LAPORAN AKHIR PROYEK

Sentiment Analysis of Song Lyrics using SVM *Algorithm*

Disusun oleh:

1. 12S17007 – Ernike Nelsi Manurung
2. 12S17021 – Inggrit Syafitri Purba
3. 12S17024 – Yohana Veronika Aritonang

11S4037 – PEMROSESAN BAHASA ALAMI

FAKULTAS INFORMATIKA DAN TEKNIK ELEKTRO

INSTITUT TEKNOLOGI DEL

2020

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

Bagian ini menyajikan latar belakang, tujuan, manfaat, dan ruang lingkup pengerjaan proyek.

1.1 Latar Belakang

Musik merupakan suatu hiburan, seni ataupun aktivitas yang dilakukan dan didengarkan oleh manusia dengan perlibatan suara yang teratur. Proses musik memunculkan suatu emosi yang memiliki keterkaitan dengan otak manusia. Musik dapat digunakan untuk menggambaran ekspresi atau perasaan. Contohnya, orang mendengar musik untuk melepaskan emosi terhadap perasaannya sekarang, melepaskan stress, menghibur diri, dan pengambaran ekspresi yang lainnya (Sakti, 2010). Lirik lagu adalah bagian penting untuk menggambarkan suatu perasaan atau emosi seseorang yang dapat digunakan dalam penentuan emosi. Lirik lagu merupakan suatu penyampaian perasaan seseorang secara tidak langsung terhadap apa yang dialami, dirasakan, dilihat, ataupun didengarnya [1].

Menentukan emosi pada suatu lagu biasanya dilakukan secara manual oleh manusia, misalnya seseorang sedang mendengarkan lagu yang mempunyai tempo yang ceria sehingga dia dapat menyimpulkan emosi yang disampaikan dari lagu tersebut adalah emosi perasaan senang, akan tetapi isi dari lagu tersebut belum tentu menggambarkan perasaan senang. Keterbatasan tersebut mendorong diciptakannya klasifikasi lirik lagu agar lebih efektif jika seseorang yang sedang ingin mendengarkan lagu sesuai dengan emosi yang diinginkan. Semakin banyaknya lagu, semakin banyak jenis emosi yang muncul maka diperlukan pengklasifikasian secara otomatis untuk menyelesaikan masalah tersebut. Pengklasifikasian emosi pada lirik lagu dilakukan berdasarkan fitur-fitur yang terdapat di dalam lagu tersebut. Fitur merupakan suatu ciri data dalam suatu objek yang membedakannya terhadap objek yang lain dan dapat digunakan dalam pengklasifikasian lirik lagu yaitu terdapat pada kata-kata yang sering muncul maupun yang jarang muncul di dalam lirik lagu tersebut [1].

Hal yang penting dalam menganalisis lirik lagu dan mengklasifikasikannya ke dalam emosi yang terdapat pada lirik lagu tersebut adalah dengan melakukan analisis sentimen. Analisis sentimen berkaitan dengan bidang yang lebih luas seperti pengolahan bahasa alami, komputasi linguistik, dan *text mining* untuk tujuan menganalisis, *sentiment*, pendapat, sikap, evaluasi, penilaian dan emosi seseorang berkenan pada topik, produk, layanan, organisasi, atau kegiatan tertentu. Sentimen adalah perasaan, sikap, evaluasi, atau emosi yang terkait dengan suatu opini

[2]. Dalam melakukan analisis sentimen mengklasifikasikan lirik lagu, diperlukan suatu metode atau algoritma klasifikasi. Klasifikasi adalah bentuk dasar dari analisis data. Konsep dasar dari klasifikasi adalah sejumlah data yang mempunyai struktur data yang hampir sama atau serupa akan menghasilkan klasifikasi yang hampir sama atau serupa juga. Klasifikasi sendiri merupakan suatu proses untuk memprediksi suatu objek yang akan diteliti.

Dalam penelitian yang dilakukan oleh Fadholi Fat Haranto dan Bety Wulan Sari yang berjudul "Implementasi *Support Vector Machine* Untuk Analisis Sentimen Pengguna Twitter Terhadap Pelayanan Telkom Dan Biznet" pada tahun 2019, bahwa penelitian sentimen pengguna twitter terhadap Biznet dan Telkom menggunakan algoritma *Support Vector Machine* menghasilkan rata-rata nilai *accuracy* 79,6%, *precision* 76,5%, *recall* 72,8%, dan f1-*score* sebesar 74,6% hasil dari Telkom, dan nilai *accuracy* 83,2%, *precision* 78,8%, *recall* 71,6%, dan f1-*score* 75% hasil dari Biznet [3]. Penelitian lainnya yang dilakukan oleh Rian Tineges, dkk.(2020) yaitu "Analisis Sentimen terhadap Layanan Indihome Berdasarkan Twitter Dengan Metode Klasifikasi *Support Vector Machine (*SVM)" yang menerapkan metode SVM pada analisis sentimen serta mengetahui seberapa puas pelanggan layanan Indihome berdasarkan Twitter dengan hasil *accuracy* 87%, *precision* 86%, *recall* 95%, dan f1-*score* sebesar 90% [4]. Selain itu berdasarkan perbandingan metode yang digunakan dalam analisis sentimen yang dilakukan oleh Elly Indrayuni(2019) yaitu "Komparasi Algoritma *Naive Bayes* dan *Support Vector Machine* Untuk Analisa Sentimen *Review* Film" menunjukkan nilai akurasi untuk algoritma *Naive Bayes* sebesar 84.50%. Sedangkan nilai akurasi algoritma *Support Vector Machine* (SVM) lebih besar dari *Naive Bayes* yaitu sebesar 90.00% [5].

Berdasarkan permasalahan dari uraian sebelumnya tim proyek memutuskan mengangkat topik judul proyek yaitu "*Sentiment Analysis of Song Lyrics using SVM Algorithm".* Proyek ini diharapkan dapat mengklasifikasikan lirik lagu ke dalam sentimen yang mengekspresikan rasa senang (*happy*) atau sedih (*sad*) yang terdapat pada lirik lagu.

1.2 Tujuan

Adapun tujuan dari pengerjaan proyek ini adalah:

1. Menerapkan metode *Support Vector Machine* (SVM) dalam menganalisis lirik lagu apakah lirik lagu tersebut termasuk ke dalam lirik lagu dengan label *happy* atau *sad.*
2. Untuk mengetahui bagaimana tingkat akurasi menggunakan metode *Support Vector Machine* (SVM) dalam melakukan analisis sentimen lirik lagu.

1.3 Manfaat

Manfaat dari pengerjaan proyek ini yaitu:

1. Bagi Mahasiswa

   Proyek ini diharapkan nantinya dapat memberikan wawasan serta pengetahuan bagi tim proyek dalam menerapkan aplikasi pemrosesan bahasa alami khususnya analisis sentiment pada lirik lagu.

2. Bagi Penikmat Musik

   Proyek ini diharapkan dapat digunakan sebagai acuan atau masukan bagi penikmat musik untuk lebih dalam memahami lagu yang mengekspresikan rasa senang (*happy*) dan sedih (*sad*) yang terdapat pada lirik lagu serta agar penikmat musik dapat mengetahui lagu mana yang yang mungkin ingin mereka dengarkan untuk mengekspresikan apa yang sedang mereka rasakan .

1.4 Ruang Lingkup

Ruang lingkup dalam pengerjaan proyek ini adalah menggunakan metode *Support Vector Machine* (SVM) dengan data sampel yang diperoleh dari github.com. Dataset yang digunakan adalah lirik lagu dalam bahasa Inggris yang telah diberikan label *happy* dan *sad.*

BAB 2
ISI

Pada bab ini dijelaskan analisis yang dilakukan terhadap data dan metode, desain pemrosesan bahasa alami yang ditampilkan dalam bentuk *flowchart* atau diagram alir, implementasi berupa kode program dan cuplikan hasil, dan hasil evaluasi kuantitatif terhadap implementasi pemrosesan bahasa alami yang dilakukan.

2.1 Analisis

Pada subbab ini dijelaskan analisis yang dilakukan terhadap data dan metode yang digunakan dalam pengimplementasian pemrosesan bahasa alami.

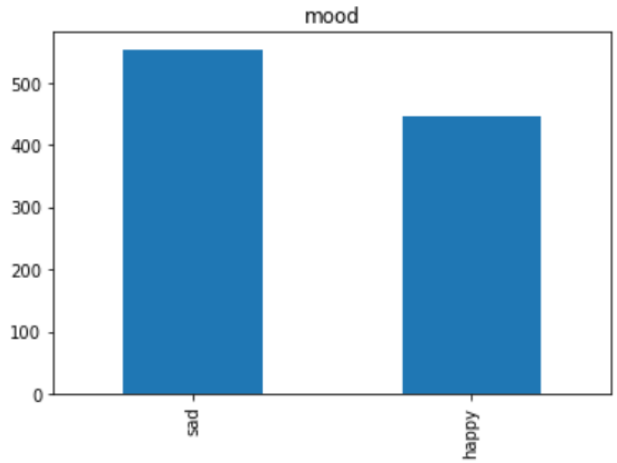
2.1.1 Analisis Data

Dataset yang digunakan dalam proyek ini diperoleh dari link berikut: https://github.com/rasbt/musicmood/tree/master/dataset. *Dataset* yang digunakan adalah train_lyrics_1000.csv sebagai data train dan valid_lyrics_200.csv sebagai data test, dimana data *train* terdiri dari 1000 *records* dan data *test* terdiri dari 200 *records*. *Dataset* tersebut telah diberikan kelas/label yaitu *happy* atau *sad* pada setiap lagu dalam data tersebut. Masing-masing data *train* dan data *test* memiliki 7 atribut yaitu *file, artist, title, lyrics, genre, mood*, dan *year*.

