari 606 *data testing*, terdapat 476 data yang diklasifikasikan secara benar yang terdiri dari 167 komentar positif, 120 komentar negatif, dan 189 komentar *crash*. Berdasarkan persamaan (4), hasil akurasi pada iterasi pertama adalah 78,55%. Selanjutnya pada iterasi kedua dan iterasi ketiga, masing-masing mendapatkan nilai akurasi 75,91% dan 71,91%. Sehingga, jika dirata-ratakan hasil akurasi *Naïve Bayes classifier* adalah 75,41%.

```

C:\Program Files\Java\jdk-9.0.4\bin>java -jar NaiveBayesClassifier.jar
Evaluasi Naive Bayes Classifier: K-fold Cross-validation
=====
Jumlah keseluruhan data = 1008
Nilai K = 3
Jumlah partisi = 3
Jumlah data tiap partisi = 666

Iterasi I:
Partisi Training = Partisi 1 + Partisi 2
Partisi Testing = Partisi 3
=====
Response Positif | Response Negatif | Response Crash |
-----
Reference Positif | TP = 167 | FN = 21 | FC = 14 |
Reference Negatif | FP = 48 | TN = 120 | FC = 14 |
Reference Crash | PP = 8 | FN = 5 | TC = 189 |

Jumlah data benar = 476 dari 666 data
Akurasi = 71,55%
Iterasi pertama selesai...

Iterasi II:
Partisi Training = Partisi 1 + Partisi 3
Partisi Testing = Partisi 2
=====
Response Positif | Response Negatif | Response Crash |
-----
Reference Positif | TP = 172 | FN = 27 | FC = 4 |
Reference Negatif | FP = 52 | TN = 119 | FC = 42 |
Reference Crash | PP = 7 | FN = 16 | TC = 179 |

Jumlah data benar = 469 dari 666 data
Akurasi = 70,59%
Iterasi kedua selesai...

Iterasi III:
Partisi Training = Partisi 2 + Partisi 3
Partisi Testing = Partisi 1
=====
Response Positif | Response Negatif | Response Crash |
-----
Reference Positif | TP = 145 | FN = 36 | FC = 21 |
Reference Negatif | FP = 27 | TN = 96 | FC = 79 |
Reference Crash | PP = 4 | FN = 4 | TC = 184 |

Jumlah data benar = 435 dari 666 data
Akurasi = 71,78%
Iterasi ketiga selesai...

Akurasi Classifier = 71,40%
Evaluasi Classifier selesai...

```

Gambar 2.a. Output Evaluasi Naive Bayes

```

C:\Program Files\Java\jdk-9.0.4\bin>java -jar SVMClassifier.jar
Evaluasi Support Vector Machine Classifier: K-fold Cross-validation
=====
Jumlah keseluruhan data = 1008
Nilai K = 3
Jumlah partisi = 3
Jumlah data tiap partisi = 666

Iterasi I:
Partisi Training = Partisi 1 + Partisi 2
Partisi Testing = Partisi 3
=====
Response Positif | Response Negatif | Response Crash |
-----
Reference Positif | TP = 151 | FN = 15 | FC = 9 |
Reference Negatif | FP = 38 | TN = 161 | FC = 3 |
Reference Crash | PP = 9 | FN = 11 | TC = 182 |

Jumlah data benar = 404 dari 666 data
Akurasi = 61,52%
Iterasi pertama selesai...

Iterasi II:
Partisi Training = Partisi 1 + Partisi 3
Partisi Testing = Partisi 2
=====
Response Positif | Response Negatif | Response Crash |
-----
Reference Positif | TP = 151 | FN = 15 | FC = 9 |
Reference Negatif | FP = 39 | TN = 163 | FC = 9 |
Reference Crash | PP = 5 | FN = 19 | TC = 179 |

Jumlah data benar = 401 dari 666 data
Akurasi = 61,39%
Iterasi kedua selesai...

Iterasi III:
Partisi Training = Partisi 2 + Partisi 3
Partisi Testing = Partisi 1
=====
Response Positif | Response Negatif | Response Crash |
-----
Reference Positif | TP = 134 | FN = 66 | FC = 2 |
Reference Negatif | FP = 25 | TN = 171 | FC = 6 |
Reference Crash | PP = 1 | FN = 12 | TC = 189 |

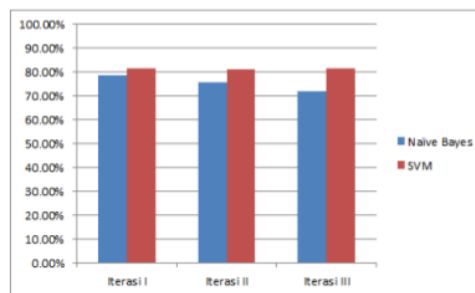
Jumlah data benar = 404 dari 666 data
Akurasi = 61,52%
Iterasi ketiga selesai...

Akurasi Classifier = 61,46%
Evaluasi Classifier selesai...

