ILKOM Jurnal Ilmiah
Vol. 12 No. 2, Agustus 2020, pp.154-161
E-ISSN 2548-7779

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

Lutfi Budi Ilmawan[a,1,*] dan Muhammad 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] *aliyazid.mude@umi.ac.id*
**corresponding author*

INFORMASI ARTIKEL

Dikirim : 29 Juni 2020
Diulas : 19 Juli 2020
Direvisi : 25 Juli 2020
Diterbitkan : 27 Agustus 2020

Kata Kunci:
Klasifikasi
Analisis Sentimen
Support Vector Machine
Naïve Bayes
Cross-validation

ABSTRAK

Pada penelitian ini, metode klasifikasi *SVM* akan dibandingkan kinerjanya denga 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 diaplikasikan untuk kalsifikasi 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%.

Keywords:
Classification
Sentiment Analysis
Support Vector Machine
Naïve Bayes
Cross-validation

ABSTRACT

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. 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 algortima 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 mengatasi kalimat negasi. Adapun fitur yang digunakan adalah fitur unigram.

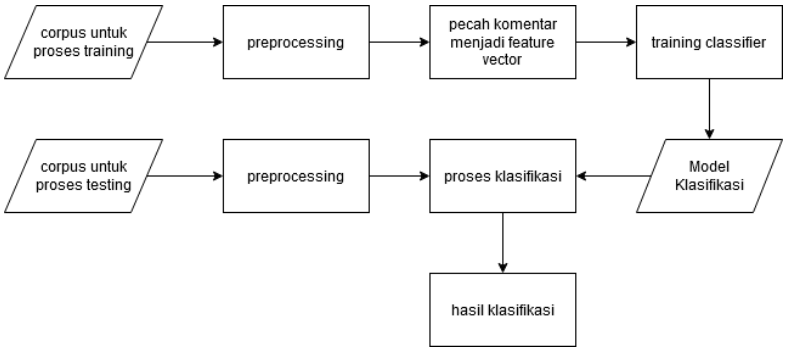
Proses evaluasi kinerja *classifier* menggunakan pendekatan *k-fold cross-validation*. Hasil klasifikasi yang diprediksi benar dan tidak benar oleh *classifier* ditabulasikan pada sebuah tabel *confusion matrix*. Informasi dalam *confusion matrix* diperlukan untuk menentukan kinerja model klasifikasi. 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 tersebut[1] sebelumnya telah diberikan label dalam 3 kelas, yaitu postif, negatif, dan *crash.*

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

1) Preprocessing

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

a) *Case folding*, yaitu penyeragaman bentuk huruf menjadi huruf kecil.

b) *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.

c) *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 pemilihan 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*).

2)* 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 biner 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.

3)* 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 :

a) 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

b) 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.

c) 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_{cj} : jumlah dokumen yang terdapa pada kelas c_j

Sesuai dengan persamaan (1), maka didapatkan hasil:

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

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

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

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

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

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

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

e) 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 pada 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	***Negaif***	***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
nyesel	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.

f) Mencari jumlah fitur unik *B* yang terdapat pada *vocabulary*. Berdasarkan Tabel 2, jumlah fitur unik *B* = 11.

g) Proses selanjutnya menentukan nilai probabilitas kondisional tiap fitur untuk tiap kelas *P*(*fitur*|*kelas*) berdasarkan persamaan (2).

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

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:

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

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

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

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

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

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

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

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	***Negaif***	***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	***Negaif***	***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):

$$c_{map} = arg \max_{c_j \in C} \log\left(P(c_j) \prod_{i=1}^{m} P(x_i|c_j)\right)$$

$$= arg \max_{c_j \in C} \log P(c_j) + \sum_{i=1}^{m} P(x_i|c_j) \quad (3)$$

Dengan mencari nilai maksimum dari nilai probabilitas akhir tiap kelas *P*(*kelas*|*komentar*).

P(*Positif*|*'percuma NOT_jelas nyesel'*) = -1.5849625007211563 + -4.0 + -4.0 + -4.0
= -13.584962500721156

P(*Negatif*|*'percuma NOT_jelas nyesel'*) = -1.5849625007211563 + -3.0 + -3.0 + -4.0
= -11.584962500721156

P(*Crash*|*'percuma NOT_jelas nyesel'*) = -1.32192809488736 + -3.8073549220576046
-3.8073549220576046 + -2.8073549220576046
= -12.00702726689397

c_{map} = max(*P*(*Positif*|*'percuma NOT_jelas nyesel'*), *P*(*Negatif*|*'percuma NOT_jelas nyesel'*), *P*(*Crash*|*'percuma NOT_jelas nyesel'*))
= max(-13.584962500721156, -11.584962500721156, -12.00702726689397)
= -11.584962500721156

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

4) 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).

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

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* dimplementasikan 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*.

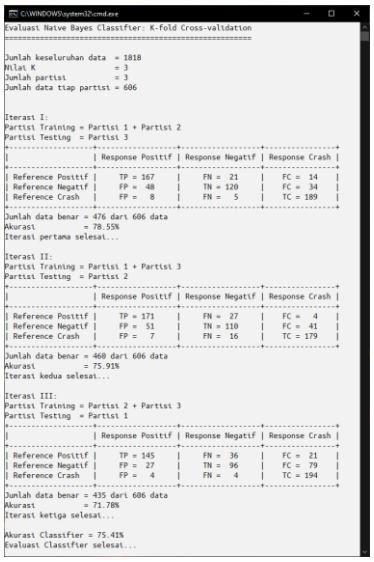
A. Data

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.*

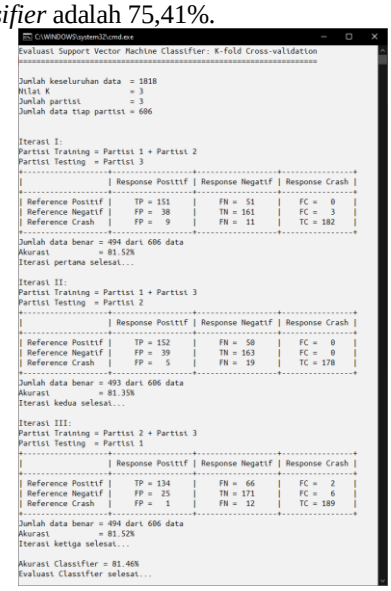
B. Evaluasi dan Pengukuran Akurasi Classifier

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%.

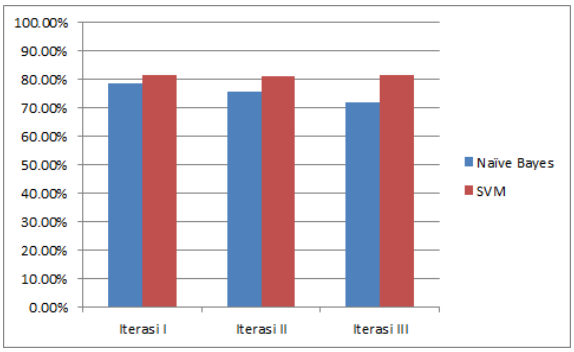


Gambar 2.a. Output Evaluasi *Naïve Bayes*



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* sebelumya. *SVM classifier* berhasil mendapatkan akurasi sebesar 81,46%.



Gambar 3. Grafik hasil akurasi *classifier* per iterasi

Perbandingan hasil akurasi dalam tiap iterasi antara *Naïve 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 *Naïve 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 *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%, 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 (IJECSCSE)*, 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.