Tabel 1. Atribut pada *Dataset*

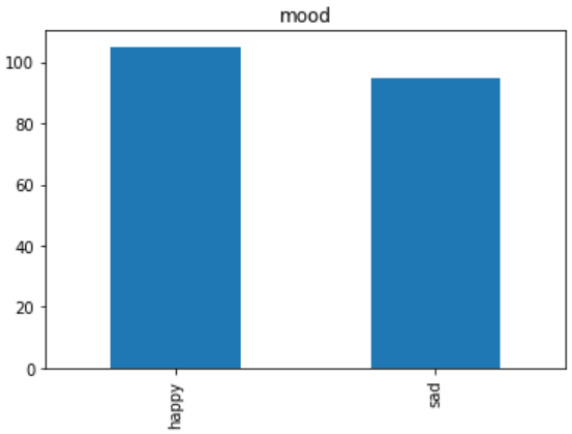
No.	Nama atribut	Tipe Atribut	Deskripsi
1.	*file*	Nominal	Nama file dari setiap lagu dan bersifat unik.
2.	*artist*	Nominal	Nama artis yang menyanyikan lagu tersebut.
3.	*title*	Nominal	Judul lagu.
4.	*lyrics*	Nominal	Lirik dari lagu tersebut.
5.	*genre*	Nominal	Pengelompokkan musik sesuai dengan kemiripannya satu sama lain.
6.	*mood*	Nominal	Emosi yang terdapat pada lagu tersebut, yaitu *happy* dan *sad*.
7.	*year*	Nominal	Tahun lagu tersebut di-*release*.

Berikut adalah distribusi *mood* lirik lagu pada *train data* ditunjukkan pada Gambar 1. Dapat dilihat bahwa *mood sad* (554) lebih banyak daripada *mood happy* (446).



Gambar 1. Distribusi Mood Lirik Lagu pada *Train Data*

Berikut adalah distribusi *mood* lirik lagu pada *test data* ditunjukkan pada Gambar . Dapat dilihat bahwa *mood happy* (105) lebih banyak daripada *mood sad* (95).



Gambar 2. Distribusi Mood Lirik Lagu pada *Test Data*

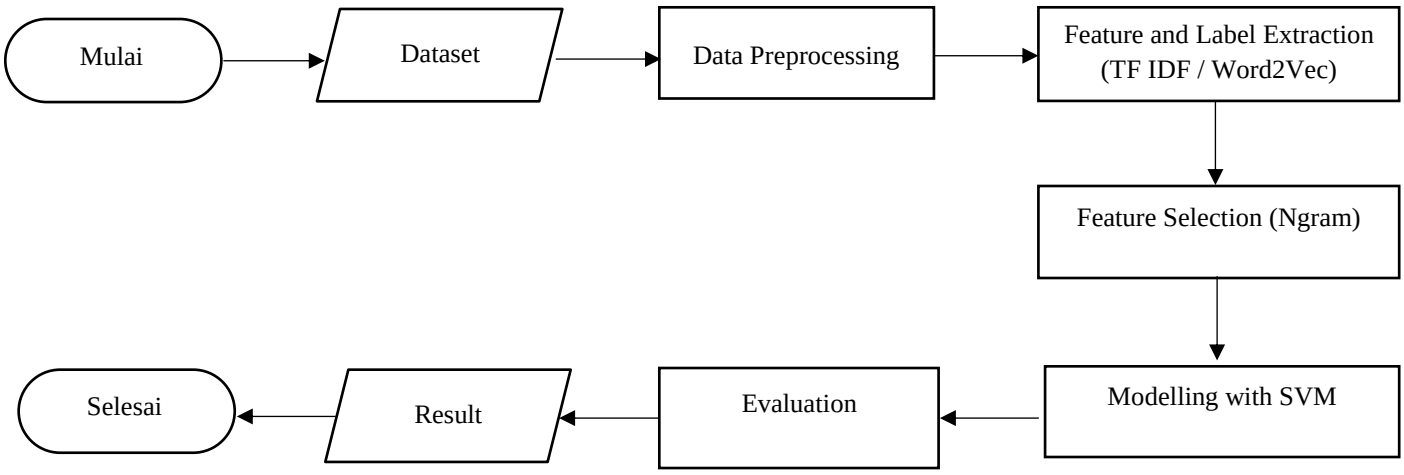
2.1.2 Analisis Metode

Analisis sentimen berkaitan dengan bidang yang lebih luas seperti pengolahan bahasa alami, komputasi linguistik, dan *text mining* untuk tujuan menganalisis, *sentiment*, pendapat, sikap, evaluasi, penilaian dan emosi seseorang berkenan pada topik, produk, layanan, organisasi, atau kegiatan tertentu. Sentimen adalah perasaan, sikap, evaluasi, atau emosi yang terkait dengan suatu opini. Dalam melakukan analisis sentimen mengklasifikasikan lirik lagu, diperlukan suatu metode atau algoritma klasifikasi. Dalam proyek ini, untuk mengklasifikasikan lagu ke dalam kategori *mood happy* dan *sad*, digunakan metode SVM (*Support Vector Machine*). Dalam SVM, objek-objek data terluar yang paling dekat dengan *hyperplane* ini disebut *support vector*. Hanya *support vector* inilah yang diperhitungkan oleh SVM untuk menemukan *hyperplane* paling optimal sedangkan objek-objek data yang lain tidak diperhitungkan sama sekali. Dengan cara ini, SVM dapat bekerja secara lebih efisien. Analisis sentimen pada lirik

lagu pada dataset (*train set* dan *test set*) diimplementasikan menggunakan pemodelan *Support Vector Machine* (SVM) dengan TF-IDF, pemodelan *Support Vector Machine* (SVM) dengan Word2Vec, *Support Vector Machine* (SVM) dengan *K-Fold Cross Validation*, dan pemodelan dengan *Linear SVM Classifier*.

2.2 Desain

Pada subbab ini dijelaskan desain pemrosesan bahasa alami yaitu analisis sentiment pada lirik lagu yang ditampilkan dalam bentuk *flowchart* atau diagram alir sepertu ditunjukkan pada Gambar 3.



Gambar 3. Desain Analisis Sentimen pada Lirik Lagu

2.2.1 *Data Preprocessing*

Data Preprocessing merupakan tahap pemrosesan awal yang dilakukan sebelum teks (*dataset*) diolah ke tahap selanjutnya. Data yang diperoleh masih dalam format yang tidak terstruktur, dimana masih terdapat *noise* pada *dataset* tersebut. Selain itu, data masih dalam format *raw data* sehingga tidak memungkinkan untuk melakukan analisis pada *raw data*. Oleh karena itu perlu dilakukan *data preprocessing* untuk menghilagkan kata-kata pada teks atau dokumen yang mengandung beberapa format yang keberadaannya tidak penting dalam *text mining* [1].

2.2.1.1 *Data Cleaning*

Data yang diperoleh dari dataset memiliki beberapa *noise* yang perlu dibersihkan, misalnya string yang kosong (*incomplete data* / *missing value*). Beberapa cara yang dapat dilakukan untuk membersihkan data adalah sebagai berikut:

- Mengabaikan data yang hilang dengan cara menghapus data tersebut, namun cara ini tidak cukup efektif ketika terdapat banyak data yang hilang.

- Mengisi *missing value*, namun cara ini tidak cukup efektif ketika datanya besar dan membutuhkan waktu yang lama.
- Memperbaiki penulisan kata yang kurang tepat.
- Menghapus karakter-karakter yang bersifat *noise* pada dataset.

2.2.1.2 *Case Folding*

Setelah data dibersihkan, pada tahapan ini dilakukan proses konversi data teks yang terdapat pada *dataset* menjadi ke dalam bentuk *lower case*. Hal ini untuk memudahkan preproses data di tahap selanjutnya. Berikut adalah contoh *case folding*:

Tabel 2. ***Case Folding***

Text	***Case Folding***
LONG time ago, there lived a King	long time ago, there lived a king

2.2.1.3 *Tokenization*

Tujuan dari tahapan ini adalah untuk memecah teks menjadi unit yang relevan, seperti *words*, simbol, *phrases*, atau elemen bermakna lainnya yang disebut juga dengan *token*. Contoh *tokenization* adalah sebagai berikut.