```

Gambar 2.b. Output Evaluasi SVM

Adapun Gambar 2.b menunjukkan output dari proses evaluasi dari *SVM Classifier*. Nilai akurasi yang dihasilkan pada *classifier* ini ternyata lebih tinggi, jika dibandingkan dengan akurasi *classifier* sebelumnya. *SVM classifier* berhasil mendapatkan akurasi sebesar 81,46%.



Gambar 3. Grafik hasil akurasi classifier per iterasi

Perbandingan hasil akurasi dalam tiap iterasi antara *Naive Bayes classifier* dan *SVM classifier* tampak pada grafik pada Gambar 3. Pada grafik tersebut terlihat bahwa dalam tiap iterasi dari 3 iterasi, *SVM classifier* selalu memiliki akurasi yang lebih tinggi. Sementara *Naive Bayes classifier* tidak pernah mendapatkan nilai akurasi yang lebih tinggi dari akurasi *SVM classifier*.

IV. Kesimpulan

Berdasarkan dari hasil penelitian dan pembahasan yang dilakukan maka diperoleh kesimpulan bahwa Hasil akurasi dari *Support Vector Machine Classifier* memiliki nilai yang lebih tinggi jika dibandingkan dengan akurasi dari *Naive Bayes classifier* untuk mengklasifikasikan ulasan tekstual berbahasa Indonesia pada Google Play Store, yakni *SVM classifier* mendapatkan akurasi sebesar 81,46% dan *Naive Bayes classifier* sebesar 75,41%, sehingga metode *SVM* lebih baik untuk dijadikan metode klasifikasi untuk proses Analisis Sentimen ulasan tekstual berbahasa Indonesia pada Google Play Store.

Daftar Pustaka

- [1] L. B. Ilmawan and A. Mude, "Analisis Sentimen untuk Text Review Berbahasa Indonesia pada Google Play Store Menggunakan Metode Maximum Entropy," Makassar, 2019.
- [2] E. Boiy, P. Hens, K. Deschacht, and M. Moens, "Automatic Sentiment Analysis in On-line Text Concepts of Emotions in Written Text Concept of Emotions," in *Electronic Publishing*, 2007, no. June, pp. 349–360.
- [3] A. Go, R. Bhayani, and L. Huang, "Twitter Sentiment Classification using Distant Supervision," *Processing*, vol. 150, no. 12, pp. 1–6, 2009.
- [4] L. Zhang, R. Ghosh, M. Dekhil, M. Hsu, and B. Liu, "Combining lexicon-based and learning-based methods for Twitter sentiment analysis," *International Journal of Electronics, Communication and*

- Soft Computing Science & Engineering (IJECSCE), vol. 89, pp. 1–8, 2015.
- [5] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-Based Methods for Sentiment Analysis," *Computational Linguistics*, vol. 37, no. 2, pp. 267–307, 2011.
- [6] P. D. Turney, "Thumbs up or thumbs down? Semantic Orientation applied to Unsupervised Classification of Reviews," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002, no. July, pp. 417–424.
- [7] M. Y. Nur and Diaz D. Santika, "Analisis Sentimen pada Dokumen berbahasa Indonesia dengan pendekatan Support Vector Machine," in *Konferensi Nasional Sistem dan Informatika*, 2011, pp. 9–14.
- [8] P. Aliandu, "Analisis Sentimen Tweet Berbahasa Indonesia di Twitter," Universitas Gadjah Mada, 2012.
- [9] I. Rish, "An Empirical Study of the Naïve Bayes Classifier," *IJCAI 2001 Work Empir Methods Artif Intell*, vol. 3, 2001.
- [10] P. Domingos and M. Pazzani, "On the Optimality of the Simple Bayesian Classifier under Zero-One Loss," *Machine Learning*, vol. 29, pp. 103–130, 1997.
- [11] L. B. Ilmawan, "Aplikasi Mobile untuk Analisis Sentimen pada Google Play," Universitas Gadjah Mada, 2014.
- [12] R. Kohavi, "Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid," *KDD*, 1997.
- [13] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *Informatica*, vol. 31, pp. 249–268, 2007.
- [14] C.-C. Chang and C.-J. Lin, "{LIBSVM}: A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1--27:27, 2011.
- [15] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2002, pp. 79–86.