Tabel 3. ***Tokenization***

Text	***Tokenization***
LONG time ago, there lived a King	*long, time, ago, there, lived, a, king*

2.2.1.4 *Stopwords Removal*

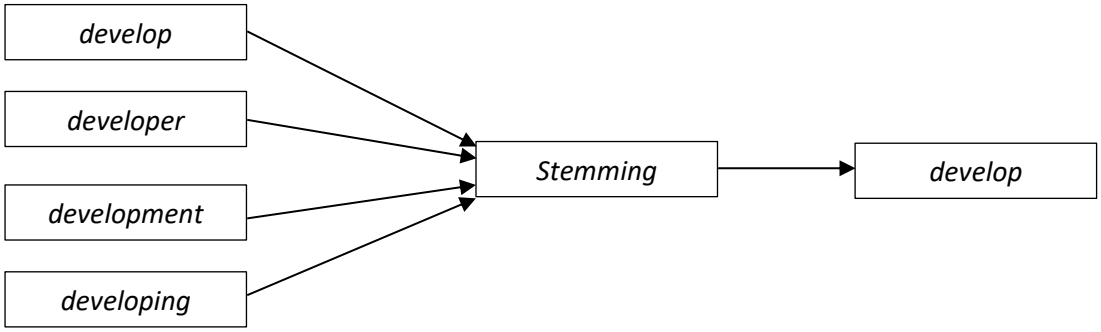
Pada tahapan ini akan dilakukan proses untuk menghilangkan *stopwords*. Kata-kata seperti “this”, “there”, “the”, “a”, “of” merupakan *stopwords* yang biasa digunakan dalam teks. Jika dibandingkan dengan kata lainnya, frekuensi kemunculan *stopwords* lebih banyak. *Stopwords* merupakan elemen penting yang perlu untuk dihilangkan karena membuat teks yang diproses menjadi lebih banyak. Selain itu, *stopwords* juga kurang dibutuhkan dalam proses *text mining*. Sehingga, menghilangkan *stopwords* akan mengurangi dimensi dari data dan akan meningkatkan kecepatan dalam memproses data. Contoh *stopwords removal* adalah sebagai berikut.

Tabel 4. ***Stopwords Removal***

Text	*Stopwords Removal*
LONG time ago, there lived a King	*LONG time ago, lived King*

2.2.1.5 *Stemming*

Stemming merupakan sebuah proses menghilangkan imbuhan pada kata untuk mendapatkan kata dasar atau *stem* dari kata tersebut. Contohnya, kata *develop, developer, development, developing* akan dipetakan ke kata dasar develop. Tujuan dari tahapan ini adalah untuk menghilangkan variasi imbuhan yang mungkin pada suatu kata, sehingga akan mengurangi jumlah kata yang diidentifikasi. Contoh *stemming* adalah sebagai berikut.



Gambar 4. ***Stemming***

2.2.1.6 *Lemmatization*

Lemmatization merupakan sebuah proses untuk mengubah kata menjadi bentuk dasarnya sesuai dengan kata-kata yang terdapat dalam kamus. Contoh *lemmatization* adalah sebagai berikut.

Tabel 5. ***Lemmatization***

Word	*Word Lemmatization*
Good	*Good*
Better	*Good*
Best	*Good*
Leafs	*Leaf*
Leaves	*Leaf*

2.2.2 *Feature Extraction (TF IDF / Word2Vec)*

Model *machine learning* tidak dapat bekerja pada *features* yang tekstual sehingga *features* tekstual tersebut perlu diproses menjadi *numerical features*. Cara yang akan digunakan untuk melatih model tekstual adalah TF IDF *Vectorization* dan Word2Vec.

(1) TF-IDF *Vectorization*

TF-IDF juga dapat digunakan untuk menghitung frekuensi kata. Dengan TF-IDF, maka dapat di-*assign* jumlah frekuensi kemunculan kata yang mengidentifikasikan pentingnya kata tersebut pada dokumen atau korpus. Nilai TF-IDF dari sebuah kata dapat ditentukan menggunakan persamaan berikut:

$$W_i = TF(w_i, d) \times IDF\ (w_i)$$

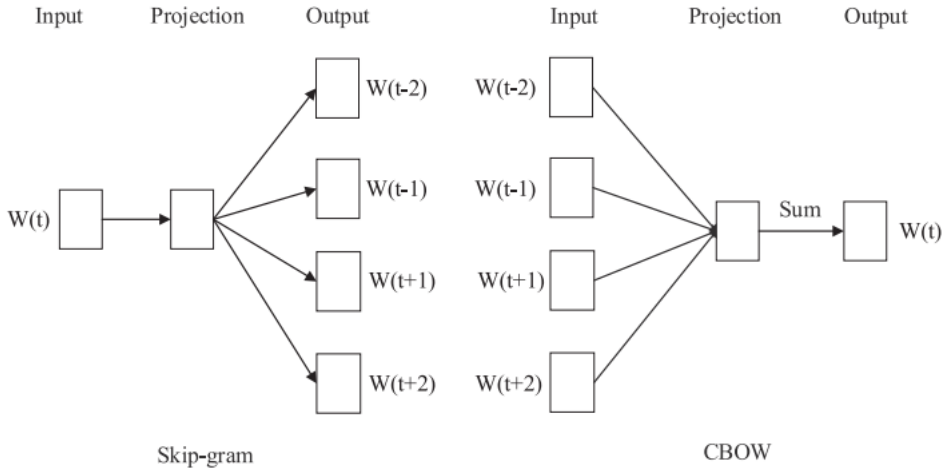
Dimana W_i adalah bobot kata w_i dalam dokumen $d \in D$, $TF(w_i, d)$ adalah *term frequency*, atau jumlah w_i di d. Dan untuk nilai IDF adalah sebagai berikut:

$$IDF(w_i) = \log\left(\frac{|D|}{DF(w_i)}\right)$$

Dimana IDF adalah *inverse document frequency* dan $DF(w_i)$ merepresentasikan kemunculan w_i dalam D. Nilai yang terbesar dari $IDF(w_i)$ terjadi ketika w_i hanya muncul dalam sebuah dokumen dan pengaruhnya sangat besar [2].

(2) Word2Vec

Word2Vec adalah salah satu metode *embedding word* yang berguna untuk merepresentasikan kata menjadi sebuah vektor dengan panjang N. *Word2Vec* mentransformasi kata ke dalam sebuah vektor dalam K-dimensional *vector space* dan menggunakan kesamaan dalam ruang vektor untuk merepresentasikan kesamaan *semantic* dari kata tersebut. *Word2Vec* terdiri dari 2 model, yaitu *Skip-gram* dan *Continuous Bag Of Words* (CBOW). Kedua model pada *Word2Vec* tersebut akan mengkonversi kata ke dalam vektor kata dengan mengadopsi bobot jarak antara setiap kata dalam *context* dan *middle word* (kata tengah) [7].



Gambar 5. Skip-gram dan CBOW pada Model Word2Vec

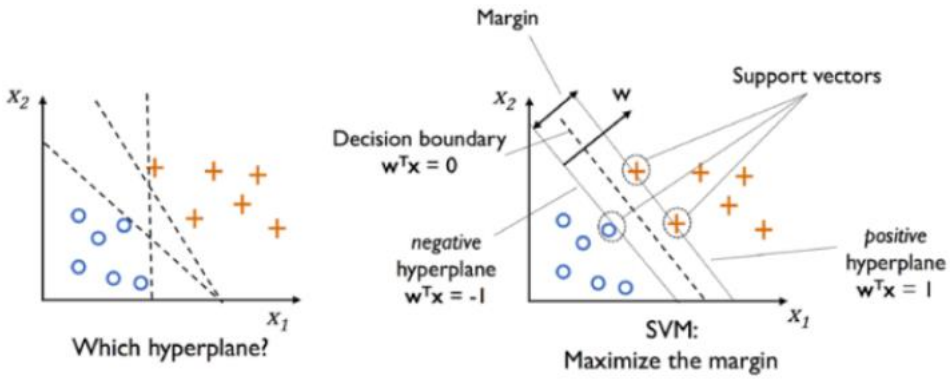
Pada arsitektur *Skip-gram* menggunakan sebuah kata atau disebut juga dengan *current word* untuk memprediksi konteks kata disekitarnya. Arsitektur ini dapat dimodelkan dengan probabilitas kata-kata sebelum W_t yaitu W_{t-h} dan kata-kata W_t yaitu W_{t+h} diketahui/diberikan W_t. Sementara itu, pada arsitektur *Continuous Bag Of Words* memprediksi *current word* (sebagai target) dari konteks (sebagai input) di sekitarnya. Arsitektur ini dapat dimodelkan dengan probabilitas W_t diketahui/diberikan kata-kata sebelum W_t yaitu W_{t-h} dan kata-kata setelah W_t yaitu W_{t+h}.

2.2.3 *Feature Selection (Ngrams)*

Teks yang telah diekstrak selanjutnya perlu diproses dengan *feature selection* untuk mengetahui pola kata yang digunakan, urutan kata, dll. Urutan kata (*sequence of words*) merupakan elemen penting dalam melakukan analisis sentimen. Misalnya, kata "*i am sorry*" merupakan 3-gram yang artinya lebih masuk akal dibandingkan ketika analisis dilakukan per kata secara individu, misalnya "i", "am", "sorry". Dalam pengerjaan proyek ini, akan dilakukan *feature selection* dengan unigram, bigram, dan trigram untuk menghasilkan *document term matrix* dengan N Gram.

2.2.4 *Modelling with SVM*

Setelah *feature-feature* yang akan digunakan diseleksi, selanjutnya adalah melakukan *train* pada dataset dengan menggunakan algoritma *Support Vector Machine* (SVM). SVM merupakan salah satu metode dalam *supervised learning* yang digunakan untuk klasifikasi (*Support Vector Classification*) dan regresi (*Support Vector Regression*). SVM bekerja dengan mencari *hyperplane* terbaik untuk memaksimalkan jarak antar kelas. *Hyperplane* adalah sebuah fungsi yang dapat digunakan sebagai pemisah antar kelas [3].



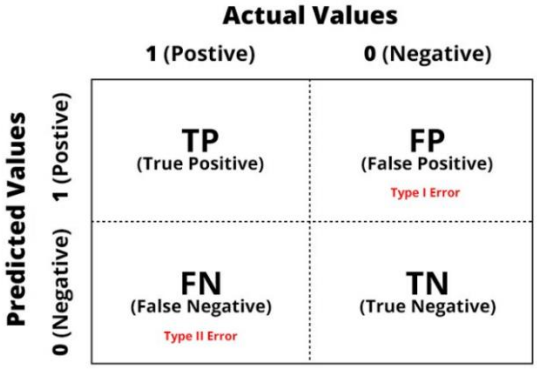
Gambar 6. *Support Vector Machine*

Teknik pemodelan dengan SVM akan menggunakan 4 cara yaitu:

- *SVM Model with TF IDF*
- *SVM Model with Word2Vec*
- *SVM Model with K-Fold Cross Validation*
- *Linear SVM Classifier*

2.2.5 *Evaluation and Results*

Setelah model selesai dibangun, selanjutna adalah melakukan evaluasi. Pada tahapan ini proses evaluasi dari hasil yang didapatkan, dilakukan dengan *confusion matrix*. *Confusion matrix* memberikan informasi perbandingan hasil klasifikasi yang dilakukan model dengan klasifikasi yang sebenarnya.



Gambar 7. *Confusion Matrix*

Terdapat 4 istilah yang digunakan untuk merepresntasikan hasil klasifikasi, yaitu:

1. *True Positive* (TP) merupakan data positif yang diprediksi benar.
2. *True Negative* (TN) merupakan data negatif yang diprediksi benar.
3. *False Positive* (FP) merupakan data negatif namun diprediksi sebagai data positif.
4. *False Negative* (FN) merupakan data positif namun diprediksi sebagai data negatif.

Untuk mengukur *performance metrics* dari *confusion matrix* dapat dilakukan dengan menghitung nilai *accuracy*, *precision*, *Recall*, dan F1-*Score*.

(1) *Accuracy* menggambarkan seberapa akurat model dapat mengklasifikasikan dengan benar.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

(2) *Precision* menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model.

$$Precision = \frac{TP}{TP + FP}$$

(3) *Recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi.

$$Recall = \frac{TP}{TP + FN}$$

2.3 Implementasi

Pada subbab ini dijelaskan pengimplementasian pemrosesan bahasa alami, yaitu *sentiment analysis* menggunakan algoritma SVM (*Support Vector Machine*).

2.3.1 *Data Preprocessing*

Pada bagian ini akan dibahas *data preprocessing* yang dilakukan sebelum digunakan dalam pemodelan, mencakup *data cleaning*, *case folding*, *stopwords removal*, *tokenization*, *stemming*, dan *lemmatization*.

2.3.1.1 *Data Cleaning*

Pada bagian ini, sebelum melakukan *data cleaning* perlu dilakukan pemeriksaan terhadap missing value dari *train data* dan *test data*. Berikut adalah kode program dalam mendeteksi *missing value* pada *train data* ditunjukkan pada Gambar 8.

```
# mendeteksi missing value dengan mengembalikan nilai boolean
(true/false) pada data train dan data test
train_data.isna()
test_data.isna()
```

Gambar 8. Kode Program Mendeteksi *Missing Value* pada *Train Data* dan *Test Data*

Hasil pendeteksian *missing value* pada *train data* ditampilkan pada Gambar 9. Pada gambar tersebut dapat dilihat bahwa pada 5 data teratas dan 5 data terakhir tidak memiliki *missing value*, sehingga nilainya *False*.

	file	artist	title	lyrics	genre	mood	year
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
995	False	False	False	False	False	False	False
996	False	False	False	False	False	False	False
997	False	False	False	False	False	False	False
998	False	False	False	False	False	False	False
999	False	False	False	False	False	False	False

1000 rows × 7 columns

Gambar 9. Hasil Pendeteksian *Missing Value* pada *Train Data*

Hasil pendeteksian *missing value* pada *test data* ditampilkan pada Gambar 10. Pada gambar tersebut dapat dilihat bahwa pada 5 data teratas dan 5 data terakhir tidak memiliki *missing value*, sehingga nilainya *False*.

	file	artist	title	genre	lyrics	mood
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...	...	...	...	...	...	...
195	False	False	False	False	False	False
196	False	False	False	False	False	False
197	False	False	False	False	False	False
198	False	False	False	False	False	False
199	False	False	False	False	False	False

200 rows × 6 columns

Gambar 10. Hasil Pendeteksian *Missing Value* pada *Test Data*

Setelah dilakukan pendeteksian, selanjutnya adalah menyimpulkan apakah pada *train data* dan *test data* terdapat *missing value*. Berikut adalah kode program untuk melakukan peringkasan *missing value* pada *train data* dan *test data.*

```
# melakukan agregasi data
train_data.isna().sum()
test_data.isna().sum()
```

Gambar 11. Kode Program Peringkasan *Missing Value* pada *Train Data* dan *Test Data*

Hasil peringkasan *missing value* pada *train data* ditunjukkan pada Gambar 12. Dapat disimpulkan bahwa pada *train data* tidak terdapat *missing value*, sehingga *data cleaning* tidak akan dilakukan pada *train data.*

```
file      0
artist    0
title     0
lyrics    0
genre     0
mood      0
year      0
dtype: int64
```

Gambar 12. Hasil Peringkasan *Missing Value* pada *Train Data*

Hasil peringkasan *missing value* pada *train data* ditunjukkan pada Gambar 13. Dapat disimpulkan bahwa pada *test data* tidak terdapat *missing value*, sehingga *data cleaning* tidak akan dilakukan pada *test data.*

```
file      0
artist    0
title     0
genre     0
lyrics    0
mood      0
dtype: int64
```

Gambar 13. Hasil Peringkasan *Missing Value* pada *Test Data*

2.3.1.2 *Case Folding*

Pada bagian ini *case folding* dilakukan untuk mengubah semua lirik lagu dalam format huruf kecil (*lowercase*). Berikut adalah kode program yang dilakukan untuk mengubah lirik lagu menjadi dalam format *lowercase* pada *train data* dan *test data*.

```
# case folding/lowercasing
train_data['lyrics'] = train_data['lyrics'].str.lower()
test_data['lyrics'] = test_data['lyrics'].str.lower()
```

Gambar 14. Kode Program *Case Folding* Lirik Lagu pada *Train Data* dan *Test Data*

Hasil *case folding* lirik lagu yang dilakukan pada *train data* ditunjukkan pada Gambar 15.

file	artist	title	lyrics	genre	mood	year
AAW128F429D538.h5	Casual	I Didn't Mean To	verse one:\n\nalright i might\nhave had a litt...	Hip Hop/Rap	sad	1994
AAEF128F4273421.h5	Adam Ant	Something Girls	adam ant/marco pirroni\nevery girl is a someth...	Rock	happy	1982
AFD128F92F423A.h5	Gob	Face the Ashes	i've just erased it's been a while, i've got a...	Rock	sad	2007

Gambar 15. Hasil *Case Folding* Lirik Lagu pada *Train Data*

Hasil *case folding* lirik lagu yang dilakukan pada *test data* ditunjukkan pada Gambar 16.

artist	title	genre	lyrics	mood
acebo	Where Is My Mind (XFM Live Version)	Pop	oooooohh\nooooohh\nooooohh\nooooohh\n\nwith you...	happy
ueens f The Stone Age	This Lullaby	Rock	where, oh, where have you been, my love? \nwher...	sad
MC5	Looking At You (Cody High School)	Rock	when it happened \nsomething snapped inside \n...	happy

Gambar 16. Hasil *Case Folding* Lirik Lagu pada *Test Data*

2.3.1.3 *Tokenization*

Pada bagian ini dilakukan tokenisasi yang merupakan proses untuk memecah lirik lagu menjadi unit yang relevan, seperti *words*, simbol, *phrases*, atau lainnya yang disebut dengan token. Berikut adalah kode program yang digunakan untuk melakukan tokenisasi dengan memanfaatkan *RegexpTokenizer* pada *train data*.

```
# melakukan tokenisasi dengan RegexpTokenizer pada kolom lyrics di
train_data
Regtokenizer = RegexpTokenizer(train_data['lyrics'])
print(Regtokenizer)
Regtokenizer = RegexpTokenizer(test_data['lyrics'])
print(Regtokenizer)
```

Gambar 17. Kode Program *Tokenization* Lirik Lagu pada *Train Data* dan *Test Data*

Hasil *tokenization* lirik lagu yang dilakukan pada *train data* ditunjukkan pada Gambar 18.

```
RegexpTokenizer(pattern=0      verse one:\n\nalright i might
\nhave had a litt...
1      adam ant/marco pirroni\nevery girl is a someth...
2      i've just erased it's been a while, i've got a...
3      little darling \nwhere you've been so long \ni...
4      lead vocal by greg\n\nwell, these late night c...
                            ...
995    its all about our love\nso shall it be forever...
996    it's time that i rain on your parade\nwatch as...
997    speakin of the devil\nlook who just walked in\...
998    born from some mother's womb\njust like any ot...
999    this wanting more from me is tearing me, it's ...
Name: lyrics, Length: 1000, dtype: object, gaps=False, disca
rd_empty=True, flags=<RegexFlag.UNICODE|DOTALL|MULTILINE: 56
>)
```