Perbandingan Metode Klasifikasi Support Vector Machine dan Naïve Bayes untuk Analisis Sentimen pada Ulasan Tekstual di Google Play Store

ORIGINALITY REPORT

23%

SIMILARITY INDEX

PRIMARY SOURCES

- | | | |
|---|---|-----------------|
| 1 | jurnal.ugm.ac.id
Internet | 473 words — 13% |
| 2 | jurnal.fikom.umi.ac.id
Internet | 45 words — 1% |
| 3 | Tri Duc Nguyen, Linh Diep-Phuong Nguyen, Tru Cao. "Sentiment analysis on medical text using combination of machine learning and SO-CAL scoring", 2017 21st Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES), 2017
Crossref | 28 words — 1% |
| 4 | Wei-Liang Tai. "Blood Cell Image Classification Based on Hierarchical SVM", 2011 IEEE International Symposium on Multimedia, 12/2011
Crossref | 26 words — 1% |
| 5 | Abd. Samad Hasan Basari, Burairah Hussin, I. Gede Pramudya Ananta, Junta Zeniarja. "Opinion Mining of Movie Review using Hybrid Method of Support Vector Machine and Particle Swarm Optimization", Procedia Engineering, 2013
Crossref | 23 words — 1% |
| 6 | digitalcommons.wayne.edu
Internet | 21 words — 1% |
| 7 | e-journal.uajy.ac.id
Internet | 19 words — 1% |
| 8 | www.slideshare.net | |

18 words — $< 1\%$

- 9 Suherman Suherman, Sunny Samsuni, Imam Lukman Hakim. "Sistem Rekomendasi Wisata Pantai menggunakan Metode Simple Additive Weighting", ILKOM Jurnal Ilmiah, 2020

Crossref

16 words — $< 1\%$

- 10 Ika Kurniawati, Hilman F. Pardede. "Hybrid Method of Information Gain and Particle Swarm Optimization for Selection of Features of SVM-Based Sentiment Analysis", 2018 International Conference on Information Technology Systems and Innovation (ICITSI), 2018

Crossref

16 words — $< 1\%$

- 11 www.scribd.com

Internet

16 words — $< 1\%$

- 12 repository.ipb.ac.id

Internet

16 words — $< 1\%$

- 13 tel.archives-ouvertes.fr

Internet

14 words — $< 1\%$

- 14 jurnal.fs.umi.ac.id

Internet

13 words — $< 1\%$

- 15 Sujata Rani, Parteek Kumar. "Rule based sentiment analysis system for analyzing tweets", 2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS), 2017

Crossref

12 words — $< 1\%$

- 16 Yu. A. Rozanov. "Markov Random Fields", Springer Science and Business Media LLC, 1982

Crossref

12 words — $< 1\%$

- 17 Md. Ferdous Wahid, Md. Jahid Hasan, Md. Shahin Alom, Shamim Mahbub. "Performance Analysis of

11 words — $< 1\%$

-
- 18 fr.scribd.com 11 words — < 1%
Internet
-
- 19 Sri Ratna Wulan, Suhono Harso Supangkat. "Semi-supervised learning self-training for Indonesian motivational messages classification", 2017 International Conference on ICT For Smart Society (ICISS), 2017 10 words — < 1%
Crossref
-
- 20 digilib.uinsby.ac.id 10 words — < 1%
Internet
-
- 21 Rizauddin Saian, Ku Ruhana Ku-Mahamud. "Hybrid Ant Colony Optimization and Simulated Annealing for Rule Induction", 2011 UKSim 5th European Symposium on Computer Modeling and Simulation, 2011 9 words — < 1%
Crossref
-
- 22 ejurnal.its.ac.id 9 words — < 1%
Internet
-
- 23 id.123dok.com 9 words — < 1%
Internet
-
- 24 text-id.123dok.com 8 words — < 1%
Internet
-
- 25 Paulina Aliandu. "Sentiment Analysis to Determine Accommodation, Shopping and Culinary Location on Foursquare in Kupang City", Procedia Computer Science, 2015 4 words — < 1%
Crossref