Gambar 18. Hasil *Tokenization* Lirik Lagu pada *Train Data*

Hasil *tokenization* lirik lagu yang dilakukan pada *test data* ditunjukkan pada Gambar 19.

```
RegexpTokenizer(pattern=0      oooooohh\nooooooh\nooooooh\noo
ooohh\n\nwith you...
1      where, oh, where have you been, my love?\nwher...
2      when it happened \nsomething snapped inside \n...
3      my eyes got blinded\nand conned by the light\n...
4      [music: v santura, lyrics: morean]\n\nawake, a...
                            ...
195    {b-side of glam slam}\nsnare drum pounds on th...
196    well i will rise\nthe morning comes\nnothing e...
197    listen when i say, when i say it's real\nreal ...
198    imagine a world where the girls, girls rule th...
199    each evening, from december to december\nbefor...
Name: lyrics, Length: 200, dtype: object, gaps=False, discar
d_empty=True, flags=<RegexFlag.UNICODE|DOTALL|MULTILINE: 56
>)
```

Gambar 19. Hasil *Tokenization* Lirik Lagu pada *Test Data*

2.3.1.4 *Stopwords Removal*

Pada bagian ini dilakukan proses untuk menghilangkan *stopwords*, seperti “this”, “there”, “the”, “a”, “of”, ”with”, “have”, dan lain-lain. Berikut adalah kode program untuk melakukan *stopwords removal* pada *train data* dan *test data*.

```
# Melakukan stopwords removal pada lyrics_tokens_preprocessing
yang ada di  train_data dan test_data


from nltk.corpus import stopwords


stop = stopwords.words('english')
```

```
train_data['lyrics_tokens_preprocessing'] =
train_data['lyrics_tokens'].apply(lambda x: [item for item in x if
item not in stop])
test_data['lyrics_tokens_preprocessing'] =
test_data['lyrics_tokens'].apply(lambda x: [item for item in x if
item not in stop])
```

Gambar 20. Kode Program *Stopwords Removal* Lirik Lagu pada *Train Data* dan *Test Data*

Hasil *stopwords removal* lirik lagu yang dilakukan pada *train data* ditunjukkan pada Gambar 21.

	file	artist	title	lyrics	genre	mood	year	lyrics_tokens	lyrics_tokens_preprocessing
0	TRAAAAW128F429D538.h5	Casual	I Didn't Mean To	verse one alright might have had little glare ...	Hip Hop/Rap	sad	1994	[verse, one, alright, might, have, had, little...	[verse, one, alright, might, little, glare, st...
1	TRAAAEF128F4273421.h5	Adam Ant	Something Girls	adam antmarco pirroni every girl is something ...	Rock	happy	1982	[adam, antmarco, pirroni, every, girl, is, som...	[adam, antmarco, pirroni, every, girl, somethi...
2	TRAAAFD128F92F423A.h5	Gob	Face the Ashes	have just erased it is been while have got wor...	Rock	sad	2007	[have, just, erased, it, is, been, while, have...	[erased, got, world, sale, walk, away, better,...
3	TRAABJV128F1460C49.h5	Lionel Richie	Tonight Will Be Alright	little darling where you have been so long hav...	R&B	happy	1986	[little, darling, where, you, have, been, so, ...	[little, darling, long, thinking, ya, feeling,...
4	TRAABLR128F423B7E3.h5	Blue Rodeo	Floating	lead vocal by greg well these late night conve...	Rock	sad	1987	[lead, vocal, by, greg, well, these, late, nig...	[lead, vocal, greg, well, late, night, convers...

Gambar 21. Hasil *Stopwords Removal* Lirik Lagu pada *Train Data*

Hasil *stopwords removal* lirik lagu yang dilakukan pada *test data* ditunjukkan pada Gambar 22.

	file	artist	title	genre	lyrics	mood	lyrics_tokens	lyrics_tokens_preprocessing
0	TRAFAIB128F426E636.h5	Placebo	Where Is My Mind (XFM Live Version)	Pop	with your feet on the air and your head on the...	happy	[with, your, feet, on, the, air, and, your, he...	[feet, air, head, ground, try, trick, spin, ye...
1	TRAFAJC128E078888B.h5	Queens Of The Stone Age	This Lullaby	Rock	where where have you been my love where where ...	sad	[where, where, have, you, been, my, love, wher...	[love, long, since, moon, gone, wreck, made, o...
2	TRAFBBP128F92F6CC9.h5	MC5	Looking At You (Cody High School)	Rock	when it happened something snapped inside made...	happy	[when, it, happened, something, snapped, insid...	[happened, something, snapped, inside, made, w...
3	TRAFBVU128F426B3F6.h5	Dimmu Borgir	The Fundamental Alienation	Rock	my eyes got blinded and conned by the light li...	sad	[my, eyes, got, blinded, and, conned, by, the,...	[eyes, got, blinded, conned, light, like, fugl...
4	TRAFEEO12903CFEC87.h5	Dark Fortress	The Silver Gate	Rock	awake angels of lunacy arise avatars of chaos ...	sad	[awake, angels, of, lunacy, arise, avatars, of...	[awake, angels, lunacy, arise, avatars, chaos,...

Gambar 22. Hasil *Stopwords Removal* Lirik Lagu pada *Test Data*

2.3.1.5 *Stemming*

Pada bagian ini dilakukan proses *stemming* untuk menghilangkan imbuhan pada kata sehingga menperoleh kata dasar atau *stem* dari kata tersebut. Berikut adalah kode program yang digunakan untuk melakukan *stemming* pada *train data* dan *test data*.

```
# Melakukan stemming pada lyrics_tokens_preprocessing yang ada di
train_data dan test_data
```

```
from nltk.stem.porter import PorterStemmer

stemmer = PorterStemmer()
train_data['lyrics_tokens_preprocessing'] =
train_data['lyrics_tokens_preprocessing'].apply(lambda x:
[stemmer.stem(y) for y in x])
test_data['lyrics_tokens_preprocessing'] =
test_data['lyrics_tokens_preprocessing'].apply(lambda x:
[stemmer.stem(y) for y in x])
```

Gambar 23. Kode Program *Stemming* Lirik Lagu pada *Train Data* dan *Test Data*

Hasil *stemming* lirik lagu yang dilakukan pada *train data* ditunjukkan pada Gambar 24.

	file	artist	title	lyrics	genre	mood	year	lyrics_tokens	lyrics_tokens_preprocessing
0	TRAAAAW128F429D538.h5	Casual	I Didn't Mean To	verse one alright might have had little glare ...	Hip Hop/Rap	sad	1994	[verse, one, alright, might, have, had, little...	[vers, one, alright, might, littl, glare, star...
1	TRAAAEF128F4273421.h5	Adam Ant	Something Girls	adam antmarco pirroni every girl is something ...	Rock	happy	1982	[adam, antmarco, pirroni, every, girl, is, som...	[adam, antmarco, pirroni, everi, girl, someth,...
2	TRAAAFD128F92F423A.h5	Gob	Face the Ashes	have just erased it is been while have got wor...	Rock	sad	2007	[have, just, erased, it, is, been, while, have...	[eras, got, world, sale, walk, away, better, d...
3	TRAABJV128F1460C49.h5	Lionel Richie	Tonight Will Be Alright	little darling where you have been so long hav...	R&B	happy	1986	[little, darling, where, you, have, been, so, ...	[littl, darl, long, think, ya, feel, real, str...
4	TRAABLR128F423B7E3.h5	Blue Rodeo	Floating	lead vocal by greg well these late night conve...	Rock	sad	1987	[lead, vocal, by, greg, well, these, late, nig...	[lead, vocal, greg, well, late, night, convers...

Gambar 24. Hasil *Stemming* Lirik Lagu pada *Train Data*

Hasil *stemming* lirik lagu yang dilakukan pada *test data* ditunjukkan pada Gambar 25.

	file	artist	title	genre	lyrics	mood	lyrics_tokens	lyrics_tokens_preprocessing
0	TRAFAIB128F426E636.h5	Placebo	Where Is My Mind (XFM Live Version)	Pop	with your feet on the air and your head on the...	happy	[with, your, feet, on, the, air, and, your, he...	[feet, air, head, ground, tri, trick, spin, ye...
1	TRAFAJC128E078888B.h5	Queens Of The Stone Age	This Lullaby	Rock	where where have you been my love where where ...	sad	[where, where, have, you, been, my, love, wher...	[love, long, sinc, moon, gone, wreck, made, oc...
2	TRAFBBP128F92F6CC9.h5	MC5	Looking At You (Cody High School)	Rock	when it happened something snapped inside made...	happy	[when, it, happened, something, snapped, insid...	[happen, someth, snap, insid, made, want, hide...
3	TRAFBVU128F426B3F6.h5	Dimmu Borgir	The Fundamental Alienation	Rock	my eyes got blinded and conned by the light li...	sad	[my, eyes, got, blinded, and, conned, by, the,...	[eye, got, blind, con, light, like, fugit, run...
4	TRAFEEO12903CFEC87.h5	Dark Fortress	The Silver Gate	Rock	awake angels of lunacy arise avatars of chaos ...	sad	[awake, angels, of, lunacy, arise, avatars, of...	[awak, angel, lunaci, aris, avatar, chao, inci...

Gambar 25. Hasil *Stemming* Lirik Lagu pada *Test Data*

2.3.1.6 *Lemmatization*

Pada bagian ini dilakukan *lemmatization* untuk mengubah kata menjadi bentuk dasarnya sesuai dengan kata-kata yang terdapat dalam kamus. Berikut adalah kode program yang digunakan untuk melakukan *lemmatization* pada *train data* dan *test data*.

```
# Melakukan Lemmatization pada lyrics_tokens_preprocessing yang
ada di  train_data dan test_data
```

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

train_data['lyrics_tokens_preprocessing'] =
train_data['lyrics_tokens_preprocessing'].apply(lambda x:
[lemmatizer.lemmatize(y) for y in x])
test_data['lyrics_tokens_preprocessing'] =
test_data['lyrics_tokens_preprocessing'].apply(lambda x:
[lemmatizer.lemmatize(y) for y in x])
```

Gambar 26. Kode Program *Lemmatization* Lirik Lagu pada *Train Data* dan *Test Data*

Hasil *lemmatization* lirik lagu yang dilakukan pada *train data* ditunjukkan pada Gambar 27.

	file	artist	title	lyrics	genre	mood	year	lyrics_tokens	lyrics_tokens_preprocessing
0	TRAAAAW128F429D538.h5	Casual	I Didn't Mean To	verse one alright might have had little glare ...	Hip Hop/Rap	sad	1994	[verse, one, alright, might, have, had, little...	[vers, one, alright, might, littl, glare, star...
1	TRAAAEF128F4273421.h5	Adam Ant	Something Girls	adam antmarco pirroni every girl is something ...	Rock	happy	1982	[adam, antmarco, pirroni, every, girl, is, som...	[adam, antmarco, pirroni, everi, girl, someth,...
2	TRAAAFD128F92F423A.h5	Gob	Face the Ashes	have just erased it is been while have got wor...	Rock	sad	2007	[have, just, erased, it, is, been, while, have...	[era, got, world, sale, walk, away, better, do...
3	TRAABJV128F1460C49.h5	Lionel Richie	Tonight Will Be Alright	little darling where you have been so long hav...	R&B	happy	1986	[little, darling, where, you, have, been, so, ...	[littl, darl, long, think, ya, feel, real, str...
4	TRAABLR128F423B7E3.h5	Blue Rodeo	Floating	lead vocal by greg well these late night conve...	Rock	sad	1987	[lead, vocal, by, greg, well, these, late, nig...	[lead, vocal, greg, well, late, night, convers...

Gambar 27. Hasil *Lemmatization* Lirik Lagu pada *Train Data*

Hasil *lemmatization* lirik lagu yang dilakukan pada *test data* ditunjukkan pada Gambar 28.

	file	artist	title	genre	lyrics	mood	lyrics_tokens	lyrics_tokens_preprocessing
0	TRAFAIB128F426E636.h5	Placebo	Where Is My Mind (XFM Live Version)	Pop	with your feet on the air and your head on the...	happy	[with, your, feet, on, the, air, and, your, he...	[foot, air, head, ground, tri, trick, spin, ye...
1	TRAFAJC128E078888B.h5	Queens Of The Stone Age	This Lullaby	Rock	where where have you been my love where where ...	sad	[where, where, have, you, been, my, love, wher...	[love, long, sinc, moon, gone, wreck, made, oc...
2	TRAFBBP128F92F6CC9.h5	MC5	Looking At You (Cody High School)	Rock	when it happened something snapped inside made...	happy	[when, it, happened, something, snapped, insid...	[happen, someth, snap, insid, made, want, hide...
3	TRAFBVU128F426B3F6.h5	Dimmu Borgir	The Fundamental Alienation	Rock	my eyes got blinded and conned by the light li...	sad	[my, eyes, got, blinded, and, conned, by, the,...	[eye, got, blind, con, light, like, fugit, run...
4	TRAFEEO12903CFEC87.h5	Dark Fortress	The Silver Gate	Rock	awake angels of lunacy arise avatars of chaos ...	sad	[awake, angels, of, lunacy, arise, avatars, of...	[awak, angel, lunaci, aris, avatar, chao, inci...

Gambar 28. Hasil *Lemmatization* Lirik Lagu pada *Test Data*

2.3.2 *Feature Extraction*

Pada bagian ini dilakukan *feature extraction* dari lirik lagu menggunakan TF IDF dan Word2Vec. Berikut adalah kode program yang digunakan untuk melakukan *feature extraction* dengan TF IDF ditunjukkan pada Gambar 29.

```
# tf-idf
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer

def do_nothing(tokens):
    return tokens
```

```
count_vect = CountVectorizer(tokenizer=do_nothing,
lowercase=False)
X_train_tf =
count_vect.fit_transform(train_data['lyrics_tokens_preprocessing']
)

tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_tf)
```

Gambar 29. *Feature Extraction* Lirik Lagu dengan TF IDF

Selain itu *feature extraction* juga dilakukan dengan Word2Vec. Berikut adalah kode programnya ditunjukkan pada Gambar 30.

```
# skipgram - word2vec
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from gensim.models import Word2Vec

w2v_model   =   Word2Vec(train_data['lyrics_tokens_preprocessing'],
size=100, sg=1)

# fungsi untuk membuat averaged sentence vektor
def buildWordVector(tokens, size):
    vec = np.zeros(size).reshape((1, size))
    count = 0.
    for word in tokens:
        try:
            # melakukan penjumlahan terhadap vektor
            vec    +=    w2v_model.wv.__getitem__(word).reshape((1,
size))
            count += 1.
        except KeyError: # handling the case where the token is not
                         # in the corpus. useful for testing.
            continue
    # melakukan pembagian total word dalam kalimat
    if count != 0:
        vec /= count
    return vec

# melakukan konversi train_data['lyrics_tokens_preprocessing'] ke
dalam list of vectors
X_train_w2v  =  np.concatenate([buildWordVector(z,  100)  for  z  in
map(lambda x: x, train_data['lyrics_tokens_preprocessing'])])

# Melakukan normalisasi dengan MinMax Normalization
scaler = MinMaxScaler()
X_train_w2v = scaler.fit_transform(X_train_w2v)
```

Gambar 30. *Feature Extraction* Lirik Lagu dengan Word2Vec

2.3.3 *Feature Selection*

Pada bagian ini dilakukan *feature selection* dengan unigram, bigram, dan trigram. Berikut adalah kode program yang digunakan untuk melakukan *feature selection* dengan unigram

ditunjukkan pada Gambar 31, dengan bigram ditunjukkan pada Gambar 32, dan dengan trigram ditunjukkan pada Gambar 33.

```
# unigrams

for words in train_data['lyrics_tokens_preprocessing']:
    unigrams = ngrams(words, 1)
    for i, gram_rev in enumerate(unigrams):
        print(gram_rev)

for words in test_data['lyrics_tokens_preprocessing']:
    unigrams = ngrams(words, 1)
    for i, gram_rev in enumerate(unigrams):
        print(gram_rev)
```

Gambar 31. *Feature Selection* Lirik Lagu dengan Unigram

```
# bigrams
for words in train_data['lyrics_tokens_preprocessing']:
    bigrams = ngrams(words, 2)
    for i, gram_rev in enumerate(bigrams):
        print(gram_rev)

for words in test_data['lyrics_tokens_preprocessing']:
    bigrams = ngrams(words, 2)
    for i, gram_rev in enumerate(bigrams):
        print(gram_rev)
```

Gambar 32. *Feature Selection* Lirik Lagu dengan Bigram

```
# trigrams
for words in train_data['lyrics_tokens_preprocessing']:
    trigrams = ngrams(words, 3)
    for i, gram_rev in enumerate(trigrams):
        print(gram_rev)

for words in test_data['lyrics_tokens_preprocessing']:
    trigrams = ngrams(words, 3)
    for i, gram_rev in enumerate(trigrams):
        print(gram_rev)
```

Gambar 33. *Feature Selection* Lirik Lagu dengan Trigram

Hasil *feature selection* yang diperoleh ditunjukkan pada Gambar 34, Gambar 35, dan Gambar 36.

```
('vers',)
('one',)
('alright',)
('might',)
('littl',)
('glare',)
('stare',)
('ya',)
```

Gambar 34. Hasil *Feature Selection* Lirik Lagu dengan Unigram

```
('vers', 'one')
('one', 'alright')
('alright', 'might')
('might', 'littl')
('littl', 'glare')
('glare', 'stare')
('stare', 'ya')
('ya', 'ho')
```

Gambar 35. Hasil *Feature Selection* Lirik Lagu dengan Bigram

```
('vers', 'one', 'alright')
('one', 'alright', 'might')
('alright', 'might', 'littl')
('might', 'littl', 'glare')
('littl', 'glare', 'stare')
('glare', 'stare', 'ya')
('stare', 'ya', 'ho')
('ya', 'ho', 'know')
```

Gambar 36. Hasil *Feature Selection* Lirik Lagu dengan Trigram

2.3.4 *Modeling with SVM*

Pada bagian ini dilakukan pemodelan terhadap analisis sentimen dengan algoritma SVM. Dalam penerapan algoritma SVM, terdapat 4 teknik yang dilakukan, yaitu:

2.3.4.1 *SVM Model with TF IDF*

Pada bagian ini, SVM model yang dibangun memanfaatkan TF IDF. Berikut adalah kode program yang digunakan, ditunjukkan pada Gambar 37.

```
# melakukan pembangunan model SVM dengan TF-IDF
from sklearn import svm
from sklearn.multiclass import OneVsRestClassifier
model_svm_tfidf = OneVsRestClassifier(svm.SVC(gamma=0.02, C=100.,
probability=True, class_weight='balanced', kernel='linear'))
model_svm_tfidf.fit(X_train_tfidf, train_data['mood'])
```

Gambar 37.* Kode Program *SVM Model with TF IDF

2.3.4.2 *SVM Model with Word2Vec*

Pada bagian ini, SVM model yang dibangun memanfaatkan Word2Vec. Berikut adalah kode program yang digunakan, ditunjukkan pada Gambar 38.

```
# melakukan pembangunan model SVM dengan Word2Vec
from sklearn import svm
from sklearn.multiclass import OneVsRestClassifier
model_svm_w2v = OneVsRestClassifier(svm.SVC(gamma=0.02, C=100.,
probability=True, class_weight='balanced', kernel='linear'))
model_svm_w2v.fit(X_train_w2v, train_data['mood'])
```

Gambar 38. Kode Program *SVM Model with Word2Vec*

2.3.4.3 *SVM Model with K-Fold Cross Validation*

Pada bagian ini, SVM model yang dibangun menggunakan K-Fold *Cross Validation*. Berikut adalah kode program yang digunakan, ditunjukkan pada Gambar 39.

```
# melakukan pembangunan model SVM dengan K-Fold Cross Validation

# Import dependencies
from sklearn.multiclass import OneVsRestClassifier
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score,
train_test_split
from sklearn.pipeline import Pipeline

category_svm = Pipeline([('vect',
TfidfVectorizer(ngram_range=(1,2))),
                     ('svm', SGDClassifier(loss='hinge',
penalty='l2', alpha=1e-4,
                                           random_state=123))])

category_svm = category_svm.fit(train_data['lyrics'],
train_data['mood'])
cross_val_score(estimator=category_svm, X=train_data['lyrics'],
y=train_data['mood'], cv=7).mean()
```

Gambar 39. Kode Program *SVM Model with K-Fold Cross Validation*

2.3.4.4 *Linear SVM Classifier*

Pada bagian ini, SVM model yang dibangun menggunakan *linear SVM classifier*. Pada *linear SVM classifier* sebelumnya perlu didefinisikan label yang akan digunakan dalam analisis sentimen dengan kode program berikut ini.

```
# fungsi untuk mendefinisikan label yang digunakan dalam analisis
sentimen
def sentiment2target(sentiment):
    return {
        'happy': 1,
        'sad' : 2
    }[sentiment]
targets = train_data['mood'].apply(sentiment2target)
```

Gambar 40. Kode Program Mendefiniskan Label dalam Analisis Sentimen

Selanjutnya dilakukan pemisahan terhadap *train data* dan *test data* seperti ditunjukkan pada Gambar 41.

```
# Melakukan pemisahan terhadap train_data dan test_data
from sklearn.model_selection import train_test_split
train_data, test_data, targets_train, targets_test =
train_test_split(indexed_data, targets, test_size=0.03,
random_state=10)
```

```
train_data_index = train_data[:,0]
train_data = train_data[:,1:]
test_data_index = test_data[:,0]
test_data = test_data[:,1:]
```

Gambar 41. Kode Program Pemisahaan *Train Data* dan *Test Data*

Berikut adalah kode program yang digunakan, ditunjukkan pada Gambar 42.

```
# Membangun SVM model dengan OneVsRestClassifier
from sklearn import svm
from sklearn.multiclass import OneVsRestClassifier
clf = OneVsRestClassifier(svm.SVC(gamma=0.02, C=100.,
probability=True, class_weight='balanced', kernel='linear'))
clf_output = clf.fit(train_data, targets_train)

# Menampilkan nilai akurasi dari model SVM yang telah dibangun
dengan train_data
clf.score(train_data, targets_train)
```

Gambar 42. Kode Program *Linear SVM Classifier*

2.4 Hasil

Pada subbab ini dijelaskan evaluasi kuantitatif berdasarkan implementasi pemrosesan bahasa alami, yaitu *sentiment analysis* menggunakan algoritma SVM (*Support Vector Machine*). Untuk mengukur *performance metrics* dilakukan dengan menghitung nilai *precision, Recall,* dan F1-*Score*.

(1) *Precision* menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model.

$$Precision = \frac{TP}{TP + FP}$$

(2) *Recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi.

$$Recall = \frac{TP}{TP + FN}$$

(3) F1-*Score* menggambarkan *harmonic mean* dari *precision* dan *recall*.

$$\frac{1}{F1} = \frac{1}{2}\left(\frac{1}{precision} + \frac{1}{recall}\right)$$

2.4.1 *Evaluation SVM Model – TF IDF*

Berikut adalah kode program evaluasi terhadap model SVM yang telah dibangun dengan *TF IDF*.

```
# Melakukan evaluasi terhadap model SVM yang telah dibangun dengan
TF IDF

from sklearn.metrics import classification_report
```

```
X_test_tf =
count_vect.transform(test_data['lyrics_tokens_preprocessing'])
X_test_tfidf = tfidf_transformer.transform(X_test_tf)

predicted = model_svm_tfidf.predict(X_test_tfidf)

y_true = test_data['mood']
y_pred = predicted

target_names = test_data['mood'].unique()
print(classification_report(y_true, y_pred,
target_names=target_names, zero_division=1))
```

Gambar 43. Kode Program *Evaluation SVM Model – TF IDF*

Hasil *evaluation SVM Model – TF IDF* yang diperoleh ditunjukkan pada Gambar 44 berikut.

	precision	recall	f1-score	support
happy	0.64	0.57	0.60	105
sad	0.58	0.64	0.61	95
accuracy			0.60	200
macro avg	0.61	0.61	0.60	200
weighted avg	0.61	0.60	0.60	200

Gambar 44. Hasil *Evaluation SVM Model – TF IDF*

2.4.2 *Evaluation SVM Model – Skipgram Word2Vec*

Berikut adalah kode program evaluasi terhadap model SVM yang telah dibangun dengan *Skipgram Word2Vec.*

```
# Melakukan evaluasi terhadap model SVM yang telah dibangun dengan
Skipgram Word2Vec

X_test_w2v = np.concatenate([buildWordVector(z, 100) for z in
map(lambda x: x, test_data['lyrics_tokens_preprocessing'])])
X_test_w2v = scaler.transform(X_test_w2v)

predicted = model_svm_w2v.predict(X_test_w2v)

y_true = test_data['mood']
y_pred = predicted

target_names = test_data['mood'].unique()
print(classification_report(y_true, y_pred,
target_names=target_names, zero_division=1))
```

Gambar 45. Kode Program *Evaluation* SVM Model – Skipgram Word2Vec

Hasil *evaluation SVM Model – Skipgram Word2Vec* yang diperoleh ditunjukkan pada Gambar 46 berikut.

	precision	recall	f1-score	support
happy	0.71	0.66	0.68	105
sad	0.65	0.71	0.68	95
accuracy			0.68	200
macro avg	0.68	0.68	0.68	200
weighted avg	0.68	0.68	0.68	200

Gambar 46. Hasil *Evaluation SVM Model – Skipgram Word2Vec*

2.4.3 *Evaluation SVM Model with K-Folds Cross Validation*

Berikut adalah kode program evaluasi terhadap model SVM yang telah dibangun dengan *K-Folds Cross Validation.*

```
# Melakukan evaluasi terhadap model SVM yang telah dibangun dengan
K-Fold Cross Validation

print(category_svm.score(y=test_data['mood'],
X=test_data['lyrics']))
preds_svm = category_svm.predict(test_data['lyrics'])
print(classification_report(y_pred=preds_svm,
y_true=test_data['mood']))
pd.crosstab(preds_svm, test_data['mood'])
```

Gambar 47. Kode Program *Evaluation SVM Model with K-Folds Cross Validation*

Hasil *evaluation SVM Model with K-Folds Cross Validation* yang diperoleh ditunjukkan pada Gambar 48 berikut.

0.68

	precision	recall	f1-score	support
happy	0.80	0.52	0.63	105
sad	0.62	0.85	0.72	95
accuracy			0.68	200
macro avg	0.71	0.69	0.67	200
weighted avg	0.71	0.68	0.67	200

Gambar 48. Hasil *Evaluation SVM Model with K-Folds Cross Validation*

2.4.4 *SVM Model with Misclassify*

Pada bagian ini dilakukan evaluasi terhadap data yang tidak bisa dikelompokkan secara benar (*misclassify*). Berikut adalah kode program untuk evaluasi terhadap data yang *misclassify*.

```
# Which songs did the svm model misclassify?
test_data['preds_svm'] = preds_svm
misclasses_svm = test_data[test_data.preds_svm !=
test_data['mood']]
misclasses_svm['misclass_combo'] = misclasses_svm.apply(lambda x:
x['mood']+'-'+x['preds_svm'], axis=1)
```

Gambar 49. Kode Program *Check Data* yang *Misclassify*

Selanjutnya dilakukan perhitungan terhadap jumlah data yang *misclassify* seperti yang ditunjukkan pada Gambar 50 berikut.

```
# Melakukan perhitungan terhadap jumlah data yang misclassify
berdasarkan model yang telah dibangun
misclasses_svm.misclass_combo.value_counts()
```

Gambar 50. Kode Program Perhitungan Jumlah data yang *Misclassify*

Hasil perhitungan berdasarkan kode program pada Gambar 50 adalah jumlah data yang *misclassify* dimana nilai *actual*-nya adalah *happy* namun diprediksi *sad* adalah 38 sementara jumlah data yang *misclassify* dimana nilai *actual*-nya adalah *sad* namun diprediksi *happy* adalah 21. Hasil perhitungan tersebut dapat dilihat pada Gambar 51 berikut.

```
happy-sad    38
sad-happy    21
Name: misclass_combo, dtype: int64
```

Gambar 51. Perhitungan Jumlah data yang *Misclassify*

Berikut merupakan kode program yang digunakan untuk menampilkan data yang *misclassify* dimana nilai *actual*-nya adalah *happy* namun diprediksi *sad.*

```
# Menampilkan data yang yang misclassify dimana nilai actualnya
adalah happy, namun diprediksi sad
misclasses_svm[misclasses_svm.misclass_combo=='happy-sad']
```

Gambar 52. Kode Program Menampilkan Data yang Misclassify (*Nilai Actual happy*, namun diprediksi *sad*)

Berikut merupakan kode program yang digunakan untuk menampilkan data yang *misclassify* dimana nilai *actual*-nya adalah *sad* namun diprediksi *happy.*

```
# Menampilkan data yang yang misclassify dimana nilai actualnya
adalah sad, namun diprediksi happy
misclasses_svm[misclasses_svm.misclass_combo=='sad-happy']
```

Gambar 53. Kode Program Menampilkan Data yang Misclassify (*Nilai Actual sad*, namun diprediksi *happy*)

2.4.5 *Accuracy SVM Model with OneVsRestClassifier*

Pada bagian ini dilakukan pembangunan model SVM dengan *OneVsRestClassifier*. Berikut adalah kode program untuk model SVM dengan *OneVsRestClassifier.*

```
# Membangun SVM model dengan OneVsRestClassifier
from sklearn import svm
from sklearn.multiclass import OneVsRestClassifier
clf = OneVsRestClassifier(svm.SVC(gamma=0.02, C=100.,
probability=True, class_weight='balanced', kernel='linear'))
```

```
clf_output = clf.fit(train_data, targets_train)

# Menampilkan nilai akurasi dari model SVM yang telah dibangun
dengan train_data
clf.score(train_data, targets_train)
```

Gambar 54. Kode Program *Accuracy* SVM Model with *OneVsRestClassifier train_data*

```
# Menampilkan nilai akurasi dari model SVM yang telah dibangun
dengan test_data
clf.score(test_data, targets_test)
```

Gambar 55. Kode Program *Accuracy* SVM Model with *OneVsRestClassifier test_data*

Hasil nilai *accuracy SVM* dengan *OneVsRestClassifier* pada train_data yang diperoleh ditunju kkan pada Gambar 55 berikut. Hasil menunjukkan bahwa nilai *accuracy*-nya adalah 0.997938 1443298969

```
0.9979381443298969
```

Gambar 56. *Accuracy SVM Model with OneVsRestClassifier train_data*

Hasil nilai *accuracy SVM* dengan *OneVsRestClassifier* pada test_data yang diperoleh ditunjuk kan pada Gambar 55 berikut. Hasil menunjukkan bahwa nilai *accuracy*-nya adalah 0,7333333 333333333

```
0.7333333333333333
```

Gambar 57. *Accuracy SVM Model with OneVsRestClassifier test_data*

BAB 3
PENUTUP

Pada bab ini dijelaskan mengenai pembagian tugas dan tanggung jawab dalam pengerjaan proyek, kesimpulan yang diperoleh, dan saran terhadap proyek ke depannya.

3.1 Pembagian Tugas dan Tanggung Jawab

Pada subbab ini dijelaskan pembagian tugas dan tanggung jawab dari setiap anggota dalam pengerjaan proyek.

Tabel 6. Pembagian Tugas dan Tanggung Jawab

Name	*Role*	*Task*
Ernike Manurung	*Data Analyst*	Berperan dalam mengumpulkan, mengidentifikasi, menafsirkan serta menganalisis data, model, dan strategi yang efisien untuk digunakan dalam pengerjaan proyek.
	Programmer	Berperan dalam mengimplementasikan code untuk membangun sistem dan melakukan pengujian terhadap sistem yang sudah dibangun.
Inggrit Purba	*Data Analyst*	Berperan dalam mengumpulkan, mengidentifikasi, menafsirkan serta menganalisis data, model, dan strategi yang efisien untuk digunakan dalam pengerjaan proyek.
	Programmer	Berperan dalam mengimplementasikan code untuk membangun sistem dan melakukan pengujian terhadap sistem yang sudah dibangun.
Yohana Aritonang	*Data Analyst*	Berperan dalam mengumpulkan, mengidentifikasi, menafsirkan serta menganalisis data, model, dan strategi yang efisien untuk digunakan dalam pengerjaan proyek.
	Programmer	Berperan dalam mengimplementasikan *code* untuk membangun sistem dan melakukan pengujian terhadap sistem yang sudah dibangun.

3.2 Kesimpulan

Penggunaan *sentiment analysis* pada *song lyrics* dengan menerapkan metode *Support Vector Machine* (SVM) dapat menganalisis lirik lagu yang ditampilkan pada *dataset* dengan label *happy dan sad*. Analisis sentimen pada lirik lagu pada dataset (*train set* dan *test set*) diimplementasikan menggunakan pemodelan *Support Vector Machine* (SVM) dengan TF-IDF, pemodelan *Support Vector Machine* (SVM) dengan Word2Vec, *Support Vector Machine* (SVM) dengan *K-Fold Cross Validation*, dan pemodelan dengan *Linear SVM Classifier*. Setelah dievaluasi analisis sentimen dengan pemodelan tersebut menghasil hasil akurasi secara berurutan 0,60, 0,68, 0,70, dan 0,73. Berdasarkan hasil pemodelan tersebut pemodelan yang lebih akurat dan lebih sesuai untuk menganalisis lirik lagu yang diberikan label *happy dan sad* tersebut adalah pemodelan dengan *Linear SVM Classifier* dengan hasil akurasi 0,73.

3.3 Saran

Beberapa saran sekiranya dapat membantu meningkatkan efektivitas dari analisis sentimen pada lirik lagu sebagai berikut:

1. Penambahan jumlah data sekiranya dapat membantu dalam memperbanyak kosa kata dalam *data training.*
2. Penambahan kategori pada kelas label *mood* dengan kategori lainnya, misalnya *relaxed, angry*, dll.
3. Pemodelan yang dilakukan dapat dilakukan dengan menggunakan algoritma yang lainnya untuk menghasilkan prediksi analisis sentimen pada lirik lagu yang lebih akurat, misalnya dengan menggunakan Bi-LSTM, Maximum Entropy, dll.

DAFTAR PUSTAKA

[1] F. S. Sinaga, Indriati and B. Rahayudi, "Klasifikasi Emosi Lirik Lagu menggunakan Improved K-Nearest Neighbordengan Seleksi Fitur dan BM25," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* vol. 3, no. 6, pp. 5697-5702, 2019.

[2] B. Liu, Sentimen Analysis : Mining Opinions, Sentiments, and Emotions, Chicago: Cambridge University Press, 2014.

[3] F. F. Haranto and B. W. Sari, "Implementasi Support Vector Machine Untuk Analisis Sentimen Pengguna Twitter Terhadap Pelayanan Telkom Dan Biznet," *Jurnal PILAR Nusa Mandiri,* vol. 15, no. 2, pp. 171-176, 2019.

[4] R. Tineges, A. Triayudi and I. D. Sholihati, "Analisis Sentimen terhadap Layanan Indihome Berdasarkan Twitter Dengan Metode Klasifikasi Support Vector Machine (SVM)," *Jurnal Media Informatika Budidarma,* vol. 4, no. 3, pp. 650-658, 2020.

[5] E. Indrayuni, "Komparasi Algoritma Naive Bayes dan Support Vector Machine Untuk Analisa Sentimen Review Film," *Jurnal PILAR Nusa Mandiri,* vol. 14, no. 2, pp. 175-180, 2018.

[6] S. Kannan and V. Gurusamy, "Preprocessing Techniques fo Text Mining," *ResearchGate,* 2015.

[7] I. P. Windasari, F. N. Uzzi and K. I. Satoto, "Sentiment Analysis on Twitter Posts: An Analysis of Positive or Negative Opinion on Gojek," *Proc. of 2017 4th Int. Conf. on Information Tech., Computer, and Electrical Engineering (ICITACEE),* 2017.

[8] D. K. Srivastava and L. Bhambhu, "Data Classification Using Support Vector Machine," *Journal of Theoretical and Applied Information Technology,* 2005 - 2009.